KRZYSZTOF WOJTUSZKIEWICZ

URZĄDZENIA TECHNIKI KOMPUTEROWEJ
CZĘŚĆ 1
JAK DZIAŁA KOMPUTER?

Projekt okładki: **Michał Rosiński**

Redakcja: **Matylda Pawłowska**

Skład komputerowy: **Krzysztof Świstak**

Książka przeznaczona jest dla słuchaczy szkół policealnych o specjalności technik informatyk, dla uczniów technikum o specjalności systemy mikroprocesorowe oraz dla każdego, kto jest zainteresowany tym jak działa komputer. Cała ość pracy składa się z dwóch części - pierwsza omawia architekturę i działanie komputera, druga działanie urządzeń peryferyjnych i ich współpracę z jednostką centralną.
Obecne wydanie książki zostało gruntownie zmienione w celu uwzględnienia zarówno zmian w technice komputerowej jak i dostosowania podręcznika do wymagań nowej formy egzaminu potwierdzającego kwalifikacje zawodowe w zawodzie technik informatyk.

ISBN-13:978-83-01-15035-8
ISBN-10: 83-01-15035-8

Wydawnictwo Naukowe PWN SA
00-251 Warszawa, ul. Miodowa 10
tel.(0 22)69 54 321
faks (0 22) 69 54 031
e-mail: pwn@pwn.com.pl
www.pwn.pl

Wydawnictwo Naukowe PWN SA
Wydanie pierwsze
Arkuszy drukarskich 20
Druk ukończono w lutym 2007 r.
Druk i oprawa:
Drukarnia Wydawnictw Naukowych Sp. z o.o.
90-450 Łódź, ul. Żwirki 2

Spis treści

Marii

Podziękowania

Pragnę w tym miejscu podziękować wszystkim, którzy przyczynili się do powstania tej książki. W szczególności dziękuję Panu mgr inż. Tadeuszowi Główczyńskiemu za udostępnienie podzespołów komputera typu laptop, i mojemu synowi Marcinowi za pomoc w przygotowaniu zdjęć do książki.

Dziękuję także za bardzo miłą współpracę oraz za wyrozumiałość paniom z redakcji PWN-Mikom: Pani Redaktor Naczelnej Iwonie Krajewskiej-Kranas, Pani Magdalenie Scibor i Pani Dorocie Świstak.

Specjalne podziękowanie chcę przekazać zającowi Maciusiowi za cierpliwe pozowanie do zdjęcia oraz jego wielbicielce, Pani Marii Pikulskiej-Woźniczce za zdjęcia tego udostępnienie.

Chciałbym także podziękować wszystkim moim słuchaczom z Policealnego Studium Zawodowego za wyrozumiałość i współpracę. To dzięki Wam narodził się pomysł napisania tej książki i powstało wiele pomysłów dotyczących jej treści.

Krzysztof Wojtuszkiewicz

Przedmowa

Od pierwszego wydania książki *Urządzeniu techniki komputerowej. Jak działa komputer* upłynęło już 7 lat, od jej ostatniego uaktualnienia 4. Siedem lat to w technice komputerowej cała epoka. Dlatego też nowe wydanie tej książki wymagało gruntownych zmian. Zmieniony został układ treści książki. Niezmienione zostały, z oczywistych powodów, fragmenty książki dotyczące podstaw techniki cyfrowej i podstaw działania komputera. Natomiast rozdziały dotyczące budowy komputera typu IBM-PC i jego podzespołów zostały przeze mnie gruntownie przepracowane. Rozbudowany został rozdział dotyczący pamięci półprzewodnikowych. Całkowicie zmieniony i praktycznie napisany od nowa jest rozdział o rodzinie procesorów x86. Znacznych zmian wymagał także rozdział o płytach głównych. W książce dodano dwa nowe, krótkie rozdziały, o związku budowy komputera z działaniem systemów operacyjnych, i o zasilaczach komputerów typu IBM-PC. Ponadto w części rozdziałów dodana została sekcja *Praktyka,* w której starałem się pokazać w sposób praktyczny budowę komputera typu IBM-PC i jego podzespołów.

W podręczniku starałem się uwzględnić wszelkie zmiany w programie nauczania przedmiotu „Urządzenia techniki komputerowej" wykładanego w ramach specjalności technik-informatyk, a także wymagania nowej formy egzaminów potwierdzających kwalifikacje zawodowe w zawodzie technik-informatyk.

Przedmowa do wydania pierwszego

Pisząc książkę *Urządzenia techniki komputerowej. Juk działu komputer,* postawiłem sobie dwa zadania. Pierwsze z nich wynika z mojego przekonania, że komputer mimo swej fizycznej złożoności jest urządzeniem zbudowanym i działającym w sposób bardzo logiczny. Wynika z tego, że jeżeli problem budowy komputera oraz jego działania rozbijemy na wiele prostszych zagadnień, które następnie połączymy w logiczną i spójną całość, to zrozumienie działania komputera nie powinno sprawić kłopotu nawet użytkownikom o zainteresowaniach zdecydowanie humanistycznych. To, czy z zadania tego udało mi się wywiązać, pozostawiam ocenie Czytelników. Z tego też względu będę bardzo wdzięczny za wszelkie uwagi, szczególnie krytyczne, które proszę przesyłać na jeden z niżej podanych adresów. Przyczynią się one do ulepszenia ewentualnych przyszłych wydań niniejszej książki.

Drugie z zadań wiąże się z przedmiotem „Urządzenia techniki komputerowej" wykładanym w policealnym studium zawodowym o specjalności „technik informatyk". Książka ta jest pomyślana jako pierwsza część dwutomowego podręcznika do tego przedmiotu. Jest ona zgodna z bieżącym programem i pokrywa cały materiał wykładany w ramach pierwszego roku. Druga część podręcznika znajduje się obecnie w opracowaniu. Pomysł napisania podręcznika wziął się stąd, że przedmiot ten wykładam już od siedmiu lat (od początku powstania specjalności „technik informatyk"). Jednym z problemów, jakie napotkałem przy jego prowadzeniu, był brak jednej książki pokrywającej większość materiału. Potrzebne wiadomości, nawet jeżeli są dostępne, znajdują się w wielu różnych publikacjach. Opracowanie to jest więc próbą ułatwienia pracy zarówno wykładowcom tego przedmiotu, jak i uczniom.

Niniejsza edycja jest drugim wydaniem tej książki. Pierwsze ukazało się pod tytułem *Jak działa komputer PC* w serii „Systemy mikroprocesorowe" wydawnictwa CKP z Wrocławia. W tym wydaniu wprowadzono niewielkie zmiany uaktualniające treść oraz poprawiono kilka dostrzeżonych błędów.

Drugi tom podręcznika będzie dotyczył działania urządzeń peryferyjnych komputera, takich jak napędy dyskowe, monitory, adaptery graficzne i wiele innych. Dla poszczególnych urządzeń zostanie przedstawiona zasada ich działania, parametry oraz sposób komunikacji z systemem, co w sposób naturalny będzie wiązało się z materiałem z pierwszej części.

Na zakończenie pragnę stwierdzić, że dołożyłem wszelkich starań, aby wiadomości zawarte w książce były jak najbardziej aktualne. Elektronika i informatyka są jednak dziedzinami tak szybko rozwijającymi się, że muszę prosić Czytelników o wybaczenie, jeżeli nie znaleźli w książce wyjaśnienia interesujących ich najnowszych zagadnień.

Krzysztof Wojtuszkiewicz

Ewentualną korespondencję proszę nadsyłać na jeden z poniższych adresów:

- Elektroniczne Zakłady Naukowe, ul. Ostrowskiego 22, 53-238 Wrocław
- e-mail: kawojt@masters.ckp.pl

Wstęp

Zamierzeniem podręcznika Urządzenia techniki komputerowej - Jak działa komputer jest przedstawieni w sposób logiczny i uporządkowany wiadomości potrzebnych do zrozumienia funkcjonowania komputera, przy założeniu, że czytelnik nie posiada żadnych wiadomości na temat zarówno techniki komputerowej jak i cyfrowej. Z założeń tych wynika treść i układ poszczególnych rozdziałów.

Rozdział pierwszy jest pierwszym spojrzeniem na urządzenie, które jest przedmiotem książki. Jednakże już tu staramy się pokazać, że jeżeli spojrzy się na komputer w sposób uporządkowany, to jego budowa, w przeciwieństwie do pierwszego wrażenia, jakie często odnosimy, wyda się zdecydowanie prostsza.

Rozdział drugi zawiera trzy bloki informacji. Pierwszy z nich, rozdział 2.1, to podstawy funkcjonowania układów cyfrowych. Drugi blok, rozdział 2.2, przedstawia działanie podstawowe układy cyfrowe, istotnych dla zrozumienia funkcjonowania bardziej skomplikowanych układów komputera, takich jak na przykład mikroprocesor. Wreszcie trzeci blok zajmuje się budową i funkcjonowaniem pamięci, będących jednymi z bardzo ważnych części składowych komputera.

Rozdział trzeci przedstawia podstawy architektury komputerów. Prezentowane w nim wiadomości i pojęcia dotyczą ogólnie komputerów, nie są więc związane tyko z komputerami typu IBM-PC. Rozdział ten zawiera między innymi podstawowe wiadomości o zasadach działania procesora i współdziałaniu jego poszczególnych elementów. Zrozumienie i dobre opanowanie materiału tego rozdziału umożliwia spojrzenie w sposób logiczny i uporządkowany na funkcjonowanie komputera, będącego systemem mikroprocesorowym. W rozdziale wyjaśnione są między innymi takie ważne pojęcia jak przerwania, bezpośredni dostęp do pamięci, pamięć cache i pamięć wirtualna.

Począwszy od rozdziału czwartego, przedstawiane wiadomości są ukierunkowane na budowę i funkcjonowanie komputera typu IBM-PC. Dlatego też w rozdziale tym zapoznajemy z budową mikroprocesorów rodziny Intel x86,stosowanych w tych komputerach. Budowę tych procesorów prezentujemy od najprostszego i jednocześnie najstarszego procesora tej rodziny - Intel'a 8086, a kończymy na procesorach Pentium 4 i Itanium. Taki sposób prezentacji tych procesorów ma dwojaki cel. Po pierwsze pozwoli w sposób względnie prosty wyjaśnić budowę tych procesorów, zaczynając od prostego, a następnie pokazując coraz nowsze rozwiązania. Po drugie sposób ten pozwoli ocenić kierunki zmian, jakie zachodzą w budowie procesorów.

Rozdział piąty jest próbą spojrzenia na związki pomiędzy pewnymi funkcjami realizowanymi przez system operacyjny a rozwiązaniami sprzętowymi wprowadzonymi przede wszystkim w mikroprocesorach.

Rozdział szósty porusza zagadnienia związane z budową i działaniem płyt głównych. Po kolei omawiane są: standard ISA, rola i zadania BIOSu, chipsety, standardy magistrali rozszerzających, standard Plug and Play oraz formaty płyt głównych.

Rozdział siódmy, ostatni, omawia krótko budowę zasilaczy stosowanych w komputerach typu IBM-PC.

Załączona na końcu książki bibliografia umożliwia odnalezienie bardziej szczegółowych informacji na poszczególne tematy. Odnośniki do poszczególnych pozycji bibliografii znajdują się w tekście książki.

1. Komputer PC od zewnątrz

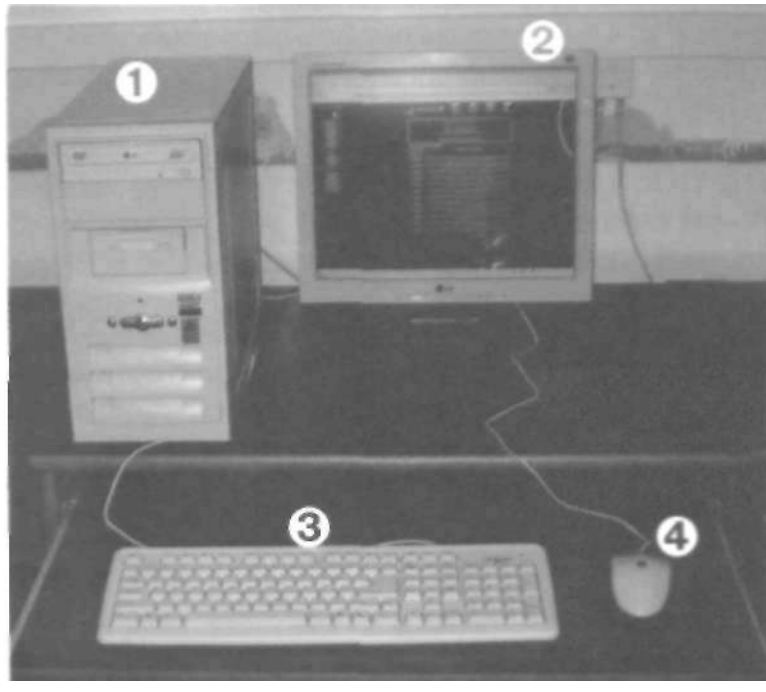
1.1. Elementy zestawu komputerowego

Nasze spotkanie z techniką komputerową rozpoczniemy od zapoznania się z wyglądem i elementami typowego zestawu tworzącego komputer osobisty. Rysunek 1.1 przedstawia komputer stacjonarny, na rysunku 1.2 pokazujemy komputer typu notebook.

Komputer stacjonarny

Podstawowe, najczęściej spotykane elementy stacjonarnego komputera osobistego pokazane na rysunku 1.1 to:

1. jednostka centralna, często nazywana po prostu komputerem,
2. monitor,
3. klawiatura,
4. mysz.



Rysunek 1.1. Komputer stacjonarny

Jednostce centralnej poświecimy za chwile więcej uwagi. Trzy następne elementy służą do komunikowania się człowieka z komputerem. Monitor wys'wietla informacje dostarczane przez działające na komputerze programy komputerowe (aplikacje, system operacyjny itp.). Klawiatura pozwala wprowadzać człowiekowi do komputera dane mogące być liczbami, poleceniami i innymi rodzajami informacji. Mysz jest tak zwanym urządzeniem wskazującym, które w połączeniu z ekranem monitora i wyświetlanym na nim graficznym interfejsem użytkownika (ang. *graphical user interface - GUI)* także pozwala na wprowadzanie informacji do komputera.

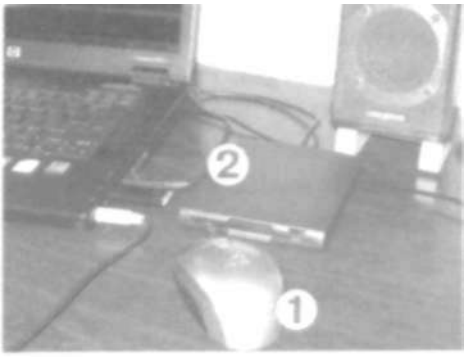
Monitor, mysz i klawiatura są przykładami urządzeń peryferyjnych (ang. *peripherals, peripheral devices).* Innymi przykładami urządzeń peryferyjnych są: drukarka, skaner, modem, ploter, napęd dysku twardego, stacja dysków elastycznych (dyskietek), napęd DVD i wiele innych. Proszę zauważyć, że niektóre z wymienionych urządzeń są z reguły osobnymi urządzeniami (na przykład drukarka czy ploter), inne są zwykle zamontowane w jednostce centralnej (na przykład napęd dysku twardego czy stacja dysków elastycznych, choć tu możliwe są nieliczne odstępstwa), wreszcie pewna grupa urządzeń równie często będzie urządzeniem zewnętrznym lub zamontowanym w jednostce centralnej (modemy: wolnostojące i w postaci kart). Urządzeniom peryferyjnym pos'więcona jest druga część podręcznika „Urządzenia techniki komputerowej", zatytułowana „Urządzenia peryferyjne i interfejsy".

Komputer typu notebook

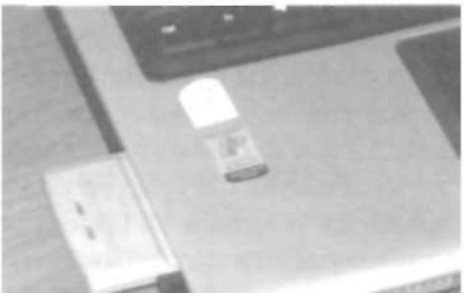
Wygląd przykładowego komputera osobistego typu notebook przedstawia rysunek 1 . 2 . Możemy tu wyróżnić elementy podobne do komputera stacjonarnego, lecz tym razem stanowią one nierozłączną całość. Są to: jednostka centralna (1) z umieszczoną na jej wierzchu klawiaturą i urządzeniem wskazującym, zwanym po polsku gładzikiem (2) - częściej jednak, zwłaszcza w żargonie, używana jest nazwa angielska touch pad. Gładzik zastępuje w tym zestawie mysz, którą jednak można także podłączyć do notebooka. Na przykład na rysunku 1 . 3 pokazana jest mysz bezprzewodowa (1) podłączona przez magistralę USB.

Elementem zestawu jest także ekran ciekłokrystaliczny (3). Inaczej przedstawiają się możliwości zmiany konfiguracji sprzętowej notebooka. Nieco mniejsze możliwości zmian w jego wnętrzu (brak kart) kompensowane są częściowo możliwością podłączenia dodatkowych urządzeń przez złącza kart magistrali PCMCIA, Express Card (złącza te widać na rysunku 1.10) czy jako zewnętrzne urządzenia obsługiwane przez magistralę USB. Na rysunku 1 . 3 a jest to przykładowo stacja dysków elastycznych (2), a na rysunku 1.3c bezprzewodowa karta sieciowa.

a)



b)



Rysunek 1.3. *(a)* Mysz bezprzewodowa i stacja dysków elastycznych podłączone przez magistralę USB, (b) widok bezprzewodowej karty sieciowej PCMCIA (PC Card) i (c) karta w złączu

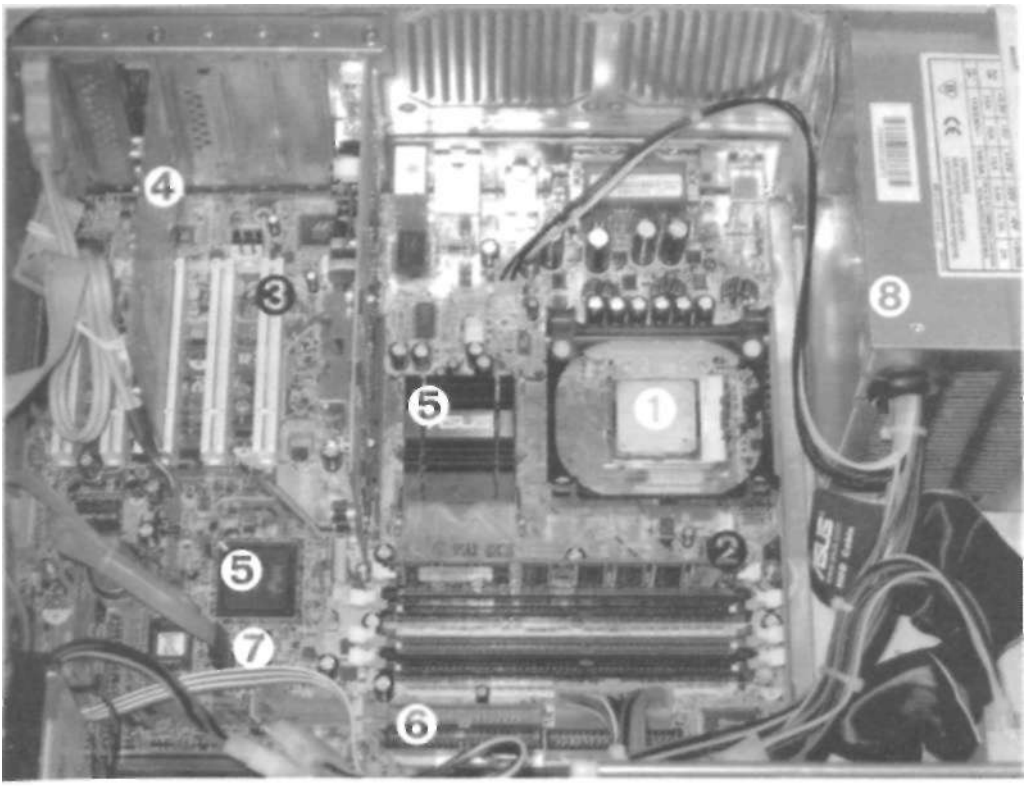
1.2. Podzespoły wchodzące w skład jednostki centralnej

Obecnie zajmiemy się koncepcją budowy jednostki centralnej komputera osobi- I stego. Pierwsza części książki „Urządzenia techniki komputerowej" zajmuje się właśnie podstawami działania dużej części układów jednostki centralnej, takich jak mikroprocesor, pamięć, płyta główna, układ przerwań i tak dalej. Terminy tu wymienione, być może nieznane Czytelnikowi, zostaną omówione w tej książce.

Podstawową ideą, którą ma realizować koncepcja budowy komputera typu PC, jest elastyczność jego konfiguracji sprzętowej, czyli możliwość dostosowania jego budowy do wymagań właściciela czy użytkownika. Wymagania te wynikają z celów, do jakich jest przeznaczony komputer, oczekiwań co do jego nowoczesności oraz ze względów ekonomicznych (niestety dwa ostatnie czynniki często są ze sobą sprzeczne).

W celu zapewnienia możliwości dostosowywania budowy komputera do wymagań użytkownika ma on budowę modułową, czyli składa się z bloków, które mogą być wymieniane. Podstawowym elementem jednostki centralnej będącym nośnikiem kluczowych elementów systemu komputerowego jest płyta główna (ang. *main board, matherboard),* której przykład pokazano na rysunku 1.4. Elementami tymi są (w nawiasach podajemy numery odnoszące się do rysunku 1.4): mikroprocesor (1), pamięć półprzewodnikowa (obecnie w postaci modułów) (2), gniazda tak zwanej magistrali rozszerzającej (ang. *expansion slots)* (3), w których można instalować wybrane urządzenia w postaci płytek elektronicznych zwanych kartami (4). Płyta główna zawiera także układy sterujące (na przykład w postaci chipsetów) (5) (jeden z nich znajduje sie pod radiatorem) i gniazda wybranych interfejsów. Na naszym zdjęciu są to przykładowo: EIDE (6) lub SATA (7), omówione dokładniej w rozdziale 6. w części „Praktyka". Na rysunku 1.4 widać też blok zasilacza (8) zamontowany w obudowie komputera.

Elastyczność budowy PC jest zapewniana między innymi przez umieszczanie pewnych elementów w specjalnych gniazdach, w których mogą być łatwo wymieniane. Przykładem jest tu gniazdo procesora, gniazda modułów pamięci czy właśnie gniazda magistrali rozszerzającej. Ewolucję gniazd pamięci krótko omawiamy w podrozdziale 2.3.6, a magistrale rozszerzające omówione są w rozdziale 6. w punkcie 6.4 i części „Praktyka". Wygląd przykładowych złączy interfejsów na płytach głównych i sposób ich połączenia za pomocą pasm i kabelków z urządzeniami peryferyjnymi montowanymi wewnątrz jednostki centralnej pokazujemy w rozdziale 6. dotyczącym płyt głównych w części „Praktyka". Tam też omawiamy dokładniej wybrane elementy płyty głównej, takie jak procesor czy chipsety. Rodzaje złączy pojawiające się na ściance tylnej lub frontowej jednostki centralnej (komputera) prezentowane są w tym rozdziale.

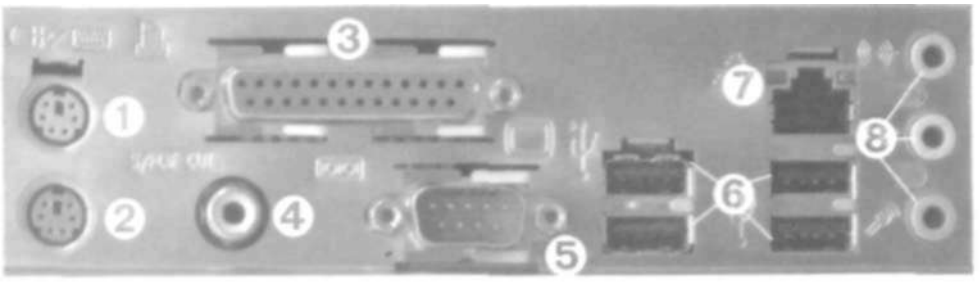


Rysunek 1.4. Płyta główna

- Praktyka

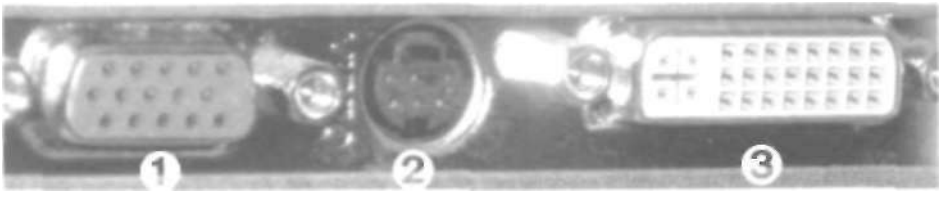
Na kolejnych zdjęciach pokazujemy złącza występujące na tylnej, a czasami i przedniej ściance komputera stacjonarnego oraz w komputerze typu notebook. Gniazda te budzą często panikę, szczególnie u początkujących użytkowników komputera. Jest to jednak odruch błędny. Gniazda są dobierane i skonstruowane tak, że jeżeli tylko rezygnujemy z użycia siły, to nieprawidłowe wykonanie połączenia jest praktycznie niemożliwe. Proszę zwrócić uwagę, co podkreślamy w opisie gniazd, że różnią się one kształtem i wyglądem, a wtyki urządzeń do nich podłączanych są do nich ściśle dopasowane. To właśnie praktycznie uniemożliwia podłączenie urządzenia do niewłaściwego gniazda.

Na rysunku 1 . 5 widać następujące gniazda: myszy (1) i klawiatury (2) w standardzie PS/2, portu równoległego (3) (oznaczane skrótem LPT - *line printer,* gdyż dawniej obsługiwało przede wszystkim drukarkę), wyjście liniowe audio (4) mogące pracować w standardzie S/PDIF, port szeregowy RS 232C (5) (oznaczany często skrótem COM), cztery porty USB (6), gniazdo adaptera sieciowego (7) (RJ 45 - Ethernet - skrętka) oraz trzy gniazda kart dźwiękowej (8).



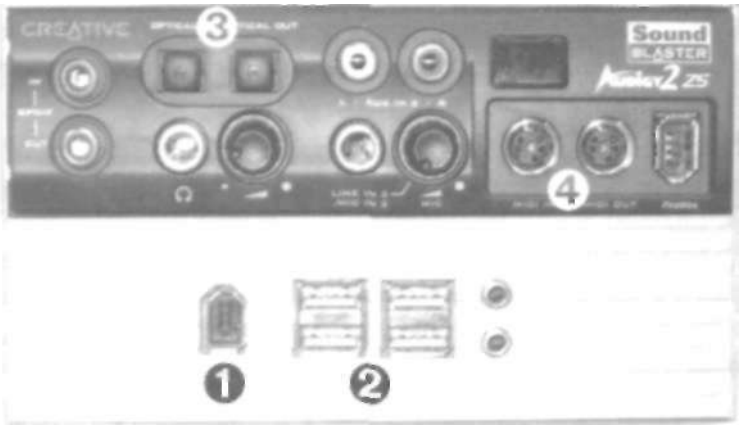
Rysunek 1.5.

Pewne obawy budzą czasami gniazda typu DB, na przykład (3), (5) czy też (1) na rysunku 1 . 6 . Wydają się podobne. Proszę jednak zwrócić uwagę na to, że gniazda te mogą być typu męskiego (z wystającymi bolcami) lub żeńskiego (z otworami). Dodatkowo gniazdo (5) z rysunku 1 . 5 i gniazdo (1) z rysunku 1 . 6 różnią się liczbą r z ę dów w złączu (pierwsze - 2, drugie - 3 rzędy). Wszystko to stanowi cechy, które pozwalają jednoznacznie zidentyfikować gniazdo.



Rysunek 1 . 6 .

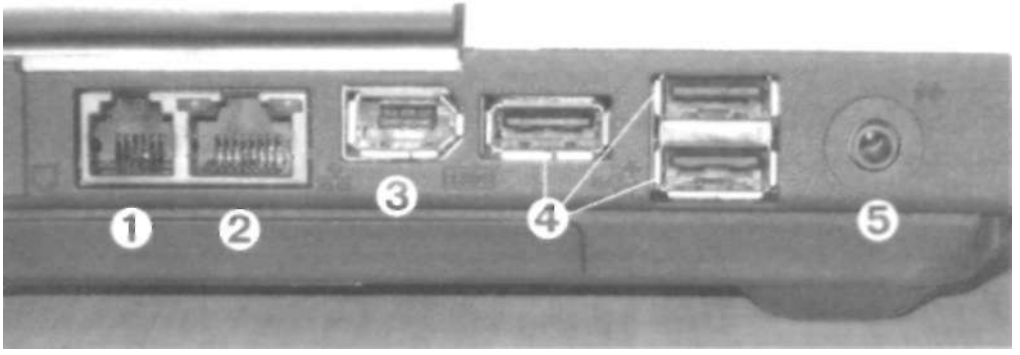
Na rysunku 1.6 widzimy kolejno gniazdo monitora (karty (S)VGA (1) (nazywane czasami D-SUB), gniazdo wyjs'cia S Video (2) oraz gniazdo interfejsu cyfrowego monitora - DVI (3).



Rysunek 1 . 7 .

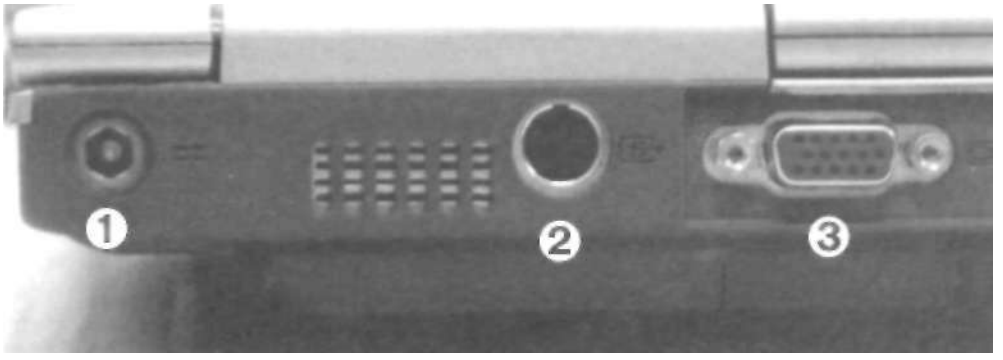
Na rysunku 1.7 w dolnej części widzimy gniazdo interfejsu FireWire (1) i cztery porty USB (2). Powyżej widać panel profesjonalnej karty dźwiękowej Sound Blaster Audigy, między innymi z wejściem i wyjściem optycznym (3) oraz gniazdami MIDI (4) (o kartach dźwiękowych piszemy w drugiej części książki).

Dalsze zdjęcia dotyczą komputera typu laptop/notebook (nazwy te są obecnie używane zamiennie, choć w istocie oznaczały komputery nieco różniące się wyglądem i wielkością).



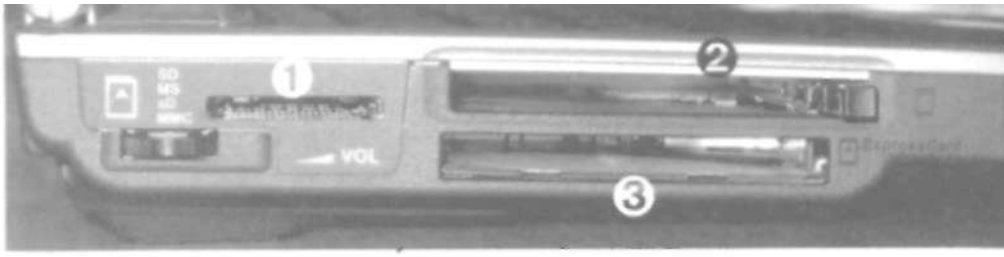
Rysunek 1.8.

Kolejne gniazda na rysunku 1.8 to: gniazdo modemu (1), gniazdo sieciowe (Ethernet) (2), interfejs IEEE 1394 (FireWire) (3), trzy gniazda USB (4) i gniazdo wyjścia liniowego (S/PDIF) (5).



Rysunek 1.9.

Na zdjęciu 19 widoczne jest gniazdo zasilania (I), wyjście SVideo (2) i wyjście na monitor VGA (D SUB) (3)



Rysunek 1.10.

Na rysunku 1.10 widoczny jest czytnik kart pamięci (SD, MS, xD, MMC) (1) oraz złącza kart typu PC Card II (PCMCIA) (2) i Express Card (3), umożliwiające podłączanie dodatkowych urządzeń. Przykładem takiego urządzenia jest karta sieci bezprzewodowej pokazana na rysunku 1.11.

Rysuacfc 1.11.

2. Układy cyfrowe

Wstęp

Przełom dwudziestego i dwudziestego pierwszego wieku można śmiało nazwać erą informatyki i mikroprocesorów. Obydwa te obszary zrewolucjonizowały nasze życie. Systemy i układy mikroprocesorowe są wszechobecne, od komputerów począwszy, przez sprzęt powszechnego użytku, motoryzację, do telekomunikacji. Wszystkich dziedzin zresztą wymienić nie sposób.

Mikroprocesory są układami cyfrowymi. Kolejną tendencją, którą łatwo zauważyć, jest wypieranie w elektronice, w wielu dziedzinach, techniki analogowej przez technikę cyfrową. Może więc należałoby powiedzieć, że żyjemy w wieku techniki cyfrowej i informatyki (która bez techniki cyfrowej pozostałaby prawdopodobnie teorią).

Poniższy podrozdział w sposób bardzo skrótowy odpowie na pytanie, jakie są przyczyny tak wielkiej popularności techniki cyfrowej, i wyjaśni podstawy jej działania.

2.1. Podstawy działania układów cyfrowych

W celu przedstawienia podstaw techniki cyfrowej omówimy kolejno: ideę działania układów cyfrowych, jej konsekwencje dotyczące postaci informacji, sposób realizacji układów cyfrowych i ich podział.

2.1.1. Idea działania układów cyfrowych

Idea funkcjonowania układów cyfrowych oparta jest na założeniu, ze wszelka m formacja i wszelkie wielkości przetwarzane przez te układy reprezentowane są przez dwa stany. Stany te możemy umownie nazywać zerem (0) i jedynką (1) lub stanem niskim (L) i wysokim (H). Tak przedstawioną informację lub wielkości nazywamy dyskretnymi, w przeciwieństwie do informacji lub wielkości analogowych, które reprezentowane są przez bardzo wiele położonych blisko siebie stanów. Okazało się, że informacja dyskretna, czyli informacja w postaci cyfrowej (dwójkowej, binarnej), ma w zastosowaniach elektronicznych bardzo istotne zalety. Należą do nich:

- Prostota układów elektronicznych realizujących przetwarzanie takiej postaci informacji. Elementy półprzewodnikowe, które są podstawą współczesnej elektroniki, mają z natury rzeczy duże rozrzuty swoich parametrów. Chcąc zniwelować wpływ tych rozrzutów na właściwości układów elektronicznych, które z nich bu-

dujemy, należy stosować odpowiednie rozwiązania. Okazuje się, że można to zrobić znacznie prościej dla układów cyfrowych, które muszą rozróżniać jedynie dwa stany elektryczne (o czym piszemy dokładniej w następnym podpunkcie). W przypadku układów rozróżniających więcej stanów lub dla układów analogowych jest to znacznie bardziej skomplikowane. Prowadzi to więc do prostoty układów cyfrowych, czego konsekwencją jest możliwa do uzyskania duża gęstość upakowania (na przykład we współczesnych procesorach kilkaset milionów tranzystorów w jednym układzie scalonym) oraz (co też niezmiernie ważne) relatywnie niska cena.

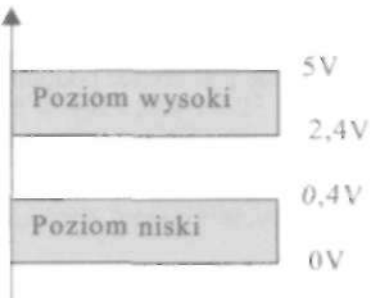
- Łatwość transmisji i odporność na zakłócenia. Informację reprezentowaną przez dwa stany, czyli informację cyfrową, można znacznie łatwiej przesyłać i regenerować, czyli odtwarzać jej niezakłóconą, niezmienioną postać. Wiąże się to nie tylko z samą postacią informacji, ale także z dodatkowymi możliwościami, oferowanymi na przykład przez informatykę, przykładowo w postaci tak zwanych kodów korekcyjnych. Dowodem tego jest chociażby zapis cyfrowy dźwięków - Compact Disc Digital Audio (czyli płyty CD) czy też cyfrowy zapis obrazów zarówno ruchomych, jak i nieruchomych. Kopiowanie takiej informacji nie powoduje utraty jakości, czego nie da się powiedzieć o zapisie analogowym.
- Łatwość konstruowania układów pamiętających. Jest to na pewno własność wynikająca w dużej mierze z punktu pierwszego. Jednak należy tu mieć na względzie także urządzenia zwane pamięciami masowymi, takie jak dyski twarde czy płyty DVD, pozwalające obecnie przechowywać ogromne ilości informacji, właśnie w postaci cyfrowej.

Wymienione zalety powodują, że technika cyfrowa sprawdza się w coraz większej liczbie zastosowań, w różnych dziedzinach. Wystarczy przyjrzeć się takim, jak telefonia, zwłaszcza komórkowa, sprzęt powszechnego użytku czy też właśnie technika komputerowa, wszechobecna w naszym życiu.

2.1.2. Poziomy logiczne

Układy cyfrowe są układami elektronicznymi i jako takie używają do reprezentowania informacji wartości wielkości elektrycznych: napięcia lub prądu. Jednak jak już powiedziano, wszelka informacja dla układów cyfrowych ma być przedstawiona za pomocą dwóch stanów, umownie zwanych na przykład **zerem** i **jedynką** lub **poziomami logicznymi.** Prostota układów cyfrowych wynika właśnie z faktu, że muszą rozróżniać tylko dwa stany. Należy więc określić, jakie wartości lub zakresy wartości będą oznaczać **zero,** czyli **poziom logiczny niski,** a jakie **jedynkę,** czyli **poziom logiczny wysoki.** W dużym przybliżeniu można to zrobić, mówiąc na przykład: jest napięcie - jedynka, nie ma napięcia (napięcie zerowe) - zero. W rzeczywistości trzeba oczywiście precyzyjnie określić, jakie zakresy napięć odpowiadają zeru, a jakie jedyn-

ce. Zakresy te mogą być różne dla różnych wykonań, czyli serii technologicznych układów cyfrowych. Nie powinny też być zbyt wąskie, gdyż wtedy wymagałoby to bardzo precyzyjnej realizacji układów, co czyniłoby je skomplikowanymi i drogimi. Rysunek 2.1 pokazuje, jak określono poziomy logiczne dla jednej z najbardziej znanych serii technologicznych układów cyfrowych - TTL.



Rysunek 2.1. Określenie poziomów logicznych

W celu zmniejszenia możliwości wystąpienia błędów, na przykład w wyniku występowania zakłóceń, okres'la się inne zakresy wartości poziomu niskiego i wysokiego dla wejs'ć, a inne dla wyjść układów, uzyskując tak zwany margines zakłóceń, co jest zilustrowane na rysunku 2.2.

Rysunek 2.2. Poziomy logiczne dla wejścia i wyjścia

Istnieje oczywiście możliwość innego określenia poziomów logicznych. Nie jest to jednak przedmiotem tej książki. Zainteresowanych odsyłamy do wielu pozycji książkowych dotyczących układów cyfrowych, na przykład [30].

2.1.3. System dwójkowy i szesnastkowy

Ponieważ każda informacja, która ma być przetwarzana przez układy cyfrowe, musi być reprezentowana przez dwie wartości, zwane na przykład zerem i jedynką logiczną, naturalnym staje się zainteresowanie systemem liczbowym dwójkowym, opartym właśnie na takim zapisie liczb. Pozwoli nam to zapisywać i przetwarzać liczby za pomocą układów cyfrowych. Opisujemy tu konstrukcję systemu dwójkowego, czyli binarnego, oraz szesnastkowego, czyli heksadecymalnego. Potrzeba i zastosowanie tego drugiego wyjaśnią się pod koniec podpunktu.

System dwójkowy

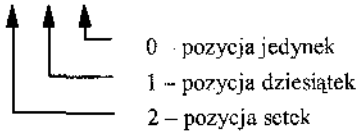
Ludzie w sposób naturalny przyzwyczajeni są do liczenia w systemie dziesiętnym, dlatego też konstrukcję i użycie systemu dwójkowego przedstawiamy przez analogię do systemu dziesiętnego.

Do zapisu dowolnej liczby bez znaku system dziesiętny wykorzystuje dziesięć symboli graficznych, zwanymi cyframi: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Przy ich użyciu jesteśmy w stanie przedstawić dowolną liczbę. System dziesiętny, podobnie jak dwójkowy, jest systemem pozycyjnym. Liczbę 425p (D oznacza zapis liczby w systemie dziesiętnym) możemy przedstawić jako następującą sumę:

$$425_D = 4 * 100 + 2 * 10 + 5 * 1$$

czyli:

$$4\ 2\ 5_D = 4 * 10^2 + 2 * 10^1 + 5 * 10^0$$



Widzimy więc, że cyfra na danej pozycji mnożona jest przez odpowiednią potęgę liczby 10, przy czym wykładnik tej potęgi zależy od położenia (pozycji) danej cyfry w liczbie. Uwaga! Pozycje cyfr w liczbie numerujemy od 0 (najmłodsza cyfra). Poszczególne mnożniki, zwane inaczej wagami, w systemie dziesiętnym noszą nazwę odpowiednio: jedynek ($10^0 = 1$), dziesiątek ($10^1 = 10$), setek ($10^2 = 100$) i tak dalej. Poszczególne wagi w systemie dziesiętnym są potęgami liczby 10, dlatego jest ona zwana podstawą tego systemu (p = 10). Podsumowując, formalny zapis $a_{n-1} \dots a_0$ w systemie dziesiętnym oznacza:

$$a_{n-1} \dots a_0{}_D = a_{n-1} * 10^{n-1} + a_{n-2} * 10^{n-2} + \dots + a_0 * 10^0 = \sum_{i=0}^{n-1} a_i * 10^i$$

gdzie i jest numerem pozycji w liczbie, natomiast a_i oznacza dowolną z cyfr od 0 do 9, a n jest liczbą cyfr (pozycji) w liczbie.

W systemie dziesiętnym dysponujemy dziesięcioma cyframi do zapisania dowolnej liczby bez znaku, w systemie dwójkowym musimy do tego celu używać jedynie dwóch cyfr: 0 i 1. Jak już wspomniano, obydwa systemy są **pozycyjne,** co oznacza, że cyfrę na danej pozycji mnoży się przez określoną wagę. Dla systemu dwójkowego podstawą jest liczba 2 (p = 2) i wagami są odpowiednie potęgi tej liczby. Kolejne pozycje liczby zwane są więc pozycjami jedynek, dwójek, czwórek, ósemek i tak dalej. Zapis w systemie dwójkowym, zwanym inaczej **systemem binarnym,** liczby 10100_B (litera B sygnalizuje liczbę w systemie dwójkowym) oznacza:

$$10100_B = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 =$$
$$1 * 16 + 0 * 8 + 1 * 4 + 0 * 2 + 0 * 1 = 16 + 4 = 20_D$$

Uogólniając, zapis $a_{n-1}.....a_{0B}$ w systemie dwójkowym będzie oznaczał:

$$a_{n-1}\cdots a_{0B} = a_{n-1} * 2^{n-1} + a_{n-2} * 2^{n-2} + ... + a_0 * 2^0 = \sum_{i=0}^{n-1} a_i * 2^i$$

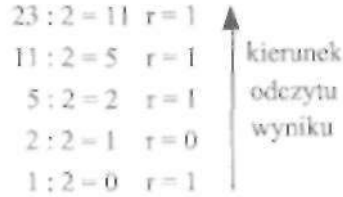
Wzór ten, określający sposób zapisu liczby w systemie dwójkowym, pozwala jednocześnie na dokonanie **konwersji** (przeliczenia) liczby zapisanej w systemie dwójkowym na liczbę zapisaną w systemie dziesiętnym.

Jedną z metod konwersji liczby dziesiętnej na dwójkową pokażemy na przykładzie, pomijając uzasadnienie jej poprawności. Metoda ta polega na wykonywaniu kolejnych dzieleń całkowitoliczbowych (wynik jest liczbą całkowitą) przez liczbę 2, z zapisem reszty. Rozpoczynamy od podzielenia liczby przeliczanej przez 2. Kolejne dzielenie wykonujemy na liczbie będącej ilorazem (wynikiem) poprzedniego dzielenia. Postępowanie kontynuujemy aż do momentu otrzymania jako wyniku 0. Reszty dzieleń ustawione w odpowiedniej kolejności dają poszukiwaną liczbę binarną.

Przykład

Dokonać konwersji liczby 23_D na liczbę binarną.

Rozwiązanie



A zatem $23_D = 10111_B$.

System heksadecymalny

System heksadecymalny, czyli szesnastkowy, nie jest używany bezpośrednio przez układy cyfrowe, stanowi natomiast wygodny, zwarty sposób zapisu liczb binar-

nych. Stosowany jest często przez programistów czy też przy wys'wietlaniu informacji f cyfrowej na ekranie (na przykład w programach typu *debuger).*

W systemie heksadecymalnym do zapisu dowolnej liczby dysponujemy szesnastoma cyframi. Ponieważ symboli graficznych oznaczających liczby arabskie jest i dziesięć, brakuje symboli sześciu cyfr. Przyjęto więc, że będą oznaczane początkowymi literami alfabetu (dużymi lub małymi). Zatem A oznacza dziesiątkę, B jedenastkę, aż do cyfry F, która oznacza piętnastkę. Pełny zestaw cyfr heksadecymalnych jest następujący:

$$a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

gdzie a_i oznacza cyfrę heksadecymalną. Jak łatwo sprawdzić, cyfr heksadecymalnych jest szesnas'cie. Liczba szesnaście jest też podstawą tego systemu. Formalny zapis $^{a}n-1\ldots{}^{a}0\,H$ oznacza więc:

$$a_{n-1}\cdots a_0{}_H = a_{n-1}*16^{n-1} + a_{n-2}*16^{n-2} + \ldots + a_0*16^0 = \sum_{i=0}^{n-1} a_i*16^i$$

gdzie a_i jest dowolną cyfrą heksadecymalną.

Przykład

Znaleźć liczbę dziesiętną odpowiadającą liczbie heksadecymalnej 4c2

Rozwiązanie

Zgodnie z podanym wzorem oraz wcześniejszymi informacjami:

$$4c2_H = 4 * 16^2 + C * 16^1 + 2 * 16^0 =$$
$$= 4 * 256 + 12 * 16 + 2 * 1 = 450_D$$

Konwersji liczby dziesiętnej na heksadecymalną można dokonać metodą analogiczną do pokazanej dla systemu dwójkowego, wykonując kolejne dzielenia z resztą przez liczbę 16. Należy jednak pamiętać, że reszty z dzielenia zapisujemy w postaci cyfr heksadecymalnych, czyli np. resztę 14 zapisujemy jako E.

Najistotniejszą cechą systemu heksadecymalnego jest łatwość przechodzenia od zapisu binarnego do heksadecymalnego i na odwrót, przez co zapis heksadecymalny staje się zwartym zapisem liczb binarnych. Pokażemy to na przykładzie.

Przykład

Zapisać liczbę binarną 1001011010_B w postaci liczby heksadecymalnej.

Rozwiązanie

Przy przejściu od liczby binarnej do heksadecymalnej wykorzystujemy fakt, że każdej cyfrze heksadecymalnej odpowiada określona kombinacja czterech cyfr binarnych i na odwrót. Pokazuje to tabela 2.1.

Przeliczaną liczbę binarną dzielimy od końca (czyli od najmłodszej pozycji) na czwórki, a następnie każdą z nich zapisujemy w postaci jednej cyfry heksadecymalnej, zgodnie z tabelą 2.1. Jeżeli ostatni fragment liczby nie jest pełną czwórką, możemy ją dopełnić do czwórki zerami. Tak więc dla liczby binarnej 001001011010:

$$001010101\ I\ 1010_{B} = 25A_{H}$$

Podobnie możemy postąpić przy przeliczaniu w drugą stronę. Wówczas każdą cyfrę heksadecymalną zapisujemy w postaci czwórki cyfr binarnych. Ewentualne nieznaczące zera na początku liczby binarnej można w wyniku pominąć.

Tabela 2.1. Cyfry heksadecymalne i odpowiadające im liczby binarne

Cyfra hex	Liczba binarna	Cyfra hex	Liczba binarna
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Przykład

Zapisać liczbę heksadecymalną $7cd5_{H}$ w postaci liczby binarnej.

Rozwiązanie

$$7cd5_{H} = 0111\ |\ 1100\ |\ 1101\ |\ 0101_{B} = 1111100110101011_{B}$$

Prosimy porównać długości liczb heksadecymalnych i odpowiadających im liczb dwójkowych. Wyjaśni to wygodę i sens stosowania zapisu heksadecymalnego.

2.1.4. Kodowanie informacji

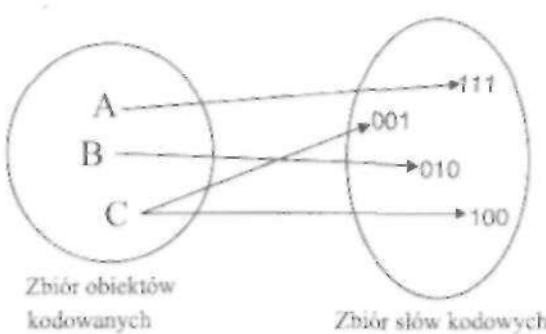
Komputer jest urządzeniem służącym do przetwarzania informacji. Informacją są liczby, ale także inne obiekty, takie jak litery, wartości logiczne, obrazy itp. Ponieważ komputer jest urządzeniem zbudowanym z **układów** cyfrowych to, jak powiedziano, każda informacja przetwarzana przez niego musi być reprezentowana za pomocą dwóch stanów - wysokiego i niskiego. Duża część tej informacji to liczby, stąd przyjęło się nazywać te stany **jedynką** i **zerem (1** i **0).** Możemy zatem stwierdzić, że

wszelka informacja w komputerze musi występować w postaci zerojedynkowej, czyli binarnej. Potrzebne są więc także reguły przekształcania różnych postaci informacji na informację binarną (dokładniejsza definicja informacji binarnej zostanie podana nieco później). Proces przekształcania jednego rodzaju postaci informacji na inną postać nazywamy **kodowaniem.**

Definicja

Kodowaniem nazywamy przyporządkowanie poszczególnym obiektom zbioru kodowanego odpowiadających im elementów zwanych słowami kodowymi, przy czym każdemu słowu kodowemu musi odpowiadać dokładnie jeden element kodowany.

Zbiorem kodowanym może być zbiór dowolnych obiektów, przykładowo liter, i symboli graficznych czy np. stanów logicznych. Proces kodowania poglądowo przedstawiony jest na rysunku 2.3.



Rysunek 2.3. Graficzna interpretacja procesu kodowania

Zgodnie z rysunkiem litera A będzie reprezentowana przez słowo kudowe (w skrócie kod) 111, litera B przez 010, a litera C przez 001 lub 100. Fakt, że literze C odpowiadają dwa słowa kodowe, nie przeszkadza w poprawnym przetwarzaniu informacji, aczkolwiek stanowi pewne utrudnienie procesu kodowania. Sytuacja odwrotna, gdy jedno słowo kodowe odpowiadałoby dwóm literom (na przykład A - 001 i B - 001), byłaby niedopuszczalna. Jeżeli w procesie przetwarzania informacji otrzymalibyśmy jako wynik kod 001, nie bylibyśmy w stanie określić przy dekodowaniu, czy odpowiada on literze A, czy B.

Sposób określenia kodu, czyli procesu kodowania, może być różnoraki. Może to być opis słowny, wzór, tabela przekodowująca lub każdy inny sposób zapewniający spełnienie warunków podanych w definicji.

Jak już wspomniano, informacja kodowana w komputerze jest bardzo różnorodna. Mogą to być teksty (czyli ciągi znaków), polecenia do wykonania przez komputer (na przykład instrukcje dla procesora), wartości logiczne czy też liczby. W ostatnim przypadku będziemy mówić o tak zwanych **kodach liczbowych.** Będą to kody przedstawia-

jące liczby, z reguły dziesiętne, w postaci binarnej. Poniżej podajemy definicje kodu liczbowego, kilka przykładów najczęściej używanych kodów, wśród nich kodów liczbowych. Dodatkowo przedstawiamy sposób przetwarzania tak zwanej informacji analogowej, czyli ciągłej, na informację cyfrową. Przykładem takiej sytuacji jest wczytywanie do komputera dźwięków za pomocą mikrofonu i karty dźwiękowej (muzycznej).

2.1.4.1. Przykłady kodów liczbowych

Definicja

Kodem liczbowym nazywamy taki kod, który liczbom dowolnego systemu będzie przyporządkowywał słowa kodowe w postaci zerojedynkowej.

Przykład

Naturalny kod binarny (NKB)

Definicja

Jeżeli dowolnej liczbie dziesiętnej przyporządkujemy odpowiadającą jej liczbę binarną, to otrzymamy **naturalny kod binarny** (NKB).

Kilka przykładowych wartości liczb kodowanych i odpowiadających im słów kodowych (przy założeniu długości słów kodowych równej 4) zawiera tabela 2.2.

Tabela 2.2. Przykłady słów kodu NKB

Liczba kodowana	Kod NKB
7	0111
0	0000
14	1110
9	1001

Kod prosty BCD

Sposób konstruowania słowa kodowego w kodzie prostym BCD jest następujący:

1. Każdej cyfrze dziesiętnej przyporządkowujemy czterocyfrową liczbę dwójkową (zwaną tetradą) w kodzie NKB (gdyby zamiast słów kodu NKB został użyty inny kod, np. Graya, wówczas otrzymalibyśmy kod BCD Graya). Przyporządkowanie to przedstawione jest w tabeli 2.3.

Tabela 2.3. Przyporządkowanie cyfr dziesiętnych tetradom NKB

Cyfra dziesiętna	Tetrada NKB	Cyfra dziesiętna	Tetrada NKB
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

2. Słowo kodowe w kodzie prostym BCD odpowiadające danej liczbie otrzymujemy, zapisując każdą cyfrę tej liczby w postaci czwórki cyfr binarnych, zgodnie z tabelą 2.3.

Przykład

Znaleźć słowa kodu prostego BCD odpowiadające liczbom 463_D i 67_D

Rozwiązanie

Jeśli zapiszemy każdą cyfrę liczby w postaci tetrady NKB, otrzymamy:

$$463_D = 0100\ 0110\ 0011_{BCD}$$

$$67_D = 0110\ 0111_{BCD}$$

2.1.4.2. Kod ASCII i jego następcy

Inny przykład stanowi kod służący do kodowania tekstów i przesyłania ich pomiędzy urządzeniami cyfrowymi. Nosi nazwę kodu ASCII (ang. American Standard Code for Information Interchange). Koduje oprócz znaków alfanumerycznych tak zwane znaki sterujące, służące do sterowania transmisją i pracą drukarki lub dalekopisu czy też ruchami kursora na ekranie. Kod ten podamy w postaci tabeli zawierającej kodowane obiekty i odpowiadające im słowa kodowe (tabela 2.4). Jak widać, tabela oprócz znaków alfanumerycznych zawiera znaki sterujące. Pełne zestawienie znaków sterujących wraz z ich znaczeniem zawiera tabela 2.5.

Do zakodowania liter alfabetu łacińskiego, cyfr arabskich, znaków przestankowych i podstawowych znaków arytmetycznych oraz poleceń sterujących wystarczy 128 pozycji, stąd kod ASCII do kodowania tych obiektów używał 7 bitów (2^7=128). Dlatego też początkowo ósmy bit był nieużywany lub służył jako bit kontroli parzystości. Później, jako rozwinięcie kodu ASCII, powstał rozszerzony kod ASCII, do którego dołączono tak zwane znaki semigraficzne, czyli proste znaki graficzne pozwalające rysować ramki i inne obiekty. Wykorzystano do tego celu ósmy bit, przy czym dla znaków semigraficznych miał on wartość 1.

Tabela 2.4. Kod ASCII

Numery bitów słowa			8	Bit kontroli parzystości							
			7	0	0	0	0	1	1	1	1
			6	0	0	1	1	0	0	1	1
			5	0	1	0	1	0	1	0	1
4	3	2	1								
0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	≠	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	↑	n	~
1	1	1	1	SI	US	/	?	O	←	o	DEL

Tabela 2.5. Znaki sterujące kodu ASCII

Symbol	Pełna nazwa znaku	Polska nazwa znaku
SOH	Start of heading	Początek nagłówka
STX	Start of text	Początek tekstu
ETX	End of text	Koniec tekstu
EOT	End of transmision	Koniec transmisji
ENQ	Enquiry	Zapytanie
ACK	Acknowledge	Potwierdzenie

S)tnt>oJ	Pitna nazwa znaku	Pokka na/wa znaku
DLI;	Data linę escape	ana interpretacji okresloiiej liczb du
/ NAK	itiM[1] acknowta	Hr.ik potwierdzenia
S[1]!	Synchronotu fik	Znak •≫ nchronizujayy
FTB \	i od <>/ *tmmaańoB* bt<vk	Koniec transmisji bloku
NUL	Nult	•V. pUttJ
BEL	Bell	Dzwonek
SO	Shift out	z kodu
SI	Shift m	Pininuiii*ktxlu
CAN	Cucd	Anulowanie danycJi zawartych * bloku
EM	Em) of medium	Koniec nośnika informacji
• i 3	SubatŚO	/.unuiuhtęducgii maku
ESC	i I|V	/miana ki>du
DII	Dekle	Unieważnienie
BS	Bacfcapac*	• jedną i*v
HT	Horizwntai tabulatuui	Tabulacja pozioma
LP	feed	Zmian.! wiersza
VT	. tical tarmlatinn	Tabulacja pionowa
CR	Caniage return	Powrót karetki
FE	• Mil II'I'd	Zmiana formatu, arkusza, strony
LS	!iU-sepaiatin	. . . . .>rów
OS	Group separalor	Separata n
RS	Record separatoi	Separator zapisów
US	Unii separator	Separator jednostek
SP	Sp	Spacja
DCI	De\ice it»niri)l 1	Sterowanie urządzenia 1
DC2	Dewce contro] 2	Steniwanie urządzenia -
D O	Device conm.il 3	• rowanie urządzenia
DC4	Deróe contro) 4	Steni*anie urządzenia 4

Kod ASCII jest oczywiście przystosowany do pisowni anglosaskiej. Ponieważ inne narodowości chciały używać w tekstach znaków specyficznych dla danego języka (na przykład znaków diakrytycznych: polskie ś, ć czy niemieckie a, ii), powstał problem narodowych stron kodowych. Próbowano go rozwiązać, zastępując potrzebnymi znakami niektóre znaki semigraficzne, jednak dawało to czasami dość nieoczekiwane efekty na rysunkach. Sytuację komplikował jeszcze fakt istnienia innych alfabetów poza łacińskim (hebrajski, cyrylica, katakana i inne). Przejściowym rozwiązaniem było stosowanie dla tego typu kodów dodatkowego bajtu, jednak nie było wygodne dla programistów (wyjaśnienie tej kwestii i dokładniejszy opis wspomnianego niżej Unikodu zawiera na przykład pozycja [29]). Rozwiązaniem okazało się wprowadzenie tak zwanego Unikodu, kodującego znaki 16 bitami. Daje to możliwość zakodowania 65536 znaków, co jest wystarczające do obdzielenia wszystkich narodów na świecie. Jednocześnie zapewniono kompatybilność Unikodu z kodem ASCII. Ten ostatni jest podzbiorem Unikodu, w którym starszy bajt jest równy zeru.

2.1.4.2. Kodowanie informacji ciągłej

Poniżej omawiamy sposób postępowania przy przekształcaniu tak zwanej informacji ciągłej, czyli analogowej, na informację cyfrową, który to sposób pozwala kodować w postaci binarnej takie wielkości jak napięcie (będące wielkością wyjściową wielu przetworników, na przykład mikrofonów, termometrów cyfrowych i wielu innych). Informacja analogowa charakteryzuje się tym, że może przybierać wiele wartości, przy czym zmiany pomiędzy tymi wartościami są płynne. Zmiany wielkości analogowej na wykresie przedstawiamy w postaci ciągłej krzywej.

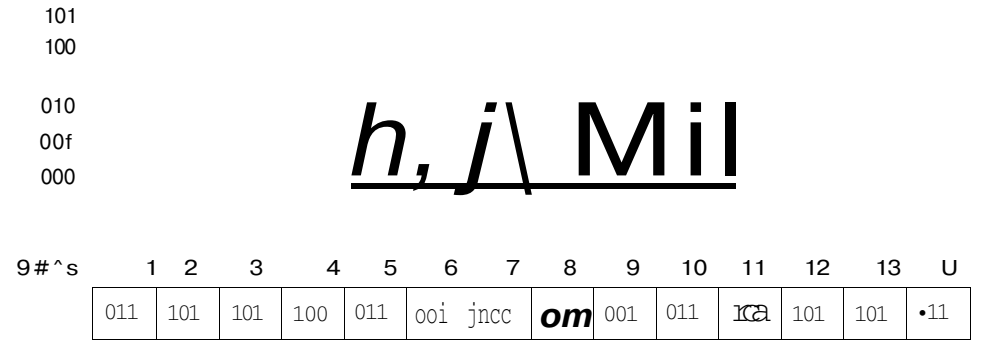
Proces kodowania informacji analogowej, czyli ciągłej, jaką jest przykładowo dźwięk (dźwięk jest zmianą ciśnienia akustycznego rozchodzącą się w powietrzu lub innym materiale), wymaga realizacji kilku etapów. Są to:

próbkowanie, polegające na cyklicznym (inaczej: z określoną częstotliwością) sprawdzaniu wartości przebiegu analogowego i zwykle czasowym zapamiętaniu tej wartości,

kwantyzacja, polegająca na podziale całego obszaru zmienności wielkości analogowej na określoną liczbę przedziałów i stwierdzeniu, w którym przedziale znajduje się dana pobrana próbka,

kodowanie, polegające na przyporządkowaniu każdemu przedziałowi zmienności wielkości analogowej określonej kombinacji zerojedynkowej (zwykle interpretowanej jako numer czy też wartość danego przedziału) i podaniu kodu tego przedziału, w którym znajduje się nasza próbka.

Interpretacja wymienionych operacji przedstawiona jest na rysunku 2.4.



Rysunek 2.4. Kodowanie informacji analogowej

Jak widać, po przeprowadzeniu opisanych operacji wartości wielkości analogowej (w naszym przypadku sinusoidalnie zmiennej) udało się zapisać w „pamięci" w postaci cyfrowej, a więc nadającej się do przetwarzania przez komputer. Przejście takie realizowane jest przez układ zwany **przetwornikiem analogowo-cyfrowym** (w skrócie **przetwornikiem a/c,** ang. ADC - *Analog to Digital Converter),* poprzedzone ewentualnie układem próbkująco-pamiętającym (nie każdy przetwornik go wymaga). Z przetwarzaniem analogowo-cyfrowym i jego dokładnością wiążą się pewne problemy, które przedstawiamy w drugiej części książki.

Podane przykłady nie wyczerpują oczywiście możliwości kodowania informacji. Prócz kodowania informacji czysto liczbowej czy też tekstów możemy przykładowo kodować (i przetwarzać cyfrowo) obrazy (np. w postaci bitmap), dźwięk, wielkości fizyczne, takie jak temperatura, ciśnienie (odpowiednie czujniki plus przetworniki a/c), oraz wiele innych. Przykłady niektórych kodów (m.in. U2) zostaną podane później, wraz z przykładowymi zastosowaniami.

2.1.5. Bramki logiczne i operatory (działania) logiczne

Jedną z ważnych grup działań wykonywanych podczas przetwarzania informacji są działania logiczne. Wykonywanie tych działań wiąże się z operowaniem dwoma wartościami logicznymi zwanymi prawdą (ang. *true)* i fałszem *(ang. false)*. Wartości te są w logice pojęciami pierwotnymi, czyli nie są to pojęcia definiowane. Działania logiczne operują na wartościach logicznych i ich wynikiem również jest wartość logiczna (podobnie jak działania arytmetyczne operują na liczbach, dając w wyniku liczbę).

W układach cyfrowych wartości logiczne muszą być reprezentowane (jak każdy rodzaj informacji) przez dwa stany elektryczne: wysoki - H i niski - L. Możemy

przykładowo przyporządkować (czyli zakodować) wartości logicznej prawda stan H, a wartości logicznej fałsz stan L. Posługujemy się wówczas logiką prostą. Przy odwrotnym przyporządkowaniu mamy do czynienia z logiką odwróconą. Jak wspomniano, w technice komputerowej stan H jest zwykle zapisywany jako 1, a stan L jako 0, dlatego przy prezentacji działań logicznych będziemy się posługiwać właśnie tymi oznaczeniami.

• Przedstawiając działania logiczne, posłużymy się dwiema metodami. Pierwsza z nich to opis słowny. Druga z nich to tak zwana tabela prawdy, która wymaga krótkiego wyjaśnienia. Polega na przedstawieniu w tabeli wszystkich możliwych kombinacji argumentów i odpowiadających im wartości logicznych wyniku danego działania. Przypomina więc przykładowo określenie działania zwanego w arytmetyce mnożeniem w postaci tabliczki mnożenia.

W technice cyfrowej działania logiczne wykonywane są przez układy cyfrowe zwane **bramkami.** Wyjaśnienie, skąd wziął się termin „bramka", odkładamy na później. Bramki są podstawowymi układami cyfrowymi będącymi cegiełkami, z których buduje się bardziej skomplikowane układy logiczne. Przy opisie bramek, a później przy opisie wielu innych układów cyfrowych, będziemy posługiwać się tak zwaną metodą **czarnej skrzynki.** Polega ona na tym, że nie zastanawiamy się, dlaczego układ działa w podany sposób ani jaka jest jego wewnętrzna budowa, a interesuje nas jedynie jego zachowanie zewnętrzne. Inaczej mówiąc, będziemy podawać, jak przy określonych sygnałach wejściowych (pobudzeniach) będzie się zachowywać wyjście układu. Dla niektórych typów układów (na przykład dla bloków mikroprocesora) jest to praktycznie jedyny możliwy sposób opisu (ze względu na stopień komplikacji tych układów). Zobaczymy ponadto, że w przypadku bramek do ich opisu będzie można zastosować tabelę prawdy.

Przedstawiając działania logiczne, będziemy też używać terminu zmiennej logicznej.

Definicja

Zmienną logiczną nazywamy zmień ną, która może przyjmować jedną z dwóch wartości logicznych: prawdę lub fałsz.

W zapisie stosowanym w układach cyfrowych zmienna taka będzie więc przyjmować wartość 1 bądź 0, przy czym pamiętajmy, że w tym wypadku są to zakodowane wartości logiczne.

Zwyczajowo zmienne logiczne oznacza się małymi literami z końca alfabetu. Działania logiczne można przedstawić jako działania na zmiennych logicznych. Po tych krótkich wyjaśnieniach przechodzimy do zdefiniowania podstawowych działań logicznych: iloczynu logicznego, sumy logicznej, negacji (zaprzeczenia) oraz alternatywy wykluczającej (Ex-OR). Określenie pozostałych działań logicznych (na przykład

działań złożonych, takich jak funkcja NAND, NOR lub EX-NOR) można znaleźć na przykład w pozycji [25].

1. Iloczyn logiczny - bramka **AND.**

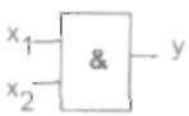
Iloczyn logiczny dwóch zmiennych zapisujemyjako:

j czytamy „> nmiui MC K_r i *£**. W układach cyfrowych częste zamiast jynibołu „A" *użyv>n* sic >\,!iibołu zwykłego mnożenia, dlatego powyitze działanie nożna Zd^ jako:

$$y = x_1 \cdot x_2$$

a z kontekstu tego zapisu wynika, czy jest to działanie logiczne, czy arytmetyczne.

Symbol graficzny układu cyfrowego realizującego iloczyn logiczny, czyli symbol bramki AND, przedstawia rysunek 2.5, tabela 2.6 jest zaś tabelą prawdy opisującą zarówno zachowanie się tej bramki, jak i własności dwuargumentowego iloczynu logicznego.



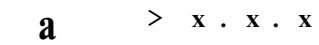
Rysunek 2.5. Symbol bramki A N D

Tabela 2.6. Tabela prawdy dwuwejściowej bramki A N D

x_1	x_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Z tabeli tej łatwo można odczytać, że wartość iloczynu logicznego dwóch zmiennych jest równa 1 (czyli prawdzie) tylko wtedy, gdy obydwie wartości argumentów wynoszą 1. W pozostałych przypadkach otrzymujemy 0. Można to uogólnić na wiele argumentów, stwierdzając, że iloczyn logiczny jest prawdziwy, gdy wszystkie argumenty tego iloczynu są prawdziwe. Iloczyny wieloargumentowe realizowane są przez

bramki wielowejściowe. Symbol przykładowej czterowejściowej bramki AND oraz zależność sygnału wyjściowego od sygnałów wejściowych przedstawia rysunek 2.6.



Rysunek 2.6. Czterowejściowa bramka A N D

2. Suma logiczna - bramka OR.

Tabela 2.7 pokazuje wartości dwuargumentowej sumy logicznej w zależności od wartości jej argumentów, a rysunek 2.7 przedstawia symbol graficzny dwuwejściowej bramki OR realizującej to działanie.

Tabela 2.7. Tabela prawdy dwuwejściowej bramki OR

''I	*2	J
0	0	'
0	i	1
1	0	
	1	1

Sumę logiczną zapisujemy jako:

$$x_1 \vee x_2 = y \qquad \text{lub} \qquad x_1 + x_2 = y$$

i czytamy jako „x_1 lub x_2". Podobnie jak w poprzednim punkcie, możemy działanie to uogólnić na wiele argumentów, podając następujące jej określenie. Suma logiczna jest równa zeru tylko wtedy, gdy wszystkie argumenty są równe 0. W pozostałych przypadkach wynikiem działania jest 1.

Rysunek 2.7. Symbol bramki OK

3. Negacja - bramka N O T.

Operacę negacji, czyli zaprzeczenia, oznaczamy następująco:

y s ~x lub > - x

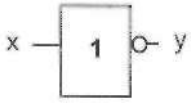
Pierwsze oznaczenie stosowane jest głównie przez matematyków, drugie jest bardzo często wykorzystywane w układach cyfrowych (choć niezbyt chętnie jest używane przez składających teksty). Firma Intel sygnały zanegowane oznacza jeszcze inaczej, a mianowicie:

' y = x #

W każdym przypadku czytamy „nie x" lub „nieprawda, że x".

Tabela 2.8. Tabela prawdy bramki NOT

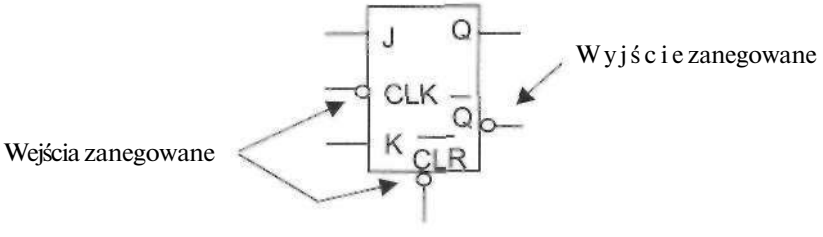
x	y
0	i
l	**o**



Rysunek 2.8. Symbol bramki NOT

Określenie negacji jest bardzo proste. Jeżeli wartość argumentu jest równa 0 (fałsz), to w wyniku otrzymujemy 1 (prawda) i odwrotnie. Negacja działa zawsze na jeden argument (choć możemy zanegować na przykład sumę). Dlatego bramka NOT jest zawsze jednowejściowa. Jej symbol oraz tabelę prawdy przedstawiają rysunek 2.8 i tabela 2.8.

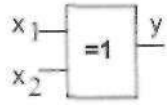
Symbol negacji wymaga krótkiego komentarza. Ponieważ sygnały logiczne mogą być negowane zarówno na wejściach, jak i na wyjściach układów, przyjęto, że operację negacji oznacza w symbolu bramki NOT kółko. Dlatego wejścia bądź wyjścia sygnału zanegowanego w układach cyfrowych są oznaczane tak, jak to pokazano na rysunku 2.9.



Rysunek 2.9. Sposób oznaczania wejść i w y j s ' ć zanegowanych

4. Bramka **Ex-OR**

Ostatnie z działań logicznych, które prezentujemy, jest bardzo użyteczne. Jego polska nazwa to alternatywa wykluczająca, używane jest też nazwa suma modulo 2. W żargonie często mówimy po prostu Ex-OR (od ang. *Exclusive OR* - alternatywa wykluczająca). Działanie to jest zawsze dwuargumentowe, jego wynik przedstawia tabela 2.9, symbol zaś bramki realizującej to działanie jest pokazany na rysunku 2.10.

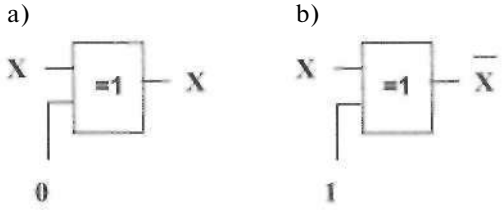


Rysunek 2.10. Symbol bramki Ex-OR

Tabela 2.9. Tabela prawdy bramki Ex-OR

*I	x2	>
0	{}	i!
0	1	1
1	{)	1
1	1	0

Jedno z ważnych zastosowań tej bramki pokazane jest na rysunku 2.11.



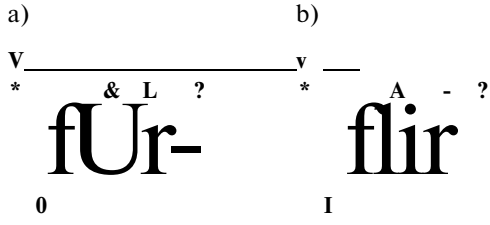
Rysunek 2.11. Zastosowanie bramki Ex-Or

Bramka ta w zależności od wartości logicznej podawanej na jedno z jej wejść neguje lub nie wartość podawaną na drugie wejście. Inaczej mówiąc, jedno z wejść tej bramki możemy traktować jako wejście sterujące, które powoduje negowanie sygnału na drugim wejściu lub przekazywanie go bez negacji (czyli jako sygnał prosty).

Na koniec tego podpunktu spróbujemy na przykładzie uzasadnić nazwę układu „bramka".

Przykład

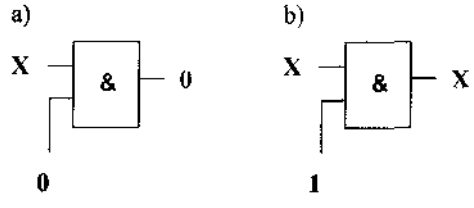
Określić, jaki będzie stan na wyjściu bramki pokazanej na rysunku 2.12, gdy na jedno z jej wejść podajemy sygnał binarny X, a na drugie wejście: a) stan 0; b) stan 1.



Rysunek 2.12. Rysunek do przykładu wyjaśniającego pochodzenie terminu „bramka"

Ko/wiązanie

Zgodnie z tabelą prawdy (lub definicją iloczynu), jeżeli choć na jednym wejściu bramki jest 0, to na wyjściu jest także zero, niezależnie od stanu pozostałych wejść. W przypadku gdy na jednym z wejść bramki dwuwejściowej jest sygnał 1, wówczas stan wyjścia tej bramki zależy od stanu drugiego wejścia, czyli od wartości sygnału X. Jeżeli na tym wejściu jest 0, to i na wyjściu mamy 0, jeżeli natomiast sygnał X przyjmuje wartość 1, to na wyjściu również jest 1. Możemy więc stwierdzić, że na wyjściu tej bramki występuje sygnał X. Tłumaczy to wynik pokazany na rysunku 2.13:



Rysunek 2.13. Rozwiązanie przykładu

Traktując stan drugiego wejścia jako pewien sygnał sterujący, widzimy, że gdy ma on wartość 1, wówczas sygnał z drugiego wejścia jest przenoszony na wyjście, a gdy sygnał sterujący ma wartość 0, sygnał X jest blokowany, czyli... **bramkowany.** Stąd nazwa układu.

2.1.6. Przykładowe parametry układów cyfrowych

Układy cyfrowe, z którymi mamy obecnie do czynienia, są monolitycznymi półprzewodnikowymi układami scalonymi. Układy scalone mają pewne wielkości liczbowe charakteryzujące ich działanie, zwane ich parametrami. Parametry układów scalonych można podzielić na dwie podstawowe grupy: **parametry graniczne,** ina-

czej dopuszczalne, i **parametry charakterystyczne.** Parametry graniczne podają pewne wartości wielkości dla układów scalonych, których nie wolno przekraczać, gdyż spowoduje to co najmniej błędne działanie układu, a zwykle także jego uszkodzenie. Parametry charakterystyczne opisują pewne właściwości układu istotne dla jego prawidłowego zastosowania. Celem naszej książki nie jest przedstawienie pełnej informacji na temat parametrów układów cyfrowych (taka informacja jest istotna dla elektroników), a jedynie zwrócenie uwagi na pewne właściwości i związane z nimi parametry mogące mieć znaczenie dla prawidłowego działania systemu komputerowego.

2.1.6.1. Parametry graniczne

Doskonałym przykładem parametru granicznego jest maksymalna moc, jaka może być wydzielona na danym rodzaju układu scalonego. Ograniczenie wydzielanej mocy wiąże się z nagrzewaniem układu. Przekroczenie tego parametru może spowodować przegrzanie się układu i w efekcie jego trwałe termiczne uszkodzenie.

Inny parametr graniczny to maksymalna dopuszczalna temperatura struktury półprzewodnika. Parametr ten jest w pewien sposób związany z poprzednim oraz ze sposobem chłodzenia układu. Jeżeli zapewnimy sprawniejsze chłodzenie układu, na przykład stosując radiator czy wentylator, możemy zwiększyć moc wydzielaną na układzie scalonym. Dzieje się tak dlatego, że uszkodzenie układu następuje dopiero po przekroczeniu dla niego temperatury granicznej. Maksymalna moc wydzielana na układzie jest więc zwykle podawana w określonych warunkach chłodzenia.

2.1.6.2. Parametry charakterystyczne

Parametry charakterystyczne określają własności układu istotne dla prawidłowego jego działania i eksploatacji. Przykładem takiego parametru jest czas dostępu do pamięci. Mówiąc w sposób uproszczony, to czas, który musimy odczekać, aby pamięć zakończyła wykonywanie żądanej operacji (dokładniejsze określenie czasu dostępu do pamięci podajemy w podrozdziale o pamięciach). Nieuwzględnienie tego parametru nie spowoduje uszkodzenia układu pamięci, a jedynie błędne jego działanie skutkujące tak zwanymi przekłamaniami, czyli błędami w odczycie lub zapisie informacji.

Innym przykładem parametru charakterystycznego jest czas propagacji sygnału, czyli opóźnienie, jakie wprowadza układ, transmitując dany sygnał. Nieuwzględnianie tego opóźnienia przy projektowaniu układów cyfrowych może prowadzić do błędnego ich działania.

Parametrem charakterystycznym może też być maksymalna częstotliwość zegara taktującego (choć parametr ten może być także parametrem granicznym, gdyż w niektórych seriach technologicznych układów cyfrowych ze wzrostem częstotliwości taktowania wiąże się wzrost wydzielanej energii, a co za tym idzie, silniejsze nagrzewanie się układu). Przekroczenie maksymalnej częstotliwości przebiegu taktującego

ponownie spowoduje błędy, gdyż układy nie będą nadążać z wykonywaniem żądanych operacji.

Podane przykłady mają jedynie uzmysłowić Czytelnikowi rodzaj problemów występujących podczas projektowania układów cyfrowych i mikroprocesorowych. Zainteresowanych stroną elektroniczną stosowania tych układów odsyłamy przykładowo do pozycji [25] lub [30].

2.1.7. Podział układów cyfrowych

W zależności od przyjętego kryterium możemy wyróżnić kilka sposobów podziału układów cyfrowych. Poniżej podamy dwa z nich związane ze sposobem funkcjonowania układów cyfrowych oraz podział ze względu na stopień ich upakowania, czyli scalenia. Pomijamy natomiast podział związany z technologią ich wykonania, który jest istotny dla elektroników.

Układy cyfrowe ze względu na sposób działania podzielimy na układy kombinacyjne i sekwencyjne oraz na układy asynchroniczne i synchroniczne.

2.1.7.1. Układy kombinacyjne i sekwencyjne

Definicja
Układem kombinacyjnym nazywamy taki układ cyfrowy, w którym stan wejść jednoznacznie określa stan wyjść układu.

Oznacza to, że aby określić stan na wyjściach takiego układu, nie potrzebujemy żadnej dodatkowej informacji poza stanem wejść i rodzajem układu. Najprostszym przykładem układów kombinacyjnych są bramki. Jedną z cech układów kombinacyjnych jest możliwość przedstawienia ich działania (opisu) w postaci tabeli prawdy. Jest to oczywiste, gdyż tabela prawdy podaje właśnie zależność sygnałów wyjściowych od wejściowych. Inną cechą układów kombinacyjnych jest możliwość ich realizacji przez proste połączenie odpowiedniej liczby i rodzaju bramek bez sprzężeń zwrotnych (czyli prowadzenia sygnałów wstecz, od wyjścia do wejścia).

Inne własności będą miały układy sekwencyjne.

Definicja
Układem sekwencyjnym nazywamy układ cyfrowy, w którym stan wyjść zależy od stanu wejść oraz od poprzednich stanów układu.

Oznacza to, że układy sekwencyjne są układami z pamięcią. Klasycznym przykładem może być tu licznik. Znajomość stanu jego wejścia zliczającego - „pojawił się kolejny impuls do zliczenia" - nie pozwala jeszcze określić, jaka liczba zliczonych

impulsów pojawiła się na jego wyjściu. Do określenia tej wielkości potrzebna jest nam znajomość liczby impulsów, które wcześniej zliczył licznik (i którą musiał pamiętać). Najprostszymi układami z pamięcią, czyli najprostszymi układami sekwencyjnymi, są przerzutniki, których definicję i przykłady podamy w podrozdziale 2.2.3.1.

2.1.7.2. Układy asynchroniczne i synchroniczne

, Podamy teraz definicje dwóch kolejnych klas układów cyfrowych związanych z podziałem na układy asynchroniczne i synchroniczne.

Definicja

Synchronicznym nazywamy taki układ cyfrowy, dla którego w dowolnym momencie jego pracy stan wejść oddziałuje na stan wyjść.

Dla układów tych możemy zatem stwierdzić, że ich własności nie zależą od przebiegu czasowego.

Definicja

Synchronicznym nazywamy taki układ cyfrowy, dla którego stan wejść wpływa na stan wyjść jedynie w określonych odcinkach czasu pracy układu zwanych **czasem czynnym,** natomiast w pozostałych odcinkach czasu zwanych **czasem martwym** stan wejść nie wpływa na stan wyjść. Odcinki czasu czynnego i martwego wyznaczane są przez podanie specjalnego przebiegu zwanego przebiegiem zegarowym lub taktującym na wejście zegarowe lub taktujące.

Własności układów synchronicznych zależą więc od zmian przebiegu zegarowego w funkcji czasu. Przebieg ten dla układów cyfrowych jest z reguły prostokątny, czyli przyjmujący dwa poziomy, wysoki i niski, rozdzielone momentami zmian zwanymi **zboczami: narastającym** (zmiana od stanu niskiego do wysokiego) i **opadającym** (zmiana od stanu wysokiego do niskiego). Przebieg tego typu pokazany jest na rysunku 2.14.

Rysunek 2.14. Cyfrowy przebieg zegarowy

W zależności od jednego z czterech wyróżnionych fragmentów tego przebiegu używanych do wyznaczania czasu czynnego układu możemy mówić o układach

synchronicznych reagujących na poziom wysoki bądź niski lub na zbocze narastające bądź opadające. Oznaczenia każdego typu wejścia zegarowego podaje tabela 2.10.

Dla układów synchronicznych mamy możliwos'ć dokładnego wyznaczenia momentu wpływu sygnałów wejściowych na stan wyjść układu. Ma to duże znaczenie praktyczne, gdyż pozwala na przykład unikać pewnych zakłóceń czy też wyznaczać moment reakcji układu dopiero wtedy, gdy sygnały wejściowe znajdują się w stanie ustalonym. Bardzo wiele układów opisywanych w dalszej części książki będzie układami synchronicznymi.

Tabela 2.10. Oznaczenia wejść zegarowych układów cyfrowych

Rodzaj wejścia zegarowego	Symbol graficzny
układ reaguje na poziom wysoki	
układ reaguje na poziom niski	
układ reaguje na zbocze narastające	
układ reaguje na zbocze opadające	

2.1.7.3. Stopień scalenia układów cyfrowych

Trzeci podział wiąże się z tak zwanym stopniem scalenia układów scalonych, czyli z liczbą elementów elektronicznych tworzących dany układ przypadającą na jednostkę powierzchni.

Ze względu na stopień scalenia układy cyfrowe (podobnie jak i inne układy scalone) możemy podzielić na:

- układy SSI - Smali Scale Integration - około kilkudziesięciu tranzystorów w układzie scalonym,
- układy MSI - Medium Scale Integration - od kilkudziesięciu do kilku tysięcy tranzystorów w układzie scalonym,
- układy LSI - Large Scale Integration - od kilku tysięcy do setek tysięcy tranzystorów w układzie scalonym,
- układy VLSI - Very Large Scale Integration - od setek tysięcy tranzystorów w układzie scalonym.

Podane liczby tranzystorów proszę traktować jedynie orientacyjnie.

2.2. Cyfrowe układy funkcjonalne

W rozdziale tym omawiamy bardziej złożone układy cyfrowe, tworzone zwykle przy użyciu podstawowych układów poznanych w poprzednim rozdziale. Nazywamy je układami funkcjonalnymi, gdyż pełnią w systemach cyfrowych okres'lone funkcje, takie jak wykonywanie działań arytmetycznych, krótkoterminowe przechowywanie informacji czy dekodowanie adresów. Będą one, jak poprzednio, omawiane w większości wypadków jako czarne skrzynki, bez opisywania sposobu, w jaki zostały zrealizowane za pomocą elementów podstawowych. Nieliczne przykłady realizacji układów funkcjonalnych mają na celu jedynie pokazanie możliwości takiej realizacji.

Przedstawiamy także dalsze wiadomości określające podstawy funkcjonowania komputera (inna nazwa komputera, o której warto pamiętać, to maszyna cyfrowa - nazwa ta podkreśla silny związek podstaw działania komputera z pewnymi działami matematyki), takie jak arytmetyka maszyn cyfrowych czy kolejne przykłady kodów liczbowych i sposób reprezentacji liczb w komputerze.

W kolejnych podrozdziałach przedstawiamy: wybrane pojęcia i przykłady związane z arytmetyką komputerów, przykłady układów arytmetycznych, jednostkę arytmetyczno-logiczną, przerzutniki, rejestry i liczniki, bramki trójstanowe, dekodery i kodery priorytetu oraz multipleksery. Omawiamy też kod U2, w którym z reguły realizowane są działania arytmetyczne na liczbach całkowitych ze znakiem, oraz bardzo ważne pojęcie magistrali, występujące przy opisie działania systemu mikroprocesorowego.

2.2.1. Arytmetyka dwójkowa

W rozdziale przedstawimy wybrane zagadnienia arytmetyki dwójkowej, przy czym skupimy się na wyjaśnieniu zasad wykonywania określonych operacji bez teoretycznego uzasadniania ich poprawności. Kolejno omówimy dodawanie dwójkowe, zapis binarny liczb ze znakiem (dodatnich i ujemnych), zapis ułamków oraz reprezentację binarną liczb wymiernych (mających część całkowitą i ułamkową).

2.2.1.1. Dodawanie binarne

Prezentację arytmetyki dwójkowej rozpoczniemy od dodawania liczb zapisanych w kodzie **NKB,** czyli (przypominamy) liczb całkowitych nieujemnych. Ponieważ dodawanie dwójkowe może wydać się nienaturalne, postaramy się je przedstawić przez analogię do dodawania dziesiętnego wykonywanego „pisemnie".

Wykonując dodawanie dziesiętne z rysunku 2.15a, zauważmy, że do jego realizacji niezbędna jest umiejętność kilku prostych czynności.

a)

Rysunek 2.15. Dodawanie dziesiętne i binarne

Pierwszą z nich jest umiejętność sumowania dwóch cyfr (na przykład 4 + 8 = 12). Wynikiem tego dodawania jest cyfra zapisywana na danej pozycji (u nas 2) oraz cyfra będąca przeniesieniem na następną, wyższą pozycję (w naszym przykładzie 1). Kolejną więc umiejętnością jest sumowanie trzech cyfr, dwóch cyfr na określonej pozycji dodawanych liczb oraz przeniesienia z pozycji poprzedniej. Obydwie umiejętności pozwolą już nam dodać „pisemnie" dowolne liczby. Ze względu na naturalność systemu dziesiętnego nie uczymy się tabliczki dodawania (co innego z mnożeniem), jednak dla systemu dwójkowego taką tabliczkę musimy poznać.

Zaczniemy od dodawania dwóch cyfr. W przypadku dodawania dwójkowego dwóch cyfr występują cztery możliwe kombinacje wartości tych cyfr pokazane na rysunku 2.16, przy czy a i b są oznaczeniami dodawanych cyfr, wynik dodawania zapisujemy zaś jako cyfrę sumy na danej pozycji s i cyfrę przeniesienia na następną pozycję c.

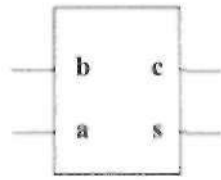
Rysunek 2.16. Dodawanie dwóch bitów

Przykładowo wynik ostatniego dodawania można więc opisać następująco: suma na danej pozycji jest równa 0, a przeniesienie wynosi 1 **(bt) I >**
W pozostałych dodawaniach przeniesienie było równe 0. Wszystkie cztery przypadki ujęto w tabeli 2.11, gdzie a i b to sumowane bity, s - wynik sumowania na danej pozycji, c - przeniesienie.

Tabela 2.11. Tabela dodawania binarnego

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Układ cyfrowy realizujący taką funkcję, którego symbol przedstawiony jest na rysunku 2.17, nazywamy półsumatorem jednobitowym.



Rysunek 2.17. Symbol pólsumatora jednobitowego

Przejdziemy teraz do dodawania dwójkowego trzech cyfr binarnych. Jeśli uwzględnimy, że dodawanie jest przemienne, czyli kolejność sumowanych cyfr nie ma znaczenia, to mamy cztery możliwe kombinacje wartości tych cyfr pokazane poniżej:

$a_i \quad b_i \quad c_{i+1} \quad s_i$

$0 + 0 + 0 = 0 \quad 0$

$0 + 0 + 1 = 0 \quad 1$

$0 + 1 + 1 = 1 \quad 0$

$1 + 1 + 1 = 1 \quad 1$

Wynikiem takiego sumowania będzie oczywiście wartość sumy na danej pozycji S| (jedna cyfra) i przeniesienie na następną pozycję C_{i+1} (druga cyfra, czasami równa 0). Wynik sumowania trzech cyfr binarnych dla wszystkich możliwych konfiguracji argumentów podaje tabela 2.12.

Tabela 2.12. Tabela dodawania trzech cyfr binarnych

«i	N	"(	*l	^ 1
0	0	0	**0**	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
!	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

W tabeli tej przyjęto następujące oznaczenia:

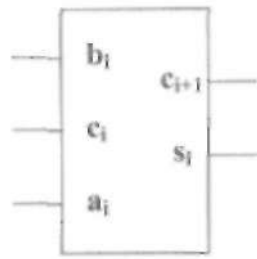
a_i – cyfra pierwszej liczby na i-tej pozycji

b_i – cyfra drugiej liczby na i-tej pozycji

s_i – cyfra sumy na i-tej pozycji

c_{i+1} - cyfra przeniesienia na pozycje i+1 (czyli następną)

Układ cyfrowy realizujący dodawanie trzech bitów zgodnie z podaną tabelą n a - zywamy sumatorem pełnym jednobitowym. Jego symbol przedstawia rysunek 2.18.



Rysunek 2.18. Symbol sumatora pełnego jednobitowego

Umiejętność wykonania dla cyfr dwójkowych opisanych czynności pozwoli n a m zsumować **dowolne całkowite** nieujemne liczby dwójkowe.

Przykład

Wykonaj binarnie dodawanie liczb 11 i 7.

a)
$$\begin{array}{r} 11 \\ +\ 7 \\ \hline 18 \end{array}$$

b)
$$\begin{array}{r} 1011 \\ +\ 111 \\ \hline 10010 \end{array}$$

Rysunek 2.19. Przykład dodawania binarnego

Liczby te pokazane są na rysunku 2.19a w zapisie dziesiętnym, a na rysunku 2.1% w zapisie binarnym. Dodając cyfry na najmłodszej pozycji (1 + 1), otrzymujemy w wyniku sumę na danej pozycji 0 i przeniesienie 1. Sumując następną pozycję (1 + 1 + 1 - dwie cyfry z danej pozycji i przeniesienie z poprzedniej), zgodnie z tabelą 2.11 otrzymujemy sumę na danej pozycji 1 i przeniesienie 1. Kontynuując takie postę- puttanii-. oir/wimjfim wynik: 10010*:= Igg.

2.2.1.2. Zapis liczb ze znakiem

Kod NKB umożliwia kodowanie liczb całkowitych nieujemnych. Obecnie przechodzimy do przedstawienia sposobów kodowania liczb całkowitych dodatnich i ujemnych, czyli liczb całkowitych ze znakiem. Kolejno opiszemy kod znak moduł (ZM) i kod uzupełnienia do dwóch (U2).

Kod znak moduł

Kod ten pozwala kodować liczby całkowite ze znakiem. Jego idea jest prosta. Najstarszy bit słowa tego kodu reprezentuje znak liczby, na przykład 0 - „+", 1 - „-"

$$\begin{cases} 0 - „+” \\ 1 - „-” \end{cases}$$

Pozostałe bity słowa reprezentują wartość bezwzględną, czyli moduł zapisywanej liczby, przy czym należy oczywiście określić, w jakim kodzie jest on zapisany.

Przykładowo, jeżeli moduł zapisujemy w kodzie NKB, to zapis 010111_{ZM} reprezentuje w zapisie ZM liczbę 23, a 110111_{ZM} - liczbę -23.

Wykonanie dodawania i odejmowania w kodzie znak moduł prowadzi do algorytmu, w którym musimy podejmować określone decyzje, na przykład sprawdzać znak obu składników. Dokładne określenie takiego algorytmu, przykładowo dla liczb dziesiętnych, pozostawiamy jako zadanie dla Czytelnika.

Kod uzupełnienia do dwóch

Pożądaną sytuacją byłby stan, w którym operacje dodawania i odejmowania liczb binarnych ze znakiem (czyli zarówno dodatnich, jak i ujemnych) dałoby się sprowadzić do prostych do wykonania (dla układów cyfrowych) operacji. Takimi operacjami są: dodawanie, przedstawione powyżej, i negacja wszystkich bitów liczby (słowa). Kodem, który umożliwia takie rozwiązanie, prowadząc do bardzo prostego algorytmu, jest kod uzupełnienia do dwóch (U2).

Definicja

Kodem uzupełnienia do dwóch (kodem **U2**) nazywamy kod wagowy, który dowolnej całkowitej liczbie dziesiętnej przyporządkowuje słowo binarne $a_{n-1}.....a_0$ takie, że spełniony jest poniższy wzór:

$$X_{U2} = a_{n-1}.....a_0 = -a_{n-1} * 2^{n-1} + a_{n-2} * 2^{n-2} + ... + a_0 * 2^0 = X_0$$

Nie będziemy dokładnie wyjaśniać, jak został skonstruowany kod U2, natomiast podamy niektóre jego ciekawe własności.

Załóżmy, że wykonywać będziemy działania na liczbach czterobitowych, czyli n=4 (jak się niedługo okaże, założenie długości słów, na których wykonujemy działania w kodzie U2, ma bardzo duże znaczenie).

Najpierw zwracamy uwagę, że jeżeli najstarszy (u nas czwarty) bit liczby zapisanej w kodzie U2 jest równy 1, to mamy do czynienia z liczbą ujemną. Prosimy jednak zauważyć, że bit ten nie jest wyłącznie bitem znaku, lecz niesie wraz ze swoją w a g ą (całą) wartość ujemną.

Przykład

Liczba 1101_{U2} zgodnie ze wzorem podanym w definicji odpowiada liczbie dziesiętnej –3, bo:

$$1101_{U2} = -1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = -8 + 4 + 1 = -3_D$$

Następnie łatwo zauważyć, że zapis liczb dodatnich w kodzie U2 i w kodzie NKB jest identyczny.

Przykład

Liczba 7_D w kodzie NKB (patrz rozdział 1.) i w czterobitowym kodzie U2 reprezentowana jest przez ciąg 0111, bo:

$$0111_{NKU} = 0*2^J + 1*2^2 + 1*2' + 0*2'' - 4 + 2 + 1 = 7_D$$
$$0111_{UJ} = -G*2^{\cdot} + 1*2^2 + 1*2^1 + 0*2^n = 4 + 2 + i = 7_D$$

Na przykładzie tym łatwo zauważyć, dlaczego istotne jest założenie, na jakiej długości słów operujemy. Bez niego nie dałoby się stwierdzić dla kodu U2, który bit niesie wartość ujemną.

Kolejną bardzo ważną własnością kodu U2 jest łatwość wyliczenia przez procesor liczby przeciwnej (o przeciwnym znaku) do liczby danej. Operacja ta sprowadza się do użycia dwóch wymienionych działań, czyli do dodawania i negowania bitów (operację negacji oznaczamy w przykładzie znakiem ~). W celu otrzymania w kodzie U2 liczby przeciwnej do danej liczby negujemy wszystkie bity tej liczby i do otrzymanego wyniku dodajemy 1. Własność tę ilustruje kolejny przykład.

Przykład

Podaną metodą obliczymy liczbę przeciwną do liczby 0111_{U2}, reprezentującą dla czterobitowego kodu U2 liczbę 7_D:

$$\begin{array}{r} \sim 0111_{U2} \\ \hline 1000 \\ +\quad 1 \\ \hline 1001_{U2} \end{array}$$

i sprawdzamy:

$$1001_{U2} = -1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = -8+1 = -7_D$$

Jak widzimy, zgodnie z oczekiwaniem, otrzymaliśmy liczbę -7. Czytelnik może sprawdzić, że ponowne przeprowadzanie tej procedury na liczbie 1001_{U2} doprowadzi do otrzymania kodu U2 liczby 7_D.

' Umiejętność dodawania i obliczania liczby przeciwnej do danej pozwala na wykonywanie dowolnych działań dodawania lub odejmowania na liczbach ze znakiem, gdyż przykładowo:

$$a - b = a + (-b)$$
$$-a + b = (-a) + b$$
$$-a - b = (-a) + (-b)$$

i tak dalej. Symbol (-b) oznacza liczbę przeciwną do b (nie musi to być liczba ujemna, na przykład liczbą przeciwną do -7 jest liczba 7).

Na zakończenie prezentacji kodu U2 podamy sposób doboru długości słów kodu U2 do żądanego zakresu liczb, dla których chcemy wykonywać działania, oraz przykład działania w kodzie U2.

Zależność zakresu liczb dziesiętnych, jakie można zapisać za pomocą słów kodu U2, od liczby bitów w tym słowie podaje następujący wzór:

$$-2^{n-1} \le X_{U2} \le 2^{n-1} - 1$$

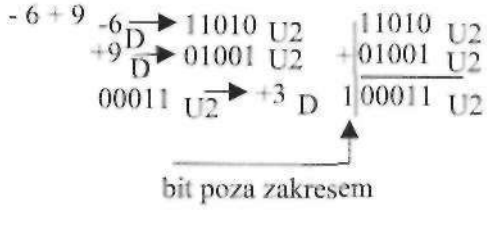
gdzie: $X_{U2} = a_{n-1}.....a_0 \qquad a_i \in \{0,1\}$

n – liczba bitów słowa kodu U2,

X_{U2} – kod U2 liczby dziesiętnej.

Przykładowo dla n=5 możemy zapisać w kodzie U2 liczby od -16_D (10000_{U2}) do $+15_D$ (01111_{U2}). Zwracamy uwagę, że w wyliczonym zakresie powinny się zmieścić nie tylko argumenty, ale także wynik.

Oto przykład wyjaśniający drugi powód, dla którego istotne jest, jaką długością słów kodu U2 posługujemy się podczas wykonywania działań:



Odczytanie wyniku uwzględniające bit numer 6 przy założeniu n=5 spowoduje otrzymanie błędnego wyniku (-29).

W kodzie U2, podobnie jak na przykład w kodzie NKB, istnieją reguły, które pozwalają stwierdzić przekroczenie zakresu w wyniku wykonania działania. W kodzie NKB jest to wystąpienie przeniesienia z najstarszego bitu. W kodzie U2 będą dwa takie przypadki, gdyż zakres możemy przekroczyć od strony liczb dodatnich (liczba dodatnia „za duża") lub od strony liczb ujemnych (liczba „zbyt ujemna" - liczba ujemna o zbyt dużej wartości bezwzględnej). Poniżej podajemy obie reguły. Zgodnie z nimi jest ustawiany w procesorach tak zwany **znacznik przepełnienia** OV (ang. *overflow),* sygnalizujący przekroczenie zakresu przy wykonywaniu działań w kodzie U2.

Zakładamy, że wykonujemy działania na słowach n-bitowych, czyli na ciągach

-.....V

> Przekroczenie zakresu od strony liczb dodatnich ma miejsce, gdy występuje przeniesienie z bitu n-2 na bit n-1, ale nie ma przeniesienia z bitu n-1 na bit n.

> Przekroczenie zakresu od strony liczb ujemnych ma miejsce, gdy występuje przeniesienie z bitu n-1 na bit n, ale nie ma przeniesienia z bitu n-2 na bit n-1.

Czytelnik może na samodzielnie dobranych przykładach sprawdzić funkcjonowanie tych reguł.

Na koniec ważna uwaga dotycząca głównie procesorów. Procesor wykonuje działania arytmetyczne zawsze tak samo, niezależnie od rodzaju kodu w jakim z a p i s a n e są argumenty. Interpretacja zarówno wartości argumentów, jak i wyników działania należy do programisty piszącego program.

2.2.1.3. Zapis części całkowitej i ułamkowej

Do tej pory w zapisie dwójkowym lub kodzie NKB mogliśmy zapisywać jedynie liczby całkowite, zgodnie z wzorem:

$$a_{n-1}\cdots a_0 = a_{n-1}\cdot 2^{n-1} + \ldots + a_0 \cdot 2^0$$

gdzie każde a, oznacza cyfrę 0 lub 1.

Jeżeli w powyższym wzorze dopuścimy występowanie ujemnych potęg, u z y skamy możliwość zapisu także ułamków, zgodnie z wzorem:

$$a_{n-1}\cdots a_0 a_{-1}\cdots a_{-k} = a_{n-1}\cdot 2^{n-1} + \ldots + a_0 \cdot 2^0 + a_{-1}\cdot 2^{-1} + \ldots + a_{-k}\cdot 2^{-k}$$

Przykładowo, zapis 1011,101 B oznacza liczbę 1 1,625 u, gdyż:

$$1011{,}101 = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} =$$

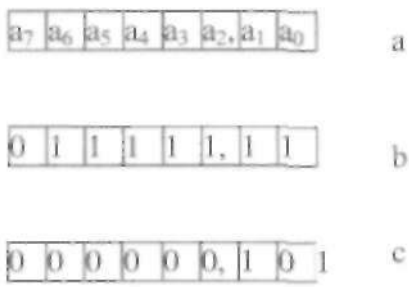
$$= 8 + 2 + 1 + \frac{1}{2} + \frac{1}{8} = 8 + 2 + 1 + 0{,}5 + 0{,}125 = 11 + 0{,}625 = 11{,}625$$

Liczby w pamięci komputera zapisywane są przy użyciu określonej liczby bitów. Decyduje o niej kompilator, czyli program tłumaczący instrukcję języka wysokiego poziomu lub instrukcje asemblera na język maszynowy, czyli binarne kody instrukcji i danych|Zapis ułamków wymaga określenia, pomiędzy którymi bitami zapisu występuje (umowny) przecinek. Jeżeli miejsce to jest ustalone dla wszystkich zapisywanych liczb, wówczas mówimy o zapisie stałoprzecinkowym (lub inaczej stałopozycyjnym). W przypadku gdy do zapisu liczb o różnych wartościach używa się notacji wykładniczej i w konsekwencji przecinek w zapisie liczby w pewnym sensie zmienia swoją pozycję, mówimy o zapisie zmiennoprzecinkowym (ang. *floating point)* (lub zmiennopozycyjnym). Poniżej krótko przedstawiamy oba sposoby zapisu, ich zalety i wady.

2.2.1.4. Zapis stało- i zmiennoprzecinkowy

Zapis statoprzecinkowy

Załóżmy (dla obu przykładów), że do zapisu liczby dysponujemy ośmioma bitami. (W zapisie stałoprzecinkowym liczbę będziemy zapisywać w kodzie znak moduł, a więc najstarszy bit będzie bitem znaku. Pozostałe bity wykorzystamy następująco: dwa najmłodsze bity będą częścią ułamkową, pozostałych pięć stanowi część całkowitą liczby. Inaczej mówiąc, pozycję przecinka ustalamy pomiędzy bitami numer 2 i 3 (bity numerujemy od zera). Sytuację tę przedstawia rysunek 2.20a.



Rysunek 2.20. Przykład zapisu stałoprzecinkowego

Największą liczbą, którą da się w ten sposób zapisać, jest 011111,11 (rys. 2.20b), czyli 32,75, najmniejszą liczbą dodatnią 000000,01 (rys. 2.20c), czyli 0,25 (patrz wzór

na stronie 54). Zapis liczby 0,625 będzie j u ż zapisem przybliżonym, gdyż nie m a m y możliwości zapisania 0,125 (czyli 1/8 lub inaczej 2^{-3}), co pokazuje rysunek 2.20c. Błąd zapisu wyniesie w tym wypadku:

$$\frac{0,625-0,5}{0,625} \cdot 100\% = 20\%$$

Maksymalny błąd względny w tym zapisie może wynieść nawet 50 proc. N a j g o r szym przypadkiem jest zapis 0,01111.. odpowiadający liczbie dziesiętnej 0,5 (dokładniej 0,4999...), który daje w wyniku niemożności zapisania wytłuszczonych jedynek błąd:

$$\frac{0,5-0,25}{0,5} \cdot 100\% = 50\%$$

Wynik ten porównamy za chwilę z zapisem zmiennoprzecinkowym.

Zapis zntiennoprzecinkowy

(W zapisie zmiennoprzecinkowym wykorzystuje się formę zapisu liczb znaną (na przykład w kalkulatorach) pod nazwą notacji wykładniczej lub notacji naukowej. Liczbę zapisuje się w postaci iloczynu tak zwanej mantysy i wagi będącej potęgą podstawy określonego systemiyPrzykładowo liczbę dziesiętną 31,75 możemy zapisać jako $0,3175 * 10^2$ Ostatni zapisjest właśnie notacją wykładniczą. Ogólna postać tej notacji wygląda następująco:

$$L = M \cdot B^E$$

gdzie: L - zapisywana liczba, M - mantysa, B - podstawa, E - wykładnik.

W przypadku zapisu liczb dziesiętnych używamy podstawy 10, w komputerze podstawą będzie oczywiście 2.

Jak Czytelnik z pewnością łatwo zauważy, tę samą liczbę można w notacji wykładniczej zapisać na wiele sposobów. Przykładowo 31,75 to $0,3175*10^2$, $31,75*10^0$, $3175*10^{-2}$ itd.

Nim prześledzimy korzyści zapisu wykładniczego, wprowadzimy jeszcze jedną definicję.

Definicja

Mantysą znormalizowaną będziemy nazywać taką mantysę, która przed przecinkiem nie ma cyfr znaczących, a pierwsza cyfra po przecinku jest cyfrą znaczącą (czyli nie jest zerem).

Dla każdej liczby istnieje tylko jedna mantysa znormalizowana. W przykładzie powyżej mantysą znormalizowaną jest 0,3175 (i nie ma innej!).

Wróćmy do naszego przykładu zapisu liczb dwójkowych, które zapisujemy za pomocą os'miu bitów. Tym razem pola tego zapisu definiujemy następująco: najstarszy bit jest bitem znaku mantysy, kolejne trzy bity będą wartos'cią bezwzględną (modułem) znormalizowanej mantysy, a cztery najmłodsze bity będą zapisem wartości wykładnika w kodzie U2 (wykładniki dodatnie i ujemne).

Jaki zakres liczb możemy zapisać w tym przypadku? Ponieważ wykładnik jest zapisany w kodzie U2 za pomocą czterech bitów, to może przyjmować wartości od -8 do +7. Największą liczbą w naszym zapisie jest 01110111, czyli $+0,875*2^7 = 112$ - wartość 0,875 wzięła się z sumy

$$2^{-1} + 2^{-2} + 2^{-3} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = 0,5 + 0,25 + 0,125 = 0,875$$

Najmniejszą liczbą w tym zapisie jest 00011111, czyli $0,125*2^{-8}$, co daje liczbę około 0,000488. Porównajmy zakres liczb 0,000488+112 z zakresem stałoprzecinkowym 0,25-32,75.

Następną różnicą jest maksymalny błąd względny, jaki może wystąpić w naszym zapisie zmiennoprzecinkowym. W najgorszym przypadku jest to 0100111...xxxx. Wytłuszczone jedynki są cyframi mantysy, których nie uda się zapisać, a których wartość w zapisie dwójkowym nie przekroczy 0,001 binarnie, czyli 0,125 dziesiętnie. Maksymalny błąd względny nie przekroczy wiec wartości

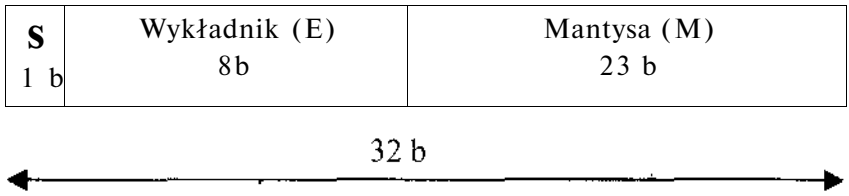
$$\frac{0,625 - 0,5}{0,625} \cdot 100\% = 20\%$$

W rzeczywistości w notacji wykładniczej przy zapisie dwójkowym i znormalizowanej mantysie sytuacja jest jeszcze korzystniejsza. Jeżeli mantysa jest znormalizowana, to pierwsza cyfra po przecinku musi być jedynką (i nie może być żadną inną cyfrą). Jeżeli tak, to nie ma sensu jej zapisywać i przykładowo w naszym przypadku mantysa będzie miała cztery pozycje (a nie trzy), mimo że zapisujemy tylko trzy z nich (nie jest to możliwe w żadnym innym systemie liczbowym). Dlatego zakres liczb zapisywanych w naszym przykładzie wynosi 0,000244 /120, a maksymalny błąd względny zapisu nie przekracza 11%.

2.2.1.5. Norma IEEE Standard 754

Zgodnie z tą normą zapisywane są przez większość kompilatorów przykładowo wbudowane typy danych (np. w C/C++ float i double) będące zapisem liczb rzeczywistych o tak zwanej pojedynczej i podwójnej precyzji.

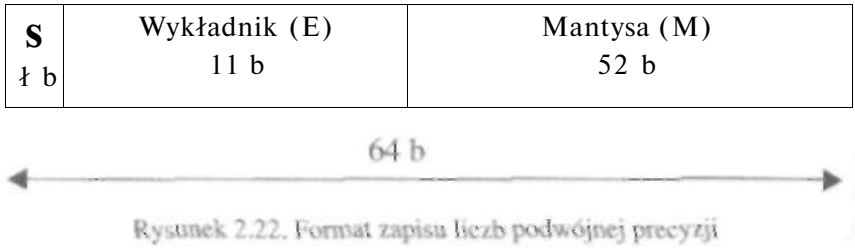
Format zapisu liczb w pojedynczej precyzji wygląda następująco:



Rysunek 2.21. Format zapisu liczb pojedynczej precyzji

Dla liczby typu float są rezerwowane w pamięci 32 bity (4 bajty). Mantysa jest znormalizowana i zapisywana w kodzie znak moduł (kod ten jest wygodny przy operacjach mnożenia i dzielenia). Długos'ć mantysy wynosi więc 24 bity (binarna mantysa znormalizowana!), a bit S jest bitem znaku mantysy. Wykładnik jest zapisywany na 8 bitach w kodzie U2, więc zakres wykładnika wynosi od - 128 do +127, przyczym wartość wykładnika jest przesunięta o 128.

Dla zapisu liczb w podwójnej precyzji zmieniają się jedynie szerokos'ci odpowiednich pól, ich znaczenie zaś nie zmienia się. Szerokos'ci tych pól podane są na rysunku 2.22.



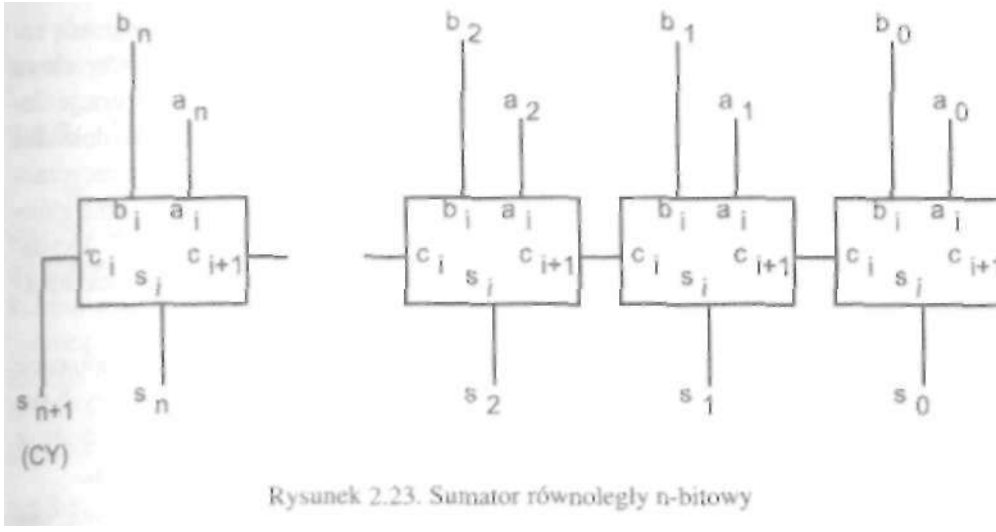
Rysunek 2.22. Format zapisu liczb podwójnej precyzji

2.2.2. Przykładowe układy arytmetyczne

2.2.2.1. Sumator równoległy n-bitowy

Za pomocą sumatorów pełnych i półsumatora można już zbudować układ, który będzie sumował dwie n-bitowe liczby binarne. Układ ten nazywany jest sumatorem kaskadowym i został przedstawiony na rysunku 2.23.

Przeniesienie z ostatniego sumatora można traktować jako kolejny bit wyniku (s_{n+1}) lub jako sygnalizację przekroczenia zakresu na wyjściu (bit przeniesienia CY), tak jak to ma miejsce w przypadku używania magistrali.



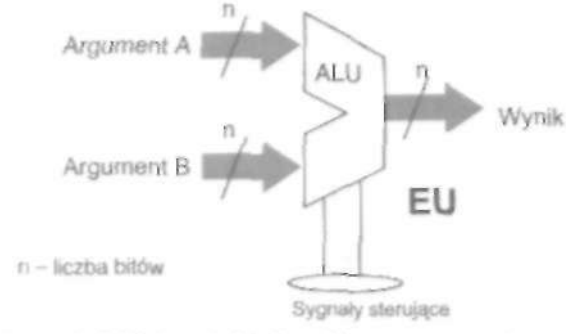
Rysunek 2.23. Sumator równoległy n-bitowy

2.2.2.2. Jednostka arytmetyczno-logiczna

Definicja

Jednostką arytmetyczno-logiczna (ALU, ang. *Arithmetic-Logic Unit)* nazywamy uniwersalny układ cyfrowy przeznaczony do wykonywania operacji arytmetycznych i logicznych.

Przykładowy symbol oznaczający jednostkę arytmetyczno-logiczna (niestety istnieje tu pewna dowolność) przedstawiono na rysunku 2.24.



Rysunek 2.24. Symbol jednostki arytmetyczno-logicznej

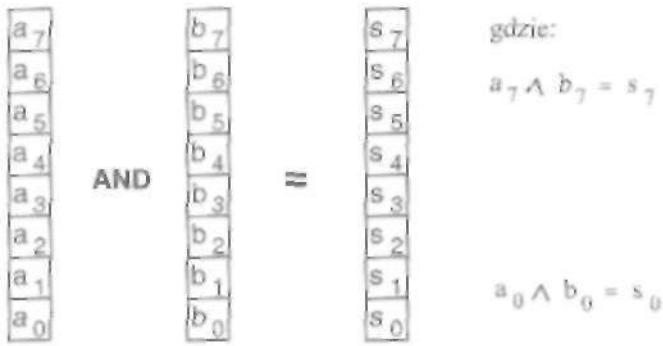
Słowo „uniwersalny" w definicji ALU oznacza, że zestaw operacji, które potrafi zrealizować jednostka arytmetyczno-logiczna, powinien być **funkcjonalnie pełny.** Zestaw operacji jest funkcjonalnie pełny, jeżeli za jego pomocą jesteśmy w stanie zrealizować dowolny algorytm przetwarzania informacji.

Do zestawu operacji wykonywanych przez jednostkę arytmetyczno-logiczną należą najczęściej: dodawanie i odejmowanie algebraiczne, przesuwanie bitów słowa w prawo i w lewo, porównywanie (komparacja) wartości dwóch słów, operacje iloczynu i sumy logicznej, negacji i alternatywy wykluczającej. Oczywiście dokładna lista wykonywanych operacji zależy od konkretnego typu jednostki arytmetyczno-logicznej. W naszej książce interesujemy się jednostką arytmetyczno-logiczną głównie jako elementem składowym mikroprocesora. Wówczas jej „umiejętności" decydują o dość dużym fragmencie tak zwanej listy rozkazów procesora, o której dokładniej piszemy w rozdziale 3.3.4.

Aby możliwe było dokonanie wyboru operacji realizowanej w danym momencie przez jednostkę arytmetyczno-logiczną, musi ona zawierać zestaw wejść zwanych **sterującymi.** Pozwalają one dokonać tego wyboru na drodze elektrycznej, przez podanie na nie określonej kombinacji zer i jedynek przyporządkowanej żądanej operacji.

Jednostka arytmetyczno-logiczną nie ma własnych układów pamiętających, dlatego musi współpracować z pewnym zestawem rejestrów. Dwa z nich pełnią szczególne funkcje i dlatego też mają swoje nazwy. Rejestr, który zawiera jeden z argumentów operacji i do którego ładowany jest wynik wykonywanej operacji, zwany jest **akumulatorem.** Dodatkowe cechy wyniku wykonywanej operacji, takie jak przeniesienie czy też przekroczenie zakresu dla działań w kodzie U2, są przechowywane w postaci określonych bitów (np. o nazwach CY czy OV) w rejestrze zwanym **rejestrem flagowym** lub **znaczników.** Dokładniej o tych rejestrach piszemy w rozdziale 3.3.2 poświęconym rejestrom roboczym mikroprocesora.

Pewnych wyjaśnień wymaga realizacja przez jednostkę arytmetyczno-logiczną operacji logicznych. W sposób naturalny operacje te dotyczą pojedynczych bitów (1 lub 0 możemy interpretować jako prawdę lub fałsz). Jednak nie każda jednostka arytmetyczno-logiczna potrafi operować na pojedynczych bitach. Jak więc operacje te są wykonywane w przypadku argumentów będących słowami (np. bajtami)? Wyjaśnienie znajduje się na rysunku 2.25.



Rysunek 2.25. Sposób wykonywania przez ALU działań logicznych na bajtach

Jak widzimy, przykładowa realizacja operacji mnożenia logicznego w przypadku argumentów bajtowych jest równoznaczna obliczeniu ośmiu iloczynów logicznych.

2.2.3. Układy z pamięcią

W podziale układów cyfrowych wprowadzono między innymi pojęcia układów cyfrowych kombinacyjnych i sekwencyjnych. Te ostatnie, jak stwierdzono, są układami z pamięcią, co pozwala przechowywać poprzednie stany układu, od których zależy kolejna reakcja układu. Najprostszymi układami z pamięcią są przerzutniki. Za ich pomocą można budować większe układy pamiętające, na przykład rejestry lub pewne rodzaje pamięci. Omówimy jedynie przykładowe, najprostsze przerzutniki. Zainteresowanych układami cyfrowymi odsyłamy do pozycji na temat układów cyfrowych, na przykład [25] lub [26].

2.2.3.1. Przerzutniki

Przerzutniki są najprostszymi układami z pamięcią i mogą być traktowane jako układy podstawowe (cegiełki, z których budujemy bardziej skomplikowane układy). Przedstawiamy je jednak w rozdziale opisującym układy funkcjonalne.

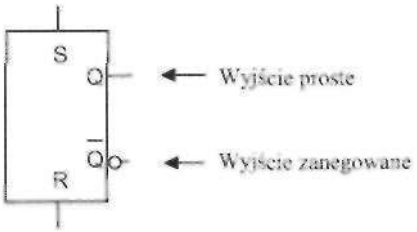
Definicja

Przerzutnikiem nazywamy układ cyfrowy umożliwiający przechowywanie najmniejszej porcji informacji, czyli jednego bitu.

W zależności od sposobu wprowadzania informacji do przerzutnika wyróżniamy kilka podstawowych rodzajów przerzutników oznaczanych symbolicznie skrótami literowymi: RS, JK, D, T. Niektóre z nich mogą być układami asynchronicznymi, jednak w większości są to układy synchroniczne. Poniżej opiszemy dla przykładu przerzutniki: asynchroniczny RS i synchroniczny D z wejściem reagującym na poziom (czyli tak zwany przerzutnik typu *latch* - zatrzask). Oba przerzutniki opiszemy, używając metody czarnej skrzynki.

Asynchroniczny przerzutnik RS

Symbol graficzny takiego przerzutnika przedstawia rysunek 2.26, jego działanie zaś opisuje tabela 2.13. Podaje ona, jaki stan zostanie wpisany do przerzutnika w zależności od różnych kombinacji wartości sygnałów wejściowych R i S. Oznaczenie wejścia „S" pochodzi od angielskiego słowa *set,* czyli „ustaw", i oznacza wpis jedynki. „R" pochodzi od *reset* i oznacza zerowanie, czyli wpis zera.



Rysunek 2.26. Symbol przerzutnika RS

Jak widać na rysunku, prezentowany przerzutnik ma dwa wyjs'cia, **proste** i **zanegowane.** Stan wyjścia prostego powinien być zawsze przeciwny do stanu wyjs'cia zanegowanego.

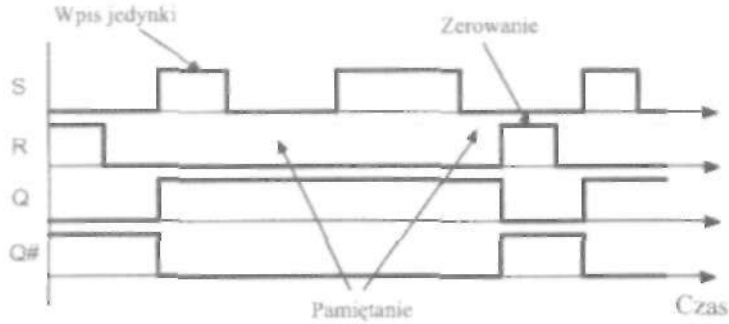
Tabela 2.13. Tabela charakterystyczna przerzutnika RS

R	**S**	Qn_{+1}
0	0	Qn
0	1	1
1	0	0
1	1	—

W tabeli Q_u oznacza stan na wyjściu przed zmianą sygnałów sterujących, natomiast Q_{n+1} stan po ich zmianie. Pozioma kreska to tak zwany stan logicznie zabroniony.

Interpretacja tabeli jest następująca. Jeżeli na obydwu wejs'ciach układu są zera, nie żądamy od przerzutnika wykonania żadnej operacji i znajduje się on wówczas w stanie **pamiętania.** Oznacza to, że stan przerzutnika nie zmienia się, czyli że $Q_{n+1} = Q_{N.}$ Gdy podajemy stan 1 na wejście S, żądamy wpisu jedynki i wówczas $Q_{n+1} = 1$. Podanie stanu 1 na wejście R oznacza zerowanie przerzutnika, czyli $Q_{n+1} = 0$. Podanie jedynek na obydwa wejścia przerzutnika jest żądaniem operacji niewykonalnej, a mianowicie jednoczesnego wpisu zera i jedynki, co jest sprzecznością (także logiczną). Rzeczywiste przerzutniki będą na taki stan reagować niezgodnie z naszymi wymaganiami (określonymi w tabeli ich działania). Przykładowo na wyjściach Q i Q 1 pojawi się ten sam stan, co jest niezgodne z definicją tych wyjść.

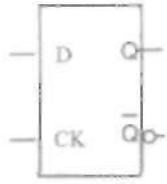
Działanie przerzutnika można też przedstawić za pomocą tak zwanych diagramów czasowych, czyli zmian wartości sygnałów na jego wyjściach w czasie w zależności od zmieniających się w czasie sygnałów wejściowych. Przykład takich diagramów dla asynchronicznego przerzutnika RS przedstawia rysunek 2.27.



Rysunek 2.27. Przebiegi czasowe dla przerzutnika synchronicznego RS

Przerzutnik D typu latch

Przerzutnik ten jest przerzutnikiem synchronicznym reagującym na poziom (niski lub wysoki w zależności od rodzaju wejścia zegarowego). Symbol i tabela opisująca jego działanie przedstawione są na rysunku 2.28 i w tabeli 2.14.



Rysunek 2.28. Symbol przerzutnika D

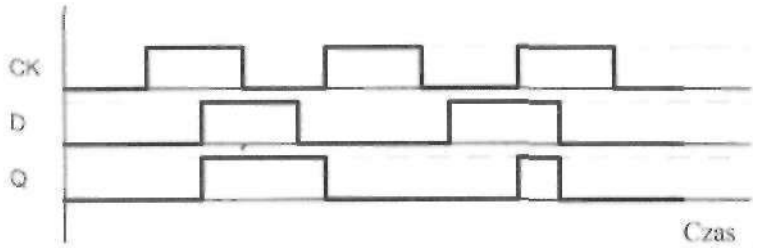
Należy pamiętać, że przerzutnik ten jest układem synchronicznym, a zatem reakcje zgodne z tabelą 2.14 będą zachodzić tylko w jego czasie czynnym. W czasie martwym stan przerzutnika nie będzie ulegał zmianie, czyli będzie on w stanie pamiętania.

Tabela 2.14. Tabela charakterystyczna przerzutnika D

D	Q_{n+1}
0	0
1	1

Zawartość tabeli oznacza, że jeżeli na wejściu zegarowym jest stan 1 (czas czynny przerzutnika), to gdy na wejściu D mamy stan 1, jest on przepisywany na wyjście (podobnie dla stanu 0). Inaczej mówiąc, w stanie czynnym przerzutnik typu *latch* powtarza na wyjściu kształt przebiegu z wejścia D, natomiast w momencie przejścia sygnału zegarowego ze stanu aktywnego w nieaktywny (zbocze opadające) stan z wejścia D jest zapamiętywany i nie zmienia się aż do kolejnego zbocza nara-

stającego. Przykładowe diagramy czasowe opisujące działanie tego przerzutnika pokazane są na rysunku 2.29.



Rysunek 2.29. Przebiegi czasowe dla przerzutnika typu *latch*

Przerzutniki tego typu są używane między innymi do budowy rejestrów typu *latch* (rejestry zatrzaskowe) pełniących ważne funkcje w układach techniki komputerowej.

2.2.3.2. Rejestry

Rejestry są układami cyfrowymi pozwalającymi zapamiętać nieco większe porcje informacji. Występują w wielu układach związanych z techniką cyfrową i komputerową, między innymi są elementami mikroprocesora. Na przykładzie rejestrów wprowadzamy też niezwykle ważne pojęcia szeregowego i równoległego przesyłania czy też postaci informacji. Pojęcia te związane są z rodzajami interfejsów w komputerze.

Definicja

Rejestrem nazywamy układ cyfrowy przeznaczony do krótkoterminowego przechowywania niewielkich ilości informacji lub do zamiany postaci informacji z równoległej na szeregową albo odwrotnie.

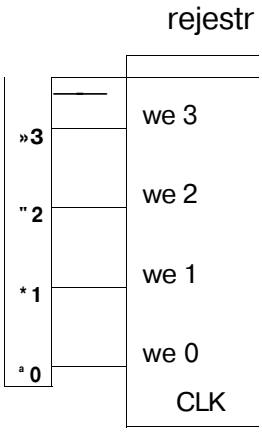
W definicji rejestru wystąpiło pojęcie szeregowej i równoległej postaci informacji. Jest ono intuicyjnie dość jasne, poniżej podajemy jednak jego dokładniejsze określenie.

Pojęcie postaci informacji jest związane z wprowadzaniem, wyprowadzaniem i przesyłaniem informacji. Określenie postaci informacji wystarczy podać dla jednego z tych przypadków, gdyż następnie łatwo je uogólnić na pozostałe przypadki.

Definicja

Wejściem cyfrowym równoległym nazywamy takie wejście, które umożliwia wprowadzenie do układu cyfrowego wszystkich bitów słowa w jednym takcie zegarowym.

Z określenia tego wynika, że liczba zacisków wejściowych w wejściu równoległym musi być równa liczbie bitów we wprowadzanym słowie. Równoległe wprowadzanie informacji pokazane jest na rysunku 2.30.

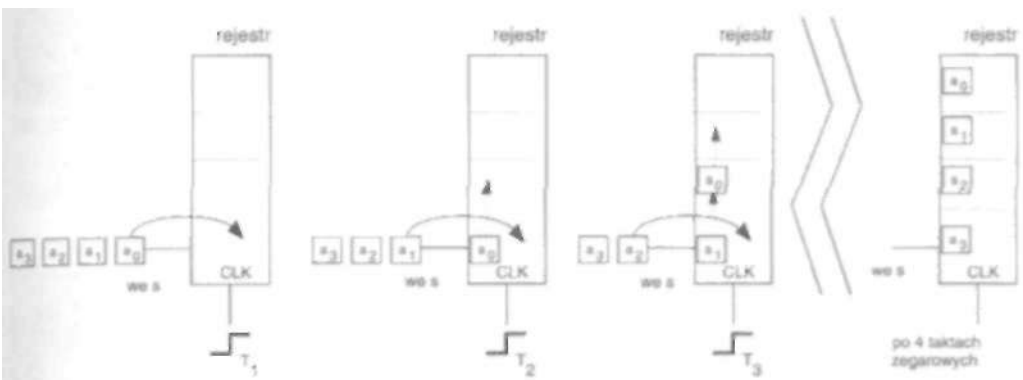


Rysunek 2.30, Równolegle wprowadzanie informacji do układu

Definicja

Wejściem cyfrowym szeregowym nazywamy takie wejście, które umożliwia wprowadzanie informacji do układu bit po bicie. Do wprowadzenia słowa n-bitowego potrzeba n taktów zegara.

Wejście szeregowe wymaga więc tylko jednego zacisku wejs'cia danych. Kolejne etapy wprowadzania do układu cyfrowego słowa 4-bitowego przedstawione są na rysunku 2.31.



Rysunek 2.31. Szeregowe wprowadzanie informacji do układu

Definicja

Jeżeli informację wprowadzamy, wyprowadzamy lub przesyłamy bit po bicie (jeden bit na jeden takt zegara), to taką postać informacji nazywamy **szeregową**. Jeżeli wprowadzamy, wyprowadzamy lub przesyłamy wszystkie bity słowa informacji jednocześnie, w jednym takcie zegara, to taką postać informacji nazywamy **równoległą**.

Ze sposobem wprowadzania i wyprowadzania informacji w rejestrach wiąże się podział rejestrów na następujące grupy:

- rejestry z wejściem i wyjściem równoległym - PIPO (ang. *parallel input, parallel output)*. Rejestry te często nazywane są rejestrami buforowymi lub (w przypadku reagowania wejścia taktującego na poziom) **rejestrami zatrzaskowymi** (ang. *latch);*
- rejestry z wejściem i wyjściem szeregowym - SISO (ang. *serial input, serial output)*. Są to tak zwane rejestry przesuwające;
- rejestry z wejściem szeregowym i wyjściem równoległym - SIPO (ang. *serial input, parallel output);*
- rejestry z wejściem równoległym i wyjściem szeregowym - PISO (ang. *parallel input, serial output).*

Przykłady zastosowania rejestrów buforowych podane są w podrozdziale 2.3.5 dotyczącym sposobu adresowania pamięci DRAM. Rejestry SIPO i PISO stosuje się do zamiany postaci informacji z szeregowej na równoległą i odwrotnie. Potrzeba takiej zamiany istnieje podczas stosowania układu transmisji szeregowej (UART lub USART, ang. *Universal Synchronous/Asynchronous ReceiverlTransmitter).* Układ ten przesyła informację na zewnątrz komputera w sposób szeregowy. Z kolei wewnątrz systemu informacja przesyłana jest w sposób równoległy. Układ USART musi więc dokonać konwersji postaci informacji i mogą być do tego użyte rejestry SIPO i PISO. Innym przypadkiem potrzeby tego typu konwersji są magistrale szeregowe, na przykład PCI Express.

2.2.3.3. Liczniki

Definicja

Licznikiem nazywamy układ cyfrowy, na którego wyjściu pojawia się zliczona przez licznik zakodowana liczba impulsów podanych na wejście zliczające licznika.

Definicja ta wymaga pewnych dodatkowych założeń. Po pierwsze zakładamy, że licznik zawsze zaczyna liczyć od tej samej wartości początkowej, najczęściej od zera.

Po drugie, zakładamy, że licznik może zliczyć dowolną liczbę impulsów, co w praktyce nie jest prawdą.

Podstawowymi parametrami charakteryzującymi licznik są jego pojemność oraz kod, w którym jest podawana liczba zliczonych impulsów. Pojemność określa maksymalną liczbę impulsów, którą może zliczyć licznik. Po przekroczeniu tej wartości licznik zaczyna zliczanie impulsów od początku.

' Liczniki w systemach mikroprocesorowych są w zasadzie układami pomocniczymi. Pewną formą licznika jest tak zwany wskaźnik instrukcji, pełniący bardzo ważną funkcję w działaniu mikroprocesora, opisany w rozdziale 3.3.2.3. Proszę jednak zachować ostrożność! Zawartość wskaźnika instrukcji, zwanego też licznikiem rozkazów, jest bardzo ściśle zdefiniowana.

2.2.4. Dekodery i kodery priorytetu

Dekodery i kodery priorytetu pełnią w systemach mikroprocesorowych ważne funkcje pomocnicze. Dekodery stosowane są w tych miejscach, gdzie za pomocą liczby zwanej adresem wybieramy jeden z wielu obiektów, na przykład określone miejsce w pamięci czy też określony układ wejścia/wyjścia. Dlatego układy te spotkamy także w dalszych częściach książki, przykładowo w rozdziale o pamięciach półprzewodnikowych. Układ kodera priorytetu występuje w takich układach jak sterownik przerwań czy sterownik DMA, opisywanych w dalszej części książki.

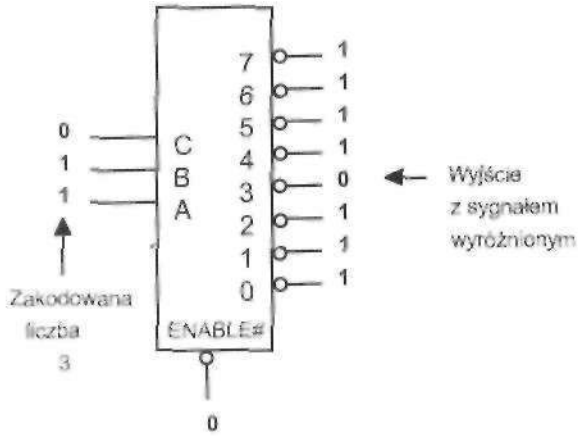
Dekodery

Definicja

Dekoderem nazywamy układ cyfrowy mający n wejść oraz k wyjść, przy czym $k \leq 2^n$. Na wejście dekodera podajemy zakodowany numer wyjścia, na którym ma się pojawić wyróżniony sygnał o wartości np. **0**. Na pozostałych wyjściach dekodera powinien występować stan przeciwny do sygnału wyróżnionego o wartości np. **1**.

Symbol dekodera przedstawiono na rysunku 2.32. Podano tam też kombinację sygnałów wejściowych i odpowiadające im sygnały wyjściowe.

Działanie dekodera jest proste. Na jednym z jego wyjść pojawia się wyróżniona wartość (może to być 0 lub 1). Na pozostałych wyjściach występują wartości przeciwne do wyróżnionej. O tym, na którym wyjściu pojawi się wartość wyróżniona, decyduje kombinacja sygnałów podawana na wejściu. Kombinacja ta jest zakodowanym numerem wyjścia, na którym wystąpi wyróżniony sygnał. Teoretycznie numer ten może być zakodowany w dowolnym kodzie. Praktycznie w dekoderach używa się niemal wyłącznie kodu NKB (jako najprostszego).



Rysunek 2.32. Symbol dekodera wraz z przykładową k o m b i n a c j a , sygnałów

W przykładowym dekoderze przedstawionym na rysunku 2.32 występuje d o d a t - kowe wejście zezwalające (sterujące) ENABLE. Brak zezwolenia, czyli w n a s z y m przypadku wartość 1, powoduje, że dekoder jest nieaktywny i na żadnym z jego wyjść nie ma stanu wyróżnionego.

Warunek na wartość k występujący w definicji wynika stąd, że musimy mieć co najmniej tyle różnych numerów, ile jest wyjść dekodera, a liczba różnych liczb n - b i t o - wych wynosi 2".

Koder priorytetu

Koder priorytetu jest pewną modyfikacją układu zwanego koderem, d l a t e g o n a j - pierw przedstawimy działanie kodera, a następnie kodera priorytetu.

Działanie kodera jest w pewnym sensie odwrotne do działania dekodera.

Definicja

Koderem nazywamy układ cyfrowy o n wyjściach i $k < 2^n$ wejściach, przy czym na wyjściu pojawia się zakodowany numer tego wejścia, na którym występuje wyróż- niony sygnał.

Podobnie jak dla dekoderów na wyjściu, dla koderów zakładamy, że na ich w e j- ściu pojawia się tylko jeden wyróżniony sygnał.

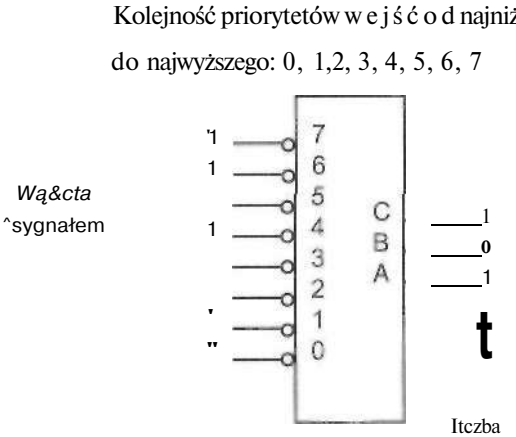
W systemach cyfrowych zachodzi niejednokrotnie potrzeba przyjmowania sy- gnałów zgłoszeń od wielu urządzeń i następnie zdecydowania, które z nich b ę d z i e obsługiwane. Realizację tych czynnos"ci umożliwia układ kodera priorytetu.

Definicja

Koderem priorytetu nazywamy układ kodera, w którym wprowadzono następujące zmiany:

1. Na jego wejściu może pojawić się więcej niż jeden sygnał wyróżniony.
2. Każdemu wejściu przyporządkowano pewien stopień ważności, zwany priorytetem.
3. Na wyjściu pojawia się zakodowany numer tego wejścia z wyróżnionym sygnałem, które ma najwyższy priorytet.

Symbol kodera priorytetu wraz z przykładową kombinacją sygnałów wejściowych i wyjściowych przedstawione są na rysunku 2.33.

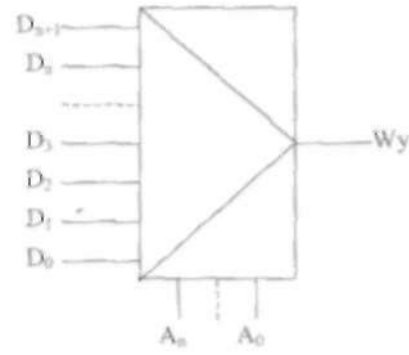


Rysunek 2.33. Symbol i przykładowe wartości sygnałów dla kodera priorytetu

Na wyjściu układu z rysunku 2.33 pojawił się kod liczby 5, ponieważ wyróżniony sygnał jest podany na wejścia o numerach 3 i 5 i spośród nich wejście 5 ma najwyższy priorytet.

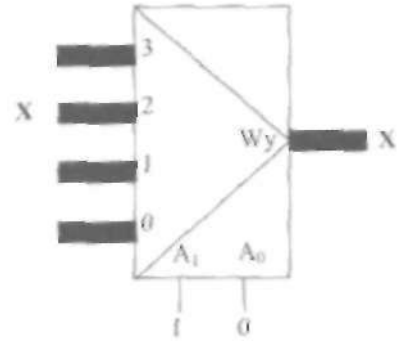
2.2.5. Multipleksery

Multipleksery są kolejnymi układami pomocniczymi występującymi w układach techniki komputerowej, przykładowo w opisie kart graficznych lub pamięci dynamicznych. Zadaniem multipleksera jest przekazywanie jednego z wielu sygnałów wejściowych na wyjście, przy czym wyboru, który sygnał ma się pojawić na wyjściu, dokonujemy za pomocą wejścia będącego rodzajem wejścia adresowego. Na wejście to podajemy zakodowany dwójkowo numer tego wejścia, którego stan ma być przekazywany na wyjście. Wyjście multipleksera, a co za tym idzie i jego wejścia, mogą być jedno- lub wielobitowe. Symbol multipleksera przedstawiony jest na rysunku 2.34a. Na rysunku 2.34b podajemy przykład działania multipleksera.



Rysunek 2.34a. Symbol multipleksera

Jeżeli na wejściu adresowym multipleksera z rysunku 2.34b podamy wartości 10, co jest zakodowaną binarnie (w kodzie NKB) liczbą 2, to wówczas na wyjście będzie przekazywany sygnał z wejścia numer 2 multipleksera, co pokazano na rysunku, Sygnał ten może być jedno- lub wielobitowy.

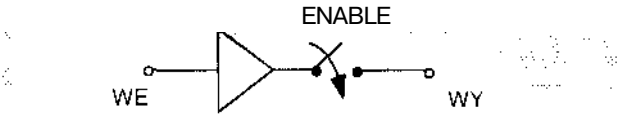


Rysunek 2.34b. Przykład działania multipleksera

2.2.6. Bramki trójstanowe

W układach cyfrowych, a szczególnie w układach i systemach mikroprocesorowych (a więc i w komputerach) występuje często potrzeba odseparowania elektrycznego dwóch lub więcej punktów w systemie, na przykład wyjścia pewnego układu i wspólnego przewodu, którym przesyłamy informację. Odseparowanie elektryczne oznacza, że wartości wielkości elektrycznych (takich jak napięcie czy prąd) w obu punktach nie wpływają wzajemnie na siebie i mogą przyjmować dowolne dozwolone wartości. Inaczej mówiąc, w układach cyfrowych stan logiczny w jednym punkcie, odseparowanym od innego punktu, nie wpływa na niego i nie jest z nim w żaden sposób związany.

Układami umożliwiającymi odseparowanie dwóch punktów w układzie są tak zwane bramki trójstanowe. Schemat logiczny takiej bramki i tabela przedstawiająca jej działanie pokazane są na rysunku 2.35 i w tabeli 2.15.



Rysunek 2.35. Schemat logiczny bramki trójstanowej

Tabela 2.15. Tabela prawdy bramki trójstanowej

WE	ENABLE	WY
0	1	0
1	1	1
X	0	Z

Klucz (przełącznik) występujący w schemacie tej bramki jest oczywiście kluczem elektronicznym. Sposób elektronicznej realizacji bramek trójstanowych nie jest dla nas istotny i nie będziemy go opisywać, Stan Z występujący w tabeli opisującej działanie bramki trójstanowej oznacza stan wysokiej impedancji, czyli włas'nie brak wzajemnego wpływu wartości elektrycznych na wejściu na wartos'ci elektryczne na wyjściu bramki. Stan Z jest więc w pewnym sensie trzecim stanem, oprócz stanów 0 i 1, w którym może się znajdować bramka, stąd też bierze się jej nazwa. W przypadku zamknięcia klucza bramka ta transmituje wartość sygnału logicznego z wejścia na wyjście.

Przykładowe użycie bramek trójstanowych zostanie pokazane w następnym podpunkcie dotyczącym pojęcia magistrali.

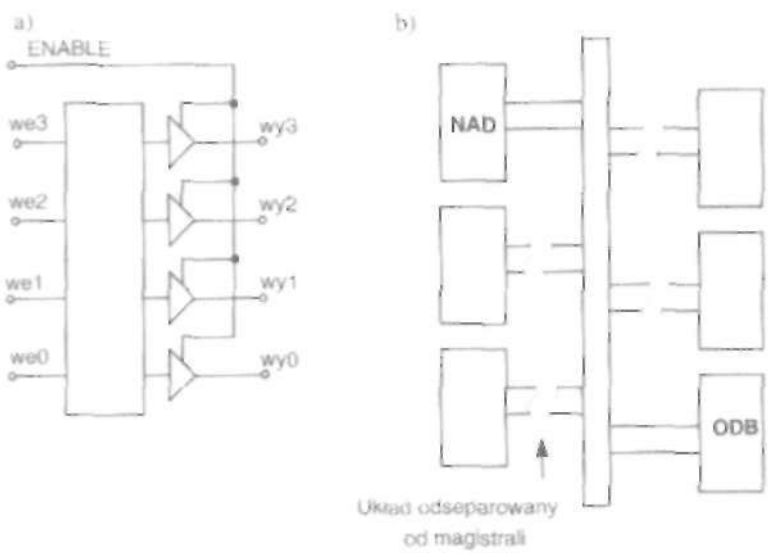
2.2.7. Pojęcie i zasada działania magistrali

W systemach mikroprocesorowych istnieje konieczność zapewnienia komunikacji pomiędzy wieloma układami. Przykładami tych układów są: mikroprocesor, pamięć RAM, ROM i układy wejścia/wyjścia. Połączenie wielu układów metodą każdy z każdym doprowadziłoby do nadmiernej liczby połączeń i jest praktycznie nierealne. Dlatego stosuje się sposób połączenia tych układów za pomocą tak zwanej magistrali. Magistrala jest wspólną drogą, dzięki której komunikują się pomiędzy sobą poszczególne układy wchodzące w skład systemu. Aby jednak zapewnić poprawne jej funkcjonowanie i brak kolizji pomiędzy połączonymi nią układami, magistrala obsługiwana jest według ściśle określonych, podanych w definicji reguł.

Definicja

Magistralą nazywamy zestaw linii oraz układów przełączających, łączących dwa lub więcej układów mogących być nadajnikami lub odbiornikami informacji. Przesyłanie informacji zachodzi zawsze pomiędzy dokładnie jednym układem będącym nadajnikiem a dokładnie jednym układem będącym odbiornikiem przy pozostałych układach odseparowanych od linii przesyłających.

Jak wynika z powyższej definicji, ukiady dołączone do magistrali muszą mieć możliwość elektrycznego odseparowania się od linii, którymi przesyłana jest informacja. Wynika to między innymi z konieczności zapewnienia jednoznacznego stanu każdej z linii. W przypadku jednoczesnego dołączenia do linii przesyłającej dwóch i nadajników mógłby wystąpić konflikt, gdyby przykładowo jeden z nich próbował wymusić stan logiczny linii równy 0, a drugi 1. Układami zapewniającymi możliwość separacji są właśnie bramki trójstanowe opisane w poprzednim podpunkcie. Przykładowy rejestr z wyjściem trójstanowym zapewniającym taką separację przedstawiony jest na rysunku 2.36a. Ideę współpracy kilku układów za pośrednictwem magistrali ilustruje rysunek 2.36b.



Rysunek 2.36. Idea działania magistrali

Na rysunku tym układ oznaczony jako NAD wymusza stan linii magistrali, czyli jest nadajnikiem informacji. Układ ODB czyta stan linii magistrali, czyli jest odbiornikiem informacji. Pozostałe układy są nieaktywne, odseparowane elektrycznie od linii magistrali i nie biorą w danym momencie udziału w transmisji.

2.3. Pamięci

Pamięci półprzewodnikowe są jednym z kluczowych elementów systemów cyfrowych. Służą do przechowywania informacji w postaci cyfrowej. Liczba informacji, które mogą przechowywać pojedyncze układy scalone pamięci, zawiera się w zakresie od kilobajtów do gigabajtów.

' W pierwszym podpunkcie tego rozdziału podajemy ogólne informacje na temat pamięci, takie jak ich podział ze względu na własnos'ci użytkowe i różnice technologiczne, podstawowe parametry oraz sposób łączenia pojedynczych układów pamięci w większe bloki. Kolejny podrozdział dotyczy działania i własności pamięci dynamicznych RAM, ze szczególnym uwzględnieniem ich zastosowania w komputerach. W kolejnym podpunkcie omawiamy moduły pamięci stosowane w technice komputerowej. Ostatni podpunkt jest poświęcony krótkiemu przeglądowi własności pamięci ROM.

2.3.1. Podstawowe definicje dotyczące pamięci

Definicja

Pamięciami półprzewodnikowymi nazywamy cyfrowe układy scalone przeznaczone do przechowywania większych ilości informacji w postaci binarnej.

Podstawowymi parametrami pamięci są pojemność, czas dostępu i transfer danych.

Definicja

Pojemnością pamięci nazywamy maksymalną liczbę informacji, jaką możemy przechować w danej pamięci.

Pojemność pamięci podajemy w bitach (b) lub bajtach (B). Podkreślamy, co zresztą zostanie dokładnie wyjaśnione przy okazji omawiania organizacji pamięci, że pojemność pamięci nie jest liczbą słów, które możemy w niej przechowywać.

Definicja

Czasem dostępu do pamięci nazywamy czas, jaki musi upłynąć od momentu podania poprawnego adresu słowa w pamięci do czasu ustalenia się poprawnej wartości tego słowa na wyjściu pamięci w przypadku operacji odczytu lub w przypadku operacji zapisu - czas, jaki upłynie do momentu zapisania wartości do tego słowa z wejścia pamięci.

W technice komputerowej używane są głównie pamięci półprzewodnikowe o dostępie swobodnym (w odróżnieniu od dostępu sekwencyjnego).

Definicja

Pamięcią o **dostępie swobodnym** nazywamy pamięć, dla której czas dostępu praktycznie nie zależy od adresu słowa w pamięci, czyli od miejsca, w którym jest przechowywana informacja.

Kolejnym ważnym parametrem pamięci (szczególnie w postaci modułów) jest jej transfer danych (ang. *data transfer ratę),* zwany czasami przepustowością (ang. *throuthput).*

Definicja

Transferem danych pamięci nazywamy maksymalną liczbę danych, jaką możemy odczytywać z pamięci lub zapisywać do pamięci w jednostce czasu.

Transfer danych pamięci może być podawany w jednostkach informacji na sekundę (na przykład w MB/s), jednak znacznie częściej dla modułów pamięci stosowanych w PC podaje się częstotliwość zegara taktującego transfer lub liczbę transferów na sekundę (np. MT/s). W dwóch ostatnich przypadkach, chcąc obliczyć transfer danych, czyli przepustowość, musimy znać liczbę bitów przesyłanych w jednym transferze (obecnie zwykle 64 bity). Nieco dokładniej o transferze danych i sposobach jego oznaczania dla modułów pamięci piszemy w punkcie 2.3.5.

2.3.2. Podział pamięci

Ze względu na własności użytkowe pamięci półprzewodnikowe możemy podzielić na pamięci RAM i ROM.

Definicja

Pamięcią **RAM** nazywamy pamięć półprzewodnikową o dostępie swobodnym przeznaczaną do zapisu i odczytu. RAM jest pamięcią **ulotną,** co oznacza, że po wyłączeniu jej zasilania informacja w niej przechowywana jest tracona.

Definicja

Pamięcią **ROM** nazywamy pamięć półprzewodnikową o dostępie swobodnym przeznaczaną tylko do odczytu. ROM jest pamięcią **nieulotną.**

Z podanych własności pamięci wynikają ich zastosowania w technice komputerowej. Z pamięci RAM buduje się pamięć operacyjną komputera przeznaczoną do przechowywania, w trakcie pracy systemu, danych oraz programów (gdyż RAM jest pamięcią do zapisu i odczytu). W pamięci ROM przechowuje się programy inicjalizujące pracę komputera, gdyż muszą być one przechowywane w pamięci nieulotnej.

Ze względu na technologię wykonania pamięci RAM dzielimy na dwie podstawowe grupy:

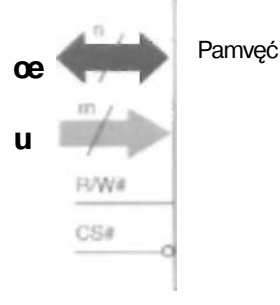
- pamięci dynamiczne - DRAM,
- pamięci statyczne - SRAM.

Pomiędzy tymi dwoma grupami pamięci występują istotne różnice w ich parametrach i własnościach użytkowych. Pamięci dynamiczne są wolniejsze od pamięci statycznych, natomiast są znacznie tańsze (szczególnie gdy uwzględnimy koszt jednego bitu). Ponadto pamięci dynamiczne znacznie łatwiej podlegają scalaniu, co oznacza, że dla porównywalnej wielkości układu uzyskujemy w nich znacznie większ7f pojemności. Wadą pamięci dynamicznych jest również fakt, że dla poprawnego ich funkcjonowania konieczny jest tak zwany proces **odświeżania.** Polega on na cyklicznym, ponownym zapisie przechowywanej informacji do komórek tej pamięci. Proces odświeżania jest nieco dokładniej opisany w podrozdziale 2.3.5.

Z porównania własności tych pamięci wynika miejsce ich zastosowania w technice komputerowej. Pamięci dynamiczne stosowane są do budowy głównej pamięci operacyjnej komputera, co wynika z ich niskiej ceny i dużych pojemności układów scalonych tej pamięci. Wadą tych pamięci w porównaniu z pamięciami statycznymi jest przede wszystkim szybkość ich działania. Jednak ze względów ekonomicznych (cena) i technologicznych (mniejszy stopień scalenia) nie można zbudować pamięci operacyjnej z pamięci statycznych. Dlatego w systemach komputerowych stosuje się t a k z w a n ą pamięć podręczną (cache - patrz rozdział 3.7), o znacznie mniejszej pojemności w porównaniu z pamięcią operacyjną. Pamięć cache buduje się z szybkich pamięci statycznych.

2.3.3. Organizacja pamięci

Podstawowe wyprowadzenia układu pamięci półprzewodnikowej są pokazane na rysunku 2.37.



Rysunek 2.37. Podstawowe wyprowadzenia układu scalonego pamięci

Szyna wejścia/wyjścia danych (DB, zaciski te często są też oznaczane jako DQ) służy do wprowadzania i wyprowadzania informacji do i z pamięci. Wejście adresowe służy do dokonania wyboru, na którym z wielu słów w pamięci zastanie wykonana operacja (zapisu bądź odczytu). Wejście sterujące R/W# informuje układ pamięci, jakiego rodzaju operacja będzie wykonywana: odczyt czy zapis. Wejście CS# służy do uaktywnienia układu pamięci. Wejście to jest używane przy budowie zespołów pamięci metodą łączenia dwóch lub więcej układów scalonych pamięci.

W dalszej części tego rozdziału będziemy używać terminów „adres" i „słowo", Mimo że terminy te nie sprawiają kłopotu naszej intuicji, podamy ich definicje.

Definicja

Adresem nazywamy niepowtarzalną liczbę (numer) przypisaną danemu miejscu (słowu) w pamięci w celu jego identyfikacji.

Definicja

Słowem w pamięci nazywamy zestaw pojedynczych komórek (pojedynczych bitów) pamięci, do którego odwołujemy się pojedynczym adresem.

Liczbę bitów w pojedynczym słowie pamięci będziemy nazywać długością słowa pamięci. Zauważmy, że długość słowa pamięci musi być równa liczbie wyprowadzeń szyny wejścia/wyjścia, gdyż słowa są wprowadzane i wyprowadzane z pamięci równolegle.

Z warunku unikalności adresu (czyli niepowtarzania się tego samego adresu - co jest oczywiste) wynika z kolei minimalna liczba linii szyny adresowej. Przy m - bitowej szynie adresowej mamy do dyspozycji 2^m różnych adresów. Jeżeli liczba słów przechowywanych w pamięci wynosi N, musi być spełniony warunek:

$$N \leq 2^m$$

lub inaczej, aby poprawnie zaadresować N słów, potrzebujemy $m - \log_2 N$ bitów adresu (lub linii adresowych).

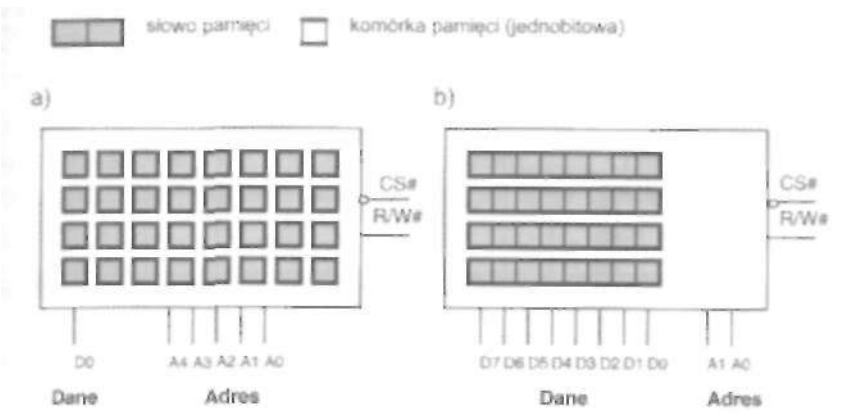
Wartość pojemności pamięci, długości słowa oraz liczby linii adresowych wiąże prosty i oczywisty wzór. Jeżeli pojemność pamięci oznaczymy przez M, długość słowa przez n, a liczbę linii adresowych przez m, to spełniona jest zależność:

$$M = n \times 2^m$$

Definicja

Organizacją pamięci nazywamy sposób podziału obszaru pamięci na słowa.

Pojęcie organizacji pamięci najprościej wyjaśnić na przykładzie. Pamięci narysowane symbolicznie na rysunku 2.38 a i b mają tę samą pojemność wynoszącą 32b, różnią się natomiast organizacją. Pamięć z rysunku a ma organizację bitową. Możemy o n i e j powiedzieć, że jest to pamięć 32 x Ib. Pamięć z rysunku b ma organizację bajtową, czyli jest to pamięć 4 x 8b (lub inaczej 4 x IB). Zwróćmy przy okazji uwagę na liczbę linii danych i adresowych dla każdej z tych pamięci.



Rysunek 2.38. Interpretacja organizacji pamięci

2.3.4. Łączenie układów pamięci

Budowa bloków (banków) pamięci polega na łączeniu układów scalonych pamięci o określonej pojemności i organizacji w ten sposób, aby uzyskać zespoły pamięci o większej pojemności i/lub o zmienionej długości słowa. Dlatego problem rozbudowy pamięci możemy podzielić na dwa podstawowe przypadki:

- zwiększanie (rozszerzanie) długości słowa przy niezmienionej liczbie słów,
- zwiększanie liczby słów przy niezmienionej długości słowa.

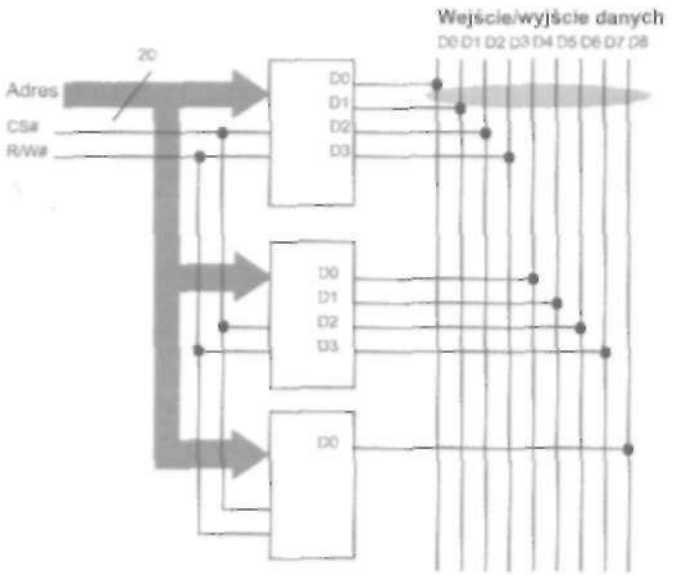
Oczywiście oba przypadki mogą występować (i w praktyce często występują; jednocześnie.

2.3.4.1. Zwiększanie długości słowa

W celu zwiększenia długości słowa pamięci szerszą magistralę danych budujemy z b i t ó w linii danych kolejnych układów scalonych pamięci, natomiast magistralę adresową i sygnały sterujące łączymy równolegle. Połączenie równoległe wejść adresowych oznacza, że we wszystkich układach, z których budujemy blok o większej długości słowa, wybieramy słowa położone w takim samym miejscu. Nie ma żadnego powodu, aby robić inaczej, gdyż jest to rozwiązanie najprostsze. Podobnie z sygnałami sterującymi. Musimy uaktywnić wszystkie układy scalone przechowujące słowa

składowe tworzące słowo o zwiększonej długości, stąd równoległe połączenie sygnałów CS#. I wreszcie na wszystkich słowach składowych wykonujemy tę samą operację, zapis lub odczyt, co wymaga równoległego połączenia sygnałów R/W#.

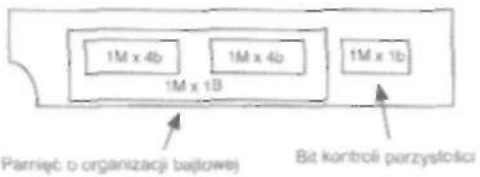
Opisany sposób najlepiej wyjaśni przykład. Załóżmy, że mamy do dyspozycji układy scalone pamięci o organizacji IM x 4b (pojemność 4 Mb) oraz IM x Ib (pojemność 1 Mb) i chcemy zbudować pamięć o organizacji IM x 9b (czyli o słowach bajtowych z bitem kontroli parzystości). Sposób połączenia układów, którymi dysponujemy, pokazuje rysunek 2.39.



Rysunek 2.39. Rozszerzanie długości słowa pamięci

Zwróćmy uwagę, że liczba linii adresowych nie zmieniła się (gdyż nie zmieniła się liczba słów), natomiast zmieniła się liczba linii danych.

Opisany sposób jest przykładowo stosowany przy budowie modułów pamięci, (na przykład SIMM, ang. *Single In line Memory Module),* co schematycznie pokazano na rysunku 2.40.

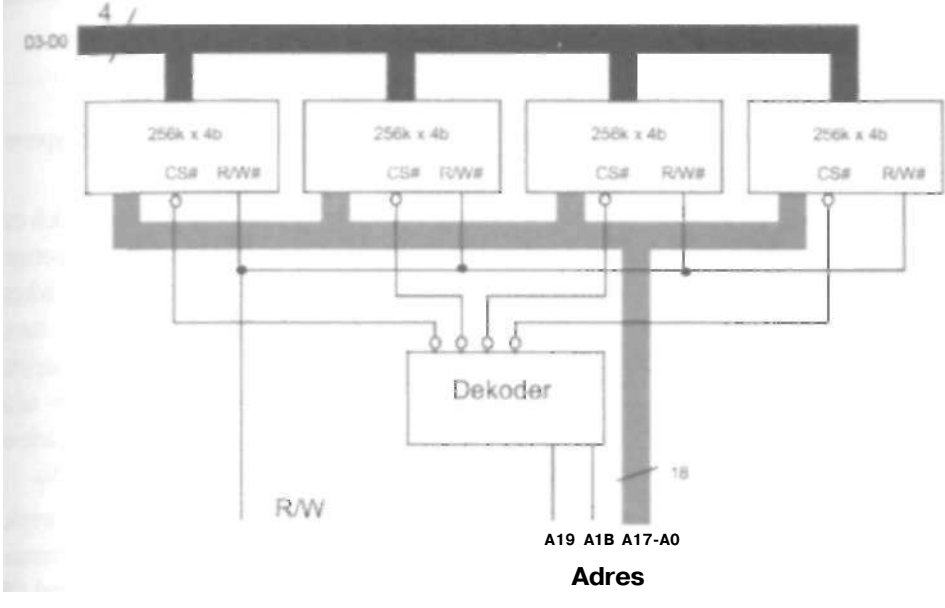


Rysunek 2.40. Wygląd modułu typu SIMM

2.3.4.2. Zwiększanie liczby słów w pamięci

Zwiększenie liczby słów pamięci oznacza zwiększenie liczby potrzebnych adresów, a co za tym idzie - rozbudowę szyny adresowej o dodatkowe bity potrzebne do uzyskania tych adresów. Przy niezmienionej długości słowa szyna danych pozostaje bez zmian. Dodatkowe bity adresu służą, przy wykorzystaniu dekodera, do wyboru jednego z łączonych układów pamięci, z którego odczytamy lub do którego zapiszemy informację. Wyboru dokonujemy przy użyciu wejs'cia CS# uaktywniającego układy scalone pamięci. Magistrale adresowe, danych i sygnały sterujące układów, z których budujemy nowy blok pamięci, łączymy równolegle.

Załóżmy, że mamy do dyspozycji układy pamięci 256k x 4b i chcemy zbudować blok pamięci IM x 4b. Do jego budowy musimy użyć czterech układów scalonych pamięci oraz dekodera. Sposób ich połączenia pokazuje rysunek 2.41.



Rysunek 2.41. Zwiększanie liczby słów pamięci

Bity A19 i A18 adresu, podawane na dekoder, uaktywniają dokładnie jedno z jego czterech wyjść. Powoduje to z kolei uaktywnienie dokładnie jednego układu scalonego pamięci. W ramach tego układu za pomocą pozostałych bitów adresu wybieramy słowo, na którym zostanie wykonana operacja zapisu bądź odczytu.

Jeżeli moduł pamięci potraktujemy jako pojedynczy układ scalony pamięci (co jest możliwe, gdyż linie adresowe i sterujące układów wchodzących w jego skład są połączone równolegle), możemy traktować budowę banku pamięci przez łączenie kilku modułów pamięci jako przypadek zwiększania liczby słów w pamięci.

2.3.5. Pamięci dynamiczne RAM

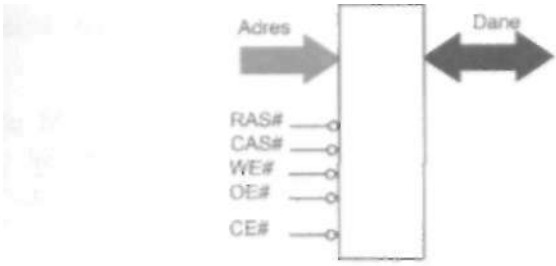
Pamięci dynamiczne RAM, w skrócie DRAM, pozwalają uzyskiwać duże pojemności w pojedynczym układzie scalonym. Zasada działania komórki pamięci dynamicznej opiera się na magazynowaniu ładunku na określonej, niewielkiej pojemności elektrycznej. Pojemność nienaładowana oznacza zero logiczne, pojemność naładowana oznacza zapisaną jedynkę logiczną. Sposób przechowywania (kodowania) stanów logicznych powoduje potrzebę odświeżania, czyli cyklicznego doładowywania tych pojemności. Proces ten opisujemy krótko w podpunkcie 2.3.5.1. Duża pojemność tych pamięci jest także przyczyną innych problemów. Stosowany jest określony sposób podawania adresu. Sposób obsługi asynchronicznej pamięci DRAM oraz jej strukturę wewnętrzną opisujemy w podpunkcie 2.3.5.1. Tam opisujemy też pewne metody przyspieszania dostępu do pamięci DRAM takie jak tak zwany tryb seryjny (ang. *burst)*. Następnie opisujemy obsługę pamięci synchronicznych DRAM na przykładzie pamięci DDR SDRAM.

2.3.5.1. Obsługa asynchronicznych pamięci DRAM

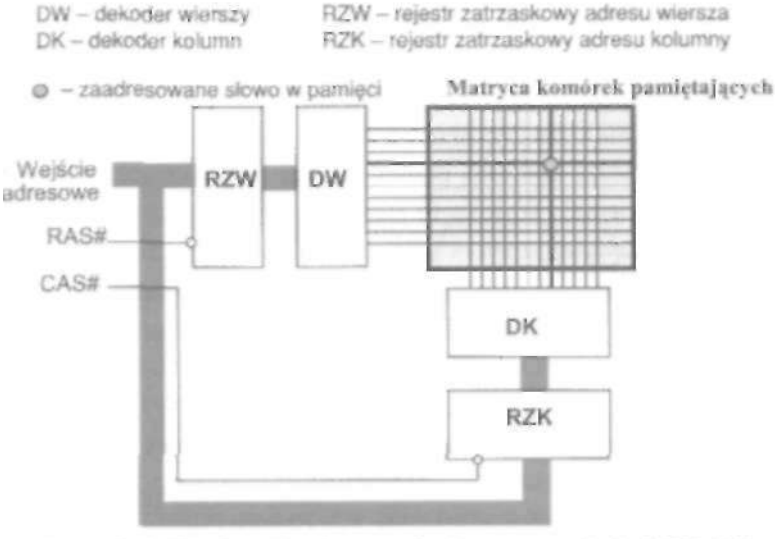
Schemat blokowy układu scalonego pamięci DRAM oraz rodzaje jego wyprowadzeń pokazane są na rysunku 2.42 a i b.

Adres słowa, na którym chcemy wykonać operację, podawany jest w dwóch częściach zwanych **adresem wiersza** i **adresem kolumny.** Zmniejsza to liczbę potrzebnych wyprowadzeń szyny adresowej i upraszcza konstrukcję dekoderów adresu. Z drugiej strony, układy logiczne kontrolera pamięci, sterujące pracą pamięci, muszą dokonać konwersji adresu podawanego przez procesor czy innego zarządcę magistral na postać wymaganą przez pamięć DRAM. Przykładowy układ dokonujący takiej konwersji pokazany jest na rysunku 2.43. Sygnał podany na wejście S multipleksera wybiera, czy starsza, czy też młodsza część adresu jest podawana na jego wyjściu.

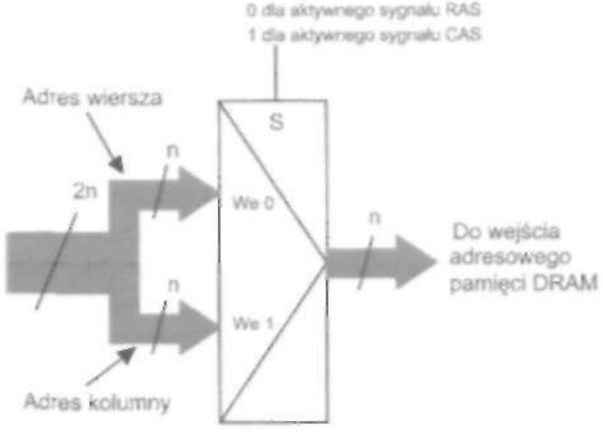
Drobne różnice występują też w wejściach sterujących pamięci. Zamiast wejścia R/W# mamy dwa wejścia: OE# - zezwolenie na wyprowadzenie (odczyt) informacji (ang. *Output Enable),* i WE# - zezwolenie na zapis (ang. *Write Enable).* Sygnał CE# (ang. *Chip Enable)* jest równoważny sygnałowi CS#. Sygnały RAS# (ang. *Row Address Strobę)* i CAS# (ang. *Column Address Strobę)* związane są z wprowadzaniem adresu do pamięci i są opisane w dalszej części tego podpunktu.



Rysunek 2.42a. Wyprowadzenia pamięci DRAM



Rysunek 2.42b. Sposób adresowania słowa w pamięciach DRAM



Rysunek 2.43. Układ konwersji adresu systemowego na adres dla pamięci DRAM

Poprawne zaadresowanie pamięci DRAM wymaga wykonania po kolei następujących czynności:

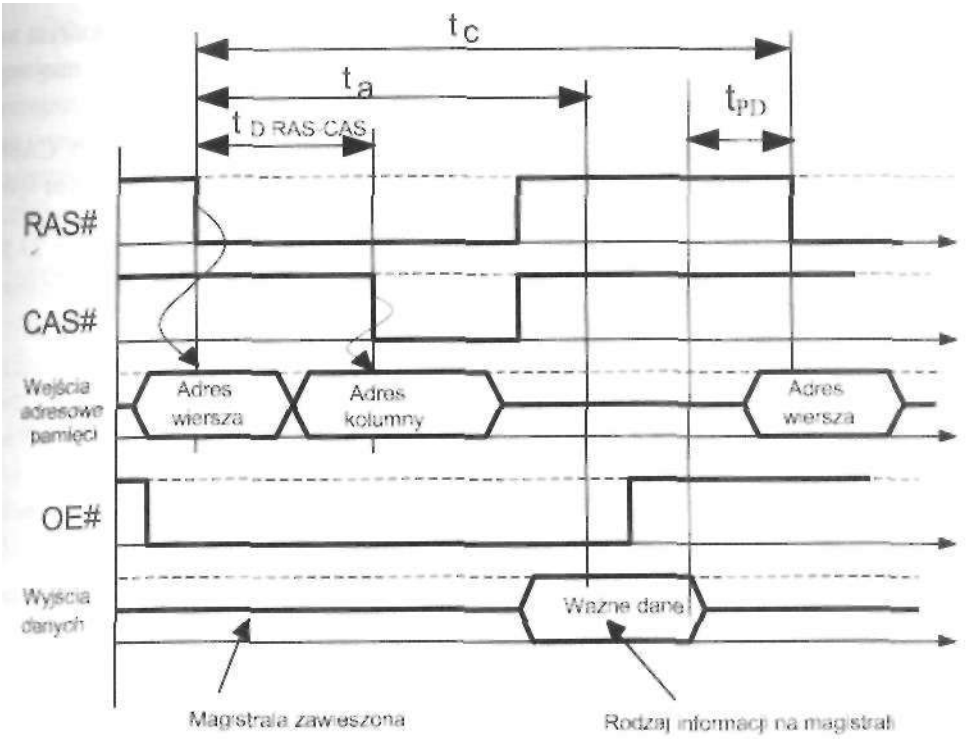
1. Podanie starszej części adresu na linie adresowe pamięci DRAM jako adresu wiersza, a następnie wytworzenie aktywnego zbocza sygnału RAS#, powodującego zapamiętanie tego adresu w rejestrze zatrzaskowym adresu wiersza.
2. Odmierzenie określonego,, wymaganego opóźnienia czasowego.
3. Podanie młodszej części adresu na linie adresowe pamięci DRAM jako adresuj kolumny i wytworzenie aktywnego zbocza sygnału CAS# powodującego zapamiętanie tego adresu w rejestrze zatrzaskowym adresu kolumny.

Następnie, zgodnie z sygnałami sterującymi OE# lub WE#, wykonywana jest operacja odczytu lub zapisu na zaadresowanym słowie. Po operacji odczytu odmierzane jest kolejne opóźnienie czasowe przed rozpoczęciem następnego cyklu, potrzebne do doładowania pojemności komórek pamiętających odczytywane słowo (w praktyce odświeżany jest cały wiersz). Wynika to stąd, że w trakcie sprawdzania stanu takiej pojemności jest ona w znacznej mierze rozładowywana.

Przebiegi na wyprowadzeniach pamięci DRAM w przypadku odczytu pokazane są na rysunku 2.44. Przy okazji wyjaśniono pewne konwencje stosowane przy rysowaniu przebiegów na magistralach.

Stan niski na wejściu OE# sygnalizuje operację zapisu. Po pojawieniu się na wejściu adresowym pamięci adresu wiersza jest on zatrzaskiwany w rejestrze zatrzaskowym adresu wiersza opadającym zboczem sygnału RAS# (zaznaczono to zygzakowatą strzałką, co jest jeszcze jedną konwencją stosowaną w tego typu rysunkach - początek strzałki to przyczyna, grot pokazuje skutek). Następnie kontroler pamięci podaje na wejście adresowe pamięci adres kolumny. Adres ten jest zatrzaskiwany w rejestrze zatrzaskowym kolumny opadającym zboczem sygnału CAS#. Zbocze opadające sygnału CAS# musi zostać opóźnione w stosunku do zbocza opadającego sygnału RAS# o określony czas, zwany opóźnieniem sygnału CAS# względem sygnału RAS#, oznaczany przez $t_{RAS-CAS}$. JEST TO związane z poprawnym zapisem adresu wiersza do rejestru oraz z ustaleniem się poprawnych wartości bitów adresu kolumny.

Po zatrzaśnięciu adresu kolumny oraz zdekodowaniu adresu wiersza i kolumny zawartość zaadresowanego słowa pojawia się na wyprowadzeniach danych układu pamięci. Przypominamy, że czas, jaki upływa od momentu podania prawidłowego adresu przez zarządcę magistral do momentu pojawienia się poprawnych danych na magistrali danych, nazywamy czasem dostępu. Oznaczany jest przez t_a. Po odczycie zawartości słowa musi upłynąć kolejny odcinek czasu t_{PD} potrzebny do doładowania komórek pamięci odczytywanego słowa (ang. *precharge delay).* Dopiero wówczas może się rozpocząć kolejny cykl dostępu do pamięci. Minimalny czas pomiędzy dwoma kolejnymi cyklami oznaczamy przez t_c (stosowane jest też oznaczenie T_{RC})



Rysunek 2.44. Operacja odczytu dla pamięci D R A M

Cykl magistrali - stany oczekiwania

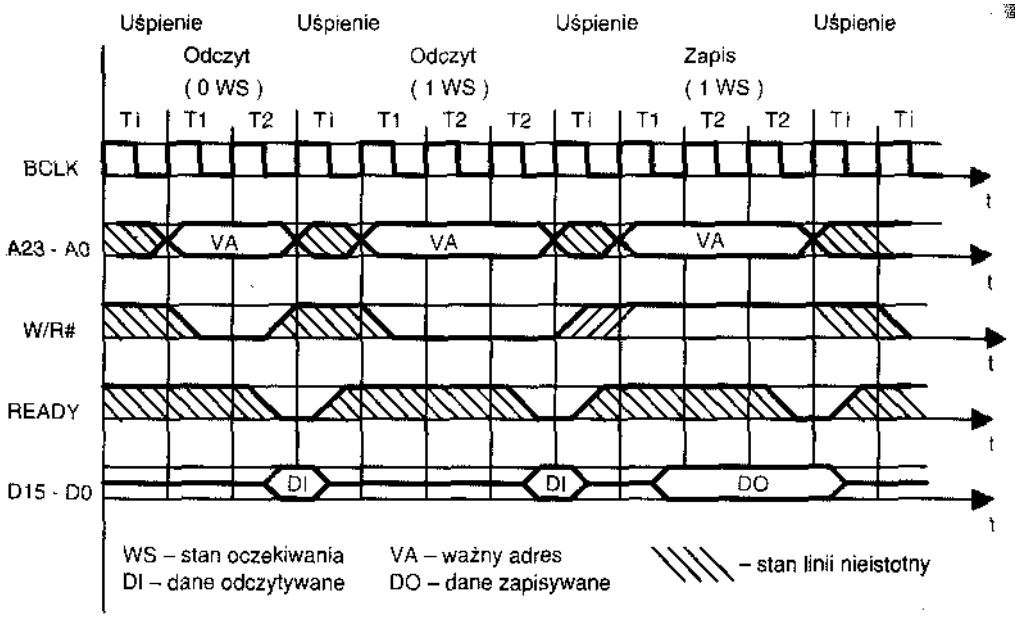
Jak wynika z poprzedniego podpunktu, dostęp do pamięci musi się odbywać z zachowaniem określonych zależności czasowych. Z drugiej strony praca magistrali taktowana jest zegarem o określonej częstotliwości, będącej zwykle podwielokrotnością częstotliwości zegara taktującego procesor. Wartość tej częstotliwości zależy też od rodzaju magistrali (ISA, PCI). Zegar magistrali oznaczany jest przez BCLK (ang. *Bus Clock).*

Rysunek 2.45 wyjaśnia pojęcie stanu oczekiwania (ang. *wait state).*

Dla magistrali ISA pojedynczy cykl magistrali realizowany jest w ciągu dwóch taktów zegara magistrali (BCLK) oznaczanych przez T_s (ang. *send status)* lub Tl oraz T_c (ang. *perform command)* lub T2. Dla większych częstotliwości tego zegara lub wolniejszych pamięci konieczne jest opóźnienie wykonania operacji na pamięci. Przykładowo w przypadku operacji odczytu musimy poczekać, aż na magistrali danych ustalą się prawidłowe wartości bitów odczytywanego słowa. Opóźnienie to jest realizowane przez wstawienie dodatkowych taktów zegara magistrali zwanych **stanami oczekiwania** (ang. *wait states).* Na rysunku 2.45 pokazano przykładowe cykle odczytu bez stanu oczekiwania i z jednym stanem oczekiwania oraz cykl zapisu

z jednym stanem oczekiwania. Cykle te mogą być rozdzielone czasem, w którym na magistrali nie są wykonywane żadne operacje. Stan taki, oznaczany przez T_i, nazywamy stanem uśpienia magistrali (ang. *idle state).*

Na rysunku 2.45. w celu jego uproszczenia, pominięte zostały sygnały **CAS#** i RAS# (tak byłoby w przypadku pamięci statycznych). Nie zmienia to w żaden sposób idei sianów oczekiwania.



Rysunek 2.45. Stany oczekiwania przy dostępie do pamięci DRAM

Odświeżanie pamięci DRAM

Odświeżanie komórek pamięci DRAM polega na cyklicznym doładowywaniu pojemności pamiętających przechowujących wartość 1. Częstotliwość odświeżania zapewniająca poprawną pracę pamięci DRAM jest podawana przez producenta jako parametr katalogowy, którego należy przestrzegać. Jak wyjaśniamy w podrozdziale 6.2.1.5, w systemie ISA za generowanie sygnału odświeżania pamięci odpowiedzialny jest timer 1. Operacja odświeżania pamięci realizowana jest przez układy logiczne odświeżania będące elementem systemu (płyty głównej).

Istnieją cztery podstawowe sposoby odświeżania pamięci dynamicznych RAM:

- sygnałem RAS (ang. *RAS only),*
- CAS przed RAS (ang. *CAS-before-RAS),*
- odświeżanie ukryte (ang. *hidden refresh),*
- samoodświeżanie (ang. *self-refresh).*

Pierwszym sposobem, obecnie niemającym już znaczenia, jest odświeżanie sygnałem RAS. Na sygnał z generatora odświeżania układy logiczne odświeżania przejmują kontrolę nad magistralami (stają się zarządcą magistral). Następnie podają na magistralę adresową zawartość tak zwanego licznika odświeżania. Licznik ten adresuje kolejne wiersze przeznaczone do odświeżenia i po każdym odświeżeniu kolejnego wiersza jest zwiększany o jeden. Po podaniu adresu generowany jest sygnał RAS powodujący odświeżenie zaadresowanego wiersza. Wysoki stan sygnału CAS# powoduje, że wyjście danych pamięci jest w stanie wysokiej impedancji.

Dwa następne sposoby wymagają obecności w układach pamięci wewnętrznego licznika odświeżania. W sposobie CAS przed RAS sterownik DRAM wytwarza aktywny sygnał CAS, a następnie sygnał RAS. W odpowiedzi na taką sekwencję układy pamięci DRAM odświeżają wiersz wskazywany przez ich wewnętrzny licznik odświeżania. W pamięciach DDR odpowiada to trybowi zwanemu autoodświeżaniem. Żądanie realizacji autoodświeżania jest generowane przez kontroler pamięci (przykładowo dla pamięci 512 Mb DDR SDRAM jest wymagany co około 7,5|i.s).

Przy odświeżaniu ukrytym po wytworzeniu aktywnych poziomów sygnałów RAS i CAS i odczycie komórki sygnał RAS zmienia kolejno stan na nieaktywny i aktywny przy stale aktywnym sygnale CAS. Powoduje to pozostawienie zawartości odczytywanej komórki na wyjściach danych przy jednoczesnym (równoległym) odświeżeniu wiersza zaadresowanego przez wewnętrzny licznik odświeżania pamięci.

Przy odświeżaniu automatycznym układy logiczne odświeżania są zawarte wewnątrz układów pamięci.

2.3.5.2. Odmiany pamięci dynamicznych

W tym punkcie omawiamy odmiany pamięci dynamicznych. Kolejne zmiany zapewniały krótszy czas dostępu i/lub lepszy transfer. Zaczynamy od omówienia metod dostępu do pamięci DRAM przyspieszających go, następnie omawiamy przeplot pamięci, pamięci SDRAM, odmiany pamięci DRAM i wreszcie przykładowe moduły pamięci.

Dostęp w trybie stronicowania i jego odmiany

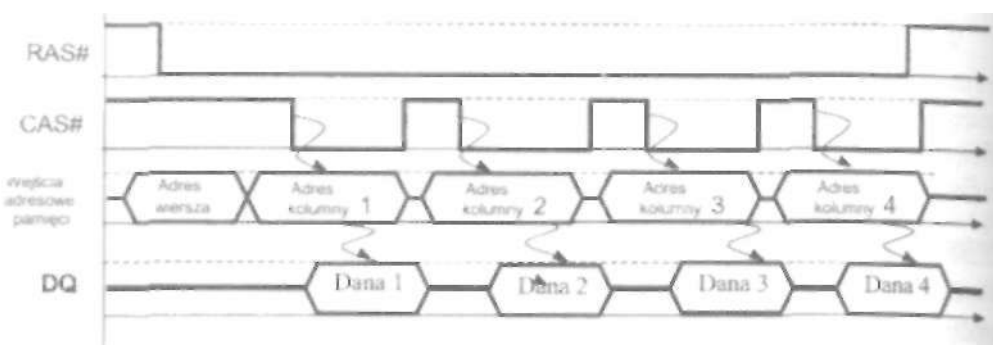
Dostęp do pamięci w trybie stronicowania jest sposobem na przyspieszenie współpracy z pamięcią DRAM. Wykorzystuje się tu dwa fakty. Po pierwsze większość odczytów dokonywana jest spod kolejnych, położonych koło siebie adresów. Oznacza to, że starsza część adresu, adres wiersza, nie zmienia się, a zmienia się jedynie adres kolumny. Wyjątkiem są tu słowa położone kolejno na początku i końcu wiersza. Drugim wykorzystywanym faktem jest to, że czas t_D RAS-CAS stanowi około 50 proc. czasu dostępu. Jeżeli przy odczytach kolejnych słów nie będziemy zmieniać adresu wiersza, a jedynie adres kolumny, to czas dostępu do pamięci ulegnie skróceniu (w przybliżeniu o czas opóźnienia sygnału CAS względem RAS). Wszystkie

odmiany trybu pracy ze stronicowaniem opierają się właśnie na tej zasadzie. Jednocześnie realizacja efektywnego trybu ze stronicowaniem wymaga od procesora możliwości adresowania potokowego, czyli podania adresu następnego słowa jeszcze i w czasie realizacji dostępu do słowa poprzedniego.

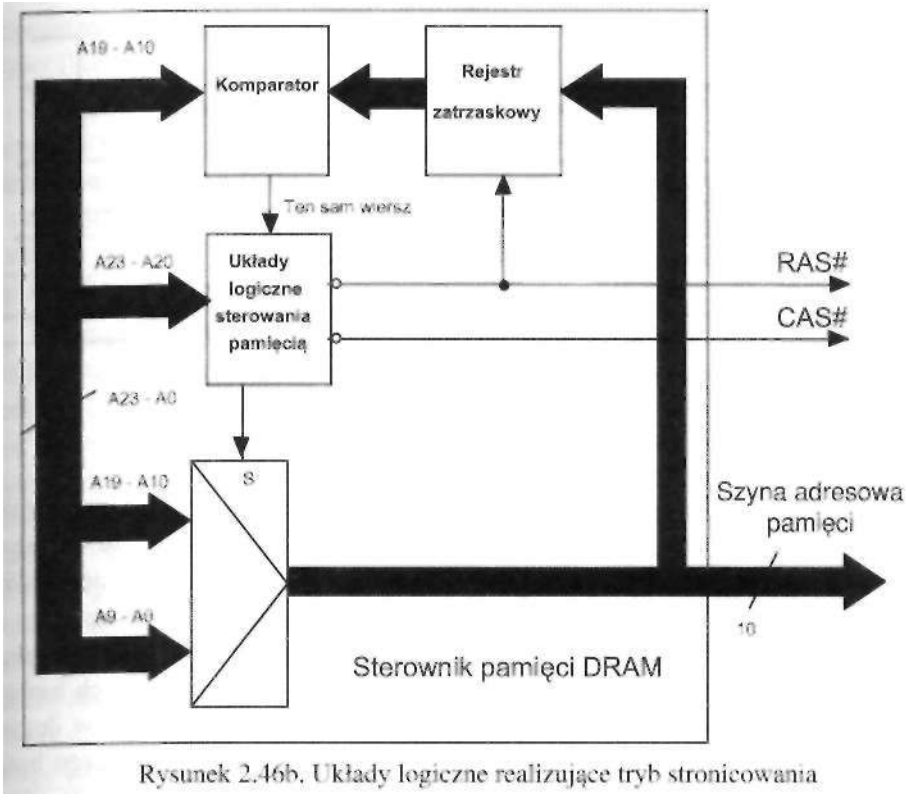
1. *Dostęp w trybie stronicowania*

Pamięć pracująca w trybie stronicowania wymaga pewnych dodatkowych układów przedstawionych na rysunku 2.46b. Początek dostępu do pamięci jest prawie identyczny jak dla zwykłych pamięci DRAM. Najpierw podawany jest adres wiersza. który jest zatrzaskiwany w rejestrze zatrzaskowym wiersza wewnątrz pamięci. Adres ten jest jednak zapamiętywany w rejestrze zatrzaskowym znajdującym się w bloku sterowania pamięcią DRAM. Następnie podawany jest adres kolumny, zatrzaskiwany w rejestrze zatrzaskowym kolumny wewnątrz układu pamięci.

Kolejny dostęp różni się od poprzedniego. Po podaniu nowego adresu przez zarządcę magistral jego część będąca adresem wiersza jest porównywana z zawartością rejestru adresu wiersza w układzie sterowania pamięcią. Jeżeli jest identyczna, układ sterowania pamięcią DRAM utrzymuje stan niski sygnału RAS# do końca bieżącego cyklu odczytu. Dla pamięci DRAM pracującej w trybie stronicowania oznacza to, że kolejny dostęp dotyczy słowa położonego w tym samym wierszu i należy wczytać jedynie adres kolumny. Powoduje to ominięcie generowania opóźnienia sygnału CAS# względem RAS# i załadowanie od razu adresu kolumny sygnałem CAS#. Przykład układu logicznego potrzebnego do obsługi pamięci DRAM oraz przebiegi czasowe przy dostępie do pamięci w trybie stronicowania pokazujemy na rysunku 2.46.



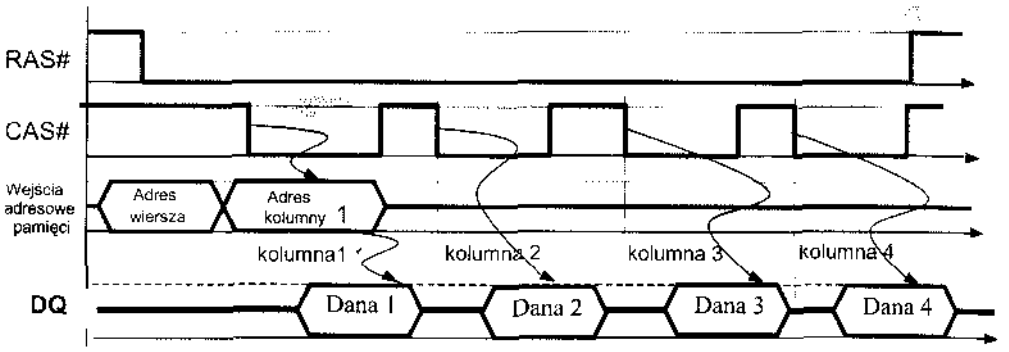
Rysunek 2.46a. Praca pamięci DRAM w trybie stronicowania



Rysunek 2.46b. Układy logiczne realizujące tryb stronicowania

2. Dostęp w trybie seryjnym (burst)

Dostęp w trybie seryjnym (ang. *burst)* stosowany jest przy współpracy pamięci głównej z pamięcią cache (opisaną w rozdziale 3.7). Pamięć ta odczytuje bądź zapisuje informacje liniami, których długość zależy od rozwiązania pamięci cache i przykładowo dla systemów z procesorem Pentium wynosi 32 bajty. Ponieważ procesor ten ma magistralę danych 64-bitową (8 bajtów), do wypełnienia linii potrzeba czterech dostępów do pamięci. Operacje te dotyczą jednak kolejnych, leżących obok siebie słów. Oznacza to, że adres wiersza nie będzie się zmieniał, a adres kolumny przy każdym kolejnym dostępie będzie większy o jeden. Jeżeli wewnątrz pamięci umieścimy układ, który będzie zwiększał wartos'ć adresu kolumny o 1 po każdym podaniu zbocza aktywnego sygnału CAS#, to taka pamięć może pracować w trybie burst. W trybie tym podajemy adres wiersza i kolumny pierwszego słowa. Następnie podajemy kolejne zbocze aktywne sygnału CAS#, nie podając kolejnych adresów kolumny, gdyż generowany jest on wewnątrz pamięci. Zysk czasowy wynika z braku konieczności zapewnienia tak zwanego czasu ustalania oraz czasu przetrzymania dla adresu kolumny. Przebiegi na wejściach adresowych oraz wejściach sterujących CAS# i RAS# pamięci pracującej w trybie burst pokazane są na rysunku 2.47.



Rysunek 2.47. Praca pamięci DRAM w trybie seryjnym (burst)

Przeplot pamięci

Inną metodą pozwalającą zwiększyć szybkość komunikacji z pamięcią jest stosowanie tak zwanego przeplotu. Pomysł bazuje ponownie na fakcie, że większość odczytów z pamięci dokonywana jest z kolejnych, położonych obok siebie słów. W przypadku odczytów następujących jeden po drugim musimy zapewnić czas na doładowanie pojemności pamiętających (patrz podrozdział 2.3.5.1). Możemy jednak sąsiadujące słowa rozmieścić na przemian w dwóch (lub czterech) różnych bankach (układach scalonych) pamięci - adresy parzyste w jednym, nieparzyste w drugim. Wówczas przy odczycie kolejnych słów po odczytaniu słowa z pierwszego banku możemy bez oczekiwania dokonać odczytu z drugiego banku, gdyż jest to odczyt z innego układu scalonego. W tym czasie w pierwszym banku zostaną doładowane pojemności pamiętające komórek odczytanego wiersza.

Pamięć SDRAM

Modyfikacja wprowadzona w pamięci SDRAM polega na zsynchronizowaniu operacji pamięci z zewnętrznym zegarem. Zmiana dotyczy więc interfejsu pomiędzy pamięcią a systemem. Opisywane do tej pory pamięci pracowały asynchronicznie w stosunku do procesora, który z kolei jest układem synchronicznym. Synchronizacja operacji pamięci z zegarem procesora pozwala osiągnąć optymalną szybkość współpracy obydwu układów oraz pozwala uniknąć przypadkowych opóźnień, na przykład gubienia niektórych cykli magistrali spowodowanych brakiem synchronizacji zegarów taktujących pamięci i procesor. Pamięć SDRAM nadaje się zwłaszcza do współpracy z pamięcią podręczną (cache). Nieco więcej informacji na temat tych pamięci podajemy w następnym punkcie.

Przegląd stosowanych pamięci DRAM

Na rynku obecnych jest kilka rodzajów pamięci dynamicznych RAM. Najstarszym, który praktycznie niemal wyszedł z użycia, są pamięci oznaczane jako FPM DRAM (ang. *Fast Page Mode DRAM).* Stosowane przy współpracy z nimi metody

przyspieszania dostępu to: adresowanie potokowe, praca z przeplotem oraz odczyt seryjny (burst).

Następcą pamięci FPM DRAM były pamięci EDO DRAM (ang. *Enhanced Data Output DRAM).* W pamięciach tych zmniejszona jest liczba cykli oczekiwania o 1 dzięki temu, że dane utrzymywane są na wyjściu pamięci po przejściu sygnału CAS# w stan wysoki (stąd ich nazwa). Pozwala to wcześniej wyznaczyć następny adres w wierszu. Dane są zdejmowane z wyjścia pamięci dopiero po ponownym przejściu sygnału CAS# w stan niski.

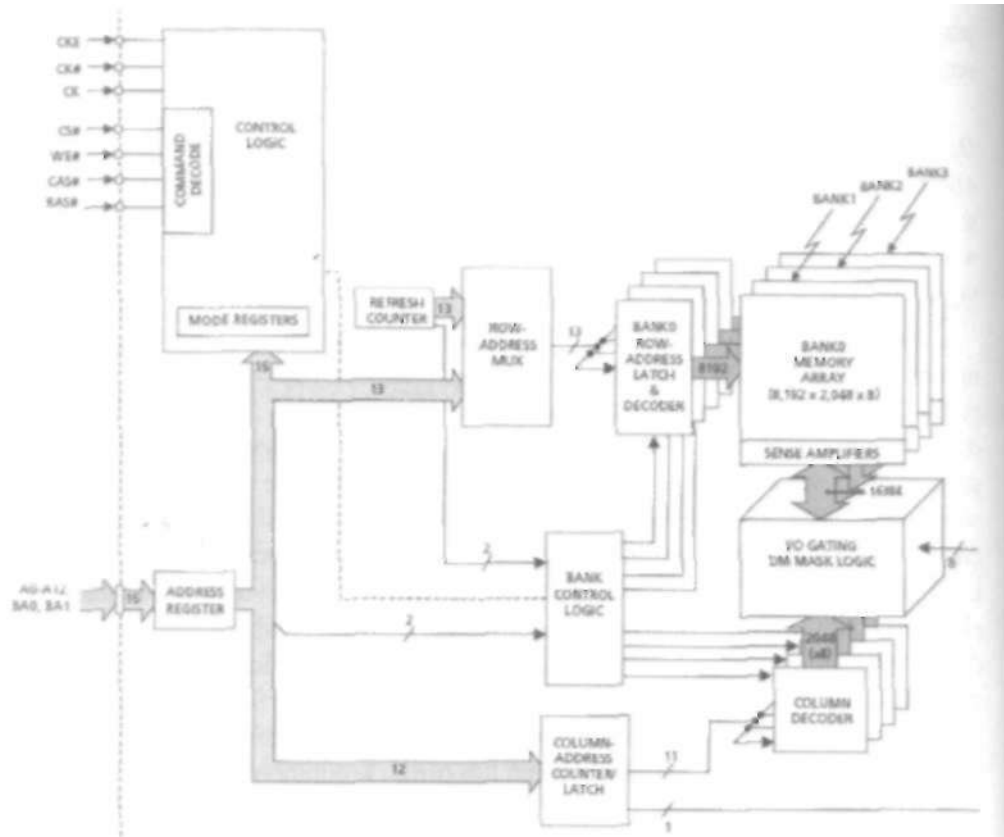
Kolejnymi układami pamięci dynamicznych są pamięci SDRAM - synchroniczne pamięci DRAM. Jak wspomniano wcześniej, jest to pamięć, na której operacje przebiegają synchronicznie z taktem zegara systemowego. Sygnały sterujące powodują wykonanie określonej operacji po pojawieniu się aktywnego zbocza (np. narastającego) na wejściu zegarowym. Taki tryb pracy eliminuje dodatkowe stany oczekiwania pojawiające się w wyniku przypadkowych przesunięć czasowych pomiędzy taktem procesora a sygnałami taktującymi działanie pamięci (dla zwykłych pamięci DRAM sygnały te nie są ze sobą zsynchronizowane, czyli przesunięcia czasowe pomiędzy nimi są przypadkowe). Moduły tych pamięci mają organizację 64-bitową, dostosowaną do szerokości magistrali procesora Pentium. Ich konstrukcja jest optymalizowana pod kątem pracy w trybie seryjnym (burst, jak zobaczymy, jest predestynowany do współpracy z pamięcią cache). Istnieje możliwość programowania długości tego odczytu (równej dwóm, czterem lub ośmiu kolumnom).

Początkowo pamięci SDRAM transmitowały informację tylko na jednym, narastającym zboczu zegarowym. Później określano te pamięci jako SDR SDRAM (ang. *Single Data Rate SDRAM).* Następne wersje pamięci SDRAM: DDR (ang. *Double Data Rate)* i DDR2, przesyłały dane już na obu zboczach przebiegu zegarowego. Wzrastała też częstotliwość tego zegara.

Funkcjonowanie pamięci SDRAM omówimy na przykładzie pamięci DDR SDRAM MT46V128M4 (32 M x 4b x 4 banki) firmy Micron Technology, Inc.

Na rysunku 2.48 przedstawiamy wewnętrzną strukturę tej pamięci (pominięto dla większej czytelności układy wyjściowe, pokazane z kolei na rysunku 2.50). Pamięć ta jest taktowana zegarem synchronicznym z zegarem procesora. Przebieg zegarowy jest symetryczny (różnicowy) i podawany na wejścia CK i CK#. Pamięć jest sterowana poleceniami wprowadzanymi przez określoną konfigurację sygnałów CS#, WS#, RAS#, CAS#, dekodowanymi przez blok COMMAND DECODE zgodnie z tabelą 2.16.

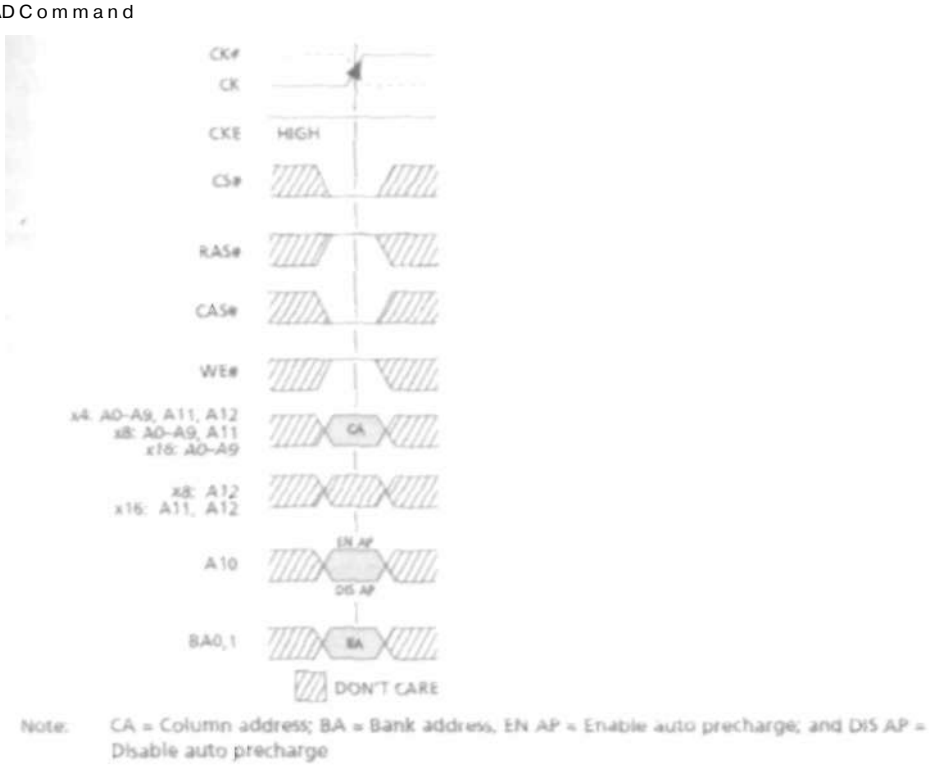
Wprowadzanie poleceń następuje na każdym narastającym zboczu zegara CK. Przykład wprowadzania polecenia odczytu (READ) pokazuje rysunek 2.49.



Rysunek 2.48. Wewnętrzna struktura pamięci DDR SDRAM (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

Tabela 2.16. Operacje pamięci DDR SDRAM (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

Name (Function)	CS#	RAS#	CAS#	WE#	Address	Notes
DESELECT (NOP)	H	X	X	X	X	1
NO OPERATION (NOP)	L	H	H	H	X	1
ACTIVE (Select bank and activate row)	L	L	H	H	Bank/Row	2
READ (Select bank and column, and start READ burst)	L	H	L	H	Bank/Col	3
WRITE (Select bank and column, and start WRITE burst)	L	H	L	L	Bank/Col	3
BURST TERMINATE	L	H	H	L	X	4
PRECHARGE (Deactivate row in bank or banks)	L	L	H	L	Code	5
AUTO REFRESH or SELF REFRESH (Enter self refresh mode)	L	L	L	H	X	6, 7
LOAD MODE REGISTER	L	L	L	L	Op-Code	8

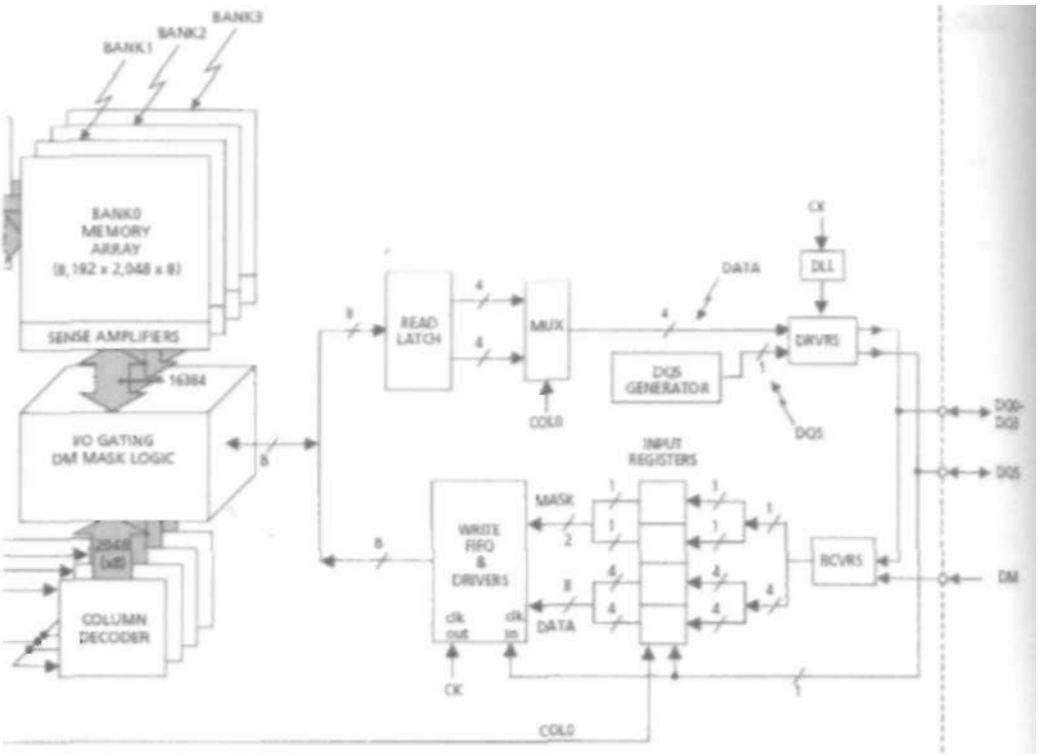


Rysunek 2.49. Wprowadzenie polecenia READ dla pamięci DDR SDRAM (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

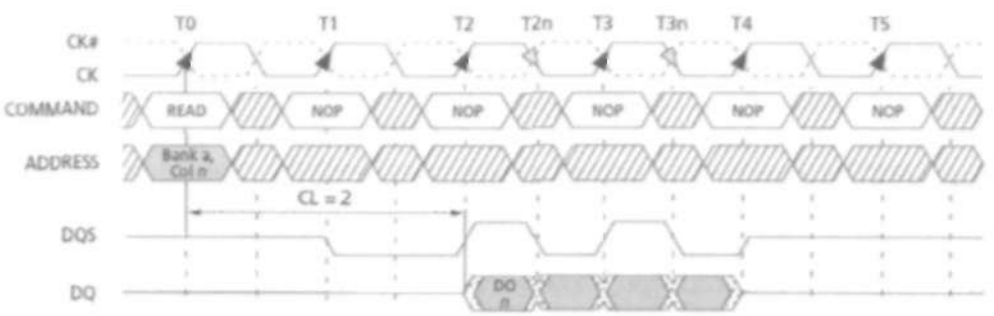
W trakcie tej operacji jest także wprowadzany do układu pamięci adres żądanej kolumny (numer banku + adres kolumny - patrz schemat blokowy pamięci).

Prezentowana pamięć to DDR SDRAM. Pamięć ta przesyła dane zarówno na zboczu narastającym, jak i opadającym zegara. Odczyty z banków pamięci następują jednak tylko dla zbocza narastającego. Oznacza to, że odczytywane słowa mają podwójną długość w stosunku do transmitowanych, i są przez układy wyjściowe pamięci dzielone na dwie równe części. Pozwala to zwiększać szybkość przesyłania bez potrzeby poprawiania czasu dostępu. Układy realizujące opisane operacje pokazane są na rysunku 2.50.

Układ oznaczony jako MUX jest dwuwejściowym, czterobitowym multiplekserem (patrz punkt 2.2.5). Opisywana pamięć ma organizację 4-bitową. Z banku pamięci odczytywanych jest jednak przy zboczu narastającym zegara 8 bitów, które następnie za pomocą multipleksera dzielone są na dwa słowa 4-bitowe wysyłane kolejno przy zboczu narastającym i opadającym zegara. Przykład odczytu pamięci w trybie seryjnym (burst) przedstawia rysunek 2.51.



Rysunek 2.50. Układy wyjściowe pamięci DDR SDRAM (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)



Rysunek 2.51. Odczyt w trybie seryjnym dla pamięci D D R S D R A M (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

Proszę zwrócić uwagę na transmisję danych zarówno dla zbocza zegara T (narastającego), jak i Tn (opadającego, ang. *negative)*.

Na rysunku 2.49 zilustrowany jest także jeden z parametrów pamięci SDRAM oznaczany jako CL (ang. *CAS Latency)*. Jest to opóźnienie pojawienia się danej w sto-

sunku do sygnału CAS# powodującego wprowadzenie adresu kolumny tej danej. Jest to więc jakby odpowiednik czasu dostępu w trybie seryjnym. Parametr CL podawany jest w taktach zegara CK.

Pewnych wyjaśnień wymagają oznaczenia modułów pamięci SDRAM, w których wprowadzono zamęt. Początkowo moduły SDR SDRAM oznaczano PC xxx, na przykład PC 100, gdzie xxx było maksymalną częstotliwością zegara taktującego. Po pojawieniu się pamięci DDR SDRAM oznaczano ich typ jako DDR yyy, na przykład DDR 200, gdzie yyy było podwojoną częstotliwością zegara, czyli częstotliwością transferu danych. Dla takich pamięci stosowano też oznaczenia PC zzzz, na przykład PC 1600. Tym razem zzzz jest transferem w MB/s. Tak więc DDR 200 i PC 1600 to dwa oznaczenia tego samego typu pamięci (wynika to z faktu, że pamięci te przesyłają w jednym transferze 8 bajtów, a więc 8 B x 200 MHz = 1600 MB/s). Kilka przykładów oznaczeń podano w tabeli 2.17.

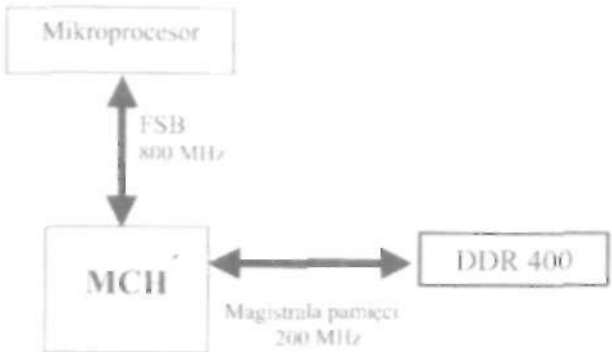
Tabela 2.17. Przykłady oznaczeń pamięci

Nazwa	Typ	**Częstotliwość zegara**	**Przepustowość**
PC 66	SDRAM	66 MHz	**0,5** GB/s
PC 133	SDRAM	133 MHz	1,06 GB/s
PC 2100	DDR 266	133 MHz	**2,1** GB/s
PC 2700	DDR 333	166 MHz	**2,7** GB/s
PC 4200	DDR 533	266 MHz	**4,2** GB/s

Dual Channel Memory

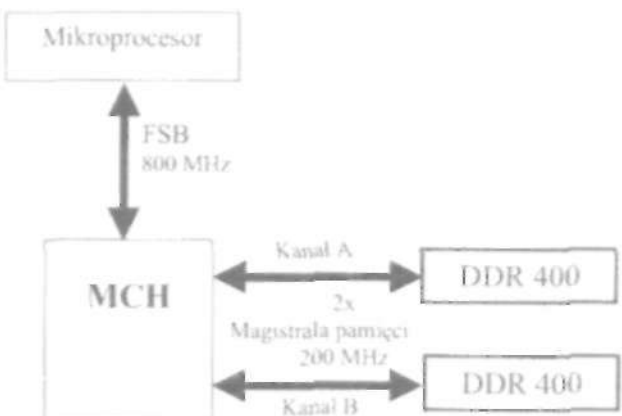
Poniżej krótko opisujemy rozwiązanie znane pod nazwą Dual Channel Memory. Nie jest to nowe rozwiązanie budowy modułów pamięci ani też stosowanych w nich układów pamięci, lecz nowy sposób komunikacji z procesorem, w której pośredniczy oczywiście kontroler pamięci będący częścią chipsetu oznaczanego MCH (ang. *Memory Control Hub* - patrz rozdział 6.3). Klasyczna komunikacja pomiędzy pamięcią a procesorem wygląda jak na rysunku 2.52.

W rozwiązaniu tym magistrala FSB (ang. *Frunt Sicie Bus)* procesora będzie przy komunikacji z pamięcią wykorzystana jedynie w połowie, jako że jej transfer wynosi 800 MHz x 8 B = 6,4 GB/s, podczas gdy transfer magistrali pamięci wynosi 200 MHz x 8 B x 2 = 3 , 2 GB/s (mnożymy przez 2, gdyż transmitujemy na obu zboczach zegara, czyli dwa razy w ciągu jednego okresu).



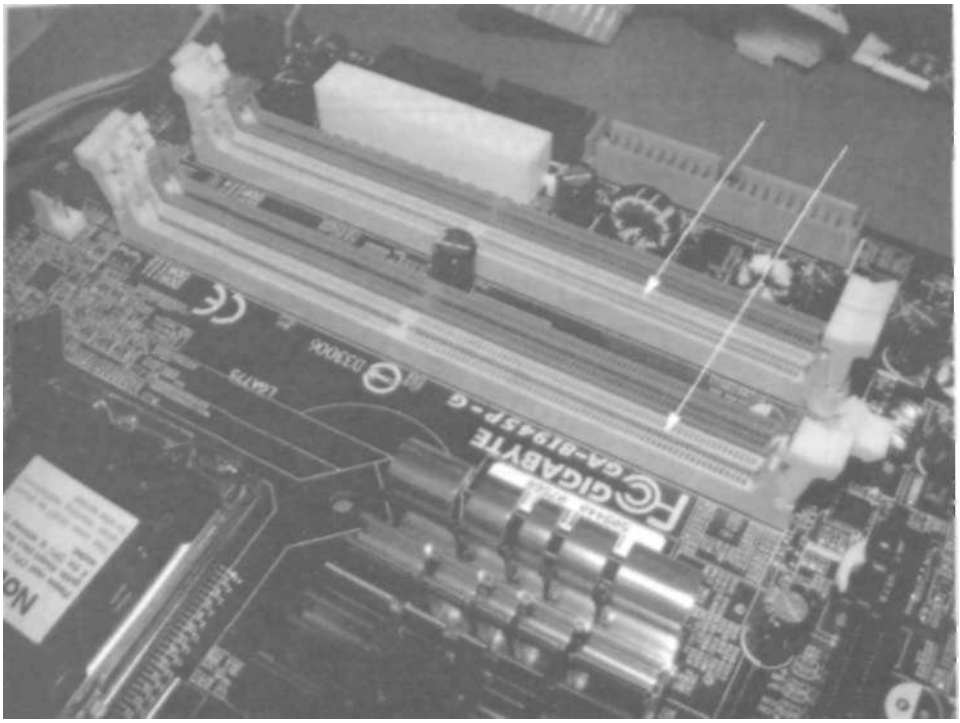
Rysunek 2.52. Klasyczny sposób komunikacji pamięci z procesorem

W płytach głównych pojawiło się więc rozwiązanie pokazane na rysunku 2.53.



Rysunek 2.53. Komunikacja procesora z pamięcią w trybie Dual Channel

Tym razem, mimo że dysponujemy modułami pamięci tego samego typu co poprzednio, w pełni wykorzystamy przepustowość magistrali FSB. Rozwiązanie to wymaga oczywiście przystosowania do niego płyty głównej, a więc między innymi chipsetu. Płyty z tym rozwiązaniem mają gniazda DIMM tego samego koloru dla kanału A i kanału B w dwóch grupach gniazd (rysunek 2.54), w celu ułatwienia i c h obsadzania. Chcąc wykorzystać pamięć pracującą w trybie Dual Channel, należy zainstalować przykładowo dwa moduły w gniazdach o tym samym kolorze w różnych grupach gniazd, tak jak to pokazano strzałkami na rysunku 2.54. Natomiast w tabeli 2.18 podajemy przykładowe transfery dla rozwiązania klasycznego i Dual Channel. W tabeli tej podajemy też maksymalną długość cyklu pracy pamięci wymaganą, a b y zrealizować daną częstotliwość zegara.



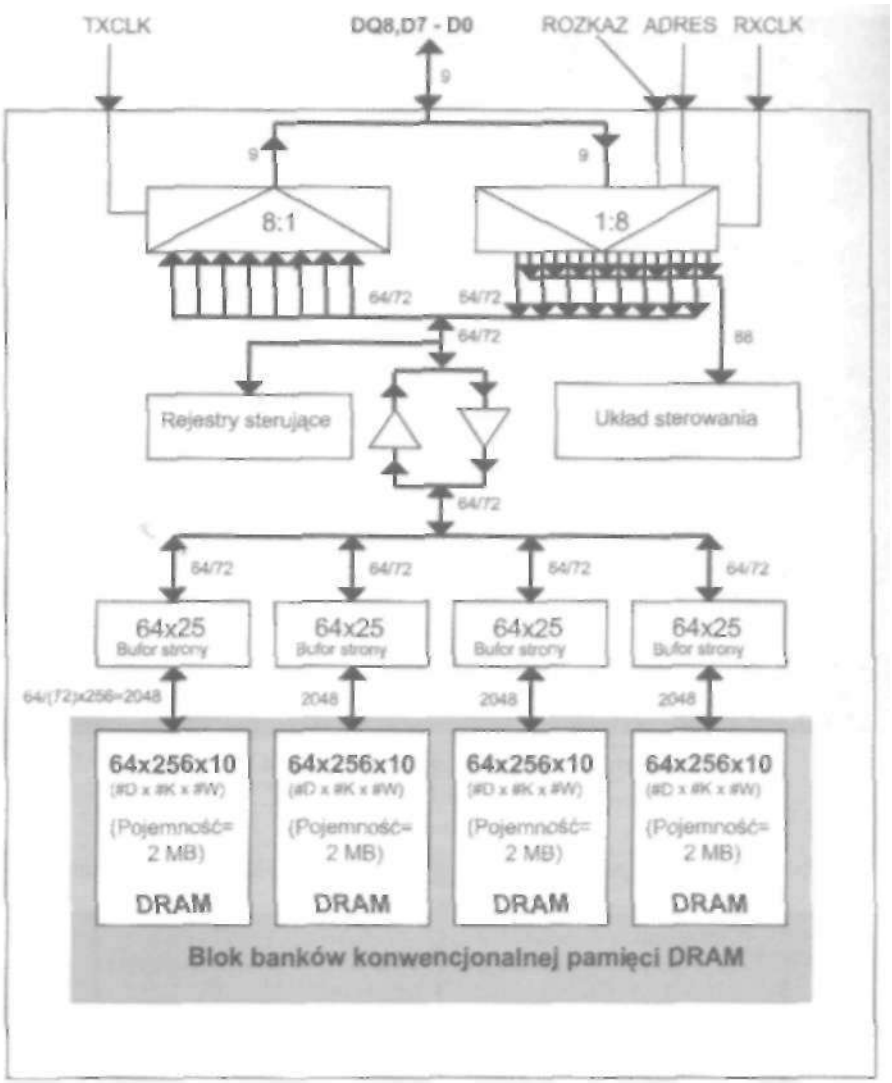
Rysunek 2.54. Płyta główna z gniazdami DIMM w standardzie Dual Channel

Tabela 2.18. Porównanie pracy pomięci w trybie zwykłym i Dual Channel

Częstotliwość taktowania magistrali pamięci	Wymagany czas cyklu	Transfer SDR SDRAM	Transfer SDR SDRAM w trybie Dual Channel	Transfer DDR SDRAM	Transfer DDR SDRAM w trybie Dual Channel
100 MHz	**10 ns**	800 MB/s	1600 MB/s	1600 MB/s	3200 MB/s
200 MHz	5ns	1600 MB/s	3200 MB/s	3200 MB/s	6400 MB/s
266 **MHz**	**1,9 ns**	2100 MB/s	4200 MB/s	4200 MB/s	8400 MB/s

Pamięci Rambus

Innym typem pamięci dynamicznych są RDRAM. Zastosowano w nich nową koncepcję funkcjonowania, którą omawiamy. Podstawowe bloki wchodzące w skład modułu pamięci RDRAM przedstawia rysunek 2.55.



Rysunek 2.55. Schemat blokowy 8 MB pamięci R D R A M

Informacja jest przechowywana w bloku standardowej pamięci DRAM. W opisywanej pamięci o pojemności 8 MB (64 Mb) składa się on z czterech banków o pojemności 2 MB każdy. Banki te są zorganizowane jako 1024 wiersze zawierające 256 kolumn 64-bitowych komórek (72 bity, uwzględniając kontrolę parzystości). Pojemność wiersza wynosi więc 2048 bitów.

Pozostałe układy opisywanej pamięci (na schemacie blokowym umieszczone poza zacieniowanym obszarem) są wykonane w technologii pozwalającej na ich pracę z bardzo dużą szybkością, która decyduje o szybkości działania tych pamięci.

Komunikacja z pamięcią RDRAM przebiega przez 8-bitową magistrale D7+D0. Przesyłane są nią pakiety (paczki) informacji zawierające adresy, dane lub rozkazy dotyczące wykonania przez pamięć okres'lonych operacji. Każdy pakiet ma pojemność 72 bitów (9x8 bajtów). Transmisja taktowana jest sygnałem RXCLK bądź TXCLK. Po wprowadzeniu do pamięci przykładowo rozkazu odczytu i adresu wiersza w żądanym banku wiersz ten jest ładowany do odpowiedniego bufora strony (ze standardowym czasem dostępu rzędu 10 ns). Z buforów tych odczytywane są (z dużą szybkością- 16 ns/bajt) żądane kolumny (64-bitowe), które za pośrednictwem multipleksera transmitowane są na zewnątrz jako pojedyncze bajty.

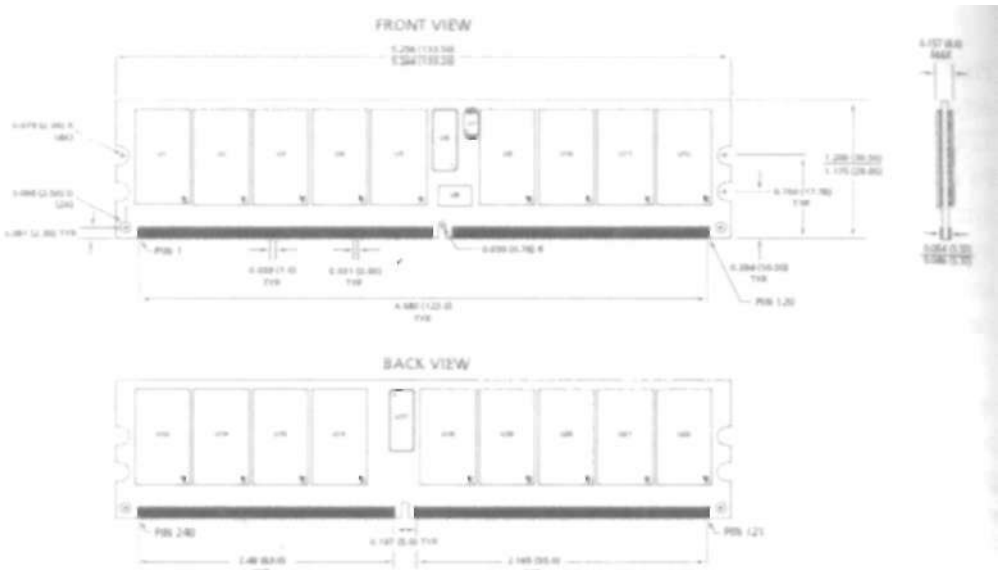
Opisana tu technologia i architektura pamięci nosi nazwę Rambus™ i została opracowana przez kalifornijską firmę o tej samej nazwie. Uzyskano w niej znaczną poprawę transferu, jednak czas dostępu nie uległ dużej zmianie. Prawdopodobnie z tego powodu, a także ze względów ekonomicznych technologia ta nie odniosła istotnego sukcesu rynkowego.

Z pamięciami Rambus™ dość skutecznie konkurują kolejne wersje synchronicznych pamięci DRAM: DDR SDRAM (ang. *Double Data Rate DRAM)* i DDR2 SDRAM.

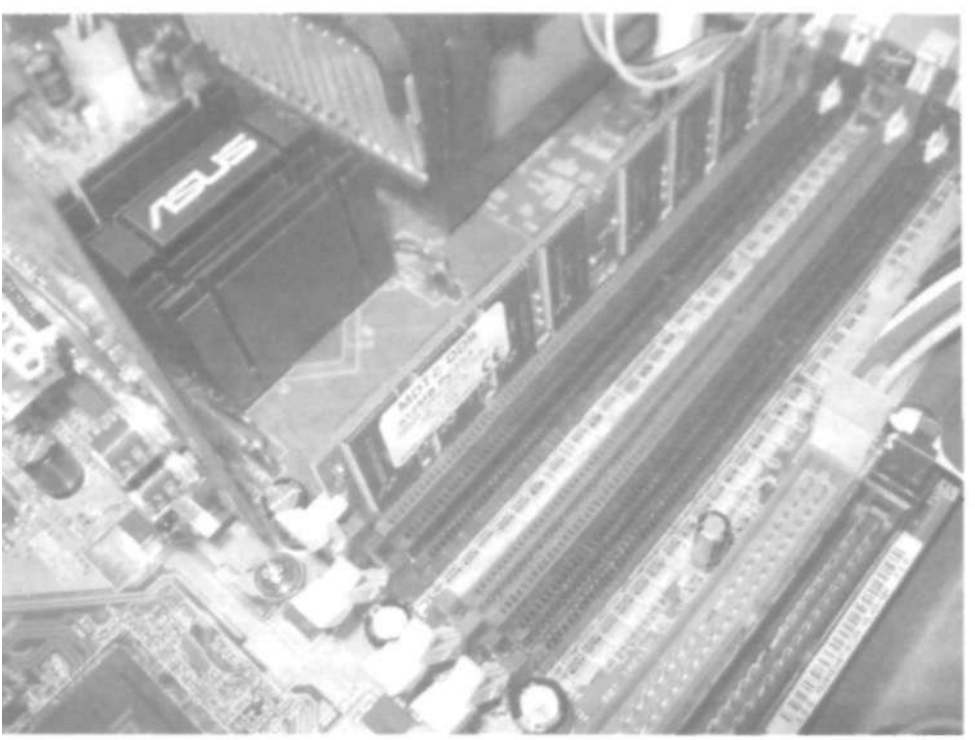
DRAM mogą być pamięciami niebuforowanymi (ang. *unbuffered),* buforowanymi (ang. *buffered)* lub rejestrowymi (ang. *registered).* Dwa ostatnie typy pamięci są zalecane do stosowania w większych zespołach pamięci. W przypadku tych pamięci w wymianie informacji pośredniczą rejestry buforowe. Dodatkowo dla pamięci rejestrowych buforowane są także polecenia sterujące. Powoduje to nieco wolniejsze, ale za to znacznie pewniejsze działanie tych pamięci.

2.3.6. Moduły pamięci

Koncepcja budowy komputera 1BM-PC zakłada elastyczność jego konfiguracji. Osiągnięto to przez jego modułową budowę. Pierwszym rozwiązaniem zapewniającym taką budowę była realizacja części podzespołów tworzących jednostkę centralną komputera w postaci oddzielnych płytek drukowanych zwanych kartami, montowanymi w specjalnie przeznaczonych do tego celu gniazdach, zwanych gniazdami rozszerzeń (ang. *expansion slots)* albo gniazdami magistrali rozszerzającej, i komunikujących się z systemem (procesorem i pamięcią) przez magistralę rozszerzającą (ang. *expansion bus).* Jednym z kolejnych kroków było zapewnienie możliwości rozbudowy pamięci. Pierwotne rozwiązanie polegające na montażu układów scalonych pamięci bezpośrednio na płycie głównej zastąpiono montażem pamięci na różnego rodzaju modułach pamięci umieszczonych w specjalnych, przeznaczonych do tego celu złączach. Moduły pamięci są płytkami drukowanymi, na których umieszczone są zespoły układów scalonych pamięci. Wygląd przykładowego modułu pamięci DDR2 SDRAM, o oznaczeniu MT18HTF25672(P)D (2GB) firmy Micron Technology, wraz z jego wymiarami przedstawia rysunek 2.56.

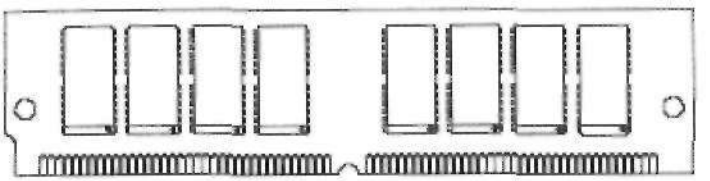


Rysunek 2.56. Wygląd modułu pamięci DDR SDRAM (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

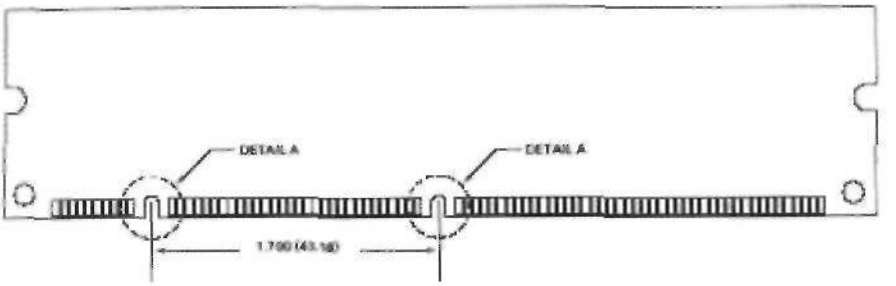


Rysunek 2.57. Moduł DIMM DDR SDRAM w gnieździe

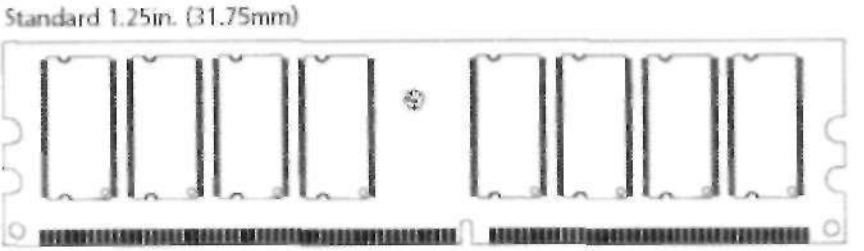
Nie będziemy prezentować całej historii rozwoju modułów pamięci dla PC. Zaczniemy od modułów SIMM 72, które obecnie także już niełatwo spotkać, a zakończymy na modułach DIMM.



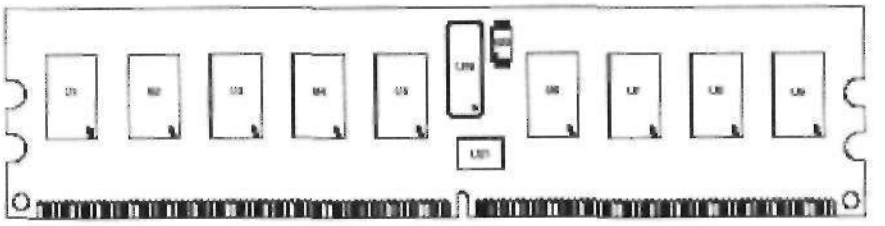
Rysunek 2.58a. Moduł SIMM 72 (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)



Rysunek 2.58b. Moduł 168 PIN DIMM (SDR SDRAM) (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)



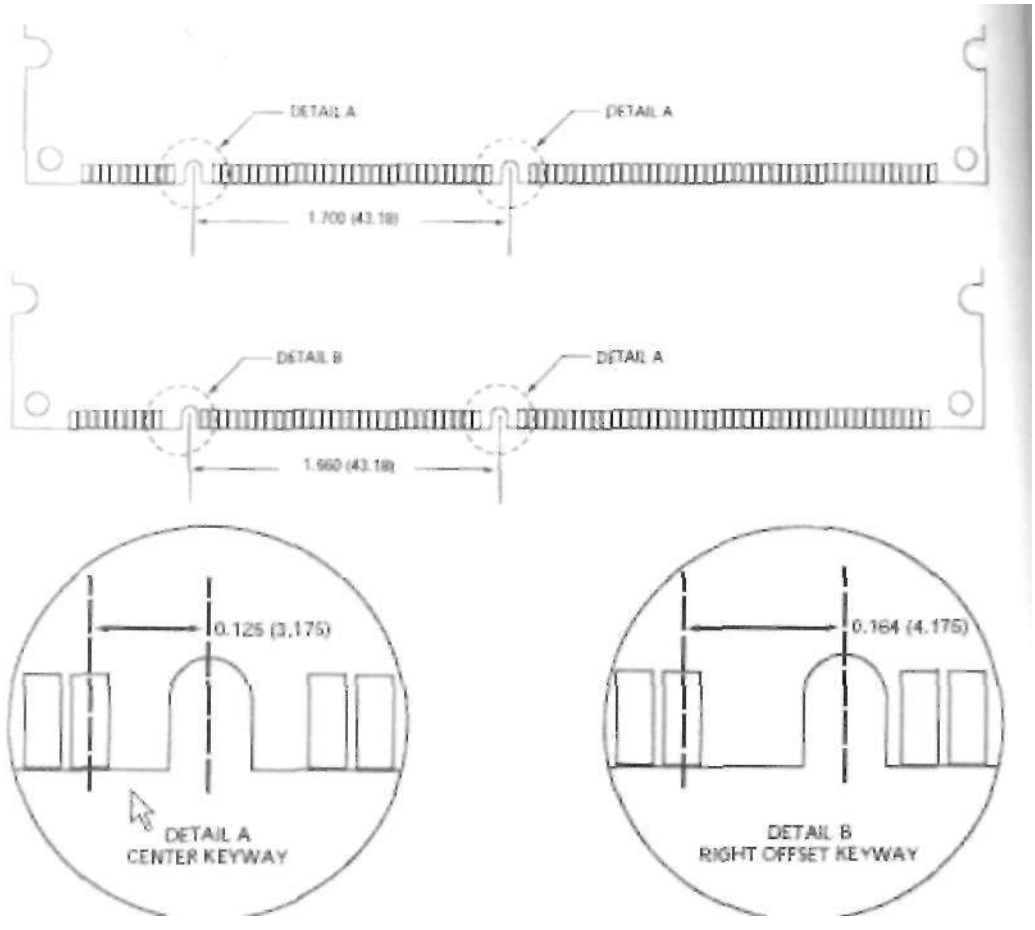
Rysunek 2.58c. Moduł 184 PIN DIMM (DDR SDRAM) (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)



Rysunek 2.58d. Moduł 240 PIN DIMM (DDR2 SDRAM) (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

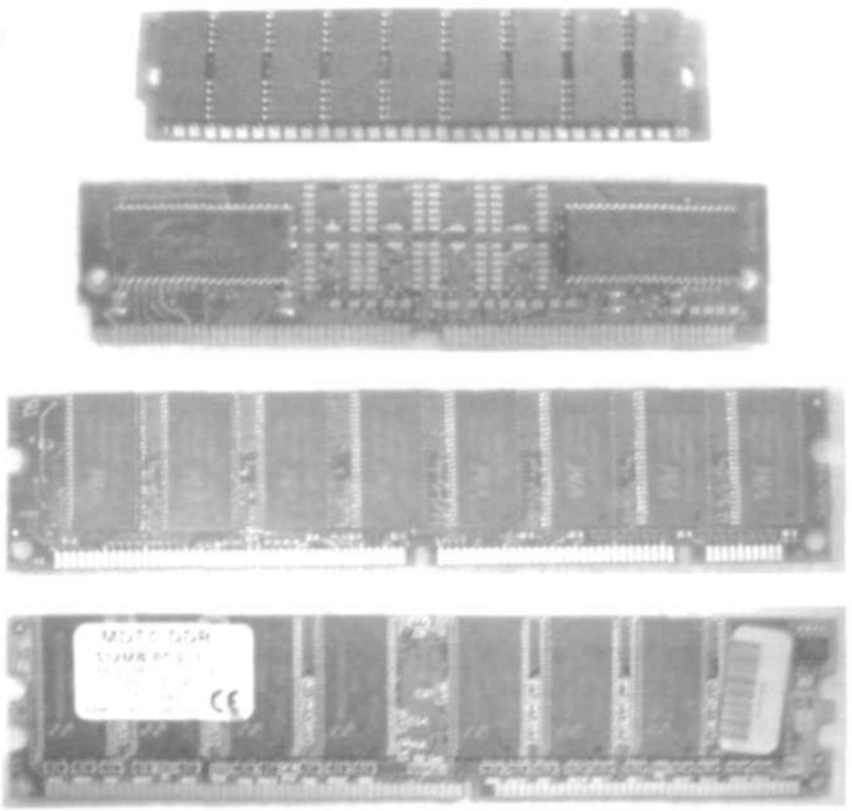
Na rysunku 2.58a pokazano moduł SIMM 72-pinowy (tak zwany „peesowy" od j nazwy komputerów PS/2, w których zastosowano je po raz pierwszy). Na tego typu modułach mogły być umieszczone przykładowo pamięci FPM lub EDO.

Kolejne rysunki przedstawiają: 2.58b - moduły DIMM dla pamięci SD RAM, / 2.58c - moduły DIMM dla pamięci DDR SDRAM i 2.56d - moduły DIMM dla pamięci DDR2 SDRAM. Zwracamy uwagę na niewielkie różnice w wyglądzie. Stosunkowo najpewniejszym ktyterium określenia rodzaju pamięci na modułach DIMM jest liczba kontaktów. Dla SDR jest to 168, z boku modułu jest jedno wycięcie (choć bywają odstępstwa), ponadto w złączu są dwa klucze. Dla DDR są 184 kontakty, dwa wycięcia z boku i jeden klucz w złączu. DDR2 różnią się liczbą kontaktów - 240. I Oczywiście klucze umieszczane są w różnych miejscach, jednak trudno to ocenić wzrokowo. Na rysunku 2.59 zamieszczamy przykład różnicy położenia klucza dla pamięci buforowanej i niebuforowanej.



Rysunek 2.59. Przykład różnicy w budowie modułu dla pamięci buforowanej i niebuforowanej (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

Na rysunku 2.60 zamieszczamy zdjęcia kolejno modułów SIMM 30, SIMM 72, DIMM SDR i DIMM DDR. Proszę zwrócić uwagę między innymi na położenie kluczy (wycięć) na krawędzi złącza modułu.



Rysunek 2.60. Zdjęcia modułów pamięci

Rysunek 2.61 przedstawia moduł SODIMM (ang. *Smali Outline DIMM)* stosowany przykładowo w komputerach typu notebook.

Rysunek 2.61. Zdjęcie modułu SODIMM (dzięki uprzejmości firmy Crucial Technology, a division of Micron Technology)

Pamięci ROM

ROM (ang. *read only memory)* jest pamięcią nieulotną, przeznaczoną tylko do odczytu. Nieulotność oznacza, że po wyłączeniu napięcia zasilania tej pamięci informacja w niej przechowywana nie jest tracona (zapominana). Określenie, że jest to pamięć tylko do odczytu nie jest równoznaczne z tym, że zawartości tej pamięci w określonych warunkach nie można zmieniać. Dla niektórych typów technologicznych pamięci ROM jest to możliwe. Sytuacja taka jest opisana w dalszej części tego punktu. Określenie „tylko do odczytu" oznacza, że do pamięci tej nie możemy zapisywać danych w trakcie jej normalnej pracy w systemie.

Podział pamięci ROM, który przedstawiamy poniżej, oparty jest przede wszystkim na własnościach użytkowych tych pamięci, choć niewątpliwie ma to związek z zasadą ich działania i technologią wykonania. Niektóre z wymienionych typów pamięci ROM nie są już używane, a podajemy je, ponieważ były pewnym etapem w rozwoju tych pamięci.

Podstawowymi typami pamięci ROM są:

- MROM (ang. *mascable ROM)* - pamięci, których zawartość jest ustalana w procesie produkcji (przez wykonanie odpowiednich masek - stąd nazwa) i nie może być zmieniana. Przy założeniu realizacji długich serii produkcyjnych jest to najtańszy rodzaj pamięci ROM. W technice komputerowej dobrym przykładem zastosowania tego typu pamięci jest BIOS obsługujący klawiaturę.
- PROM (ang. *programmable ROM)* - pamięć jednokrotnie programowalna. Oznacza to, że użytkownik może sam wprowadzić zawartość tej pamięci, jednak potem nie można jej już zmieniać. Cecha ta wynika z faktu, że programowanie tej pamięci polega na nieodwracalnym niszczeniu niektórych połączeń wewnątrz niej. Obecnie ten typ pamięci nie jest już używany.
- EPROM - pamięć wielokrotnie programowalna, przy czym kasowanie poprzedniej zawartości tej pamięci odbywa się drogą naświetlania promieniami UV. Programowanie i kasowanie zawartości tej pamięci odbywa się poza systemem, w urządzeniach zwanych odpowiednio kasownikami i programatorami pamięci EPROM. Pamięć ta wychodzi już z użycia.
- EEPROM - pamięć kasowana i programowana na drodze czysto elektrycznej. Wykonanie tych operacji wymaga użycia zewnętrznego urządzenia. Istnieje możliwość wprowadzenia zawartości tego typu pamięci bez wymontowywania jej z systemu (jeżeli oczywiście jego projektant przewidział taką opcję), choć czas zapisu informacji jest nieporównywalnie dłuższy niż czas zapisu do pamięci RAM. W tego typu pamięci przechowywany jest tak zwany Flash BIOS, czyli oprogramowanie BIOS, które może być uaktualniane (przez wprowadzanie jego nowej wersji).

> Flash - rozwinięcie koncepcji pamięci EEPROM. Przez dołączenie odpowiednich układów możliwe jest kasowanie i ponowne programowanie tej pamięci bez jej wymontowywania z urządzenia. Istnieją dwie odmiany tej pamięci oznaczane jako NOR i NAND. Pierwsza cechuje się dłuższym czasem kasowania i zapisu, ma za to dostęp swobodny, nadaje się więc do przechowywania na przykład programów. Przykładowym zastosowaniem jest przechowywanie BIOS-u, który można unowocześniać (Flash BIOS). Pamięć typu NAND przypomina własnościami dysk twardy, ma dostęp częściowo sekwencyjny i dlatego jest predestynowana do przechowywania informacji typu multimedialnego. Przykładami zastosowań są karty pamięci do aparatów fotograficznych, pamięci wymienne (tak zwane pen drive) czy też odtwarzacze plików MP3, MP4.

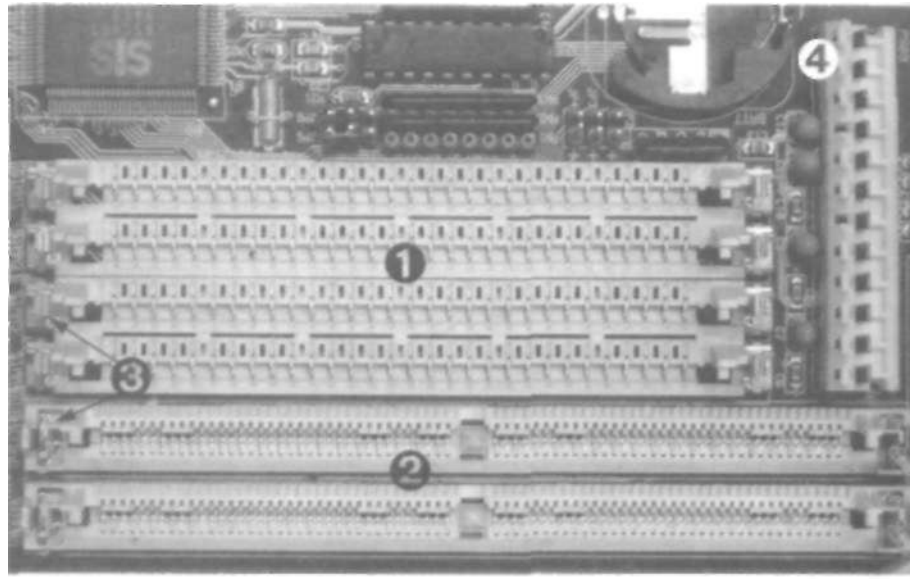
Pewną odmianą pamięci związaną z pamięciami ROM, choć nienależącą ściśle do tej grupy, jest pamięć NVRAM (ang. *non volatile RAM).* Stanowi połączenie pamięci SRAM z pamięcią EEPROM. Pamięć NVRAM może być odczytywana i zapisywana. Wprowadzona informacja może jednak zostać przepisana do pamięci typu EEPROM. Zapewnia to zachowanie zawartości tej pamięci po wyłączeniu napięcia zasilania. Czas zapisu do pamięci EEPROM jest oczywiście znacznie dłuższy niż czas dostępu do pamięci SRAM (rzędu kilkunastu ms). Nie jest to jednak istotne, gdyż przepisanie zawartości pamięci SRAM do EEPROM nie następuje po każdym zapisie do pamięci SRAM, a jedynie na żądanie, np. sygnałem STORE# (zachowaj). Przykładem zastosowania tych pamięci może być przechowywanie parametrów konfiguracji urządzeń wprowadzonych w trakcie danej sesji pracy z urządzeniem, które chcemy zachować w celu ich użycia w kolejnych sesjach.

Praktyka

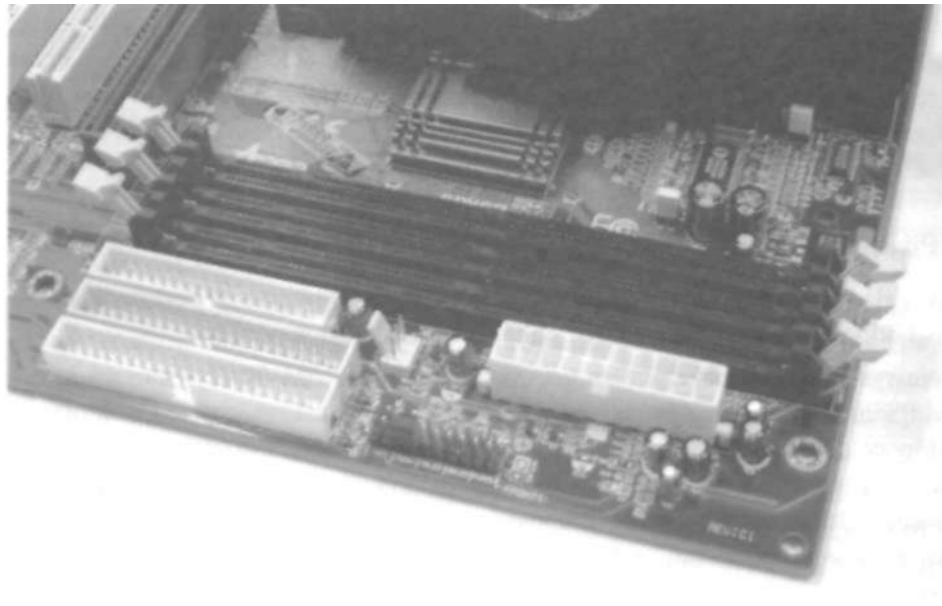
Wygląd poszczególnych modułów pamięci przedstawiono w podrozdziale 2.3.6. Wygląd odpowiadających im gniazd pokazano na zdjęciach poniżej.

Na rysunku 2.62 pokazane są gniazda modułów SIMM 30 (1) (starsze) i gniazda SIMM PS/2 (2) (nowsze). Z boku gniazd widoczne są zatrzaski mocujące (3). Montaż tych modułów polegał na wsunięciu modułu w wyżłobienie gniazda z kontaktami pod pewnym kątem, a następnie ustawienie ich w pozycji pionowej, w której następowało zapięcie zatrzasków. Demontaż wymagał odciągnięcia zatrzasków w bok i odchyleniu modułu od pozycji pionowej („położeniu" go). Obydwie operacje nie były zbyt wygodne.

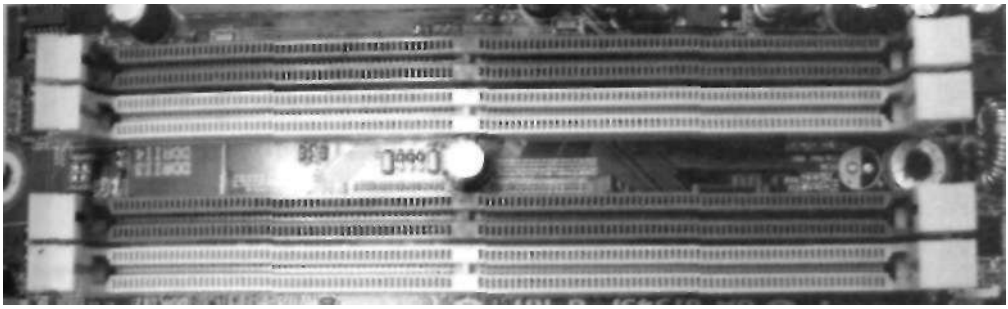
Na rysunku tym z boku z prawej strony widać też gniazdo zasilania płyty głównej AT (4).



Rysunek 2.62. Gniazda modułów SIMM 30 i SIMM PS/2



Rysunek 2.63. Gniazda modułów DIMM SDR



Rysunek 2.64. Gniazda modułów DIMM DDR

Na rysunkach 2.63 i 2.64 przedstawione są gniazda modułów DIMM, odpowiednio SDR i DDR. Zwracamy uwagę na niezbyt wielką różnicę w wyglądzie - dwa klucze (przerwy) w gniazdach modułów SDR (rys. 2.63) i jeden w gniazdach modułów DDR (rys. 2.64). Montaż tych modułów jest znacznie wygodniejszy i polega na pionowym wciśnięciu modułu w złącze aż do zapięcia się zaczepów. Moduł wyjmujemy, odginając zaczepy w bok, co powoduje wypchnięcie modułu ze złącza.

Na rysunku 2.64 widać przykładowe gniazda w standardzie Dual Channel. W celu wykorzystania tego trybu dwa moduły pamięci musimy zwykle zamontować w gniazdach o tym samym kolorze (na rysunku są to różne odcienie szarości). Dwa gniazda o dwóch kolorach tworzą parę dla jednego kanału - A lub B. Oczywiście sposób obsadzenia gniazd należy sprawdzić w dokumentacji płyty głównej.

L

3. Podstawy architektury komputera

Wstęp

Komputer jest zespołem układów cyfrowych tworzących **system mikroprocesorowy.** W jego skład wchodzą miedzy innymi układy przedstawione w poprzednich rozdziałach, na przykład pamięci, ale także układy jakościowo nowe, takie jak **mikroprocesor.** Pojęcie mikroprocesora zostanie dokładnie wyjaśnione w podrozdziale 3.3, gdzie oprócz jego budowy i działania zostanie także przedstawione jego współdziałanie z pozostałymi elementami systemu mikroprocesorowego. Podrozdział 3.1.2 przedstawia schemat blokowy systemu mikroprocesorowego, rolę poszczególnych bloków oraz ich współdziałanie. Podrozdziały 3.4 i 3.5 opisują komunikację systemu mikroprocesorowego z otoczeniem. Opisane są w nich kolejno **układy wejścia/wyjścia** i **operacje wejścia/wyjścia.**

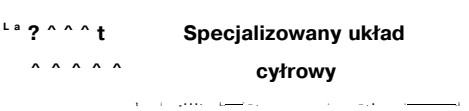
3.1. Pojęcie systemu mikroprocesorowego

3.1.1. System mikroprocesorowy a specjalizowany układ cyfrowy

Układy cyfrowe służą do przetwarzania informacji. Przetwarzanie informacji polega na dostarczeniu do układu bądź systemu **danych** poddawanych określonym działaniom, dzięki którym otrzymujemy **wyniki.** Wynikami mogą być przykładowo sygnały sterujące pracą pewnych urządzeń, obrazy, teksty i tym podobne. Tak określone przetwarzanie informacji dotyczy więc przykładowo zarówno układów automatyki, jak i komputerów.

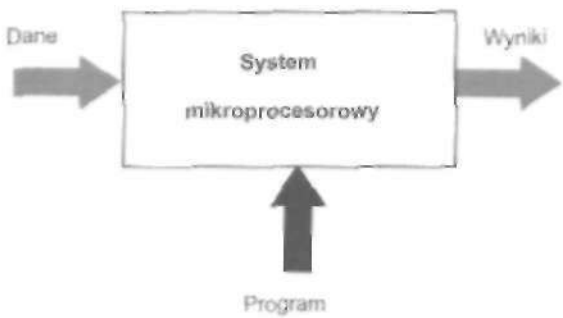
Przetwarzanie informacji przez układy cyfrowe możemy obecnie zrealizować dwoma sposobami:

1. Projektując tak zwany specjalizowany układ cyfrowy będący zestawem różnorodnych układów cyfrowych połączonych tak, aby realizowały określony sposób przetwarzania informacji. Sposób ten będzie zależał wyłącznie od użytych układów i sposobu ich połączenia, czyli od sprzętu (ang. *hardware).* Układ tego typu jest przedstawiony schematycznie na rysunku 3.1.



Rysunek 3.1. Przetwarzanie informacji za pomocą specjalizowanego układu cyfrowego

Stosując system mikroprocesorowy. Jedną z ważniejszych części tego systemu jest uniwersalny układ przetwarzający informację, czyli **procesor.** Procesor przetwarza informacje, wykonując na niej elementarne operacje zwane **instrukcjami maszynowymi** (bądź **rozkazami).** Ciąg takich instrukcji realizujący konkretne zadanie przetwarzania informacji nazywamy **programem.** Do systemu mikroprocesorowego oprócz danych wejściowych musimy więc dostarczyć także program lub zestaw programów, czyli **oprogramowanie** (ang. *software).* W przypadku systemu mikroprocesorowego sposób przetwarzania informacji jest określony głównie przez oprogramowanie. Ułatwia to w razie potrzeby zmianę sposobu przetwarzania informacji. Schematycznie przetwarzanie informacji za pomocą systemu mikroprocesorowego możemy przedstawić tak, jak na rysunku 3.2.

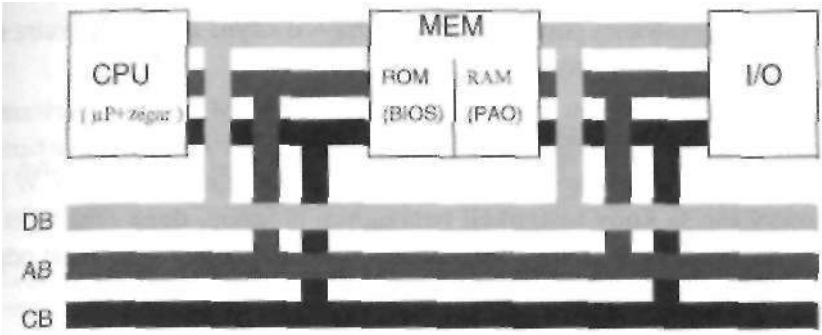


Rysunek 3.2. Przetwarzanie informacji za pomocą systemu mikroprocesorowego

Przy prezentacji sposobów przetwarzania informacji pomijamy takie możliwości jak sieci neuronowe i neurokomputery, które wciąż znajdują się w fazie eksperymentu.

3.1.2. Schemat blokowy systemu mikroprocesorowego

Jak wspomniano, jednym z elementów systemu mikroprocesorowego jest uniwersalny układ przetwarzający informacje. W naszym przypadku jest to mikroprocesor będący główną częścią CPU. Wykonuje wszelkie działania arytmetyczne i logiczne potrzebne do osiągnięcia zamierzonego wyniku. Układ ten musi jednak współpracować z dodatkowymi układami w celu utworzenia użytecznego, efektywnie pracującego systemu zwanego systemem mikroprocesorowym. Schemat blokowy takiego systemu przedstawiony jest na rysunku 3.3.



Oznaczenia bloków:
CPU - centralna jednostka przetwarzająca
RAM - pamięć do zapisu i odczytu
BIOS - podstawowy system obsługi we/wy
I/O- układy wejścia/wyjścia
AB - magistrala adresowa
MEM - pamięć
PAO - pamięć operacyjna
ROM - pamięć tylko do odczytu
DB - magistrala danych
CB - magistrala sterująca

Rysunek 3.3. Schemat blokowy systemu mikroprocesorowego

Zadaniem centralnej jednostki przetwarzającej - CPU (ang. *Central Processing Unit),* oprócz przetwarzania informacji jest sterowanie pracą pozostałych układów systemu. W skład CPU wchodzą mikroprocesor oraz układy pomocnicze, takie jak zegar czy sterownik magistral. Mikroprocesor jest układem przetwarzającym informację i kierującym pracą reszty układów. Zegar systemowy wytwarza przebiegi czasowe niezbędne do pracy mikroprocesora i systemu. Sterownik magistral jest układem, który pośredniczy w sterowaniu magistralami, wytwarzając na podstawie informacji otrzymanych z mikroprocesora (sygnałów statusowych i sterujących) sygnały sterujące pracą układów pamięci i układów wejścia/wyjścia.

Można stwierdzić, że wszystkie działania i operacje zachodzące w systemie są sterowane bądź inicjowane przez mikroprocesor. Rodzaj tych działań uzależniony jest od ciągu instrukcji dostarczanych do mikroprocesora stanowiących program. **Wynika z tego, że każde działanie wykonywane przez system (np. przez komputer) jest wynikiem realizacji określonego programu bądź jego fragmentu.** Stwierdzenie to jest bardzo istotne dla zrozumienia działania i zachowania się systemów mikroprocesorowych.

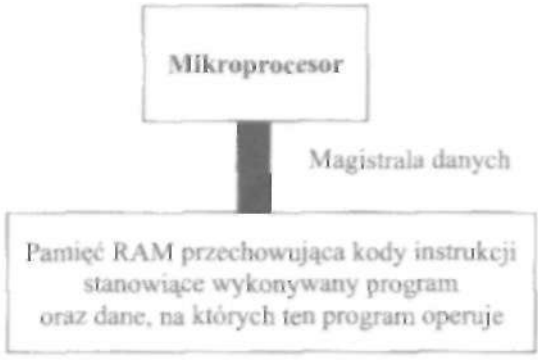
Program musi być przechowywany w miejscu, z którego mikroprocesor będzie mógł szybko, bez zbędnego oczekiwania, odczytywać kolejne instrukcje przeznaczone do wykonania. Miejscem tym jest pamięć półprzewodnikowa. Inne rodzaje pamięci, na przykład pamięci masowe, są zbyt wolne - ich czasy dostępu **w** porównaniu z szybkością pobierania kolejnych instrukcji przez mikroprocesor są za długie (przykładowe dane dotyczące zarówno procesorów, jak i pamięci zostaną przedstawione w dalszej części książki). Tak więc przed rozpoczęciem jego wykonania program jest ładowany z miejsca jego przechowywania (dysku twardego, płyty CD itp.) do pamięci

półprzewodnikowej (zwanej pamięcią operacyjną - o czym za chwilę. Patrz też rozdział 5. o systemach operacyjnych).

W bloku pamięci systemu stosowane są i pamięci RAM, i ROM. Pierwsze z nich, jak pamiętamy, przeznaczone są zarówno do odczytu, jak i zapisu, i są pamięciami ulotnymi. Tego typu układy pamięci tworzą **pamięć operacyjną** (PAO). W pamięci tej przechowywane są **kody instrukcji** tworzących program, **dane** oraz **wyniki** działania programu. Stosowane są dwa rozwiązania opisane poniżej, nazywane odpowiednio architekturą z Princeton i architekturą harwardzką.

3.1.2.1. Architektura z Princeton

Różnica pomiędzy tymi dwoma rozwiązaniami jest prosta. W przypadku architektury z Princeton zarówno dane, jak i programy są przechowywane w tym samym bloku pamięci, z którym procesor komunikuje się jedną i tą samą magistralą (co jest bardzo istotne - wyjaśnimy to za chwilę). Sytuację taką przedstawia rysunek 3.4.

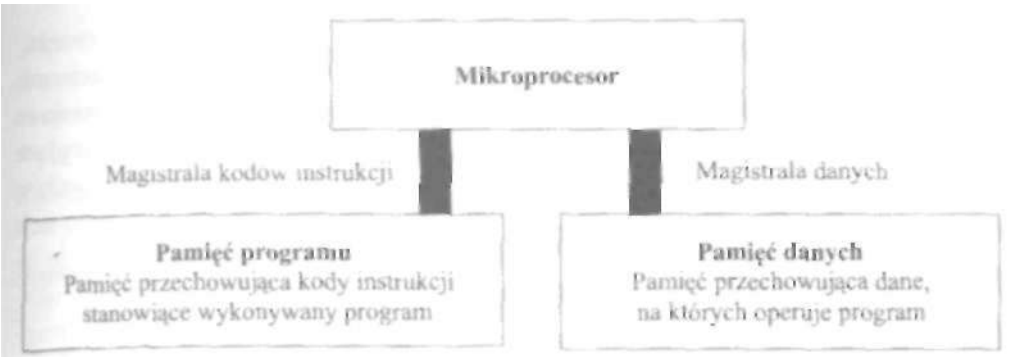


Rysunek 3.4. Architektura PAO z Princeton

3.1.2.2. Architektura harwardzka

W przypadku architektury harwardzkiej pamięć operacyjna jest tworzona z dwóch bloków pamięci zwanych pamięcią programu i pamięcią danych. Pierwszy z nich przechowuje wyłącznie wykonywany program (w przypadku niektórych systemów mikroprocesorowych takich jak układy automatyki może to być pamięć typu ROM, nie dotyczy to jednak sytuacji w komputerze!). W drugim zapisywane są dane (a więc także i wyniki działań), na których operuje program. Procesor może komunikować się z wymienionymi blokami pamięci osobnymi magistralami. Sytuacja taka jest przedstawiona na rysunku 3.5.

W przypadku rozwiązania z Harvardu możliwe jest wykonywanie jednoczesne (równoległe) dwóch operacji: odczytu kodu kolejnej instrukcji i zapisu bądź odczytu danej. Pozwala to przyspieszyć pracę systemu.



Rysunek 3.5. Harwardzka architektura PAO

W komputerze klasy IBM PC dla pamięci operacyjnej jest stosowane pierwsze rozwiązanie. Zarówno oprogramowanie, jak i dane przechowywane są w tej samej pamięci, a zadaniem systemu operacyjnego jest zapewnienie poprawności wykorzystania informacji i braku błędów (czyli przykładowo odczytania danej jako kodu instrukcji maszynowej lub odwrotnie).

Architektura harwardzka ma jednak także swoją implementację w komputerze PC. Od pewnego momentu (konkretnie od pojawienia się procesora Pentium - patrz rozdział 4.5) tak jest realizowana podstawowa pamięć cache procesorów. Powoduje to możliwość równoległego wykonywania operacji, a więc szybszą pracę procesora.

Ponieważ RAM jest pamięcią ulotną, w momencie włączenia systemu nie zawiera żadnej użytecznej informacji. Powiedzieliśmy też, że realizacja jakiejkolwiek operacji w systemie mikroprocesorowym jest wynikiem wykonania przez mikroprocesor pewnej liczby instrukcji stanowiących program (lub jego fragment). Aby więc system rozpoczął działanie, musi istnieć miejsce, gdzie przechowywany jest program inicjalizujący jego pracę. Miejsce to musi pamiętać program niezależnie od tego, czy napięcie zasilania jest włączone, czy nie. Takim miejscem jest pamięć ROM. W pamięci ROM przechowywany jest więc BIOS, czyli **podstawowy system obsługi wejścia/wyjścia** (ang. *Basic Input Output System).* Zawiera między innymi procedury inicjalizujące pracę systemu oraz umożliwiające wprowadzenie do pamięci operacyjnej dalszego oprogramowania. Prócz wymienionych zadań BIOS wykonuje zwykle jeszcze inne czynności, o których piszemy w podrozdziale 6.2.2.

Ostatnim niezbędnym blokiem systemu są **układy wejścia/wyjścia.** Pośredniczą w wymianie informacji pomiędzy mikroprocesorem i pamięcią systemu a urządzeniami zewnętrznymi w stosunku do systemu (na przykład takimi jak drukarka, monitor, stacja dysków), zwanymi **urządzeniami peryferyjnymi.** Potrzeba pośredniczenia w wymianie informacji może wynikać z konieczności translacji poziomów sygnałów elektrycznych, z potrzeby sterowania przepływem informacji w przypadku współpracy urządzeń o różnych szybkościach działania czy też z konieczności przygotowania odpowiedniego formatu informacji.

Wszystkie omówione tu bloki wymieniają informacje i współpracują ze sobą, używając wspólnych dróg przesyłania informacji zwanych magistralami. Pojecie magistrali zostało wprowadzone w rozdziale 2. W systemie występują trzy podstawowe rodzaje magistral: **magistrala danych, magistrala adresowa** i **magistrala** sterująca (patrz rysunek 3.3). Zadaniem magistrali danych jest przesyłanie danych, wyników oraz kodów instrukcji. Jest to magistrala dwukierunkowa, co oznacza, że informacja może zarówno wpływać do .mikroprocesora, jak i być przez niego przesyłana do innych układów. Magistralą adresową przesyłane są adresy komórek pamięci lub układów wejścia/wyjścia, z którymi chce się komunikować mikroprocesor. Jest to magistrala jednokierunkowa, adresy są generowane przez mikroprocesor, natomiast trafiają bądź do pamięci, bądź do układów wejs'cia/wyjs'cia. Trzecia magistrala nic jest w istocie magistralą, a raczej zestawem linii sterujących. Linie te służą do sterowania pracą układów współpracujących z mikroprocesorem oraz do sygnalizowania pewnych ich określonych stanów. Zwyczajowo jednak zestaw tych linii nazywa się magistralą sterującą.

3.2. Modułowa budowa komputera - pierwsze przybliżenie

Jak wynika z naszych rozważań, system mikroprocesorowy ma budowę blokową, modułową. Spróbujemy obecnie przyjrzeć się, jak budowa ta ma się do budowy komputera typu IBM/PC. Analizując tę budowę, zobaczymy, że elementy poszczególnych bloków systemu mikroprocesorowego umieszczone są na płycie głównej i montowane na niej bezpośrednio lub w stosownych gniazdach (sposób umieszczania określonych elementów komputera zmieniał się i zmienia się nadal). Przyjrzyjmy się krótko tym elementom.

Pierwszym modułem systemu mikroprocesorowego jest CPU, której zadaniami są: przetwarzanie informacji i sterowanie pozostałymi elementami systemu. Modułowi temu odpowiada przede wszystkim układ mikroprocesora. Przetwarza on informację, wykonując wszelkie działania arytmetyczne i logiczne. Mikroprocesor montowany jest obecnie na płycie głównej w odpowiednim gnieździe umożliwiającym jego łatwą wymianę (patrz rozdział 4.). Nie steruje jednak bezpośrednio pozostałymi elementami systemu, lecz korzysta z pomocy skomplikowanych, specjalizowanych układów, takich jak przykładowo sterownik pamięci DRAM czy sterownik magistrali PCI, tworzących układy sterujące (w żargonie zwane logiką sterującą) płyty głównej komputera. Układy te zawarte są obecnie jako część zestawu układów bardzo wielkiej skali integracji (VLSI) zwanych chipsetami (zasadniczo prawidłowa forma to liczba pojedyncza - „zwanych chipsetem", jako że termin chipset w języku angielskim oznacza właśnie zestaw układów scalonych, zestaw chipów. Obawiam się jednak, że liczba mnoga dla tego terminu tak się zadomowiła, że należy się z nią pogodzić).

Chipsety montowane są (jak na razie) bezpośrednio na płycie głównej. Możliwości chipsetów decydują w znacznym stopniu o własnościach płyty głównej, a więc w znacznej mierze i całego komputera.

Drugim modułem systemu mikroprocesorowego jest pamięć półprzewodnikowa. W komputerze IBM/PC możemy mówić o trzech jej elementach.

Pierwszy to pamięć główna komputera, zbudowana z pamięci DRAM, umieszczona obecnie na różnego rodzaju modułach pamięci i montowana w złączach opisanych w rozdziale 2. w części „Praktyka".

Drugi element to pamięć ROM przechowująca BIOS, stanowiąca osobny układ scalony umieszczony bezpośrednio na płycie głównej (czasami w podstawce umożliwiającej względnie łatwe jej wymontowanie). Obecnie jest to z reguły pamięć typu flash, której zawartość może być wymieniana bez wymontowywania układu z miejsca jego zamontowania. Umożliwia to aktualizację (unowocześnienie, ang. *upgrade)* posiadanego BIOS-u.

Trzeci element to pamięć cache (SRAM) będąca obecnie głównie elementem mikroprocesora. O pamięci cache piszemy w rozdziale 3.7.

Ostatni moduł systemu mikroprocesorowego to układy wejścia wyjścia. Tu sytuacja jest o tyle bardziej skomplikowana, że część z tych układów może znajdować się w chipsetach, a inne mogą być montowane bezpośrednio na płycie głównej bądź wreszcie na tak zwanych kartach rozszerzających, przy czym sytuacja ta zmienia się dość dynamicznie. Przykładami są kontrolery dźwięku czy też adaptery graficzne lub kontrolery dysków twardych. Mogą być umieszczone w chipsetach bądź jako osobny układ, lecz montowany bezpośrednio na płycie głównej lub mogą być umieszczone na tak zwanej karcie rozszerzeń (czyli na odpowiednio zaprojektowanej i wykonanej płytce drukowanej z układami elektronicznymi) montowanej w gnieździe magistrali rozszerzającej. Układy wejścia/wyjścia mogą też być układami wspólnymi dla wielu urządzeń, jak przykładowo kontrolery przerwań, kontrolery DMA czy układy sterujące określonymi magistralami (na przykład USB) i dedykowane określonym urządzeniom (na przykład interfejs Serial ATA - dyskom twardym).

3.3. Podstawy działania mikroprocesora

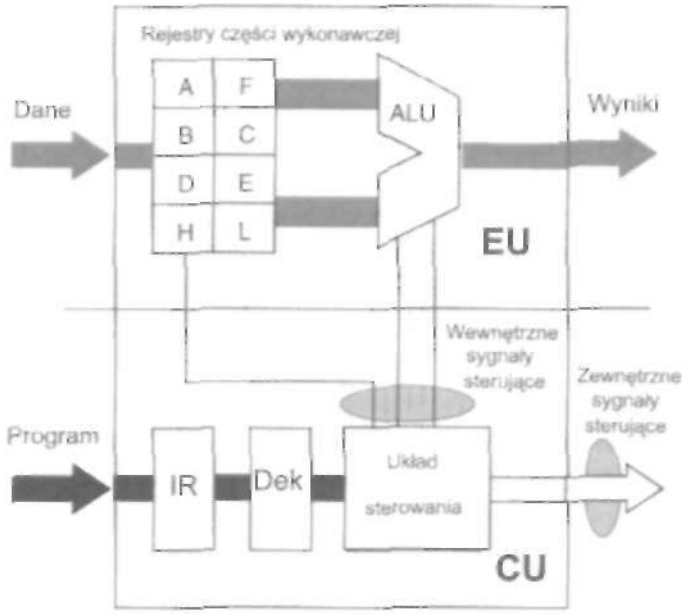
Podając określenie systemu mikroprocesorowego, stwierdzono, że jednym z jego elementów jest uniwersalny układ przetwarzający informację i sterujący pracą pozostałych elementów systemu, zwany procesorem, a w przypadku jego realizacji jako pojedynczego układu scalonego dużej skali integracji - **mikroprocesorem.** Mikroprocesor wraz z układami towarzyszącymi, takimi jak zegar systemowy i sterownik magistral, tworzy centralną jednostkę przetwarzającą, czyli CPU.

3.3.1. Schemat blokowy mikroprocesora

Schemat blokowy mikroprocesora przedstawiony jest na rysunku 3.6. Podział układów mikroprocesora na jednostkę wykonawczą i jednostkę sterującą wynika logicznie z zadań, jakie pełni mikroprocesor.

Zadaniem jednostki wykonawczej EU (ang. *execution unit)* jest przetwarzanie informacji, czyli wykonywanie wszelkich operacji arytmetycznych i logicznych. Rodzaj wykonywanych operacji zależy od wewnętrznych sygnałów sterujących wytwarzanych przez jednostkę sterującą CU. W skład jednostki wykonawczej wchodzi jednostka arytmetyczno-logiczna ALU oraz zestaw współpracujących z nią rejestrów, Informacją wejściową części wykonawczej są dane, wyjściową zaś wyniki (liczby, informacja tekstowa, sygnały sterujące pracą określonych urządzeń itp.).

W skład jednostki sterującej wchodzą: rejestr rozkazów IR, dekoder rozkazów i układ sterowania. W rejestrze rozkazów przechowywany jest kod aktualnie wykonywanego rozkazu. Kody rozkazów pobierane są do rejestru rozkazów z pamięci. Ciąg rozkazów tworzy program wykonywany przez system.



Oznaczenia:
ALU - jednostka arytmetyczno-logiczna
EU - jednostka wykonawcza
IR - rejestr rozkazów
Dek - dekoder rozkazów
CU - jednostka sterująca

Rysunek 3.6. Schemat blokowy mikroprocesora

Po pobraniu z pamięci kod rozkazu jest dekodowany w dekoderze rozkazów, czyli jest określane, jakiego rozkazu kod znajduje się w dekoderze rozkazów. Na tej

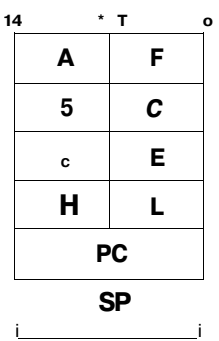
podstawie układ sterowania wytwarza wewnętrzne i/lub zewnętrzne sygnały sterujące realizujące dany rozkaz. Z punktu widzenia użytkownika mikroprocesora sposób wytwarzania sygnałów sterujących jest nieistotny, dlatego układ sterowania traktujemy jako czarną skrzynkę. Musimy jedynie założyć, że został zaprojektowany i wykonany poprawnie, co zapewnia właściwą realizację rozkazów.

3.3:2. Rejestry procesora dostępne programowo

Zgodnie ze schematem blokowym mikroprocesora, zarówno jednostka arytmetyczno-logiczna, jak i układ sterowania współpracują z określonym zestawem rejestrów. Zawartość pewnej części rejestrów z tego zestawu może być zmieniana w wyniku wykonania przez procesor określonej instrukcji. Rejestry takie nazywamy **rejestrami dostępnymi programowo.** Pozostałe rejestry są niedostępne dla użytkownika i i c h zestaw nie jest zwykle znany. Nie jest to potrzebne, podobnie jak nie musimy znać konstrukcji układu sterowania.

W rejestrach dostępnych programowo występują takie typy rejestrów, których odpowiedniki znajdują się praktycznie w każdym procesorze. Ich pojemność czy liczba mogą się zmieniać, jednak wykonywane zadania pozostają takie same. Na r y s u n k u 3.7 przedstawiamy przykładowy zestaw rejestrów oparty na prostym procesorze 8-bitowym Intel 8080. W rozdziale poświęconym procesorom stosowanym w komputerach IBM PC okaże się jednak, że każdy z tych rejestrów ma swój odpowiednik w tych procesorach.

Zadania poszczególnych rejestrów opisane są w kolejnych podpunktach.



Oznaczenia:
A - akumulator F - rejestr znaczników (rejestr flagowy)
B,C,D,E,H,L - rejestry robocze (uniwersalne) PC - licznik rozkazów
SP - wskaźnik stosu

Rysunek 3.7. Rejestry procesora 18080 dostępne programowo

3.3.2.1. Akumulator

Akumulatorem nazywamy rejestr, który zawiera jeden z operandów (argumentów) wykonywanej operacji i do którego jest ładowany wynik wykonywanej operacji. Definicja akumulatora pojawiła się po raz pierwszy przy omawianiu jednostki arytmetyczno-logicznej. W starszych procesorach definicja ta obowiązywała ściśle, w nowszych notuje się liczne odstępstwa mające na celu przyspieszenie realizacji programu. Przykładowo w procesorach rodziny 80x86 wynik dodawania może być umieszczany także w rejestrze innym niż akumulator.

3.3.2.2. Rejestr flagowy

Rejestrem flagowym nazywamy rejestr zawierający dodatkowe cechy wyniku wykonywanej operacji potrzebne do podjęcia decyzji o dalszym sposobie przetwarzania informacji. Cechami tymi mogą być przykładowo znak wyniku, wystąpienie przekroczenia zakresu czy parzystość (na przykład parzysta bądź nieparzysta liczba jedynek w wyniku). Wystąpienie określonego przypadku, na przykład wyniku dodatniego bądź ujemnego, sygnalizowane jest ustawieniem lub wyzerowaniem określonego bitu w rejestrze flagowym. Ustawiane bity nazywane są znacznikami (stąd druga nazwa tego rejestru to rejestr znaczników) lub flagami. Flagi mogą być używane przy tworzeniu rozgałęzień w programie, na przykład jako wystąpienie pewnego warunku w skokach warunkowych (więcej na temat tego typu instrukcji w kolejnych punktach rozdziału). Lista najczęściej używanych flag, ich krótka definicja oraz (tam, gdzie jest to istotne) rodzaj operacji, dla których mają znaczenie, są następujące:

1. CY lub CF (ang. *curry flag)* - flaga przeniesienia lub pożyczki. Ustawiana (czyli przyjmuje wartość 1), gdy rezultat operacji przekracza zakres długości słowa, w którym zapisywany jest wynik. Istotna dla operacji arytmetycznych na liczbach bez znaku.
2. Z lub ZF (ang. *zeru flag)* - flaga sygnalizująca, że wynikiem ostatnio wykonywanej operacji jest 0.
3. S lub SF (ang. *sign flag)* - flaga znaku. Ustawiana, gdy najstarszy bit wyniku wykonywanej operacji jest równy 1. Zasadniczo flaga ta ma znaczenie dla operacji arytmetycznych interpretowanych jako liczby ze znakiem (w kodzie U2).
4. P lub PF (ang. *parityflag)* - flaga parzystości. Sygnalizuje parzystą lub nieparzystą liczbę jedynek w wyniku (np. w jego najmłodszym bajcie). Dokładna reguła ustawiania tej flagi zależy od typu procesora.
5. OV lub OF *(ang. werflow flag)* - flaga przepełnienia. Sygnalizuje przekroczenie zakresu dla operacji arytmetycznych, gdy są interpretowane jako operacje na liczbach ze znakiem zapisanych w kodzie U2. Dokładna reguła ustawiania tej flagi podana jest w podrozdziale 2.2.1.2.

6. AC lub AF (ang. *auxiliary carry flag)* - flaga przeniesienia pomocniczego lub połówkowego. Ustawiana, gdy występuje przeniesienie lub pożyczka z najstarszego bitu pierwszej tetrady wyniku. Istotna przy interpretowaniu wyniku operacji arytmetycznych w kodzie prostym BCD.

3.3.2.3. Licznik rozkazów

Licznik rozkazów jest jednym z istotniejszych rejestrów umożliwiających zrozumienie działania mikroprocesora i systemu mikroprocesorowego. W nowszych mikroprocesorach nosi on nazwę **IP - wskaźnika instrukcji** (ang. *Instruction Pointer),* co jest chyba trafniejsze. Definicja tego rejestru jest następująca:

[Definicja

Licznikiem rozkazów (wskaźnikiem instrukcji) nazywamy rejestr mikroprocesora zawierający adres komórki pamięci, w której przechowywany jest kod rozkazu przeznaczonego do wykonania jako następny.

Z definicji tej wynika, że po wczytaniu kolejnego kodu rozkazu zawartość licznika rozkazów powinna zostać zmieniona tak, aby wskazywał kolejny rozkaz przeznaczony do wczytania do mikroprocesora. Ponieważ w przeważającej części program jest wykonywany kolejno, instrukcja po instrukcji (chociaż są od tego odstępstwa - omawiamy je w podrozdziale poświęconym liście rozkazów), to pobranie kodu rozkazu z kolejnej komórki pamięci powoduje zwiększenie zawartości licznika rozkazów o 1. W przypadku kodów rozkazów przechowywanych w kilku komórkach, np. trzech, licznik rozkazów zwiększany jest o 1 trzykrotnie, aby po pobraniu kodu całego rozkazu wskazywał na początek kodu kolejnego rozkazu.

W pewnych przypadkach zawartość licznika rozkazów jest zmieniana w wyniku wykonania określonego rozkazu (dlatego jest rejestrem dostępnym programowo). 0 tego typu rozkazach będzie mowa przy omawianiu listy rozkazów.

3.3.2.4. Wskaźnik stosu

Przed podaniem definicji wskaźnika stosu musimy najpierw wyjaśnić pojęcie **stosu.**

Definicja

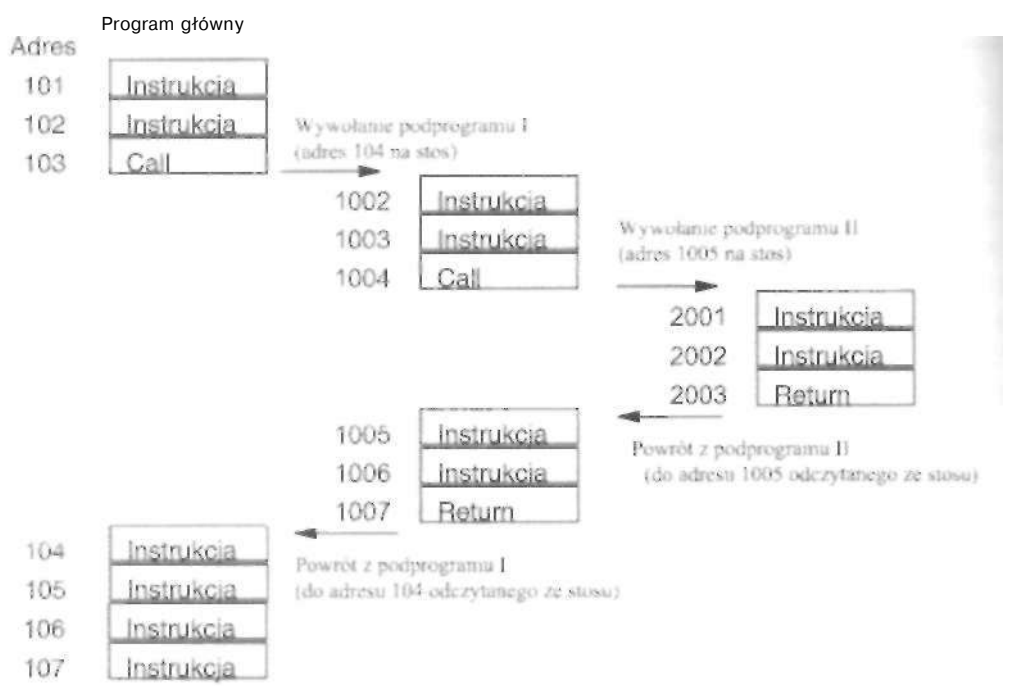
Stosem nazywamy wyróżniony obszar pamięci używany według następujących reguł:
1. Informacje zapisywane są na stos do kolejnych komórek (pod kolejnymi adresami), przy czym żadnego adresu nie wolno pominąć.
2. Odczytujemy informacje w kolejności odwrotnej do ich zapisu.
3. Informacje odczytujemy z ostatnio zapełnionej komórki, natomiast zapisujemy do pierwszej wolnej, przy czym komórkę odczytaną traktujemy jako wolną.

Stos jest więc rodzajem pamięci (czy też buforem) oznaczanej przez LIFO (ang. *last infirst out* - ostatni wchodzi, pierwszy wychodzi). Pamięć taką można porównać do stosu talerzy (stąd zresztą nazwa). Talerze dokładamy do stosu, kładąc je na wierzchu, a zabieramy, zdejmując je także z wierzchu, gdyż w przypadku zmiany kolejności grozi nam katastrofa, co odnosi się także do stosu komputerowego.

W przypadku pojęcia stosu w pamięci, zgodnie z regułą 3, konieczna jest znajomość adresu ostatniej zapełnionej komórki stosu, przy czym komórkę odczytaną uważamy za pustą (zdjęlis'my talerz). Komórka ta zwana jest wierzchołkiem stosu.

Definicja

Wskaźnikiem stosu nazywamy rejestr zawierający adres ostatniej zapełnionej komórki stosu (wierzchołka stosu).



Rysunek 3.8. Wykorzystanie stosu w obsłudze podprogramów

W rzeczywistości stos może się tworzyć w kierunku rosnących adresów, czyli „do góry", lub w kierunku adresów malejących (i wówczas wierzchołek stosu jest raczej dnem). Kierunek budowy stosu nie jest istotny i zależy od typu procesora. Z definicji wskaźnika stosu wynika, że przy założeniu narastania stosu w kierunku adresów rosnących każdy zapis na stos zwiększa zawartość wskaźnika stosu, a każdy

odczyt zmniejsza jego zawartość. Dzięki temu wskaźnik stosu pokazuje zawsze ostatnią zapełnioną komórkę stosu.

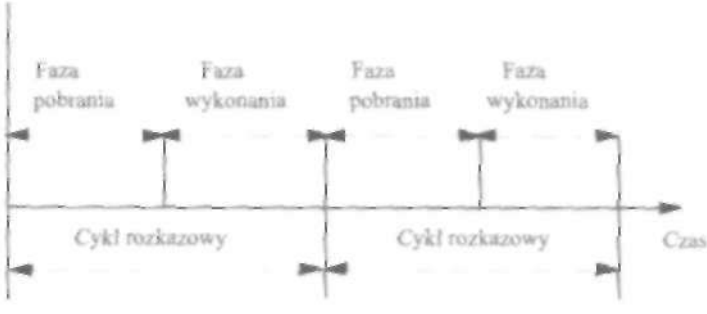
Jednym z klasycznych zastosowań stosu jest zapamiętanie adresu powrotu do programu wywołującego w przypadku wywołania tak zwanego podprogramu. Ponieważ podprogram może z kolei wywoływać inny podprogram, adresy powrotów odkładane są na stos, gdyż muszą być odczytywane w kolejności odwrotnej do kolejności ich zapisu. Ilustruje to rysunek 3.8. Prosimy zwrócić uwagę na kolejność zapisywania adresów powrotów, a następnie na kolejność ich odczytywania.

33.2.5. Rejestry robocze (uniwersalne)

Oprócz wymienionych rejestrów wykonujących ściśle określone zadania każdy mikroprocesor dysponuje pewnym zestawem rejestrów zwanych rejestrami roboczymi lub rejestrami ogólnego przeznaczenia (ang. *General Purpo.se Registers ~ GPR)*. Przykładowo rejestry takie mogą przechowywać argumenty wykonywanych operacji, wyniki czy też adresy tychże w pamięci. Rejestry te mogą pełnić pewne dodatkowe funkcje przewidziane przez projektanta mikroprocesora, na przykład liczników wykonywanych pętli. Przykładem rejestrów roboczych w mikroprocesorze 8080 są: B, C, D, E, H, L. Każdy z nich może zawierać dane do wykonywanych operacji. Ponadto para rejestrów H, L może zawierać adres komórki pamięci w przypadku używania tak zwanego trybu adresowania rejestrowego (który zostanie wyjaśniony w punkcie 3.3.4.2).

3.3.3. Cykl rozkazowy

Realizując program, system mikroprocesorowy wykonuje pewne powtarzające się czynności, polegające na cyklicznym pobieraniu kodów rozkazów z pamięci i wczytywaniu ich do układu sterowania mikroprocesora, a następnie realizacji rozkazu, którego kod został pobrany. W cyklu możemy wyróżnić dwa etapy zwane **fazą pobrania** (ang. *fetch)* i **fazą wykonania** (ang. *execution)*. Schematycznie następstwo kolejnych faz przedstawia rysunek 3.9.



Rysunek 3.9. Fazy cyklu rozkazowego

Faza pobrania polega na pobraniu kodu rozkazu z komórki pamięci o adresie przechowywanym w liczniku rozkazów, a następnie na umieszczeniu tego kodu w rejestrze rozkazów IR znajdującym się w układzie sterowania mikroprocesora. Kod rozkazu przesyłany jest do mikroprocesora magistralą danych. Następnie zawartość licznika rozkazów jest modyfikowana tak, aby wskazywał on na kolejny kod rozkazu przeznaczony do pobrania. Po zakończeniu fazy pobrania następuje faza wykonania, Polega na zdekodowaniu kodu rozkazu znajdującego się w rejestrze IR, czyli stwierdzeniu, jaki to rozkaz. Po zdekodowaniu kodu rozkazu układ sterowania wytwarza i wewnętrzne i/lub zewnętrzne sygnały sterujące realizujące dany rozkaz (patrz schemat j blokowy mikroprocesora i systemu mikroprocesorowego).

Kolejne etapy realizacji fazy pobrania i fazy wykonania rozkazu można przedstawić następująco:

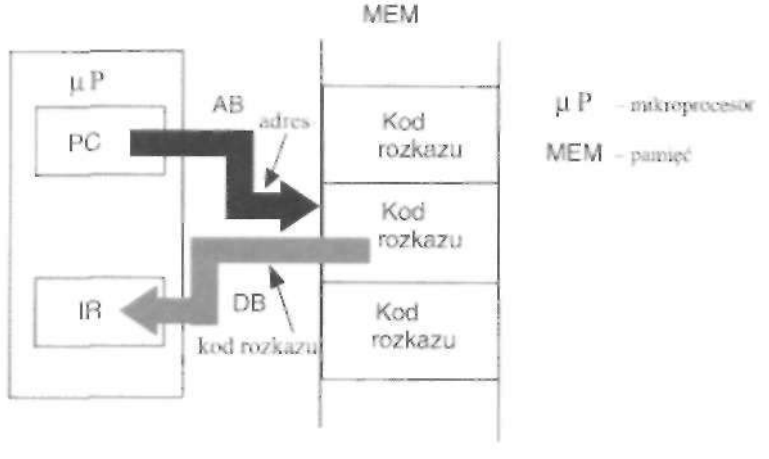
I) Faza pobrania

1. Adresowanie - podanie zawartości licznika rozkazów do magistrali adresowej: AB <-- (PC).
2. Wczytanie zawartości zaadresowanej komórki pamięci do rejestru rozkazów mikroprocesora: IR <— M(PC).
3. Zwiększenie zawartości licznika rozkazów: (PC) <— (PC) + 1.

II) Faza wykonania

4. Zdekodowanie kodu rozkazu i wytworzenie sygnałów sterujących realizujących dany rozkaz.

Obieg informacji w fazie pobrania przedstawiony jest na rysunku 3.10.

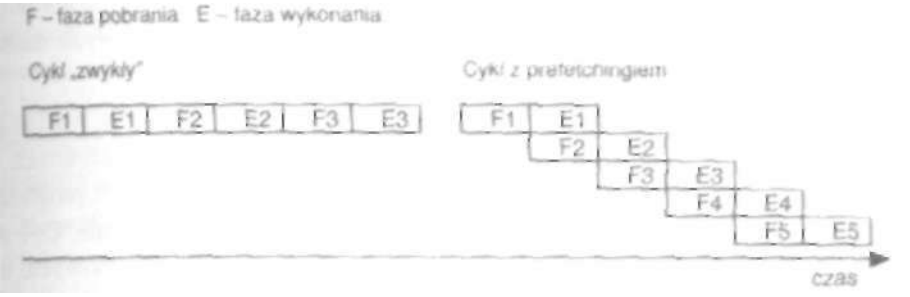


Rysunek 3.10. Przepływ informacji przy pobieraniu kodu rozkazu

Oczywiście w przypadku kodów rozkazów zajmujących kilka komórek pamięci etapy 1+3 muszą być kilkakrotnie powtórzone, zanim przejdziemy do ostatecznego wykonania rozkazu. Realizacja wszystkich wymienionych etapów wykonania rozkazu tworzy pewien cykl, zwany od nazwiska jego twórcy cyklem von Neumana. Przedstawiony jest schematycznie na rysunku 3.11. Linia przerywana oznacza wykonanie kolejnych cykli pobrania dla kodów rozkazów wielobajtowych (wielokomórkowych).

Rysunek 3.11. Cykl rozkazowy

Jak pokazano na rysunku 3.9, w cyklu rozkazowym następują po sobie na przemian faza pobrania i faza wykonania. W celu przyspieszenia pracy systemu stosuje się modyfikację tego cyklu zwaną **prefetchingiem,** czyli wstępnym pobieraniem instrukcji. Polega na równoległym wykonywaniu fazy pobrania następnego rozkazu, jeszcze w trakcie realizacji fazy wykonania rozkazu poprzedniego. Obydwa sposoby realizacji cyklu rozkazowego pokazane są na rysunku 3.12.



Rysunek 3.12. Idea prefetchingu

Jak widzimy, podczas wstępnego pobierania instrukcji zakończone zostało wykonywanie większej liczby instrukcji.

Realizacja prefetchingu wymaga oczywiście określonej budowy procesora, umożliwiającej równoległą pracę jednostek CU i EU. Ponadto wstępne pobieranie instrukcji jest ograniczane dostępnością magistral. Jeżeli faza wykonania poprzedniego rozkazu wymaga ich użycia (na przykład zapisania wyniku do pamięci), to równoległe pobranie następnej instrukcji nie jest możliwe (gdyż tymi samymi magistralami należałoby wczytać równolegle kod rozkazu).

Rozwinięciem idei prefetchingu jest praca potokowa opisana w rozdziale 6.

3.3.4. Lista rozkazów, tryby adresowania

Komputer, będący systemem mikroprocesorowym, przetwarza informację zgodnie z wykonywanym programem. Program jest ciągiem instrukcji realizujących określony algorytm działania systemu. W pamięci systemu mikroprocesorowego program przechowywany jest w postaci binarnych kodów instrukcji maszynowych (rozkazowi właściwych dla danego mikroprocesora. Poniżej podajemy definicje zarówno rozkazu, jak i listy rozkazów. Następnie omawiamy podział listy rozkazów, pojęcie formatu rozkazu, podstawowe tryby adresowania oraz sposób prezentacji rozkazu. Na koniec zamieszczamy przykłady kilku rozkazów ilustrujących poszczególne grupy rozkazów.

3.3.4.1. Lista rozkazów

Definicja

Rozkazem (instrukcją maszynową) nazywamy najprostszą operację, której wykonania programista może zażądać od procesora.

Sposób realizacji rozkazu nie jest istotny dla użytkownika systemu i z reguły nie jest znany. Nawet jeżeli wiemy, że dany rozkaz jest realizowany jako ciąg prostszych operacji, zwanych mikrooperacjami czy mikrorozkazami, i tak nie mamy żadnego wpływu na sposób realizacji rozkazu. Został on po prostu wyznaczony przez projektanta mikroprocesora. Rozkazy są więc najprostszymi operacjami, których możemy używać podczas tworzenia programów.

Tworzenie programów bezpośrednio za pomocą rozkazów jest nazywane programowaniem w asemblerze. To sposób pisania programów bardzo efektywnych, jest jednak dość żmudny i nie nadaje się do tworzenia bardzo rozbudowanych programów. W takich wypadkach używamy języków wysokiego poziomu (np. Pascala, C++ itp.). W językach tych jednej instrukcji odpowiada wiele instrukcji maszynowych, czyli rozkazów. Tłumaczeniem instrukcji języków wyższych poziomów na instrukcje maszynowe zajmują się specjalne programy zwane kompilatorami.

Zestaw instrukcji, ich liczba i rodzaj zależą od konkretnego procesora.

Definicja

Listą rozkazów nazywamy zestaw wszystkich instrukcji maszynowych (rozkazów), jakie potrafi wykonać dany procesor.

Rozkazy tworzące listę rozkazów możemy podzielić na kilka podstawowych grup w zależności od ich przeznaczenia.

Rozróżniamy:

1. rozkazy przesłań,
2. rozkazy arytmetyczne i logiczne,
3. rozkazy sterujące (skoki, wywołania podprogramów, pętle itp.),
4. inne (np. sterowanie pracą koprocesora, rozkazy testujące, operacje w trybie 'chronionym).

Rozkazy przesłań są najczęściej wykonywanymi rozkazami. Nie zmieniają wartości informacji, natomiast przenoszą ją z miejsca na miejsce. Ich duża częstotliwość wykonywania jest dość oczywista. Jeśli chcemy wykonać jakąś operację, musimy zwykle pobrać jej argumenty, a po jej wykonaniu zapisać wynik. Wśród rozkazów przesłań wyróżnia się czasami operacje na stosie (będące jednak też formą przesłań) czy instrukcje wejścia/wyjścia, przesyłające lub odczytujące dane z portów wejścia/wyjścia.

Rozkazy arytmetyczne i logiczne służą do przetwarzania informacji, czyli w wyniku ich wykonania jest ona zmieniana. Do rozkazów tych prócz wykonujących podstawowe działania arytmetyczne czy logiczne należą na przykład rozkazy cyklicznego przesuwania informacji, rozkazy porównań itp.

Rozkazy sterujące stanowią grupę rozkazów pozwalającą zmieniać kolejność wykonywania instrukcji programu. Należą do nich przykładowo rozkazy skoków bezwarunkowych i warunkowych (warunkiem w takim skoku jest wartość określonej flagi, inaczej znacznika), bezwarunkowe i warunkowe wywołania podprogramów czy też instrukcje pętli (czyli instrukcje powodujące kilkakrotne powtórzenie pewnego ciągu instrukcji).

Pozostałe instrukcje są zwykle charakterystyczne dla danego typu procesora. Przykładowo dla rodziny 80x86 mogą to być rozkazy sterujące współpracą z koprocesorem czy też instrukcje tak zwanego trybu chronionego.

3.3.4.2. Format rozkazu i tryby adresowania

Rozkazy, jak każdy inny rodzaj informacji w systemie mikroprocesorowym, są przechowywane w postaci kodów binarnych. Kod rozkazu musi zawierać informacje niezbędne do jego poprawnej realizacji. Informacje te muszą być rozmieszczone w rozkazie w określony sposób.

[Definicja j

Formatem rozkazu nazywamy sposób rozmieszczenia informacji w kodzie rozkazu.

Kod rozkazu:

1. musi zawierać określenie rodzaju wykonywanej operacji, czyli tak z w a n y k o d I operacji. Kod operacji musi być określony w początkowej częs'ci (p i e r w s z y m I bajcie lub bajtach) kodu rozkazu w celu określenia, w jaki sposób ma p r z e b i e g a l i I dalsza realizacja rozkazu przez mikroprocesor;
2. może zawierać operandy i/lub adresy operandów wykonywanych operacji (dotyczy to także adresów wyników). Oczywiście w przypadku rozkazów wymagających argumentów informacja ta musi być zawarta w rozkazie.

Z punktem 2 określającym zawartość kodu rozkazu związana jest kolejna d e f i n i c j a , j

I Definicja

Trybem adresowania nazywamy sposób określenia miejsca przechowywania a r - gumentów rozkazu.

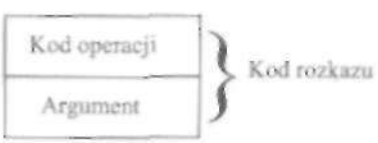
Argumenty rozkazu (przypominamy, że może to dotyczyć zarówno danych, j a k i w pewnych przypadkach wyników) m o g ą być przechowywane w rejestrach, w p a m i ę - ci lub w kodzie rozkazu. Poniżej podajemy definicje oraz interpretację podstawowych trybów adresowania.

1. Adresowanie natychmiastowe

Definicja

Przy **adresowaniu natychmiastowym** argument rozkazu zawarty jest w k o d z i e rozkazu.

Widzimy więc, że adresowanie natychmiastowe w zasadzie nie jest adresowaniern w zwykłym sensie. Argument jest umieszczony w kodzie rozkazu, z czego między innymi wynika fakt, że musi być znany w momencie pisania programu. Schematycznie ten tryb adresowania przedstawia rysunek 3.13.



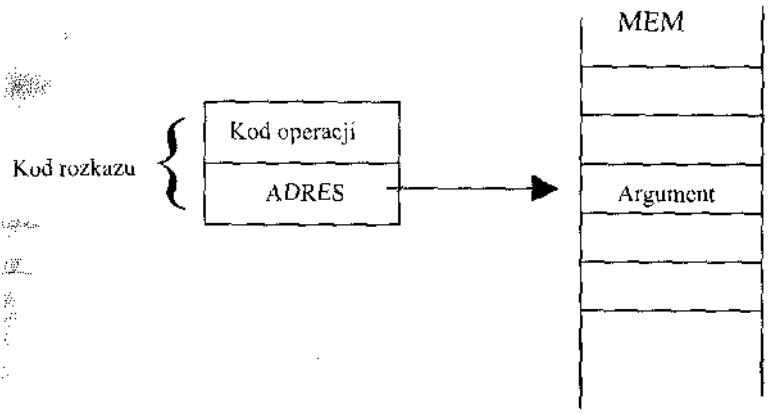
Rysunek 3.13. Adresowanie natychmiastowe

2. Adresowanie bezpośrednie

[Definicja

Przy **adresowaniu bezpośrednim** kod rozkazu zawiera adres komórki pamięci, w której przechowywany jest rozkaz.

Konsekwencją takiego określenia adresowania bezpośredniego jest to, że jeśli używamy tego adresowania, w momencie pisania programu musimy znać (lub inaczej - zarezerwować) adres przechowywania argumentu. Interpretacja adresowania bezpośredniego przedstawiona jest na rysunku 3.14.



Rysunek 3.14. Adresowanie bezpośrednie

3. Adresowanie rejestrowe

[Definicja ~~1

Przy **adresowaniu rejestrowym** w kodzie rozkazu określony jest rejestr, w którym przechowywany jest argument.

Zaletami użycia tego trybu adresowania są krótkie kody rozkazów oraz szybkie ich wykonywanie. Schematycznie tryb ten jest przedstawiony na rysunku 3.15.

Rysunek 3.15. Adresowanie rejestrowe

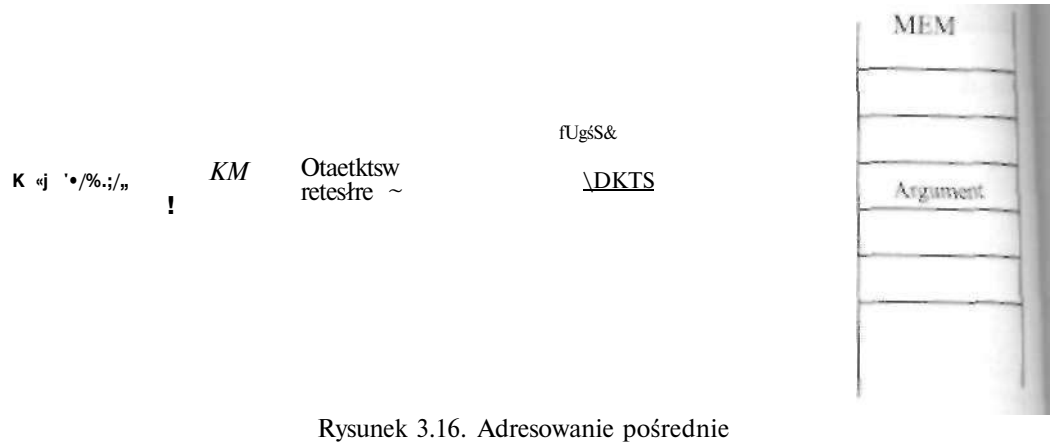
4. Adresowanie pośrednie

Adresowanie pośrednie, zwane też **adresowaniem rejestrowym** pośrednim, umożliwia modyfikację położenia argumentu w pamięci w trakcie wykonywania programu. Inaczej mówiąc, adres przechowywania tego argumentu może zostać wyliczony przez program, co jest bardzo użyteczną własnością. Ponadto kody takich rozkazów są krótkie.

Definicja

W trybie **adresowania pośredniego** kod rozkazu zawiera określenie rejestru bądź rejestrów, w których znajduje się adres komórki pamięci zawierającej argument.

Interpretacja graficzna trybu adresowania pośredniego przedstawiona jest na rysunku 3.16.



Rysunek 3.16. Adresowanie pośrednie

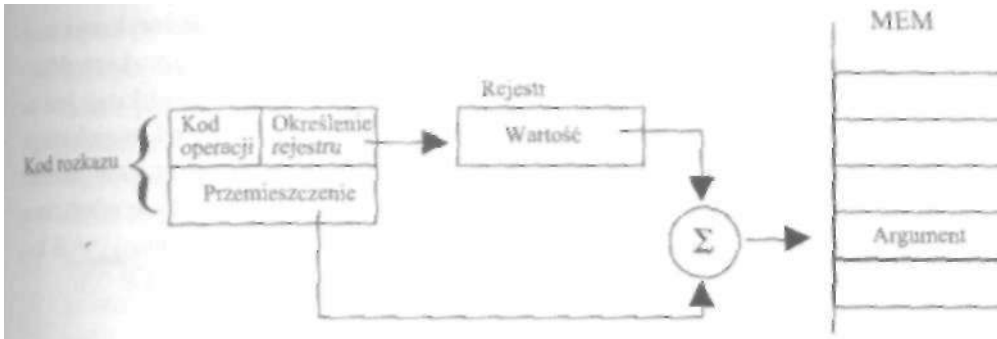
5. Adresowanie indeksowe z przemieszczeniem

W procesorach rodziny Intel 80x86 istnieje 12 trybów adresowania. Część z nich jest kombinacją dwóch lub trzech trybów podstawowych. W naszej książce nie będziemy opisywać wszystkich tych trybów, a osoby zainteresowane odsyłamy do literatury podanej na końcu, na przykład pozycji [1] lub [16]. W tym podpunkcie opisujemy ostatni z trybów adresowania, który można zaliczyć do podstawowych.

Definicja

W trybie **adresowania indeksowego z przemieszczeniem** adres argumentu przechowywanego w pamięci obliczany jest jako suma zawartości rejestru określonego w kodzie rozkazu i wartości umieszczonej w kodzie rozkazu, zwanej przemieszczeniem.

Sposób wyznaczania adresu argumentu w tym trybie pokazano na rysunku 3.17.



Rysunek 3.17. Adresowanie indeksowe

3.3.4.3. Sposób prezentowania rozkazu

Poprawne i efektywne użycie rozkazów wymaga znajomości określonego zestawu informacji na temat rozkazu. Lista rozkazów procesora powinna zawierać następujące informacje:

1. Oznaczenie symboliczne rozkazu.

   Rozkazy przechowywane są w pamięci komputera w postaci binarnej. Jednak zapis binarny rozkazu, nawet w formie heksadecymalnej, byłby niewygodny, nieczytelny i nieużyteczny. Dlatego też np. podczas pisania programów w asemblerze, a ogólnie tam, gdzie prezentuje się go człowiekowi (na przykład w programach typu debugger), używa się oznaczenia symbolicznego rozkazu. Składa się ono z **mnemonika** i pola argumentów. Rozpatrzmy przykładowy rozkaz o symbolicznym oznaczeniu:

   JMP SHORT etykieta

   Przykładowy symbol konkretnej wersji tego rozkazu mnemonik i pole argumentu - pokazany iest na rysunku 3.18.

Rysunek 3.18. Opis rozkazu JMP Short NEXT

Mnemonik jest skrótem, który powinien sugerować rodzaj operacji wykonywanej przez rozkaz. W pewnym sensie można go traktować jako nazwę rozkazu. Mnemoniki pochodzą od słów angielskich, stąd znajomos'ć języka angielskiego jest tu bardzo pomocna. W podanym przykładzie **JMP** jest skrótem od angielskiego słowa *jump* - skocz - i oznacza skok do kodu rozkazu określonego przez argument. Prosimy o porównanie czytelności oznaczenia rozkazu w postaci symbolicznej (JMP SHORT NEXT) z oznaczeniem rozkazu w postaci kodu binarnego (czyli kodu maszynowego) bądź heksadecymalnego (11101011 00000011 lub EB 03h).

2. Opis działania rozkazu.

   Opis działania rozkazu możemy podać zarówno w formie słownej, jak i symbolicznej. Opis słowny byłby następujący: wykonaj skok i pobierz kod rozkazu z komórki pamięci o adresie równym **etykieta** (u nas NEXT = 100). Argumentem rozkazu nie jest jednak wartość adresu, lecz długość skoku (interpretowana w kodzie U2). Słowo SHORT oznacza, że chodzi o tak zwany skok bliski, czyli w zakresie od 127 bajtów w górę do 128 bajtów w dół (wartość takiego skoku można zapisać w postaci jednego bajtu). W naszym przykładzie zostanie wykonany skok do komórki oddalonej o 3 bajty od kodu rozkazu JMP.

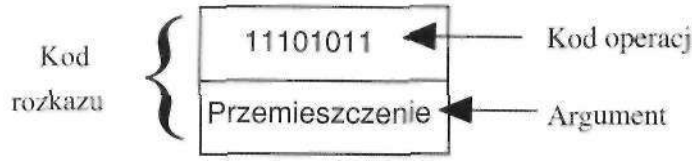
   Skok do komórki o określonym adresie zawierającej kod rozkazu i wykonanie tego rozkazu jako następnego wymaga załadowania tego adresu do licznika rozkazów. Argumentem rozkazu jest liczba komórek, o którą mamy skoczyć, więc symbolicznie wykonanie tego rozkazu możemy zapisać:

   $$(PC) \leftarrow (PC) + \text{przemieszczenie}$$

   co oznacza: w liczniku rozkazów umieść jego zawartość zwiększoną o przemieszczenie, gdzie przemieszczenie (dodatnie bądź ujemne) jest liczbą komórek, o które mamy się przesunąć.

3. Format rozkazu.

   Jak pamiętamy, jest to sposób rozmieszczenia informacji w kodzie rozkazu. W przypadku rozkazu **JMP SHORT etykieta** jest następujący:



   W formacie rozkazu oprócz sposobu rozmieszczenia informacji w kodzie rozkazu zawarta jest także informacja o jego długości. W naszym przypadku mamy do czynienia z rozkazem dwubajtowym. Długość rozkazów jest istotna wówczas, gdy chcemy utworzyć program zajmujący możliwie mało miejsca w pamięci.

4. Ustawiane flagi.

   Kolejną istotną informacją o rozkazie jest, czy i ewentualnie jakie flagi są ustawiane przy wykonaniu danego rozkazu. Reguły ustawiania poszczególnych flag podane są zwykle przy opisie rejestru flagowego. W opisie rozkazu podaje się, **które** flagi są ustawiane. Dla naszego przykładowego rozkazu opis brzmiałby: żadne flagi nie są ustawiane.

5. Szybkość wykonania rozkazu.

   Informacja ta jest istotna w przypadku optymalizacji programu pod kątem szybkości jego działania. Szybkość wykonywania rozkazów jest zwykle podawana jako liczba taktów zegara procesora potrzebnych do wykonania danego rozkazu. Przykładowo rozkaz JMP SHORT jest wykonywany przez procesor 80386 w 7 taktach (w zależności od szybkości zegara oznacza to różny czas wykonania).

3.3.4.4. Przykładowe rozkazy

W celu ilustracji zarówno sposobu opisu rozkazów, jak i ich poszczególnych grup podamy kilka przykładowych rozkazów z listy rozkazów procesora 80486. Z pełną listą rozkazów tego procesora można zapoznać się np. w pozycji [16]. Jest to jednak przydatne jedynie w przypadku programowania w języku asemblera. Pierwszy z przykładowych rozkazów opiszemy nieco dokładniej, aczkolwiek nie jest to też pełny opis - nie podajemy tu wszystkich możliwych kombinacji miejsc przechowywania argumentów. Następne rozkazy opisujemy skrótowo, podając ich oznaczenie dla konkretnego przypadku danego rozkazu, działanie rozkazu i ustawiane flagi. A oto przykładowe rozkazy.

1. **MOV**

   Instrukcja MOV przesyła dane pomiędzy dwoma miejscami. Obydwa argumenty muszą być tego samego rozmiaru.

   Składnia:

   MOV mem, accum (prześlij zawartość akumulatora do komórki o podanym adresie)

   Format:

1010011W	address low	address high

   gdzie: **address Iow, address high** - przemieszczenie względem początku segmentu

   w = 1 - operacja na słowach

   w = 0 - operacja na bajtach

   Ustawiane flagi: żadne flagi nie są ustawiane.

   Czas wykonania: 1 cykl.

2. **ADC**

Instrukcja ADC sumuje dwie liczby umieszczone we wskazanych miejscach oraz i bit CF (przeniesienia). Należy do grupy instrukcji arytmetycznych.

Przykład: ADC ax, bx - do liczby umieszczonej w ax dodaj liczbę umieszczoną w bx oraz bit CF. Wynik umieść w ax. Symbolicznie zapisujemy to następująco:

$$ax \leftarrow ax + bx + CF$$

Ustawiane Hagi: OF, SF, ZF, CF, AF, PF.

3. LOOP etykieta

Dekrementacja rejestru CX, a następnie, jeżeli CX * 0, wykonanie skoku do instrukcji umieszczonej pod adresem o nazwie symbolicznej etykieta. Jest to równoznaczne z n-krotnym wykonaniem pętli obejmującej instrukcje od instrukcji bezpośrednio po nazwie etykieta do instrukcji loop. Liczba obiegów pętli n musi przed rozpoczęciem realizacji pętli zostać załadowana do rejestru CX.

Symbolicznie możemy to zapisać:

a) $CX \leftarrow CX - 1$

b) Jeżeli $CX \neq 0$ to $IP \leftarrow$ etykieta

Ustawiane flagi: żadne flagi nie są ustawiane.

3.3.5. Magistrale i sygnały sterujące mikroprocesora

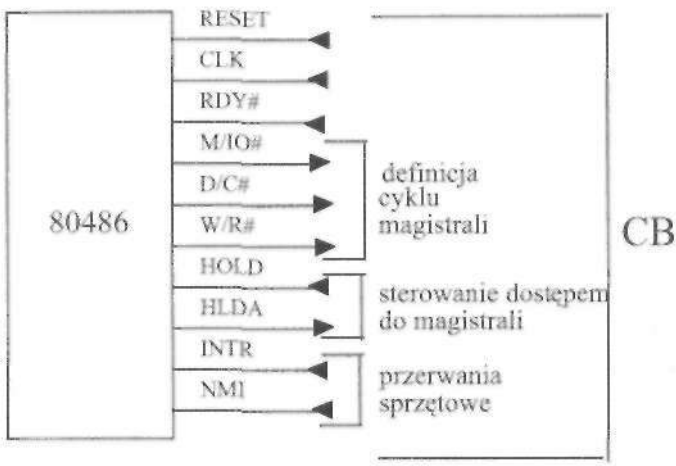
Mikroprocesor komunikuje się z pozostałymi elementami systemu za pomocą magistral. Są to: magistrala danych, magistrala adresowa i magistrala sterująca. Ich j zadania zostały opisane w podpunkcie 3.1.2. Szerokość (czyli liczba linii) dwóch pierwszych ma istotny wpływ na pewne cechy użytkowe systemu. Szerokość magistrali danych jest zwykle dostosowana do długości operandów dla jednostki arytmetyczno-logicznej, choć zdarzają się odstępstwa od tej reguły. Zwiększenie szerokości magistrali danych oznacza więc wzrost mocy obliczeniowej z powodu operowania na dłuższych argumentach i możliwości szybkiego przesyłania większych ilości informacji. Szerokość magistrali adresowej wpływa z kolei na liczbę komórek pamięci, które potrafi bezpośrednio zaadresować mikroprocesor (patrz rozdział poświęcony organizacji pamięci). Przy szerszej magistrali adresowej system dysponuje więc potencjalnie większą pamięcią.

Jednym z zadań mikroprocesora w systemie mikroprocesorowym jest sterowanie i koordynacja pracy pozostałych elementów systemu, takich jak pamięć czy układy wejścia/wyjścia. W tym celu mikroprocesor zawiera specjalną magistralę zwaną sterującą. Magistrala sterująca jest w rzeczywistości zestawem sygnałów zarówno wchodzących, jak i wychodzących z procesora. Pierwsze z nich informują procesor o określonych stanach współpracujących z nim układów, natomiast drugie sterują

pracą tych układów, czyli powodują wykonanie określonej operacji. Jak widzimy, magistrala sterująca nie jest magistralą w ścisłym tego słowa znaczeniu.

Przedstawianie pełnej magistrali sterującej któregokolwiek z procesorów używanych w komputerach kompatybilnych z IBM PC nie miałoby sensu i wykraczałoby znacznie poza ramy tej publikacji. Chcąc jednak dać pewne wyobrażenie o współpracy procesora z innymi układami, wybrano zestaw podstawowych sygnałów sterujących, stanowiący część magistrali sterującej procesora Intel 80486. Przeznaczenie części z nich Czytelnik zrozumie dokładnie dopiero po zapoznaniu się z podrozdziałem 3.5.1 dotyczącym operacji wejścia/wyjścia.

Wybrane sygnały sterujące procesora Intel 80486 przedstawia rysunek 3.19.



Rysunek 3.19. Wybrane sygnały sterujące mikroprocesora 80486

Zadania poszczególnych sygnałów są następujące:

RBSET - restart mikroprocesora (wpis do rejestrów procesora wartości początkowych i rozpoczęcie nowego cyklu rozkazowego).

CLK - (clock) przebieg taktujący (zegar) pracę procesora.

RDY# - (ready) sygnał gotowości układów współpracujących z procesorem (zwykle pamięci). # oznacza, że sygnałem aktywnym (gotowości) jest 0 (poziom niski).

M/IO# - (memory/input/output) sygnał oznaczający operację dotyczącą pamięci (1) lub układów wejścia/wyjścia (0).

D/C# - (data/code) sygnał oznaczający obecność na magistrali danych danej (1) lub kodu rozkazu (0).

W/R# - (write/read) sygnał oznaczający operację zapisu (1) lub odczytu (0).

HOLD - sygnał żądania przejścia procesora w stan zawieszenia (czyli przełączenia wejść i wyjść magistral w stan wysokiej impedancji).

HLDA - (hołd acknowledge) sygnał potwierdzenia przejścia procesora w stan zawieszenia.

1NTR - (interrupt request) sygnał żądania (zgłoszenia) przerwania maskowanego.

NMI - (non-maskable interrupt) sygnał zgłoszenia przerwania niemaskowalnego.

Pierwsze dwa sygnały nie wymagają dalszych wyjaśnień. Aktywny sygnał RDY# powoduje wstawienie do cyklu magistrali procesora stanów oczekiwania (w celu uzyskania gotowości np. pamięci). Kolejne trzy sygnały (M/I0#, D/C#, W/R#) związane są z rodzajem operacji wykonywanej na magistrali i definiują jeden z siedmiu cykli magistrali. Sygnał HOŁD jest jednym z sygnałów sterowania dostępem do magistral. Jeżeli w systemie występuje inny niż mikroprocesor **zarządca magistral** (ang. *bus master),* to sygnał ten pozwala mu przejąć kontrolę nad magistralami (układem takim może być rta przykład sterownik DMA opisany w następnych podrozdziałach). Sygnał HLDA jest odpowiedzią procesora na sygnał HOŁD. Sygnały INTR i NMI związane są z operacjami zwanymi przerwaniami, także opisanymi w kolejnych podrozdziałach.

Sygnały sterujące wytwarzane są przez układ sterowania procesora po zdekodowaniu kodu rozkazu. Mają spowodować wykonanie (realizację) rozkazu. Dlatego rodzaj sygnałów sterujących wytwarzanych w danym momencie zależy od realizowanego rozkazu (lub jego fragmentu). Przykładowo załóżmy, że realizowana jest ostatnia faza (wykonanie) rozkazu MOV AX, [BX]. Rozkaz ten oznacza „zawartość komórki pamięci o adresie umieszczonym w rejestrze BX prześlij do rejestru AX" (czyli akumulatora 16-bitowego). Zatem w fazie tej zostanie wytworzony sygnał **M/IO#** = 1 (bo operacja dotyczy pamięci, a nie układu we/wy), sygnał D/C# = 1 (bo operacja dotyczy danej, a nie kodu rozkazu) i sygnał W/R# = 0 (bo operacja odczytu, a nie zapisu), Ponadto jeżeli założymy, że pamięć jest stosunkowo wolna i mikroprocesor musi poczekać na jej gotowość (czyli ustalenie poprawnej wartości odczytywanego słowa na magistrali danych), to do czasu osiągnięcia tej gotowości na wejściu **RDY#** procesora będzie utrzymywany stan 1 (pamięć niegotowa). Sygnał ten będzie wytwarzany przez układy logiczne sterujące pracą pamięci.

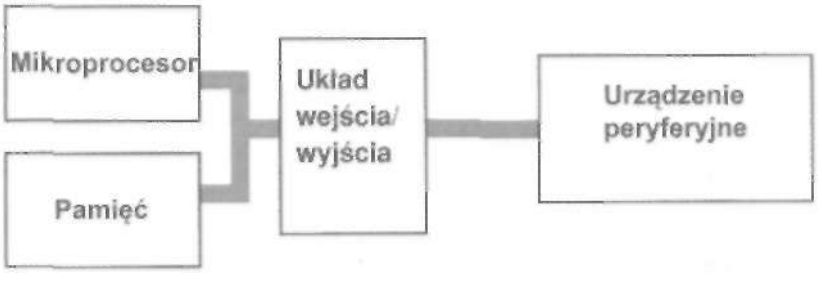
3.4. Układy wejścia/wyjścia

Układy wejścia/wyjścia są układami elektronicznymi pośredniczącymi w wymianie informacji pomiędzy systemem mikroprocesorowym a zewnętrznymi urządzeniami współpracującymi zwanymi **urządzeniami peryferyjnymi.** Urządzenia te mogą służyć przykładowo do wprowadzania, wyprowadzania bądź przechowywania informacji lub mogą być układami wykonawczymi. Przykładami urządzeń peryferyjnych

są pamięci dyskowe, klawiatura, monitor, ploter i tym podobne. Potrzeba pośredniczenia w wymianie informacji wynika z następujących faktów:

- istnieją różnice w szybkości działania współpracujących urządzeń (należy wówczas sterować przepływem informacji);
- istnieją różnice w parametrach elektrycznych współpracujących układów (trzeba dokonać na przykład translacji poziomów sygnałów);
- urządzenie wymaga podania informacji w określonym formacie wraz z pewnymi sygnałami sterującymi (na przykład należy podać treść obrazu w formie sygnału VIDEO wraz z sygnałami synchronizacji).

Koncepcję komunikacji urządzeń peryferyjnych z systemem mikroprocesorowym (procesorem i pamięcią) przedstawia rysunek 3.20.



Rysunek 3.20. Koncepcja komunikacji systemu mikroprocesorowego z urządzeniami peryferyjnymi

Układy wejścia/wyjścia mogą być przeznaczone do współpracy z konkretnym urządzeniem (na przykład karta graficzna - monitor, sterownik dysku twardego - napęd dysku twardego) lub mogą współpracować z wieloma urządzeniami (na przykład interfejs szeregowy RS 232C, sterownik przerwań).

Poniżej podajemy bardziej precyzyjną definicję układów wejścia/wyjścia, a następnie omawiamy podział układów wejścia/wyjścia na układy izolowane i współadresowalne z pamięcią operacyjną.

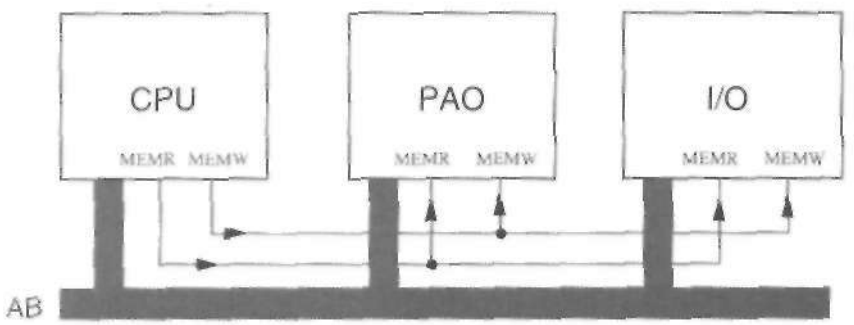
Definicja

Układem wejścia/wyjścia nazywamy układ elektroniczny pośredniczący w wymianie informacji pomiędzy mikroprocesorem i pamięcią systemu z jednej strony a urządzeniem peryferyjnym z drugiej.
Dla systemu mikroprocesorowego układ wejścia/wyjścia widoczny jest jako rejestr lub zespół rejestrów o określonych adresach oraz zestaw sygnałów sterujących.

W zależności od sposobu komunikacji z systemem, a dokładniej od sposobu, w jaki wybierany jest układ wejs'cia/wyjścia, z którym system chce się komunikować, układy wejścia/wyjścia dzielimy na układy współadresowalne z pamięcią operacyjną i układy izolowane.

3.4.1. Układy wejścia/wyjścia współadresowalne z pamięcią operacyjną

Jak wspomniano, układy we/wy możemy traktować jako zespół rejestrów, które wybieramy za pomocą adresów i na których możemy wykonywać operacje zapisu i odczytu. Blok układów we/wy będzie zatem posiadał, podobnie jak pamięć, wejście adresowe i wejście sterujące zapis/odczyt (ang. *read/write)*. Sposób podłączenia układów wejścia/wyjścia **współadresowalnych z pamięcią operacyjną** przedstawia rysunek 3.21.



Rysunek 3.21. Układy wejścia/wyjścia współadresowalne z pamięcią operacyjną

Definicja

W przypadku **układów współadresowalnych** z pamięcią operacyjną wybieramy obiekt, na którym dokonujemy operacji (komórka pamięci lub rejestr układu wejścia/wyjścia), za pomocą adresu. Sygnały sterujące są wspólne dla pamięci i układów wejścia/wyjścia.

Układy współadresowalne z PAO wymagają wydzielenia części przestrzeni adresowej (czyli zakresu przydzielonych im adresów) pamięci dla adresów układów wejścia/wyjścia. Układy wejścia/wyjścia i pamięć obsługiwane są tymi samymi rozkazami (co wynika ze wspólnych sygnałów sterujących - przypominamy, że są wytwarzane w wyniku realizacji określonego rozkazu).

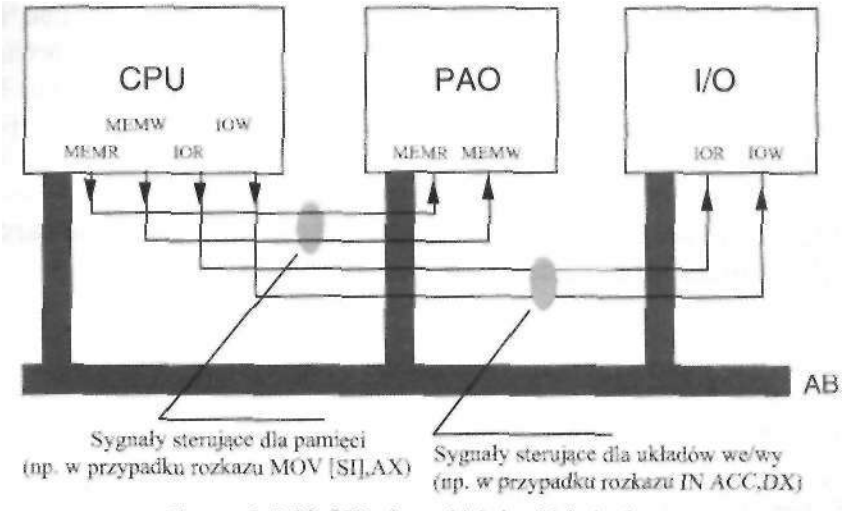
3.4.2. Układy wejścia/wyjścia izolowane

W przypadku układów wejścia/wyjścia **izolowanych** sygnały sterujące dla pamięci i układów wejścia/wyjścia są rozdzielone. Sposób ich podłączenia pokazany jest na rysunku 3.22.

Definicja

Dla **izolowanych układów wejścia/wyjścia** wybieramy obiekt, na którym dokonujemy operacji (komórka pamięci lub rejestr układu wejścia/wyjścia), za pomocą sygnałów sterujących. Przestrzenie adresowe pamięci i układów wejścia/wyjścia są rozdzielone.

Przestrzeń adresowa układów we/wy i pamięci operacyjnej mogą się pokrywać, gdyż w przypadku jednakowego adresu sygnały sterujące decydują o tym, czy zostanie wykonana operacja na układzie we/wy, czy na komórce pamięci. Wymaga to oczywiście istnienia osobnych rozkazów obsługujących pamięć, a osobnych dla układów we/wy.



Rysunek 3.22. Układy wejścia/wyjścia izolowane

W komputerach typu IBM PC stosowane są obydwa rozwiązania, choć większość układów wejs'cia/wyjścia to układy izolowane. Przykładem układu współadresowalnego z pamięcią operacyjną jest karta graficzna. Zawartość obrazu wyświetlanego na ekranie umieszczana jest przez system w tak zwanym buforze wideo (video RAM), który jest blokiem pamięci RAM umieszczonym w określonym miejscu przestrzeni adresowej pamięci operacyjnej. Przykładem układu izolowanego może być sterownik dysku twardego w standardzie IDE, który przez system widziany jest jako zestaw rejestrów o adresach zarezerwowanych **w** przestrzeni adresowej układów wejścia/wyjścia.

3.5. Operacje wejścia/wyjścia

Opisane układy wejścia/wyjścia stanowią część sprzętową komunikacji systemu mikroprocesorowego z otoczeniem. Stwierdzono jednak w podpunkcie 3.1.2, że wszystko, co zdarza się w systemie, jest wynikiem działania pewnego programu (lub jego fragmentu). Dotyczy to także wymiany informacji systemu z otoczeniem. Programy realizujące tę wymianę i wszelkie .operacje jej dotyczące nazywamy **operacjami weiścia/wyjścia.**

Definicja
Operacjami wejścia/wyjścia nazywamy całokształt działań potrzebnych do realizacji wymiany informacji pomiędzy mikroprocesorem i pamięcią z jednej strony a układem wejścia/wyjścia z drugiej.

Operacje wejścia/wyjścia mogą być realizowane od początku do końca przy udziale mikroprocesora. Przesyłana informacja przepływa wówczas przez rejestry mikroprocesora, który także steruje każdym krokiem realizacji operacji. Inną możliwością jest jedynie zainicjowanie operacji we/wy przez mikroprocesor, który następnie przekazuje nadzór nad jej realizacją innemu układowi (zarządcy magistral). Dlatego operacje wejścia/wyjścia możemy podzielić na operacje z bezpośrednim i pośrednim sterowaniem przez mikroprocesor.

3.5.1. Operacje wejścia/wyjścia z bezpośrednim sterowaniem przez mikroprocesor

W zależności od sposobu realizacji operacje wejścia/wyjścia z bezpośrednim sterowaniem przez mikroprocesor możemy podzielić na trzy grupy:

3.5.1.1. Bezwarunkowe operacje wejścia/wyjścia

Definicja
Bezwarunkową operacją wejścia/wyjścia nazywamy operację, przy której realizacji mikroprocesor nie sprawdza gotowości układu wejścia/wyjścia do tej wymiany.

Jest oczywiste, że nie z każdym układem wejścia/wyjścia możemy się w ten sposób komunikować. Istnieje jednak pewna klasa układów, dla których takie operacje są możliwe. Przykładem może być monitorowanie stanu pewnego miejsca, na przykład określonego miejsca magistrali za pomocą zestawu diod. Przesłanie informacji do wyświetlenia może się odbyć w dowolnym momencie, bez sprawdzania gotowości diod do jej wyświetlenia - diody bowiem są zawsze gotowe wyświetlić przesłaną informację. Operacje takie są najprostszymi operacjami wejścia/wyjścia.

3.5.1.2. Operacje wejścia/wyjścia z testowaniem stanu układu wejścia/wyjścia

Definicja

Podczas realizacji **operacji wejścia/wyjścia** *2* **testowaniem stanu układu wejścia/** wyjścia mikroprocesor sprawdza sygnał (np. może to być określony bit) gotowości układu wejścia/wyjścia do tej wymiany. W przypadku potwierdzenia gotowości do wymiany przez układ jest ona realizowana.

Brak gotowości układu wejścia/wyjścia do wymiany powoduje wykonywanie przez mikroprocesor tak zwanej pętli przepytywania, w której cykliczne sprawdza on gotowość układu do wymiany. Po jej potwierdzeniu pętla jest kończona i następuje realizacja wymiany.

Przykładem tego typu operacji wejścia/wyjs'cia może być współpraca systemu z przetwornikiem a/c. Pobranie wartości przetwarzanej wielkości z wyjścia przetwornika przed zakończeniem przetwarzania (jeżeli jest to w ogóle możliwe) spowodowałoby błąd - odczytalibyśmy nieprawdziwą wartość. Dlatego przetwornik a/c może mieć na przykład rejestr stanu z bitem zajętości BSY. Dopóki wartość tego bitu wynosi 1 (przetwornik nie ukończył przetwarzania), nie możemy odczytać przetwarzanej wartości. Mikroprocesor, oczekując na gotowość przetwornika, cyklicznie sprawdza stan bitu BSY. W przypadku stwierdzenia zera (przetwarzanie zakończono) kończy sprawdzanie bitu BSY i wczytuje wartość z wyjścia przetwornika do swojego rejestru lub komórki pamięci, kończąc tym samym operację wejścia/wyjścia.

Kolejną grupą operacji wejścia/wyjścia z bezpośrednim sterowaniem przez mikroprocesor są operacje wejścia/wyjścia z przerwaniem programu. Jest to tak ważna grupa, że poświęcamy jej osobny podrozdział.

3.5.1.3. Operacje wejścia/wyjścia z przerwaniem programu

Istota operacji wejścia/wyjścia z przerwaniem programu

Jedną z oczywistych wad operacji wejścia/wyjścia z testowaniem stanu układu wejścia/wyjścia jest konieczność wykonywania przez mikroprocesor pętli przepytywania w celu stwierdzenia gotowości tego układu do wymiany. Rozwiązanie tego problemu jest proste. Mikroprocesor wykonuje program główny, oczekując na sygnał gotowości do wymiany zgłoszony ze strony układu wejścia/wyjścia (nie testując w sposób programowy układu wejścia/wyjścia). W tym celu mikroprocesor dysponuje określonym wejściem zwanym **wejściem zgłoszenia przerwania** (patrz zestaw wybranych sygnałów magistrali sterującej mikroprocesora). Aktywny poziom na tym wejściu sygnalizuje tak zwane **zgłoszenie przerwania,** czyli gotowość układu wejścia/wyjścia do wymiany. Zgłoszenie przerwania powoduje przerwanie przez mikroprocesor wykonywania programu głównego po zakończeniu realizacji bieżącej instrukcji i zapamiętanie informacji potrzebnej do późniejszego powrotu do programu

głównego i jego kontynuowania. Następnie mikroprocesor przechodzi do wykonania specjalnego programu zwanego **programem obsługi przerwania** (w skrócie POP, ang. *ISR - interrupt sernice routine).* Program ten powinien zrealizować wymianę informacji z układem wejścia/wyjścia, który zgłosił gotowość. Po zakończeniu wymiany, czyli po zakończeniu programu obsługi przerwania, mikroprocesor kontynuuje przerwany program główny. Schematycznie taka operacja przedstawiona jest na rysunku 3.23.

Rysunek 3.23. Wykonanie operacji wejścia/wyjścia z przerwaniem programu

Sytuacja komplikuje się nieco w przypadku obsługiwania tą metodą kilku układów wejścia/wyjścia. Może się wówczas zdarzyć, że (zgodnie z prawami Murphy'ego - w najmniej korzystnym momencie) jednocześnie kilka układów wejs'cia/wyjścia zgłosi gotowość do wymiany. Należy wówczas zdecydować, który układ zostanie obsłużony, i o wyborze tym poinformować mikroprocesor. Ponadto, ponieważ mikroprocesor ma tylko jedno wejście zgłoszenia przerwania, musi istnieć układ pośredniczący w przyjmowaniu zgłoszeń przerwań pomiędzy mikroprocesorem a układami wejścia/wyjścia. Układem tym jest specjalizowany układ zaliczany do układów wejścia/wyjścia, zwany **sterownikiem przerwań.** Przedstawiamy zarówno podstawowe zadania i sposób działania sterownika przerwań, jak i sposób zapamiętywania informacji o programie głównym w celu późniejszej kontynuacji.

Sterownik przerwań

Zgodnie z dotychczasowymi rozważaniami, podstawowymi zadaniami sterow-

pośredniczenie w przyjmowaniu zgłoszeń przerwań pomiędzy mikroprocesorem a innymi układami wejścia/wyjs'cia;

i ' przyjmowanie zgłoszeń przerwań od wielu układów wejścia/wyjścia;

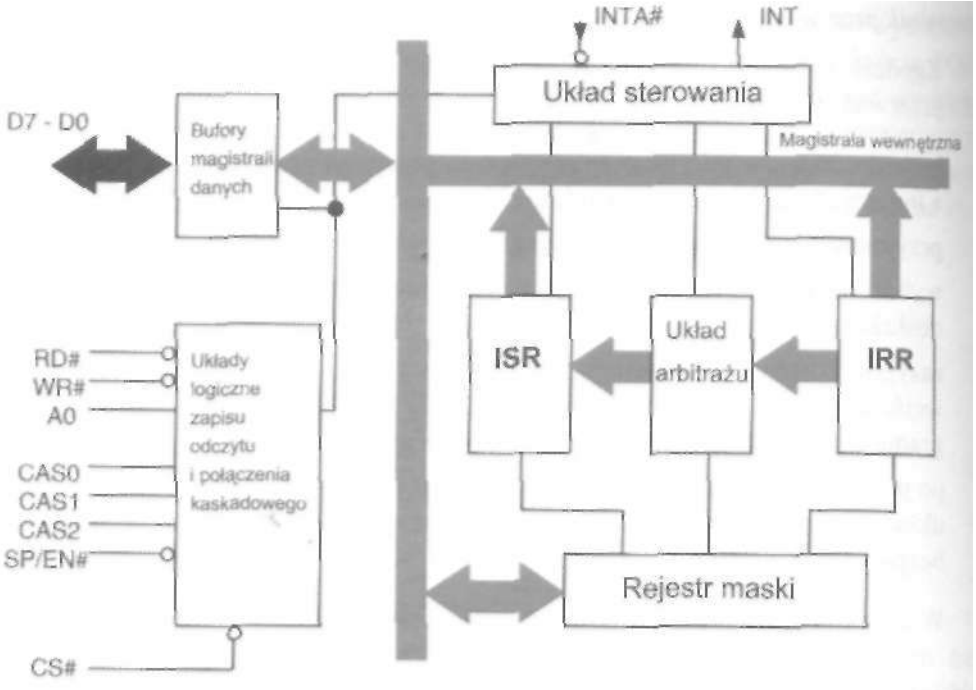
wybór spośród kilku jednocześnie zgłoszonych przerwań tego, które zostanie

zasygnalizowanie dokonanego wyboru przez podanie numeru (adresu) układu wejścia/wyjścia, z którym zostanie dokonana wymiana, a dokładniej adresu programu obsługi przerwania realizującego tę wymianę;

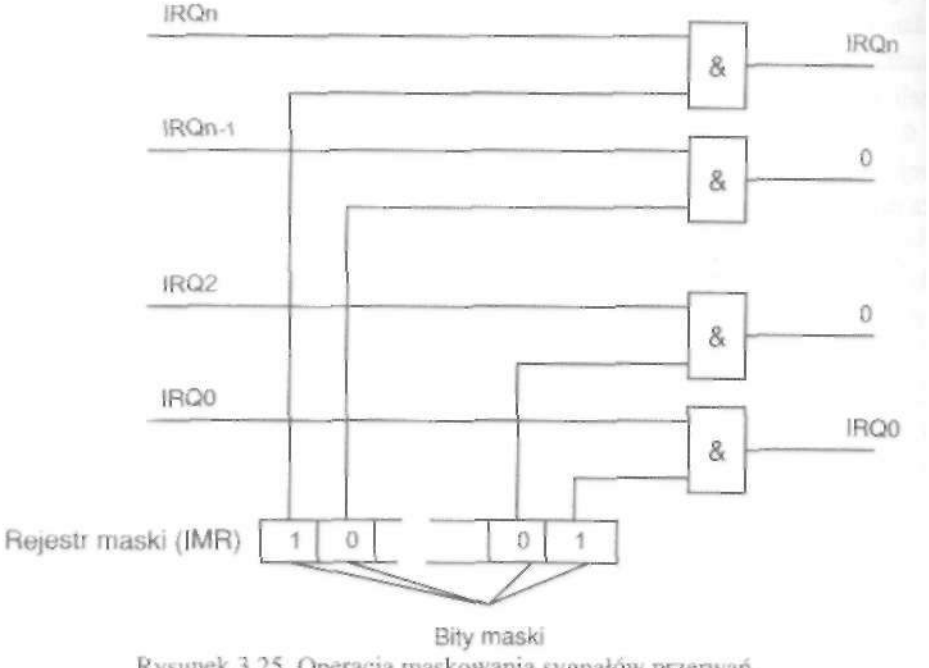
- po przyjęciu sygnałów zgłoszeń przerwań od układów wejścia/wyjścia i wyborze układu, który będzie obsłużony, wygenerowanie sygnału zgłoszenia przerwania bezpośrednio do mikroprocesora.

W celu umożliwienia przyjmowania kilku przerwań jednocześnie sterownik przerwań dysponuje określoną liczbą wejść sygnałów zgłoszeń przerwań od układów wejścia/wyjścia. .Sygnały te trafiają do rejestru zgłoszeń przerwań IRR (ang. *Interrupt Request Register).* Zgłoszone przerwania mogą być maskowane, co oznacza, że mimo ich zgłoszenia nie będą przyjmowane. Operacja maskowania jest prosta i polega na podaniu sygnałów zgłoszeń przerwań na dwuwejściowe bramki AND, co pokazano na rysunku 3.25. Na drugie wejście każdej z bramek podawany jest tak zwany **bit maski.** Jeżeli wartość tego bitu wynosi 1, przerwanie jest przyjmowane, w przypadku wartości 0 następuje zablokowanie przyjmowania przerwania. Bity maski umieszczane są w rejestrze maski IMR (ang. *Interrupt Mask Register,* patrz schemat blokowy sterownika przerwań - rysunek 3.24).

Następnie przyjmowane przerwania poddawane są arbitrażowi. **Arbitraż** polega na wyborze spośród kilku zgłoszonych jednocześnie przerwań jednego, które zostanie w danym momencie obsłużone. Wymaga to ustalenia kryterium, według którego zostanie dokonany wybór. Kryterium tym jest stopień ważności, który przydzielamy każdemu sygnałowi zgłoszenia przerwania, zwany **priorytetem.** Ze wszystkich zgłoszonych i niezamaskowanych przerwań wybierane jest to, które ma najwyższy priorytet. Priorytety przyporządkowywane są poszczególnym wejściom sterownika przerwań. W zależności od jego wersji (i intencji twórcy oprogramowania) mogą być przyporządkowane w sposób sztywny (na stałe, jak w komputerach typu IBM PC) lub programowane. Arbitraż dokonywany jest w układzie arbitrażu (ang. *priority resoher)* będącego najczęściej koderem priorytetu (którego działanie opisano w rozdziale 2.).



Rysunek 3.24. Schemat blokowy sterownika przerwań



Rysunek 3.25. Operacja maskowania sygnałów przerwań

Wynikiem arbitrażu jest numer zgłoszonego, niezamaskowanego przerwania o najwyższym priorytecie. Numer ten jest wpisywany do rejestru przerwań obsługiwanych ISR (ang. *in-sewice register)*. Przy spełnieniu określonych warunków umożliwia to przyjmowanie przerwań zagnieżdżonych, czyli przerywanie programu obsługi przerwania.

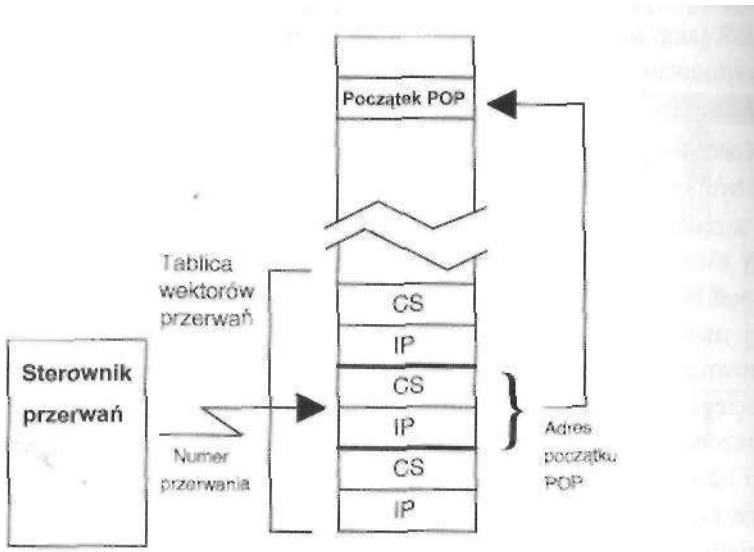
Końcowym wynikiem współpracy sterownika przerwań z mikroprocesorem powinno być uruchomienie określonego programu obsługi przerwania. Jest oczywiste, że różne urządzenia współpracujące z określonymi układami wejścia/wyjścia będą wymagały różnych programów obsługi przerwania. Przykładowo przerwanie zgłoszone przez port równoległy, do którego podłączona jest drukarka, powinno przesłać do niej kolejny znak łub blok znaków przeznaczonych do wydrukowania. Z kolei przerwanie od sterownika klawiatury powinno spowodować odczytanie przez mikroprocesor kodu naciśniętego klawisza. Programy obsługi przerwań, podobnie jak inne programy, umieszczone są w pamięci operacyjnej i zaczynają się od określonego adresu. Wybór obsługi określonego przerwania polega więc na podaniu adresu pamięci, pod którym znajduje się początek programu obsługującego dane przerwanie. Z powodów, które wyjaśniamy poniżej, sterownik przerwań w komputerach typu IBM PC nie podaje bezpośrednio tego adresu, lecz numer pozycji w specjalnej tablicy umieszczonej w pamięci zwanej **tablicą wektorów przerwań** (ang. *Interrupt Vector Tabl - IVT)*. Dopiero elementy tej tablicy zawierają właściwe adresy początków programów obsługi przerwań.

Tablica wektorów przerwań

Tablica wektorów przerwań jest strukturą umieszczoną w pamięci operacyjnej komputera. W komputerach typu IBM PC zaczyna się od adresu 0 i kończy na adresie 3FFh. Każda pozycja tej tablicy związana jest z dokładnie jednym przerwaniem i zawiera adres komórki pamięci, w której znajduje się początek (kod pierwszej instrukcji) programu obsługującego to przerwanie. Po dokonaniu arbitrażu sterownik przerwań podaje na swoim wyjściu danych numer przerwania, które ma być obsłużone. Mikroprocesor pobiera ten numer i następnie za jego pomocą odczytuje adres początku programu obsługi przerwania, który ma zostać wykonany. Adres ten, po zapamiętaniu informacji niezbędnych do powrotu do przerwanego programu głównego, jest ładowany do określonych rejestrów procesora, w tym licznika rozkazów. Powoduje to wykonanie skoku do programu obsługi przerwania umieszczonego pod adresem odczytanym z tablicy wektorów przerwań. Schematycznie sytuacja ta jest przedstawiona na rysunku 3.26.

Po zrealizowaniu wymiany następuje powrót z programu obsługi przerwania do programu głównego, który jest kontynuowany.

PAO



Rysunek 3.26. Generowanie adresu POP przy użyciu tablicy wektorów przerwań

Obsługa systemu przerwań za pomocą tablicy wektorów przerwań ma bardzo istotne zalety. W przypadku zmiany sposobu obsługi przerwania, czyli realizacji dla danego przerwania innego programu obsługi przerwania (niż np. program standardowy zawarty w BIOS-ie, w pamięci stałej), zmieniamy jedynie adres w pozycji tablicy wektorów przerwań odpowiadającej danemu przerwaniu. Oczywiście pod wprowadzonym nowym adresem musi zostać umieszczony początek nowego programu obsługującego przerwanie. Metoda taka jest stosowana przykładowo przy instalowaniu niestandardowych sterowników (driverów) dysków twardych, dostarczanych przez producenta wraz ze sterownikiem dysku.

Metody zapamiętania stanu mikroprocesora przy przejściu do obsługi przerwania

Jednym z założeń przy obsłudze układów wejścia/wyjścia za pomocą przerwań jest kontynuowanie, po zakończeniu programu obsługi przerwania, programu głównego. Wymaga to przy przejściu do realizacji programu obsługi przerwania zapamiętania wszelkich informacji o programie głównym potrzebnych do jego kontynuowania (od miejsca, w którym został przerwany). Zwykle tylko część potrzebnej informacji zapamiętywana jest automatycznie. W przypadku rodziny procesorów Intel x86 automatycznie zachowywana jest jedynie zawartość rejestru flagowego i adres komórki zawierającej kod instrukcji, od której należy kontynuować program główny (jest to zawartość dwóch rejestrów: segmentu kodu i wskaźnika instrukcji - patrz rozdział o procesorach rodziny Intel x86). Informacje te są odkładane na stos. Pozostałe informacje muszą zostać zachowane w wyniku wykonania określonych instrukcji programu

obsługi przerwania. Z reguły zachowujemy na stosie zawartość wszystkich rejestrów procesora używanych przez program obsługi przerwania.

Przerwania sprzętowe a przerwania programowe

Opisane przerwania noszą nazwę **przerwań sprzętowych.** Wynika to z faktu, że przyczyną ich realizacji jest fizyczny sygnał zgłoszenia przerwania pochodzący od określonego urządzenia. Dla systemów z mikroprocesorami rodziny Intel x86 możliwy jest też inny rodzaj przerwań, tak zwane **przerwania programowe.** Przerwania programowe są wynikiem wykonania przez mikroprocesor rozkazu INTn, gdzie n jest numerem pozycji w tablicy wektorów przerwań. W wyniku wykonania tego rozkazu mikroprocesor, tak jak dla przerwania sprzętowego, odkłada na stos zawartość rejestru flagowego oraz adres powrotu do przerwanego programu i przechodzi do wykonania programu obsługi przerwania. Adres początku tego programu określony jest przez pozycję w tablicy wektorów przerwań o numerze n podanym jako parametr rozkazu INT.

Rozkaz INT umożliwia generowanie pełnego zestawu przerwań, aczkolwiek używanie w ten sposób przerwań sprzętowych nie jest celowe. Przerwania programowe możemy podzielić na przerwania BIOS oraz DOS. Oferują one różnorodne użyteczne usługi, które możemy wykorzystać w programach pisanych w asemblerze. Opis tych przerwań można znaleźć na przykład w pozycjach [16] lub [23].

Trzecim rodzajem przerwań, które możemy zaliczyć do przerwań sprzętowych, gdyż są generowane przez mikroprocesor, są tak zwane **wyjątki** (ang. *exceptions).* Są to „wewnętrzne przerwania" generowane przez mikroprocesor w wyniku poważnych błędów lub sytuacji wymagających obsługi pojawiających się w trakcie wykonania programu. Przykładem może być tu dzielenie przez zero czy brak w pamięci operacyjnej bloku, do którego odwołuje się program. Wyjątki mają przydzielone określone pozycje w tablicy wektorów przerwań. Ich zgłoszenie powinno uruchomić określony program obsługi (ang. *handler).*

Opis wybranych fragmentów tablicy wektorów przerwań znajduje się w podrozdziale 6.2.1 opisującym architekturę systemu ISA.

3.5.2. Operacje wejścia/wyjścia z pośrednim sterowaniem przez mikroprocesor (DMA)

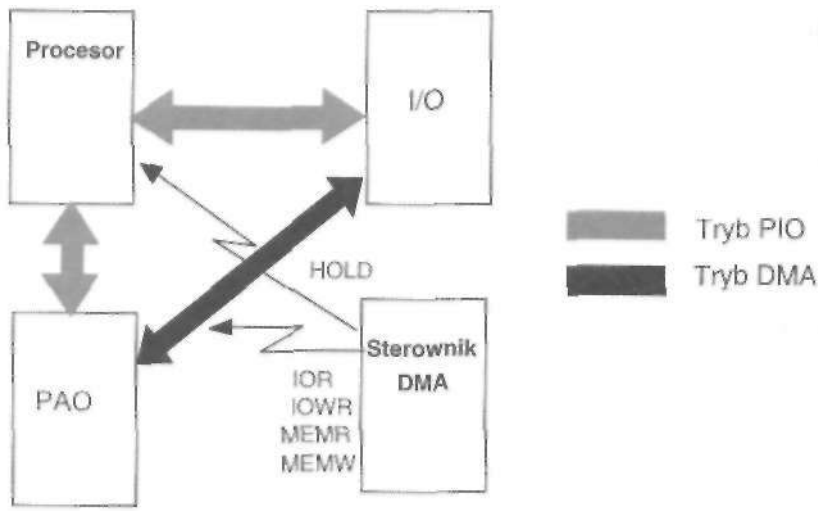
W dotychczas opisanych operacjach wejścia/wyjścia ich realizacja przebiegała od początku do końca pod nadzorem mikroprocesora. Oznacza to, że wszystkie informacje, takie jak sygnały sterujące czy adresy, są generowane przez mikroprocesor oraz że w wymianę informacji zaangażowane są pewne jego rejestry. Powoduje to, że w trakcie tej wymiany, a dokładniej w trakcie wykonywania programu realizującego daną operację wejścia/wyjścia, mikroprocesor nie może wykonywać żadnych innych czynności. Ten rodzaj operacji zwany jest często **trybem PIO.** Kolejny rodzaj opera-

cji wejścia/wyjścia, zwany **bezpośrednim dostępem do pamięci** (ang. *DMA - direct memory access),* eliminuje ten problem.

Definicja

Bezpośrednim dostępem do pamięci nazywamy operację wejścia/wyjścia jedynie inicjowaną przez mikroprocesor, który przekazuje sterowanie jej realizacją specjalizowanemu układowi zwanemu **sterownikiem DMA.**

Przy bezpośrednim dostępie do pamięci, zwanym dalej operacją DMA, transmisja informacji przebiega pomiędzy układem wejścia/wyjścia a wydzielonym obszarem buforowym w pamięci. Przebieg operacji nadzoruje sterownik DMA, co oznacza, że generuje on wszystkie sygnały sterujące i adresy potrzebne do realizacji wymiany. W tym celu sterownik DMA przejmuje na czas wymiany informacji kontrolę nad magistralami, stając się zarządcą magistral (ang. *bus master).* Koncepcja ta jest przedstawiona na rysunku 3.27. Na rysunku przedstawiono też podstawowe sygnały sterujące potrzebne do realizacji operacji DMA.



Rysunek 3.27. Koncepcja operacji DMA

Żądanie przejęcia kontroli nad magistralami jest zgłaszane do mikroprocesora za pomocą sygnału sterującego HOLD. W odpowiedzi na ten sygnał mikroprocesor przechodzi w tak zwany stan **zawieszenia,** polegający na elektrycznym odseparowaniu się od magistral przez przełączenie wyjściowych wzmacniaczy trójstanowych w stan wysokiej impedancji (patrz rozdział 2.). Przejście w stan zawieszenia jest sygnalizowane przez mikroprocesor stanem aktywnym na wyjściu HLDA (ang. *hołd acknowledge).* Zauważmy, że przejście to nie wymaga żadnych zmian stanu rejestrów mikroprocesora. Sterownik DMA zlicza także liczbę przesłanych słów w celu stwier-

dzenia zakończenia operacji po przesłaniu całego bloku (wielkość bloku jest jednym z parametrów ustalanych w trakcie inicjacji kanału DMA). Po zakończeniu transmisji (pojedynczego słowa lub bloku w zależności od trybu realizacji operacji, co jest opisane w dalszej części tego podpunktu) sterownik DMA zwraca mikroprocesorowi kontrolę nad magistralami.

W realizacji operacji DMA możemy wyróżnić następujące etapy:

1 inicjacja operacji DMA,
2. realizacja operacji DMA,
3. zakończenie operacji.

Operacja DMA, podobnie jak przerwanie, inicjowana jest na żądanie układu wejścia/wyjścia. Żądanie to jest zgłaszane do sterownika DMA sygnałem o nazwie DRQn (gdzie n jest numerem kanału DMA). Inicjacja jest realizowana przez mikroprocesor, gdyż, jak stwierdzono wcześniej, kieruje on pracą całego systemu. Inicjacja operacji DMA polega na przekazaniu do sterownika DMA (w wyniku wykonania określonych instrukcji) następujących informacji:

1. wielkość bloku do przetransmitowania (liczba bajtów lub słów),
2, adres pierwszej komórki bufora w pamięci,
3, rodzaj operacji (zapis lub odczyt),
4, sposób realizacji operacji DMA.

Ostatni punkt jest związany z realizacją operacji DMA i wymaga wyjaśnienia. Operację DMA możemy realizować w jednym z trzech trybów:

- transmisja pojedynczymi słowami,
- transmisja blokowa,
- transmisja na żądanie.

W każdym przypadku przesyłany jest blok informacji o wielkości określonej jako jeden z parametrów operacji. Sposób realizacji transmisji jest jednak różny. W przypadku transmisji pojedynczymi słowami sterownik DMA po sygnale gotowości od układu wejścia/wyjścia przejmuje sterowanie magistralami na czas jednego cyklu, realizuje operację wymiany, po czym oddaje kontrolę nad magistralami mikroprocesorowi, oczekując na kolejny sygnał gotowości od układu wejścia/wyjścia. Postępowanie takie jest kontynuowane aż do momentu przesłania całego bloku. Dopiero wówczas jest sygnalizowane zakończenie operacji DMA.

Transmisja blokowa jest realizowana w sposób ciągły aż do momentu przesłania całego bloku. Podczas jej realizacji mikroprocesor pozostaje cały czas w stanie zawieszenia, a kontrolę nad magistralami sprawuje sterownik DMA.

Przy transmisji na żądanie przesyłanie kolejnych słów trwa w sposób nieprzerwany dopóty, dopóki jest aktywny sygnał DRQn z obsługiwanego układu wejścia/wyjs'cia. W przypadku, gdy przechodzi on w stan nieaktywny, transmisja jest zawieszana, a kontrola nad magistralami zostaje przekazana do mikroprocesora. Transmisja jest kontynuowana po ponownym przejściu sygnału DRQn w stan aktywny. Podobnie jak w pozostałych przypadkach, zakończenie operacji DMA jest sygnalizowane po przesłaniu całego bloku.

Zakończenie operacji DMA sygnalizowane jest przez sterownik DMA aktywnym poziomem sygnału EOP (ang. *end of process).* Sygnał ten jest przekazywany do obsługiwanego urządzenia, dzięki czemu urządzenie zgłasza przerwanie do procesora. Przerwanie to oznacza możliwość wykorzystania danych przez mikroprocesor w przypadku transmisji DMA do pamięci i przesłanie wszystkich danych z pamięci do urządzenia zewnętrznego w przypadku transmisji DMA do urządzenia.

Budowa podsystemu DMA dla komputerów IBM PC jest przedstawiona w rozdziale 6. dotyczącym budowy płyt systemu ISA.

3.6. Pamięć wirtualna

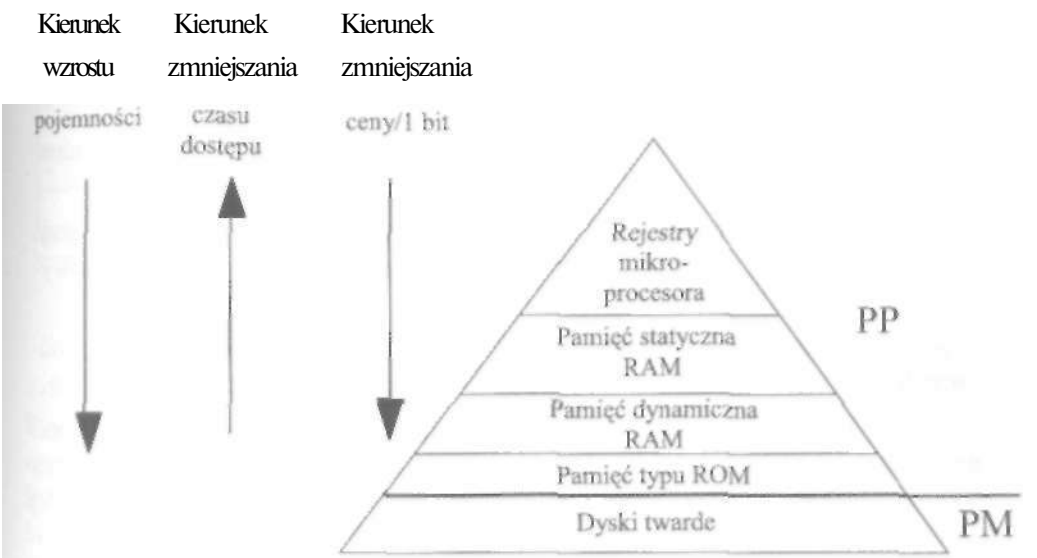
3.6.1. Hierarchia pamięci

W systemie mikroprocesorowym występuje kilka rodzajów pamięci. Na początek możemy podzielić je na pamięć masową PM i pamięć półprzewodnikową PP. Do pamięci masowych należą wszelkiego rodzaju pamięci na nośnikach magnetycznych, takie jak dyski twarde i elastyczne, streamery oraz pamięci optyczne typu CD-ROM czy też DVD-ROM. Rodzaje pamięci półprzewodnikowych to pamięci do zapisu i odczytu dynamiczne i statyczne oraz pamięci typu ROM, a także rejestry wewnątrz mikroprocesora. Pamięci te można uszeregować w hierarchiczną strukturę, biorąc pod uwagę trzy czynniki: pojemność, koszt jednego bitu oraz czas dostępu. Hierarchia pamięci uwzględniająca najistotniejsze składniki systemu przedstawiona jest na rysunku 3.28.

Jak widzimy, wzrost pojemności układów pamiętających oraz zmniejszanie się ceny 1 bitu jest (niestety) odwrotnie proporcjonalne do zmniejszania się czasu dostępu, czyli do szybkości działania tych układów. Wynikają z tego określone konsekwencje. Układami pamiętającymi o największej pojemności w systemie są pamięci masowe, na przykład dyski twarde. Są to jednocześnie najtańsze układy pamiętające, zwłaszcza jeżeli uwzględnimy cenę 1 bitu. Niestety są to układy znacznie wolniejsze od pamięci półprzewodnikowych. Ponieważ, jak wspomniano w podrozdziale 3.1.2, pamięć, z którą komunikuje się mikroprocesor, musi być pamięcią szybką, budujemy ją z pamięci półprzewodnikowych. Nie może mieć ona jednak takich pojemności jak dyski twarde ze względów zarówno technologicznych, jak i ekonomicznych. Z po-

dobnych powodów nie możemy pozwolić sobie na zbudowanie całej pamięci operacyjnej z pamięci statycznych, które są szybsze, lecz drogie i trudniej podlegają scalaniu. Wreszcie najszybszymi układami pamiętającymi w systemie są rejestry mikroprocesora, jednak ich pojemność jest najmniejsza. W systemie przyjęto więc rodzaj kompromisu. Mamy pamięć o dużej pojemności, lecz stosunkowo wolną (pamięć masową) i szybszą od niej, lecz droższą i o mniejszej pojemności pamięć operacyjną zbudowaną z pamięci DRAM. Zabieg, który pozwoli traktować programom pamięć masową jako przedłużenie pamięci operacyjnej, doprowadzi nas do pojęcia pamięci wirtualnej. Opisujemy ją w następnym podpunkcie. Podobne rozważania przeprowadzone dla pamięci DRAM i SRAM doprowadzą nas do pojęcia pamięci cache. Jak wiadomo, cache jest pamięcią bardzo szybką, lecz jej pojemność w systemie jest niewielka i wynosi od setek kB do pojedynczych MB. Koncepcja pamięci cache jest dokładniej omawiana w punkcie 3.7. Najszybszymi układami pamiętającymi są rejestry mikroprocesora. Konsekwencją tego są rozwiązania stosowane na przykład w procesorach RISC (duża liczba rejestrów roboczych).

Pamięć ROM w pokazanej hierarchii zachowuje się nietypowo (szczególnie jeśli chodzi o jej pojemność) i została tu umieszczona głównie z powodu jej czasu dostępu, który jest większy (dłuższy) niż dla pamięci DRAM. Wyjaśni to używanie w części systemów tak zwanego *shadow* BIOS-u.



Rysunek 3.28. Hierarchia pamięci w systemie komputerowym

3.6.2. Zasada działania pamięci wirtualnej

Jak już stwierdzono, mechanizm pamięci wirtualnej pozwala traktować programom pamięć masową jako przedłużenie pamięci operacyjnej. Zastanówmy się, co to oznacza. Pamięć masowa ma znacznie większą pojemność, co pozwala na używanie w programie dłuższych adresów. Z drugiej strony nie jest możliwe odczytywanie pojedynczych instrukcji w trakcie wykonywania programu z pamięci masowej, która jest dość wolna, a co ważniejsze, jej urządzenia są urządzeniami o dostępie blokowym. Dlatego też stosuje się następujące rozwiązanie. Pewna część informacji wykonywanego programu załadowana jest do pamięci operacyjnej. Pozostała część niemieszcząca się w niej jest przechowywana w pamięci masowej. Jeżeli w trakcie realizacji programu następuje odwołanie do informacji znajdującej się na dysku, to odpowiedni mechanizm powoduje wczytanie brakującego **bloku** informacji do pamięci operacyjnej, przesyłając inny blok do pamięci masowej w celu zwolnienia miejsca (chyba że blok ten, na przykład fragment programu, jest w niej już zapisany). Na wykonanie tych operacji pozwala mechanizm pamięci wirtualnej. Powinien także spowodować przetłumaczenie (translację) długich adresów (wirtualnych) w programie na krótsze adresy (fizyczne) pamięci operacyjnej. Przy okazji takiej translacji należy też sprawdzać, czy poszukiwana informacja jest dostępna w pamięci operacyjnej.

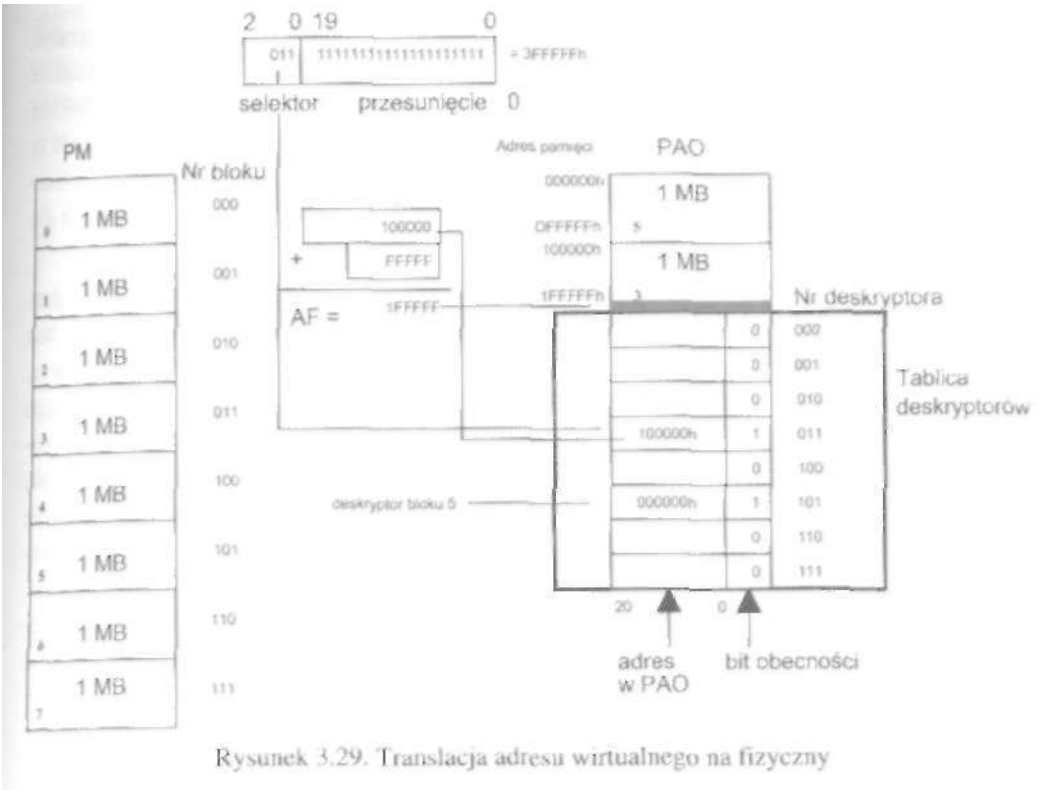
Podsumowując, mechanizm pamięci wirtualnej jest następujący (w nawiasach podajemy długości adresów występujących w procesorze 80286, w którym po raz pierwszy pojawiły się sprzętowe mechanizmy wspomagające obsługę pamięci wirtualnej):

1. Program żąda dostępu do określonej informacji, podając (długi, 32-bitowy) adres wirtualny.
2. Sprawdzana jest obecność poszukiwanej informacji w pamięci operacyjnej. Informacja o obecności konkretnych bloków w pamięci operacyjnej przechowywana jest w specjalnej tablicy.
3. W przypadku braku poszukiwanej informacji jest ona wczytywana z dysku, a odpowiednie pozycje w tablicach obsługujących pamięć wirtualną są modyfikowane.
4. Obliczany jest adres fizyczny miejsca przechowywania informacji w pamięci operacyjnej, czyli dokonywana jest translacja adresu wirtualnego na fizyczny (krótki, 24-bitowy). Translacji tej dokonuje się także przy użyciu odpowiedniej tablicy.
5. Poszukiwana informacja jest dostępna dla procesora, co zamyka cykl działania pamięci wirtualnej.

Translacja adresu wirtualnego na fizyczny oraz sprawdzenie obecności poszukiwanej informacji są realizowane za pomocą specjalnej tablicy przechowywanej w pa-

mięci operacyjnej, zwanej tablicą deskryptorów. Mechanizm jej działania ilustruje rysunek 3.29.

Przykładowe wartości użyte na rysunku 3.29 zostały utworzone na potrzeby tego przykładu, w celu jaśniejszego pokazania działania mechanizmu pamięci wirtualnej. Długości adresów i wielkości pamięci dla procesorów opartych na architekturze IA 32 (80386 do Pentium) podane są w punkcie 4.5.1.6.



Rysunek 3.29. Translacja adresu wirtualnego na fizyczny

Załóżmy, ze pojemność pamięci przydzielona jednemu zadaniu wynosi 8 MB, co oznacza, że zezwalamy na używanie w programie 23-bitowych adresów (2^{23} = 8 M). Pojemność pamięci operacyjnej wynosi 2 MB (pomijamy niewielki dodatek na system operacyjny i tablicę deskryptorów), więc długość adresu fizycznego wynosi 21 bitów. Ponieważ pojemność PAO jest mniejsza niż wielkość programu, tylko część programu będzie do niej załadowana. W naszym przypadku są to 2 MB. Program w całości będzie przechowywany w pamięci masowej, np. na dysku. Program zarówno w PAO, jak i w PM dzielimy na bloki. W naszym przypadku są to bloki o jednakowej wielkości równej 1 MB (jest to pewne uproszczenie w stosunku do rzeczywistej sytuacji, służące jaśniejszemu przedstawieniu zawartości deskryptora). Każdy z bloków ma na dysku unikalny numer, w naszym przypadku jest to liczba 3-bitowa.

W pamięci operacyjnej utworzona jest specjalna tablica, zwana tablicą deskryptorów. Liczba pozycji w tej tablicy musi być równa liczbie bloków w PM, na jakie zosta! podzielony program. Inaczej mówiąc, każdy blok programu musi mieć swój deskryptor. Każdy z deskryptorów składa się z dwóch części: bitu obecności bloku i adresu bazowego (21-bitowego) podającego, w którym miejscu w pamięci operacyjnej został umieszczony dany blok. Bit obecności równy 1 oznacza obecność danego bloku programu w PAO.

W naszym przykładzie do pamięci operacyjnej załadowane są bloki programu o numerach 5 i 3 (numery tych bloków zostały dla większej czytelności rysunku umieszczone zarówno w PM, jak i PAO). W deskryptorach odpowiadającym blokom 3 i 5 bity obecności mają wartość 1. W deskryptorach pozostałych bloków wartość ta wynosi 0. Ponadto w deskryptorze bloku 3 umieszczony jest adres 100000h, co oznacza, że blok 3 rozpoczyna się w pamięci operacyjnej od adresu 100000h. Podobnie dla bloku 5 adres bazowy wynosi 000000h.

Załóżmy teraz, że program odwołuje się do adresu wirtualnego 3FFFFFh. Adres jest dzielony na dwie części: pole selektora i pole przesunięcia. Pole selektora (u nas 3-bitowe) określa, w którym bloku programu znajduje się poszukiwana informacja, czyli wskazuje, którego deskryptora należy użyć podczas translacji adresu wirtualnego na rzeczywisty. Zauważmy, że pole to musi mieć taką długość, aby móc ponumerować wszystkie bloki przydzielone dla jednego programu, czyli aby obsłużyć całą przestrzeń pamięci wirtualnej przydzielonej programowi. W naszym przypadku pole to musi mieć długość 3 bitów, gdyż program może się składać z ośmiu bloków. Przesunięcie określa, jak daleko od początku bloku znajduje się poszukiwana informacja. W naszym przypadku selektor zawiera liczbę 3. Oznacza to, że poszukiwana informacja znajduje się w bloku 3. Kolejność postępowania systemu operacyjnego jest następująca:

1. Sprawdzany jest bit obecności bloku w deskryptorze bloku 3. W naszym przypadku wynosi on 1, czyli blok jest obecny w PAO.
2. Odczytywany jest adres bazowy bloku w PAO.
3. Do odczytanego adresu bazowego dodawane jest przesunięcie pobrane z adresu wirtualnego.
4. Otrzymany 21-bitowy adres jest fizycznym adresem w PAO, pod którym znajduje się poszukiwana informacja.

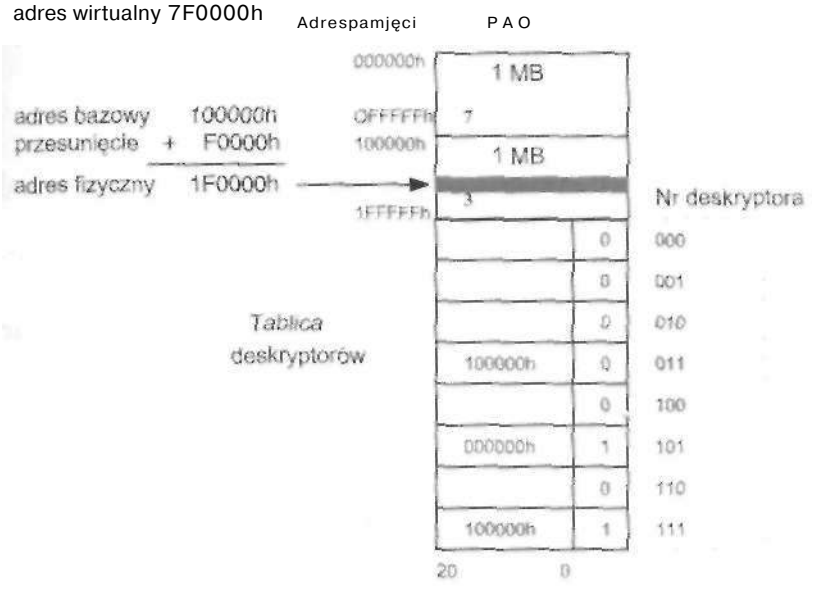
Co stanie się w przypadku, gdy bloku zawierającego poszukiwaną informację nie ma w PAO? Załóżmy tym razem, że adres wirtualny wynosi 7F0000h. Odwołujemy się więc do bloku 7, którego nie ma w pamięci. Wykonane muszą być wtedy dodatkowe czynności.

|a. Blok numer 7 wczytywany jest do pamięci operacyjnej w miejscu jednego z obecnych tam bloków. Załóżmy, że blok 7 umieścimy w miejscu bloku 3.

Ja. Modyfikowane są deskryptory usuniętego i załadowanego bloku.

Tablica deskryptorów po wykonaniu tej operacji będzie wyglądać tak, jak na rysunku 3.30.

'Dalszy ciąg postępowania jest taki sam jak w poprzednim przypadku. Zauważmy, że w deskryptorze bloku 3 wystarczyło zmodyfikować bit obecności. Powtarzający się adres bazowy nie ma znaczenia, gdyż i tak dla nieobecnego bloku nie jest czytany.



Rysunek 3.30, Przykładowa zawartość tablicy deskryptorów

3.7. Koncepcja pamięci podręcznej (cache)

Koncepcja pamięci cache jest podobna do koncepcji pamięci wirtualnej i wynika z własności pamięci SRAM i DRAM (patrz podrozdział 2.3 dotyczący pamięci oraz punkt 3.6.1 - hierarchia pamięci). Przypominamy krótko, że pamięci statyczne są szybsze od pamięci dynamicznych, natomiast pobierają więcej energii i są zdecydowanie droższe. Ponadto są układami o niższym stopniu scalenia. Z tych powodów nie jest możliwe zbudowanie całej pamięci operacyjnej z pamięci statycznych. Z drugiej strony, pamięci dynamiczne są zbyt wolne dla szybkich, współczesnych procesorów i wymagają przy dostępie stanów oczekiwania. Wynika to zarówno z dużej częstotli-

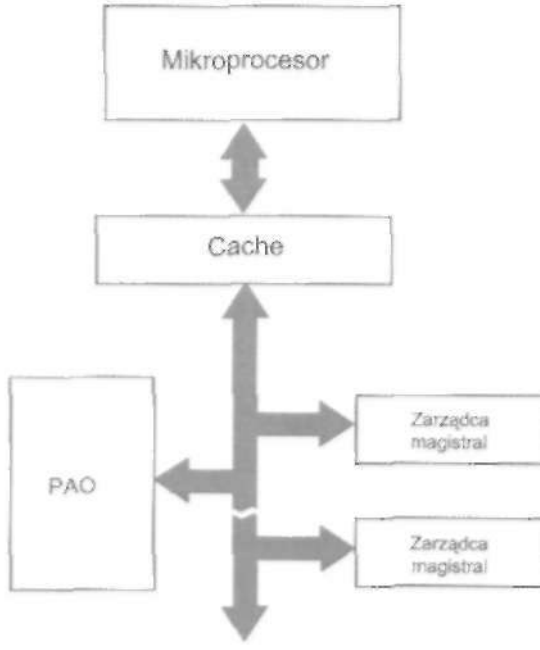
wości zegara taktującego te procesory, jak i na przykład z występującej w procesorach Pentium pracy dwupotokowej czy też dynamicznej realizacji instrukcji. Aby zmniejszyć ten efekt, wprowadzono następującą koncepcje. Pamięć systemu składa się z relatywnie dużej pamięci operacyjnej (rzędu kilkuset MB do pojedynczych GB) zbudowanej z pamięci dynamicznych oraz z mniejszej (kilka do kilkuset kB), lecz znacznie szybszej pamięci cache zbudowanej z pamięci statycznych. Ponadto w systemie musi znajdować się sterownik pamięci cache koordynujący w systemie współpracę pamięci z pozostałymi układami. W przypadku operacji na pamięci sterownik ten sprawdza, czy poszukiwana informacja znajduje się w **pamięci** cache. Jeżeli tak, operacja jest wykonywana na pamięci cache, bez stanów oczekiwania. Sytuacja taka nazywana jest trafieniem (ang. *cache hit)*. W przypadku nieobecności informacji w pamięci cache następuje dostęp do pamięci operacyjnej z koniecznymi stanami oczekiwania. Jest to tak zwane chybienie (ang. *cache miss)*.

3.7.1. Architektura systemu z pamięcią cache

We współczesnych systemach z pamięcią cache występują dwa rodzaje architektury: **Look-through** i **Look-aside.**

3.7.1.1. Architektura Look-through

Sposób podłączenia pamięci cache w przypadku architektury Look-through pokazany jest na rysunku 3.31.

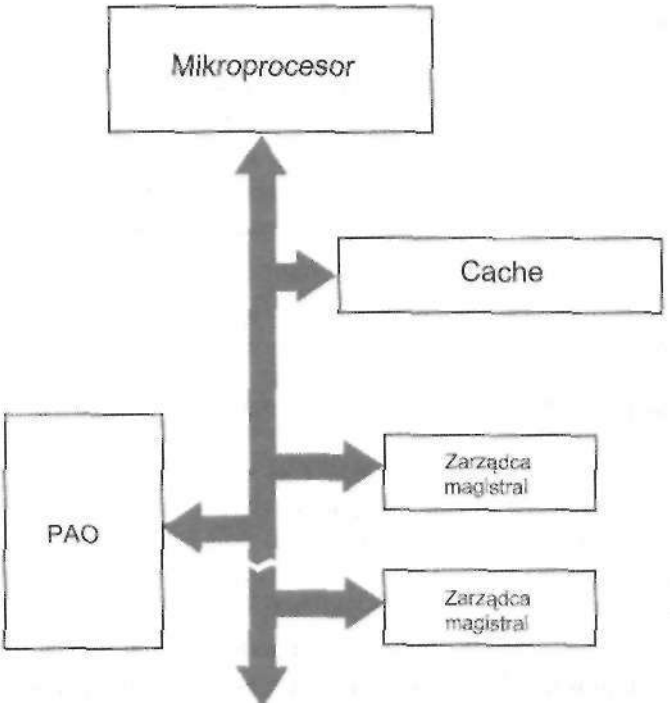


Rysunek 3.31. Architektura Look-through

W architekturze tej pamięć cache, połączona magistralą lokalną z procesorem, oddziela go od reszty systemu. W przypadku trafienia główna magistrala systemu nie jest w ogóle używana, co powoduje dodatkowe zwiększenie szybkości działania systemu. Związane jest to z możliwością korzystania z magistrali głównej przez innych zarządców magistrali (np. sterownik DMA) w trakcie realizacji operacji na pamięci cache. Żądanie dostępu do pamięci jest przekazywane do pamięci głównej (PAO) tylko w wypadku chybienia. Inicjowany jest wówczas cykl magistrali głównej z koniecznymi stanami oczekiwania.

3.7.1.2. Architektura Look-aside

Sposób współpracy pamięci cache z systemem przy zastosowaniu architektury Look-aside przedstawia rysunek 3.32.



Rysunek 3.32. Architektura Look-aside

W przypadku tej architektury procesor nie jest odizolowany od reszty układów przez cache. Dostęp do pamięci powoduje rozpoczęcie normalnego cyklu magistrali. W przypadku trafienia cykl ten jest zawieszany, a operacja jest wykonywana w pamięci cache bez stanów oczekiwania. W przypadku chybienia wykonywany jest normalny cykl magistrali ze stanami oczekiwania.

Przy zastosowaniu architektury Look-aside, nawet w przypadku trafienia, magistrala nie jest dostępna dla innych zarządców magistral. Nie są też możliwe równolegle operacje dla dwóch zarządców magistral. Zaletami architektury Look-aside są natomiast: prostsza konstrukcja, niższy koszt i nieco szybsza realizacja operacii w przypadku chybienia (normalny cykl magistrali jest rozpoczynany od razu po żądaniu dostępu do pamięci).

3.7.2. Elementy systemu pamięci cache

System pamięci cache składa się z trzech elementów:

- banku danych pamięci cache (pamięć danych),
- katalogu pamięci cache (często nazywanego TAG-RAM-em),
- sterownika pamięci cache.

W banku danych pamięci cache przechowywana jest zapisywana i odczytywana informacja, natomiast katalog pamięci cache służy do szybkiego sprawdzania, czy poszukiwana informacja znajduje się w pamięci danych cache (konkretnie, czy dany adres jest odwzorowany w pamięci cache). Stanowi to zadanie sterownika pamięci cache, który ponadto organizuje współpracę pamięci cache z systemem. Jednym z bardzo ważnych elementów tej współpracy jest zapewnienie zgodności (ang. *consistency* lub *coherency)* zawartości pamięci cache z pamięcią główną. Problemowi temu poświęcony jest następny podpunkt.

3.7.3. Sposoby zapewniania zgodności pamięci cache

Niezgodność zawartości pamięci cache z zawartością pamięci głównej występuje w dwóch przypadkach: nastąpił zapis do pamięci cache bez zapisu do pamięci głównej lub nastąpił zapis do pamięci głównej baz zapisu do pamięci cache. Pierwsza sytuacja występuje w przypadku trafienia przy zapisie. Druga występuje na przykład w przypadku transmisji DMA do pamięci głównej (gdy z magistrali głównej korzystał inny zarządca magistral niż procesor). Czasowa niezgodność pamięci jest dopuszczalna, nie wolno natomiast dopuścić do sytuacji, w której zostanie użyta nieaktualna informacja.

W przypadku zapisu do pamięci cache stosowane są następujące strategie utrzymania zgodności:

- *Write-through,*
- *buforowane Write-through,*
- *Write-back.*

W strategii Write-through każdy zapis do pamięci cache powoduje jednocześnie zapis do pamięci głównej. Jest to rozwiązanie bardzo proste i pewne, niestety powo-

dujące zmniejszenie szybkości działania systemu (każdy zapis trafia do „wolnej" pamięci głównej).

Buforowane (lub inaczej opóźnione) Write-through polega na zapisaniu informacji zarówno w przypadku trafienia, jak i chybienia do bufora sterownika cache, przy czym procesor widzi tę operację jako dostęp do pamięci bez stanów oczekiwania. W rzeczywistości operacja zapisu do pamięci głównej jest realizowana później (jest opóźniona). Stany oczekiwania występują dla procesora jedynie w przypadku kolejno po sobie następujących zapisów do pamięci. Pomiędzy zapisem informacji do pamięci cache dostęp do magistrali dla innych zarządców magistrali jest blokowany w celu uniknięcia użycia nieaktualnej informacji (dodatkowy powód tej operacji to potrzeba dostępności magistrali w celu dokonania przepisania informacji z bufora).

W przypadku strategii Write-back zawartos'ci pamięci cache i pamięci głównej są uzgadniane tylko w przypadku takiej potrzeby. Występuje ona, gdy inny zarządca magistrali chce skorzystać z komórki pamięci głównej, która zawiera nieaktualną informację, lub gdy w pamięci cache wymieniana jest linia zawierająca nową informację. Budowa pamięci cache stosującej strategię Write-back jest bardziej skomplikowana, gdyż jak wynika z podanego opisu, należy śledzić operacje magistrali w pamięci głównej dotyczące innych zarządców magistrali. W przypadku operacji w komórkach odwzorowanych w pamięci cache należy przeprowadzić jej uaktualnienie.

Drugi przypadek, w którym może dojść do niezgodności pamięci głównej i pamięci cache, występuje, gdy inny (niż procesor główny) zarządca magistrali dokonuje zapisu (lub. dla pamięci cache ze strategią Write-back, także odczytu) do pamięci. Najczęściej używaną metodą utrzymania zgodności w takim przypadku jest śledzenie przez sterownik cache operacji magistrali (ang. *bus snooping)*. Reakcja systemu pamięci cache zależy od strategii stosowanej do utrzymania zgodności przy zapisie informacji przez procesor. W przypadku strategii Write-through sterownik pamięci cache może unieważnić linię odwzorowującą w niej modyfikowaną komórkę pamięci. Powoduje to w przypadku odwołania się procesora do tej informacji w pamięci cache chybienie i w konsekwencji jej odczyt z pamięci głównej.

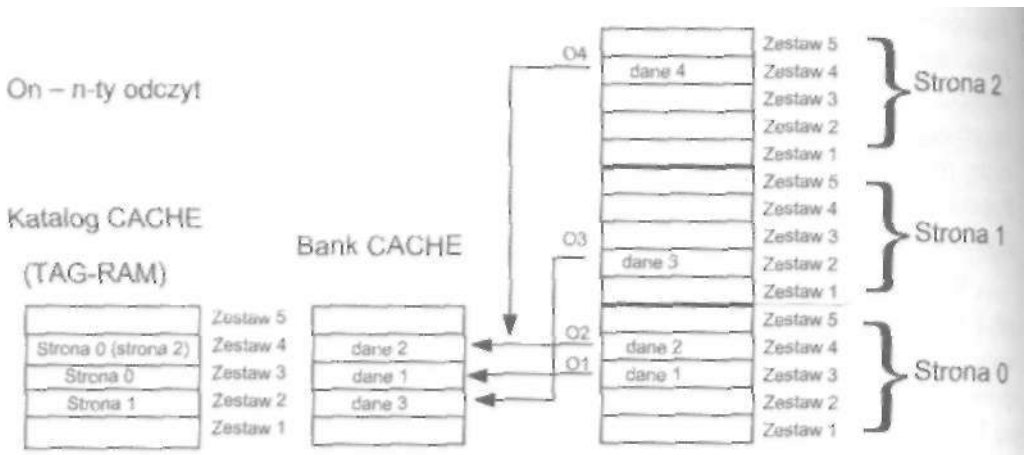
W przypadku strategii Write-back proste unieważnienie linii nie jest możliwe, gdyż może ona zawierać informacje wpisane do cache przez procesor, a nie odwzorowane w pamięci głównej, które ulegną straceniu. Sterownik pamięci cache musi więc spowodować wstrzymanie na pewien czas zapisu przez zarządcę magistrali, przepisanie linii do pamięci głównej i dopiero wtedy jej unieważnienie. Po tej operacji zarządca magistrali może wznowić operację zapisu.

Trzecia możliwość występuje zarówno dla architektury Write-through, jak i Write-back. W przypadku wykrycia zapisu do komórki, która jest odwzorowywana w pamięci cache, informacja jest przechwytywana przez sterownik pamięci cache i jednocześnie do niej zapisywana.

W przypadku stosowania strategii Write-back sterownik pamięci cache musi śledzić także operacje odczytu. Jeżeli zarządca magistrali próbuje odczytać komórkę pamięci głównej, która jest odwzorowana w pamięci cache i była zmieniana, sterownik pamięci cache musi wstrzymać cykl magistrali, uaktualnić pamięć główną i dopiero wtedy zezwolić na odczyt.

3.7.4. Organizacja pamięci,cache

W celu zapewnienia możliwości szybkiego sprawdzenia, czy komórka pamięci w której ma być wykonana operacja, jest odwzorowana w pamięci cache, dwie części pamięci cache, bank danych i katalog, tworzą tak zwaną jednoblokową pamięć asocjacyjną (ang. *One-way Set-associative* lub *Direct-mapped)*. W takiej organizacji pamięć cache stanowi jeden blok, który jest dzielony na *zestawy*. Pamięć główna dzielona jest na strony o rozmiarze równym rozmiarowi bloku pamięci cache. Strony są z kolei również dzielone na zestawy, przy czym liczba zestawów w stronie jest identyczna jak w bloku pamięci cache. Każdy zestaw w pamięci cache ma swoją pozycję w katalogu (TAG-RAM). Zawiera ona adres strony, z którego pochodzi dany zestaw. Każdy zestaw jest wpisywany na miejsce w pamięci cache do zestawu o numerze zgodnym z numerem zestawu w stronie. Umożliwia to bardzo szybkie sprawdzenie obecności zestawu - nie musimy bowiem przeszukiwać katalogu, lecz sprawdzamy adres bloku w określonej pozycji katalogu. Interpretacja graficzna struktury jednoblokowej pamieci asocjacyjnej cache oraz jej relacje z pamięcią główną przedstawia rysunek 3.33.



Rysunek 3.33. Przykład organizacji pamięci cache

W przypadku wymiany zestawu na danej pozycji modyfikowany jest adres bloku w pozycji katalogu odpowiadającej temu zestawowi. Przykładowo na rysunku 3.33 w wyniku odczytu odpowiednich komórek w pamięci cache są odwzorowane zestawy 4 i 5 ze strony 0 oraz zestaw dwa ze strony 1. Jeżeli teraz nastąpi odwołanie do zesta-

wu 4 na stronie 2, to zestaw ten zostanie wpisany w banku pamięci cache na miejsce zestawu 4 ze strony 0. Jednocześnie w pozycji katalogu cache odpowiadającej danemu zestawowi w miejsce strony 0 zostanie wpisana strona 2.

Jeżeli bank danych pamięci cache dzielimy na dwa lub cztery bloki, mówimy odpowiednio o dwu- lub czteroblokowej pamięci asocjacyjnej (ang. *two-ways set-associative memory lub four-ways set-associative memory)*.

Podsumowanie

W rozdziale 3. przedstawiliśmy podstawowe informacje o budowie i działaniu komputera. Informacje te nie były praktycznie związane z określonym systemem, choć w pewnych przykładach występowały konkretne układy. Informacje dotyczące komputerów typu IBM PC, na przykład budowa procesorów rodziny Intel x86 czy architektura systemu ISA, są przedstawione w następnych rozdziałach.

4. Procesory

Wstęp

Rodzina procesorów Intel x86 obejmuje procesory Intel 8086, 8088, 80186, 80188, 80286, 80386, 80486 oraz rodzinę Pentium™. Jej dalszym rozwinięciem jest procesor Itanium. Układy 80186 i 80188 są mikrokontrolerami, czyli w układzie scalonym oprócz procesora zawierają dodatkowe elementy, takie jak układy wejścia/wyjścia (układ przerwań, DMA) czy blok liczników. Układy te przeznaczone są głównie do zastosowań w układach automatyki i nie należą do zakresu naszego zainteresowania.

Wprowadzając nowe produkty z rodziny x86, firma Intel konsekwentnie stosuje zasadę kompatybilności wstecz. Oznacza to stosowanie takich rozwiązań w nowszych procesorach, które pozwolą im bez przeszkód wykonywać programy napisane dla procesorów starszych. Zaletą takiej polityki jest brak konieczności wymiany całego oprogramowania przy zmianie procesora. Wadą jednak jest wpływ architektury procesorów starszych na nowsze lub, inaczej mówiąc, brak możliwości radykalnych zmian w lej architekturze. Przykładowo, sposób restartu wszystkich procesorów rodziny x86 jest taki sam. Każdy z procesorów tej rodziny potrafi w trybie rzeczywistym pracować tak jak procesor 8086/88.

W rozdziale omawiamy rozwój tej rodziny, poczynając od procesora 8086 będącego protoplastą rodu, a kończąc na procesorze Pentium 4. Podejście takie pozwoli rozpocząć omawianie architektury tej rodziny od procesorów względnie prostych, aby w dalszej kolejności omawiać procesory coraz bardziej skomplikowane, przez prezentowanie nowych rozwiązań, które zostały w nich zastosowane. Pozwala to na stosunkowo proste przedstawienie rozbudowanych rozwiązań procesorów.

Poniżej omawiamy rozwój rodziny procesorów Intel x86.

4.1. Parametry wybranych procesorów

Mikroprocesor to najistotniejszy element komputera (który jest, jak powiedziano, systemem mikroprocesorowym), decydujący w znacznej mierze o jego możliwościach. Jest głównym elementem, który przetwarza informację. Szybkość przetwarzania informacji przez procesor, zwana bardzo często jego mocą obliczeniową, zależy przede wszystkim od dwóch czynników.

Tabela 4.1. Zestawienie parametrów wybranych procesorów rodziny x86

Procesor	Data powstania	Architektura	Częstotliwość zegara[1]	Liczba tranzystorów	Rozmiar rejestrów	Przestrzeń adresowa	Pamięć cache
8086	1978	prefetching	8 MHz	29 tys.	16 GP[2]	1 MB	brak
80286	1982	pipelining	12,5 MHz	134 tys.	16 GP	16 MB	brak
80386	1985	pipelining	20 MHz	275 tys.	32 GP	4 GB	zewnętrzny
80486	1989	pipelining	25 MHz	1,2 mln	32 GP 80 FPU[3]	4 GB	L1: 8 KB
Pentium	1993	pipelining	60 MHz	3,1 mln	32 GP 80 FPU	4 GB	L1: 16 KB
Pentium Pro	1995	Dynamic Execution Microarchitecture	200 MHz	5,5 mln	32 GP 80 FPU	64 GB	L1: 16 KB L2: 256 KB lub 512 KB lub 1 MB
Pentium II	1997	Dynamic Execution Microarchitecture	266 MHz	7 mln	32 GP 80 FPU 64 MMX	64 GB	L1: 16 KB L2: 512 KB
Pentium III	1999	Dynamic Execution Microarchitecture	500 MHz	8,2 mln	32 GP 80 FPU 64 MMX	64 GB	L1: 16 KB L2: 512 KB
Intel Pentium 4 Processor Supporting Hyper-Threading Technology at 90 nm	2004	Intel NetBurst Microarchitecture: Hyper-Threading Technology	3,4 GHz	125 mln	32 GP 80 FPU 64 MMX 128 XMM	64 GB	12 K µops ETC L1: 16 KB L2: 1 MB

Procesor	Data powstania	Architektura	Częstotliwość zegara[1]	Liczba tranzystorów	Rozmiar rejestrów	Przestrzeń adresowa	Pamięć cache
64-bit Intel Xeon Processor with 800 MHz System Bus	2004	*Intel NetBurst* Microarchitecture; Hyper-Threading Technology; Intel Extended Memory 64 Technology	6,6 GHz	125 mln	32, 64 GP 80 FPU 64 MMX 128 XMM	64 GB	12 K µops ETC L1: 16 KB L2: 1 MB L3: 2 MB
64-bit Intel Xeon Processor MP with 8MB L3	2005	Intel NetBurst Microarchitecture; Hyper-Threading Technology; Intel Extended Memory 64 Technology	3,33 GHz	675 mln	32, 64 GP 80 FPU 64 MMX 128 XMM	1 TB	12 K µops ETC L1: 16 KB L2: 1 MB L3: 8 MB
Intel Pentium Processor Extreme Edition 840	2005	Intel NetBurst Microarchitecture; Hyper-Threading Technology; Intel Extended Memory 64 Technology; Dual-core	3,2 GHz	230 mln	32, 64 GP 80 FPU 64 MMX 128 XMM	64 GB	12 K µops ETC L1: 16 KB L2: 2x1 MB
Intel Core Duo Processor T2600	2006	Improved Intel Pentium M Processor; Dual Core; *Intel Smart Cache*, Advanced Thermal Manager	2,16 GHz	152 mln	*32 GP* 80 FPU 64 MMX 128 XMM	*4 GB*	*L1: 16 KB* L2: 2 MB

Pierwszym z nich jest maksymalna długość argumentów, na których potrafi operować mikroprocesor. Długość ta decyduje jednocześnie o tym, do jakiej grupy procesorów należy dany procesor: 8-, 16-, 32-bitowych itd. Decydująca jest tu właśnie maksymalna długość argumentów, nie zaś na przykład szerokość magistrali danych. Rodzina x86 zmieniała się pod tym względem. Procesory 8086 i 80286 były 16-bitowe. Od 80386 do Pentium włącznie (trochę kłopotu sprawiają tu późniejsze wersje Pentium 4) są to procesory 32-bitowe, Wreszcie procesor Itanium, co do którego pojawiają się wątpliwości, czy należy go zaliczyć do rodziny x86 (co w tej książce jednak czynimy), jest 64-bitowy.

Drugim czynnikiem decydującym o szybkości przetwarzania informacji, czyli o mocy obliczeniowej mikroprocesora, jest szybkość wykonywanie instrukcji. Zależy ona między innymi od częstotliwości zegara taktującego pracę mikroprocesora, ale nie tylko. Niebagatelne znaczenie ma tu architektura procesora i, co jest z tym związane, sposób wykonywania instrukcji. Prefetching, praca potokowa, spekulatywne wykonywanie instrukcji programu są przykładami rozwiązań przyspieszających realizację instrukcji. Rozwiązania te prezentujemy w dalszej części rozdziału.

Zwiększone możliwości procesora okupowane są wzrostem komplikacji jego układów. W tabeli 4.1 zmieszczono podstawowe cechy wybranych procesorów rodziny Intel x86. Tabela pokazuje kierunek zmian i rozwoju tych procesorów.

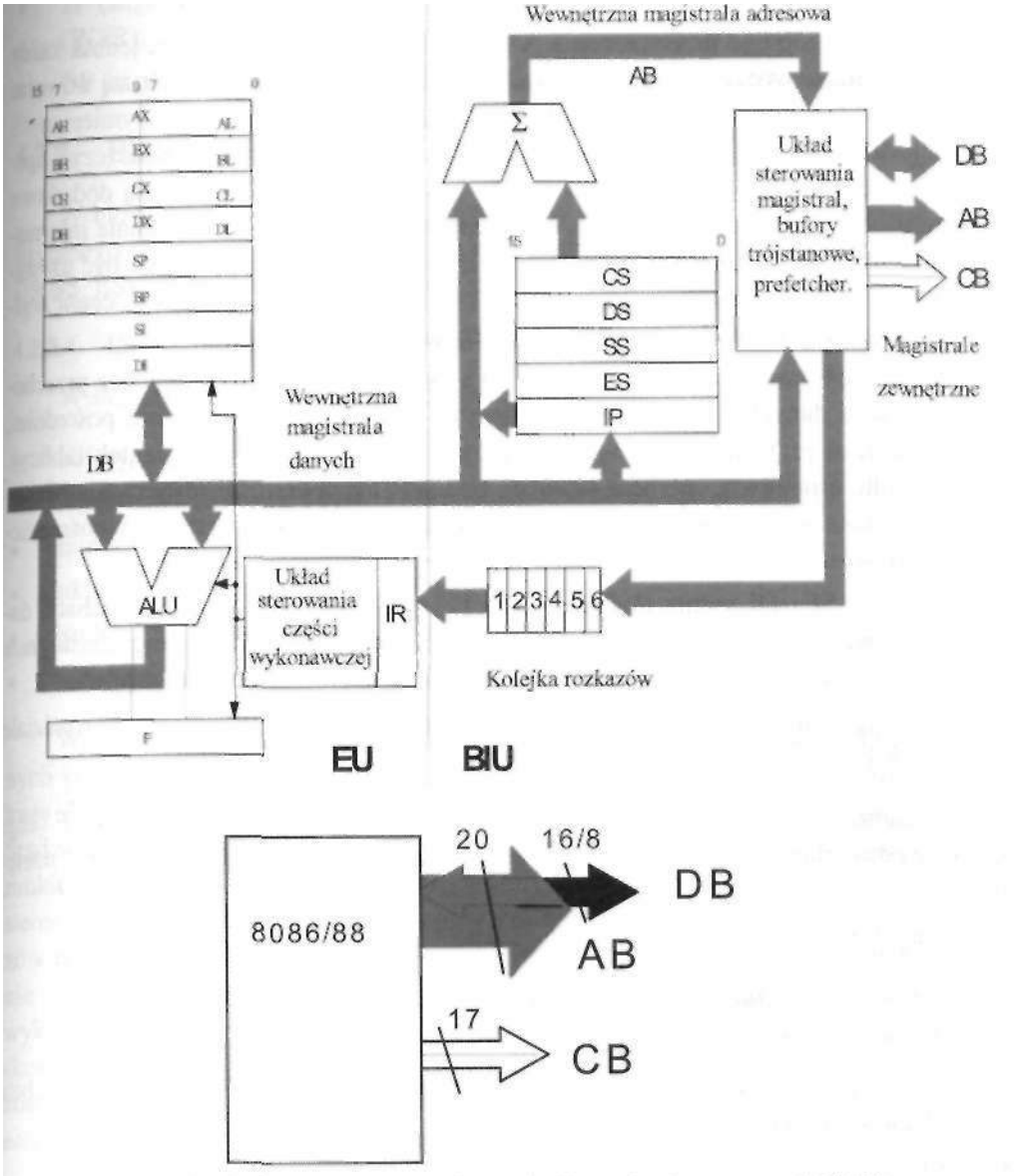
4.2. Procesor 8086/88

Mikroprocesor Intel 8086 jest procesorem w pełni 16-bitowym. Ma 16-bitową jednostkę arytmetyczno-logiczną oraz taką samą magistralę danych. Mikroprocesot 8088 różni się od niego głównie zewnętrzną 8-bitową magistralą danych (istnieją także drobne różnice wewnętrzne), przy czym wewnątrz jest także procesorem 16-bitowym. Magistrala danych dla obu procesorów jest 20-bitowa, co pozwala na zaadresowanie 1 MB pamięci (przy założeniu długości komórki pamięci równej jednemu bajtowi). Magistrala danych i adresowa są **multipleksowane,** co oznacza, że część linii magistrali adresowej jest jednocześnie liniami magistrali danych i tymi samymi liniami są przesyłane na zmianę zarówno adresy, jak i dane. Wymaga to oczywiście sygnałów sterujących mówiących, jaka informacja jest aktualnie przesyłana wspólnymi liniami.

Szerokość zewnętrznych magistral oraz schemat blokowy mikroprocesorów 8086/88 przedstawia rysunek 4.1. Na rysunku tym przedstawione są także rejestry dostępne programowo tych procesorów.

Układy mikroprocesora 8086/88 możemy podzielić na część wykonawczą EU (ang. *execution unit)* i jednostkę sterowania magistralami BIU (ang. *bus interface*

„„„;). Jednostki te mogą pracować równolegle, co umożliwia realizacje wstępnego pobierania instrukcji, czyli prefetchingu opisanego w podrozdziale 3.3.3.



Rysunek 4.1. Schemat blokowy i szerokość magistral procesora 8086/88

4.2.1. Część wykonawcza

Część wykonawcza zawiera 16-bitową jednostkę arytmetyczno-logiczną ALU I i zestaw współpracujących z nią rejestrów, które tworzą następujące grupy:

- Rejestry ogólnego przeznaczenia AX, BX, CX, DX. Są 16-bitowe, jednak każdy z nich może być używany jako dwa oddzielne rejestry 8-bitowe. Noszą wówczas I przykładowo oznaczenia AH, AL, BH, BL i tak dalej. Każdy z wymienionych rejestrów może zawierać dane, na których wykonujemy obliczenia (czyli operandy), oraz wyniki obliczeń. Ponadto poszczególne rejestry pełnią dodatkowe funkcje. AX jest akumulatorem - pośredniczy na przykład w wymianie informacji z układami wejs'cia/wyjścia. Rejestr BX (ang. *base register)* może być używany jako rejestr bazowy w adresowaniu pośrednim, a rejestr CX (ang. *count register)* może pełnić rolę licznika w instrukcjach pętli .
- Rejestr BP zwany wskaźnikiem bazy (ang. *base pointer)* oprócz przechowywania danych i wyników może być używany przy adresowaniu pośrednim, zwłaszcza przy obsłudze tablic (wskazywać może wówczas początek tablicy), Ponadto umożliwia operacje na stosie bez zmiany zawartości rejestru SP. Możliwość taka jest przykładowo wykorzystywana do przekazywania za pośrednictwem stosu argumentów do funkcji w języku C/C++ czy w Pascalu.
- Rejestry SI i DI pełnią dodatkowe funkcje przy operacjach na łańcuchach danych. Rejestr SI (ang. *source index)* zawiera adres źródła, a DI (ang. *destination index)* zawiera adres docelowy dla danych przy operacjach łańcuchowych.
- SP (ang. *stack pointer)* jest wskaźnikiem stosu, którego role opisano w rozdziale 3.3.2.4.

Ponadto z jednostką arytmetyczno-logiczną współpracuje rejestr flagowy F (ang, *flags)*. Zestaw flag, który możemy podzielić na flagi stanu i flagi kontrolne, jest następujący:

Flagi stanu:

- CF (ang. *curry flag)* - przeniesienie/pożyczka,
- PF (ang. *parityflag)* - parzystość,
- AF (ang. *uuxiliury curry flag)* - pomocnicze przeniesienie/pożyczka z 4. bitu (dla kodu BCD),
- ZF (ang. *zero flag)* - flaga wyniku zerowego,
- SF (ang. *signflag)* - flaga znaku.
- OF (ang. *overflow flag)* - flaga przepełnienia (przekroczenie zakresu w kodzie U2).

Flagi kontrolne:

, TF (ang. *trap flag)* - flaga pracy krokowej (pułapka),
- IF (ang. *interrupt enable flag)* - flaga zezwolenia na przyjmowanie przerwań (INTR),
- DF (ang. *direction flag)* - flaga kierunku wykonywania operacji łańcuchowych.

4.2.2. Blok sterowania magistralami

W bloku sterowania magistralami możemy wyróżnić układ generowania adresu fizycznego wraz ze współpracującym z nim zestawem rejestrów segmentowych (CS, DS, SS, ES) oraz układ sterowania magistralami.

4.2.2.1- Układ sterowania magistralami

W skład układu sterowania magistralami wchodzą:

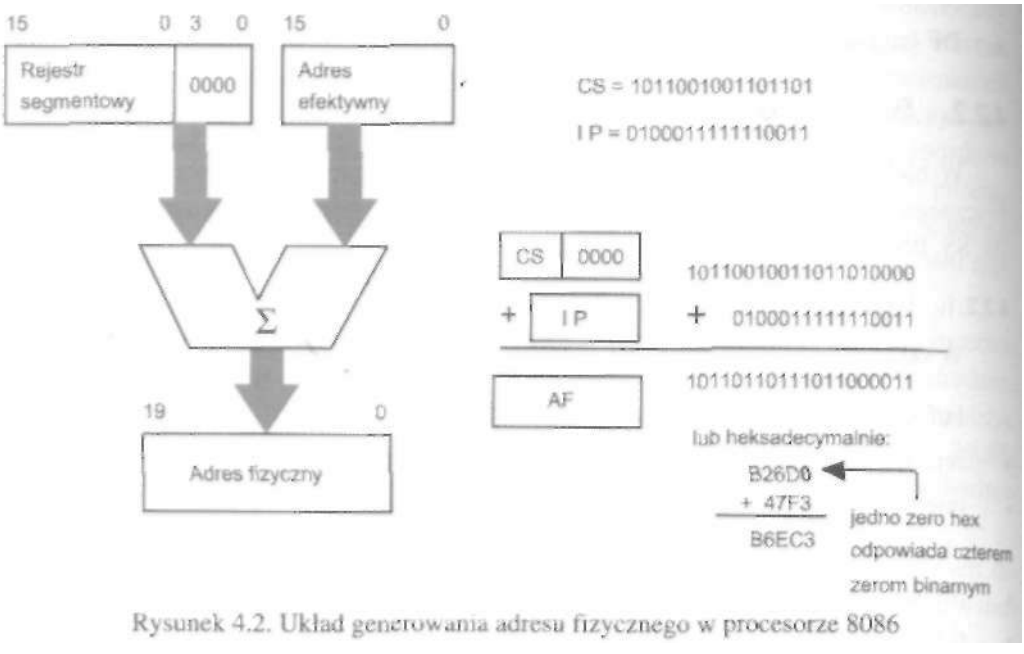
. bufory trójstanowe danych,
- rejestry zatrzaskowe i układy wyjściowe magistrali adresowej,
- interfejs koprocesora,
- układy logiczne sterowania cyklami magistrali,
- prefetcher (układ wstępnego pobierania kodów instrukcji),
- 6-bajtowa (dla procesora 8088 4-bajtowa) kolejka rozkazów.

Większa część programu tworzona jest przez instrukcje umieszczone w kolejnych komórkach pamięci i w takiej kolejności wykonywanych. Wyjątkiem są tu rozkazy skoków i wywołań podprogramów, które zdarzają się jednak stosunkowo rzadko. Stąd w celu przyspieszenia realizacji programu prefetcher *(czyt.* prefeczer), pod warunkiem niezajętości magistral, pobiera kody instrukcji z kolejnych, leżących obok siebie komórek pamięci i umieszcza je w kolejce rozkazów. Z kolejki tej kody rozkazów pobierane są przez układ sterowania części wykonawczej. Dzięki temu procesor nie musi inicjować cyklu magistral w celu pobrania kodu kolejnego rozkazu, a układ wykonawczy nie musi oczekiwać na jego wczytanie. Jedynie w przypadku instrukcji skoku lub wywołania podprogramu (czy też powrotu z niego) kolejka rozkazów musi zostać wyczyszczona i kody rozkazów są pobierane od nowa (od miejsca, do którego nastąpił skok).

4.2.2.2. Układ generacji adresu fizycznego

Sposób generowania adresu fizycznego przez procesor 8086/88 jest bardzo ważny, gdyż w kolejnych procesorach tej rodziny jest używany do generowania adresu w trybie rzeczywistym, a z kolei wszystkie procesory rodziny x86 po restarcie rozpoczynają pracę właśnie w tym trybie.

Jak łatwo zauważyć, wszystkie rejestry procesorów 8086/88 są 16-bitowe i taką szerokość ma też magistrala danych. Natomiast magistrala adresowa jest 20-bitowa Wymaga to układu, który na podstawie 16-bitowych wartos'ci pozwoliłby wygenerować 20-bitowy adres. Układ taki przedstawiony jest na rysunku 4.2



Rysunek 4.2. Układ generowania adresu fizycznego w procesorze 8086

20-bitowy adres fizyczny obliczany jest jako suma dwóch składników: zawartości jednego z rejestrów segmentowych przemnożonej przez 16 (inaczej: do której dopisano cztery zera w zapisie binarnym lub jedno zero w zapisie heksadecymalnym) oraz tak zwanego **adresu efektywnego** wynikającego z kodu aktualnie wykonywanego rozkazu i używanego w nim trybu adresowania. Zestaw rejestrów segmentowych jest następujący:

CS - rejestr segmentu programu,

DS - rejestr segmentu danych,

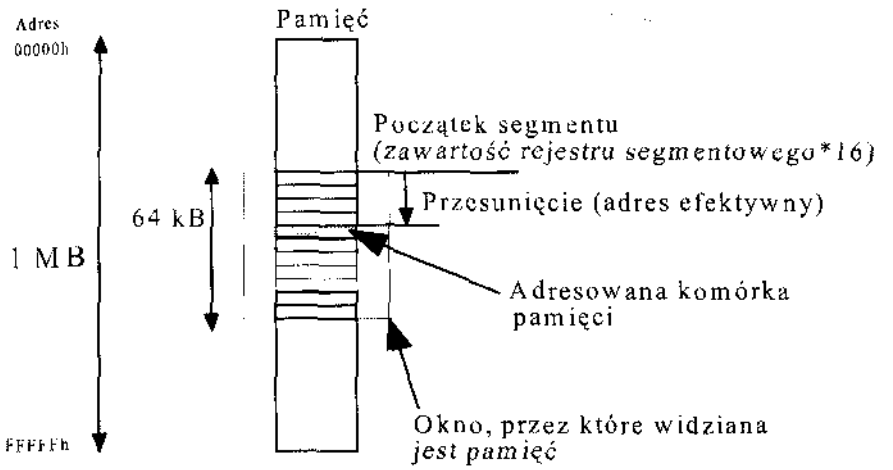
SS - rejestr segmentu stosu,

ES - rejestr dodatkowego segmentu danych.

Widzimy więc, że każdy program może mieć cztery rodzaje segmentów. Jak zobaczymy dalej, segmenty te mogą być rozłączne lub mogą się częściowo lub całkowicie pokrywać.

Interpretacja takiego sposobu generowania adresu fizycznego jest prosta i została zilustrowana na rysunku 4.3. Zawartość rejestru segmentowego przemnożona przez 16 podaje nam adres początku danego segmentu w pamięci. Od tego początku odsuwamy

się o liczbę komórek podaną w **adresie efektywnym.** Ponieważ adres efektywny jest liczbą 16-bitową, maksymalna odległość odczytywanej komórki od początku segmentu wynosi 65536 B (przy założeniu bajtowej długości komórek), czyli 64 kB. Pamięć jest więc widziana przez procesor przez 64-kilobajtowe okno. Okno to można przesuwać, przeładowując rejestry segmentowe, jednak ze skokiem nie mniejszym niż 16 B (bo zawartość rejestru segmentowego mnożymy przez 16). W ramach ustalonego 'okna komórkę, na której zostanie wykonana operacja, wybieramy za pomocą adresu efektywnego.



Rysunek 4.3. Interpretacja sposobu adresowania pamięci przez procesor 8086

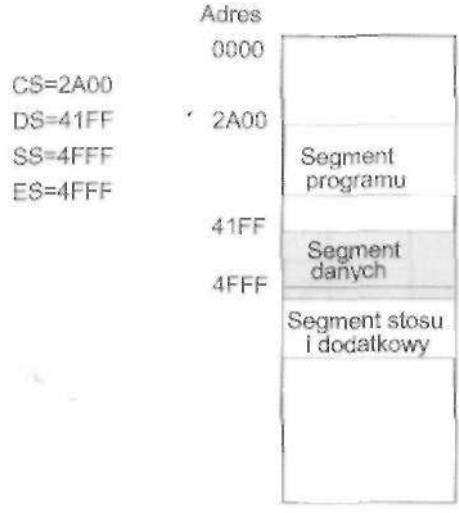
Wynikiem sumowania dwóch podanych wartości jest liczba 20-bitowa (bu du 16-bitowej wartości w rejestrze segmentowym dopisujemy cztery zera) będąca adresem fizycznym żądanej komórki. Liczba bitów adresu fizycznego jest więc zgodna *i* szerokością magistrali adresowej. Przypominamy, że 20-bitowy adres pozwala zaadresować 1 MB pamięci.

Z metody generacji adresu fizycznego wynika też sposób podawania adresu określany jako **segment:przesuniecie,** stosowany w wielu programach i publikacjach. Adres fizyczny podajemy jako dwie liczby, które należy zsumować zgodnie z regułą tworzenia adresu fizycznego. Metoda ta wprowadza pewną niejednoznaczność, gdyż ten sam adres można zapisać na wiele sposobów. Przykładowo adres B6EC3h można zapisać jako B26D:47F3 lub A76D:F7F3, gdyż:

$$B26D0h + 47F3h = A76D0h + F7F3h = B6EC3h$$

Niejednoznaczność ta nie prowadzi do problemów, gdyż po pierwsze otrzymywany adres fizyczny jest w każdym przypadku taki sam, a po drugie, w realnych sytuacjach w adresie segment:przesunięcie podajemy zwykle rzeczywistą zawartość rejestru segmentowego, co jednoznacznie określa także przesunięcie.

Do rejestrów segmentowych mogą być ładowane dowolne wartości, co pozwala na różnorodne ustawienie segmentów względem siebie. Przykładowe rozmieszczenie wszystkich czterech rodzajów segmentów w pamięci przedstawia rysunek 4.4.



Rysunek 4.4. Przykładowe położenie segmentów

Jak widzimy, segment programu jest położony oddzielnie, segment danych oraz segment stosu i dodatkowy segment danych są na siebie częściowo nałożone, natomiast segment stosu i dodatkowy segment danych pokrywają się.

O tym, który rejestr segmentowy zostanie użyty do obliczenia adresu fizycznego, decydują określone reguły umieszczone w tabeli 4.2. Pole zacieniowane oznacza, że dla tych rejestrów możemy zmienić domyślny rejestr segmentowy, używając odpowiedniej dyrektywy asemblera.

Tabela 4.2. Reguły używania rejestrów segmentowych

Nazwa rejestru	Domyślny rejestr segmentowy
IP	**zawsze CS**
SP	**zawsze SS**
BP	**SS**
pozostałe rejestry	**DS**
DI dla operacji łańcuchowych	**ES**

4.2.3. Restart procesora 8086/88

Jednym z wejść magistrali sterującej mikroprocesora 8086/88 jest RESET. Aktywny sygnał na tym wejściu powoduje wpisanie wartości początkowych do określonych rejestrów procesora i rozpoczęcie wykonywania programu od określonego, zawsze tego samego miejsca pamięci (czyli od określonego adresu). W przypadku mikroprocesora 8086/88 nazwa sygnału RESET jest o tyle niefortunna, że sugeruje mas do rejestrów wartości zerowych, co nie w każdym przypadku jest prawdą. Restart procesora 8086/88 powoduje wpisanie do rejestrów wartości początkowych podanych w tabeli 4.3.

Tabela 4.3. Zawartość rejestrów po restarcie procesora 8086

Nazwa rejestru	Wartość początkowa
F	0002h
IP	FFFOh
CS	FOOOh
SS	OOOOh
DS	OOOOh
ES	OOOOh

Jedną z bardzo ważnych konsekwencji takiego ustalenia wartości początkowych wpisywanych do rejestrów jest adres miejsca w pamięci, z którego mikroprocesor pobierze pierwszą instrukcję do wykonania (czyli miejsca, od którego rozpocznie pracę). Zgodnie z podaną regułą obliczania adresu wyniesie on:

```
CS = F000h      F0000h
IP =  FFF0h   + FFF0h
              ---------
              AF = FFFF0h
```

Oznacza to, że procesor pobierze pierwszą instrukcję z komórki pamięci odległej o 16 bajtów od końca pierwszego megabajtu pamięci (adres FFFFFh - 1 MB, FFFFOh - start pracy procesora). Fakt ten warty jest zapamiętania, gdyż ma duże znaczenie dla architektury komputerów typu IBM/PC. Powrócimy do niego w podrozdziale 6.2.3.

4.3. Procesor Intel 80286

Następcą procesorów 8086/88 był 80286. Nowe rozwiązania zastosowane w tym procesorze, choć niepozbawione wad, były znacznym jakościowym krokiem naprzód.

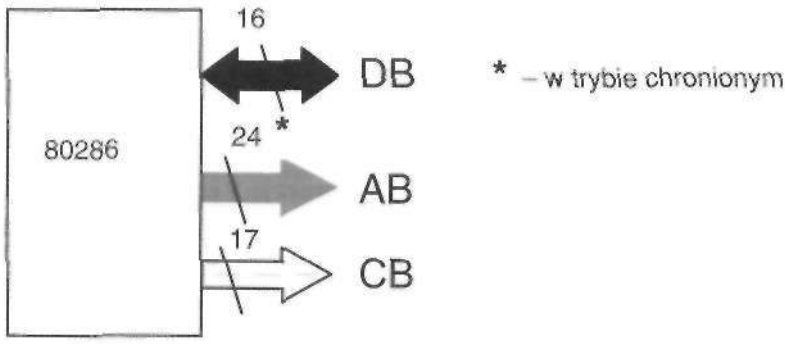
W mikroprocesorze tym pojawiły się mechanizmy sprzętowe ułatwiające realizacje pracy wielozadaniowej oraz pamięci wirtualnej. Obydwa pojęcia wiążą się z projektowaniem nowoczesnych systemów operacyjnych (patrz też rozdział 5.) i tam też głównie znalazły zastosowanie. Z drugiej strony zapewniono kompatybilność tego procesora z procesorem 8086/88.

Podstawowe własności procesora Intel 80286

Procesor 80286 może pracować w jednym z dwóch trybów:

- **Trybie rzeczywistym** (ang. *real mode)* - w trybie tym zachowuje się jak szybki procesor 8086. Między innymi używa tylko 20 bitów adresu (co pozwala zaadresować jedynie **1 MB pamięci).**
- **Chronionym trybie wirtualnym** (ang. *protected virtual mode),* zwanym dalej **trybem chronionym** lub trybem wirtualnym. W trybie tym procesor wykorzystuje swoje pełne możliwości. Używa 24 bitów adresu, co pozwala zaadresować 16 MB fizycznej pamięci. Ponadto dostępne są sprzętowe mechanizmy wspomagające obsługę pamięci wirtualnej, pracy wielozadaniowej i ochrony zasobów (ostatnie dwa mechanizmy opisujemy dokładniej w punktach 3.6.2 i 4.5.1.7), Procesor 80286 po restarcie rozpoczyna pracę w trybie rzeczywistym. Przełączenie procesora na tryb chroniony następuje po ustawieniu bitu PE (ang. *protect enable)* w rejestrze **MSW** (ang. *machine status word)* znajdującym się w zespole rejestrów sterujących w BIU. Bardziej szczegółowe informacje o trybie chronionym podamy w dalszej cześ'ci tego rozdziału.

Szerokość magistral zewnętrznych tego procesora oraz jego uproszczony schemat blokowy przedstawione są na rysunku 4.5.



Rysunek 4.5a. Szerokość magistral procesora 80286

Rysunek 4.5b. Schemat blokowy procesora 80286

Procesor 80286 ma 16-bitową magistralę danych, 24-bitową magistralę adresową oraz 17 sygnałów sterujących tworzących magistrale sterującą. Magistrala danych i adresowa są rozdzielone. Wielkos'ć pamięci, którą może zaadresować procesor 80286, zależy od trybu, w jakim pracuje, i wynosi 1 MB dla trybu rzeczywistego i 16 MB dla trybu chronionego.

Układy procesora 80286 możemy podzielić na cztery równolegle pracujące jednostki: jednostkę adresową AU, jednostkę sterowania magistralami BIU, jednostkę dekodowania instrukcji IU i jednostkę wykonawczą EU. Ponadto z jednostkami AU i BIU w trybie wirtualnym współpracuje jednostka zarządzania pamięcią MMU. Podział układów procesora na wyżej wymienione jednostki pracujące równolegle umożliwia między innymi tak zwaną pracę **potokową** (ang. *pipelining)* będącą rozwinięciem prefetchingu.

W skład jednostki wykonawczej wchodzi 16-bitowa jednostka arytmetyczno-logiczna oraz zestaw rejestrów. Zestaw ten nie różni się od tego z procesora 8086/88. Jedynym wyjątkiem jest rejestr flagowy, w którym pojawiły się dodatkowe bity: **IOPL** i **NT.** Dwa bity pola IOPL podają wymagany poziom uprzywilejowania dla dostępu do określonego układu wejs'cia/wyjs'cia, przy pracy w trybie chronionym. Bit NT jest flagą zadania zagnieżdżonego. Jest ona ustawiana w trybie chronionym, gdy aktualnie wykonywane zadanie zostało uruchomione w wyniku zgłoszenia przerwania lub wykonania instrukcji CALL w trakcie wykonywania innego zadania.

Jednostka adresowa służy do generowania adresu fizycznego. W trybie rzeczywistym adres ten jest generowany identycznie jak dla procesora 8086, za pomocą takiego samego zestawu rejestrów segmentowych (pewne odstępstwo powodujące niekompatybilność i rozwiązanie tego problemu zostało opisane w podrozdziale dotyczącym standardu ISA, w opisie sterownika klawiatury - 6.2.1.3). W trybie chronionym jednostka ta współpracuje z jednostką zarządzania pamięcią MMU, generując na podstawie adresu wirtualnego fizyczny adres pamięci. Zwracamy uwagę, że zawartość rejestrów segmentowych w tym trybie ma zupełnie inne znaczenie. Pojęcie pamięci

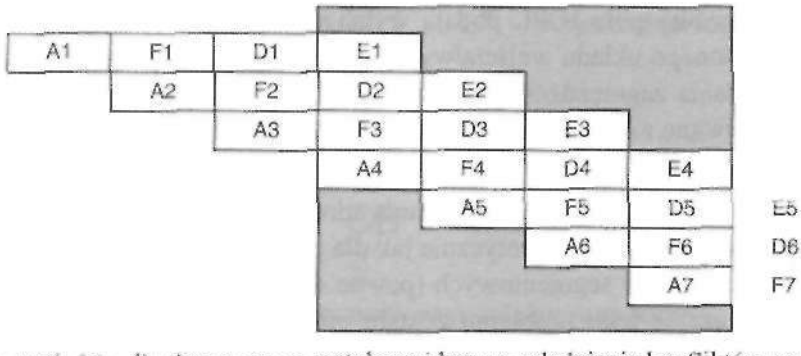
wirtualnej i prosty przykład generowania adresu fizycznego na podstawie adresu wirtualnego przedstawione są w rozdziale 3.6. Przykład generowania adresu fizycznego w trybie chronionym dla procesorów tej rodziny omawiamy na przykładzie procesora Pentium™ w podrozdziale 4.5.1.6.

Jednostka sterowania magistralami jest podobna do swojej odpowiedniczki w procesorze 8086. W jej skład wchodzą: bufory adresowe, interfejs koprocesora, prefetcher, układ sterowania magistralami, bufory danych i 6-bajtowa kolejka rozkazów. W zestawie rejestrów sterujących (dostępnych programowo) pojawił się nowy rejestr oznaczony przez MSW (ang. *machinę status word).*

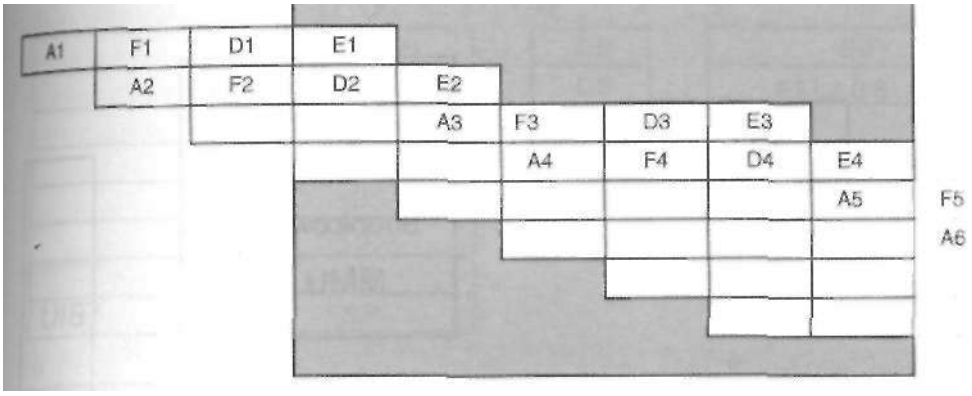
W skład jednostki dekodowania instrukcji wchodzi dekoder rozkazów oraz 3-elementowa kolejka rozkazów zdekodowanych. Jednostka dekodowania instrukcji pobiera kody instrukcji z kolejki rozkazów niezdekodowanych w BIU, dekoduje je i wynik dekodowania umieszcza w kolejce rozkazów zdekodowanych. W przypadku rozkazu skoku lub przejścia do wykonania podprogramu obie kolejki są czyszczone.

W procesorze 80286 pojawia się możliwość pracy potokowej będącej rozwinięciem koncepcji prefetehingu. Polega na równoległym wykonywaniu kilku faz realizacji rozkazu. W procesorze 80286 realizację rozkazu możemy podzielić na następujące fazy (w nawiasie podajemy realizujące je jednostki): adresowania - A (AU), pobrania - F (BIU), dekodowania - D (CU) i realizacji - E (EU). Teoretycznie można więc program realizować tak, jak to pokazano na rysunku 4.6a. W rzeczywistości konflikt dostępu do magistral (na przykład fazy pobrania i wykonania, gdy zapisujemy do pamięci) będzie powodował pracę potokową pokazaną na rysunku 4.6b.

Zacieniowane pole pokazuje szybkość realizacji instrukcji w obu przypadkach. Z rysunku wynika, że potokowa realizacja instrukcji przyspiesza 1,5-2 razy pracę procesora. Rysunek 4.6 pokazuje oczywiście dwa graniczne przypadki: gdy żaden rozkaz nie zapisuje do pamięci i gdy wszystkie rozkazy zapisują do pamięci.



Rysunek 4.6a. Realizacja pracy potokowej bez uwzględniania konfliktów magistral



Rysunek 4.6b. Rzeczywista realizacja pracy potokowej

4.4. Procesory 80386 i 80486

Procesor 80386 jest pierwszym 32-bitowym mikroprocesorem rodziny Intel x86, od którego rozpoczęło się tworzenie architektury IA 32. Architektura ta jest kompatybilna wstecz z 16-bitowymi procesorami rodziny Intel x86. Sposób realizacji instrukcji ulegał natomiast znacznym zmianom w trakcie rozwoju tej architektury. Zmiany te będziemy sukcesywnie, choć krótko, opisywać. Miały na celu głównie przyspieszenie realizacji instrukcji, a zatem zwiększenie mocy obliczeniowej procesora.

4.4.1. Procesor Intel 80386

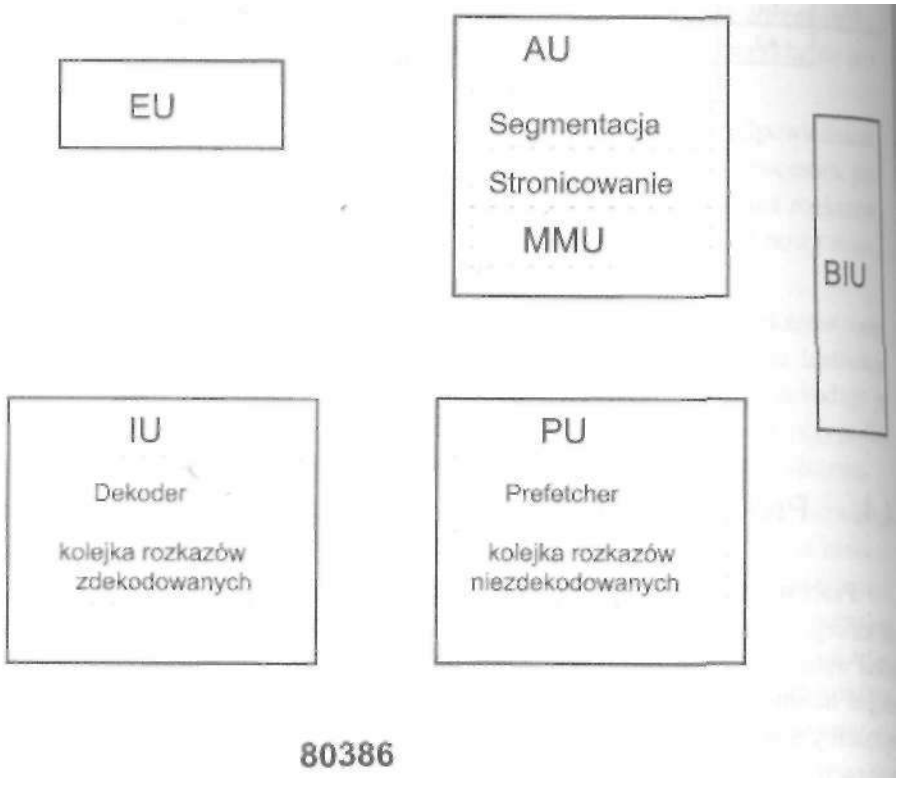
4.4.1.1. Schemat blokowy

Schemat blokowy procesora Intel 80386 jest pokazany na rysunku 4.7.

Podstawowymi blokami procesora 80386 są:

- blok adresowy AU, w skład którego wchodzą bloki segmentacji stronicowania i MMU (jednostka zarządzania pamięcią),
- blok dekodowania instrukcji IU,
- blok wstępnego pobierania instrukcji PU,
- blok sterowania magistralami BIU,
- blok wykonawczy EU.

Zestaw rejestrów dostępnych programowo procesora 80386 przedstawia rysunek 4.8.



Rysunek 4.7. Schemat blokowy procesora 80386

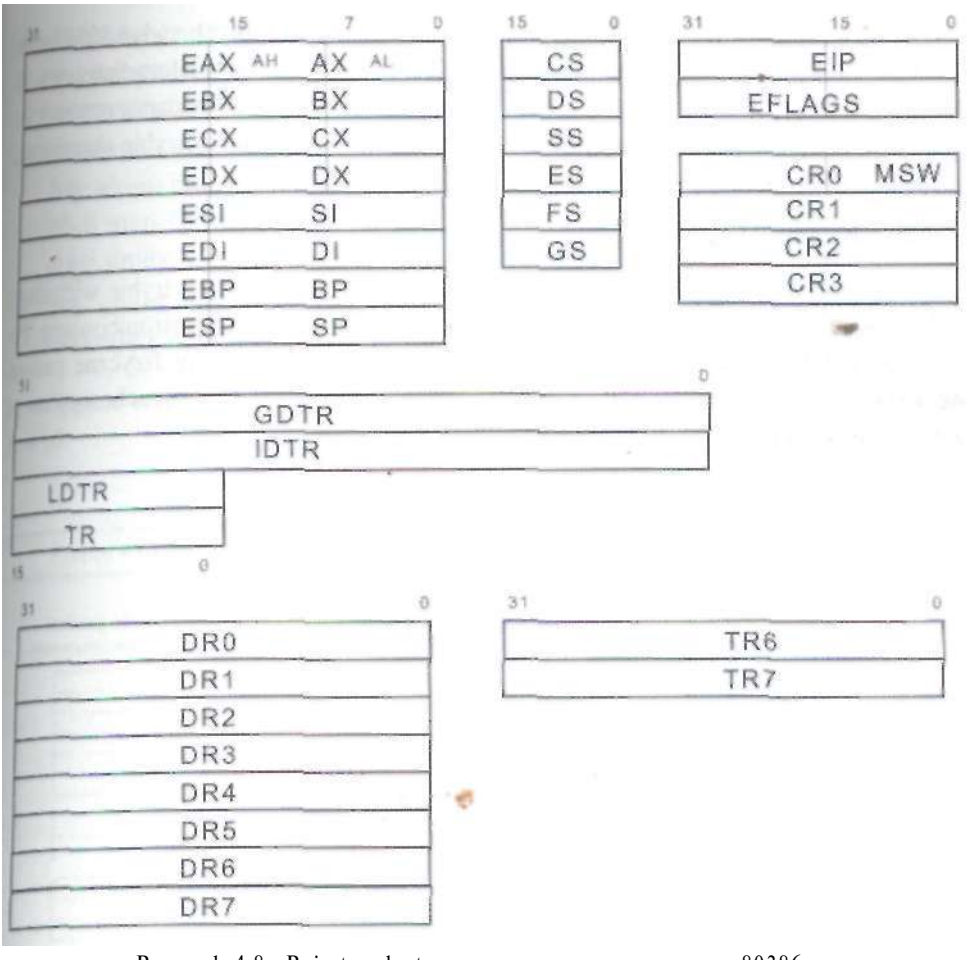
Nazwa rejestru poprzedzona literą E oznacza odwołanie do rejestru 32-bitowego. [Na przykład EAX jest nazwą 32-bitowego akumulatora. W zestawie rejestrów pojawiły się też nowe: rejestry sterujące CRx, rejestry uruchomieniowe DRx i rejestry I testowe TRx. Ponadto zmieniła się długość rejestrów IDTR i GDTR.

Proszę zwrócić uwagę, że zestaw rejestrów dostępnych programowo procesorów I 8086 i 80286 jest podzbiorem przedstawionych wyżej rejestrów, co właśnie zapewnia kompatybilność architektury IA32 z tymi procesorami. Podobnie rzecz ma się z listą B instrukcji.

4.4.1.2. Tryby pracy procesora 80386

Procesor 80386 może pracować w trzech trybach pracy:

- trybie rzeczywistym,
- chronionym trybie wirtualnym,
- trybie zadań wirtualnych 8086.



Rysunek 4.8. Rejestry dostępne programowo procesora 80386

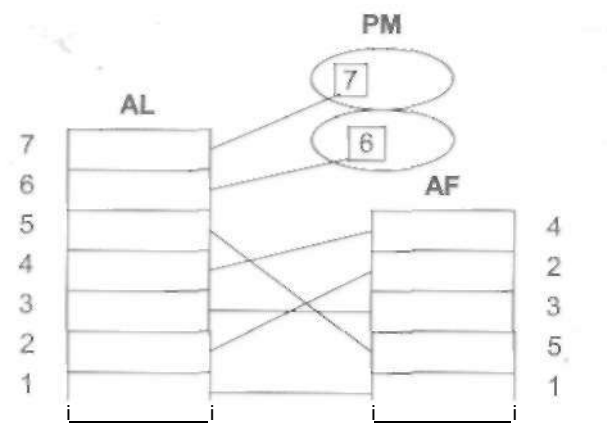
Tryb rzeczywisty procesora 80386 jest taki sam jak tryb rzeczywisty procesora 80286. Chroniony tryb wirtualny oferuje podobne, aczkolwiek ulepszone i rozszerzone możliwości jak procesor 80286. W trybie tym, w porównaniu do procesora 286, zwiększona jest przestrzeń pamięci wirtualnej przydzielona jednemu zadaniu i wynosi 64 TB. Ponadto, oprócz mechanizmu segmentacji w trybie tym dostępny jest także mechanizm stronicowania (opisany w kolejnym podpunkcie). Tak jak poprzednio, I mamy do dyspozycji mechanizmy wspomagające pracę wielozadaniową i ochronę zasobów.

Trzeci tryb jest trybem nowym, który pojawił się dopiero w procesorze 80386. Tworzy środowisko do wykonywania zadań przygotowanych dla procesora 8086 w wielozadaniowym środowisku trybu chronionego procesora 80386. Zadania procesora 8086 mogą więc być wykonywane przez procesor 80386 w dwóch trybach: rzeczywistym i wirtualnych zadań 8086. Różnica polega na tym, że w trybie rzeczy-

wistym wykonujemy jedno zadanie, podczas gdy tryb wirtualnych zadań 8086 można stosować do określonych zadań wykonywanych w środowisku wielozadaniowym. Po przełączeniu się do tego trybu w celu wykonania konkretnego zadania procesor ten zachowuje się jak 8086, lecz po powrocie z niego pracuje dalej w trybie chronionym umożliwiając pracę wielozadaniową.

4.4.1.3. Stronicowanie

W procesorze 80386 (oraz w jego następcach) pracującym w trybie wirtualnym oprócz mechanizmu segmentacji dostępny jest także mechanizm stronicowania. Pozwala on używać ciągłego adresu liniowego, podczas gdy adresy fizyczne pamięci mogą stanowić obszar nieciągły. Część informacji może też być przechowywana na dysku (pamięć wirtualna). Ilustruje to rysunek 4.9.



gdzie: AL - adres liniowy
AF - adres fizyczny
PM - pamięć masowa

Rysunek 4.9. Idea działania mechanizmu stronicowania

Stronicowanie rozwiązuje więc przykładowo pofragmentowanie pamięci czy też problem realizacji kilku aplikacji DOS w środowisku wielozadaniowym (każda chce adresować pierwszy megabajt pamięci operacyjnej) przez translację adresu liniowego AL aplikacji na adres fizyczny AF.

Ponadto, inaczej niż w mechanizmie segmentacji, mamy tu stałą wielkość stron, co jest korzystne w przypadku obsługi pamięci wirtualnej. Wielkość stron dla procesorów 80386 i 80486 wynosiła zawsze 4 kB (w procesorze Pentium może wynosić 4 kB lub 4 MB). Mechanizm stronicowania można włączać i wyłączać, ustawiając bądź zerując bit PG w rejestrze sterującym CR0.

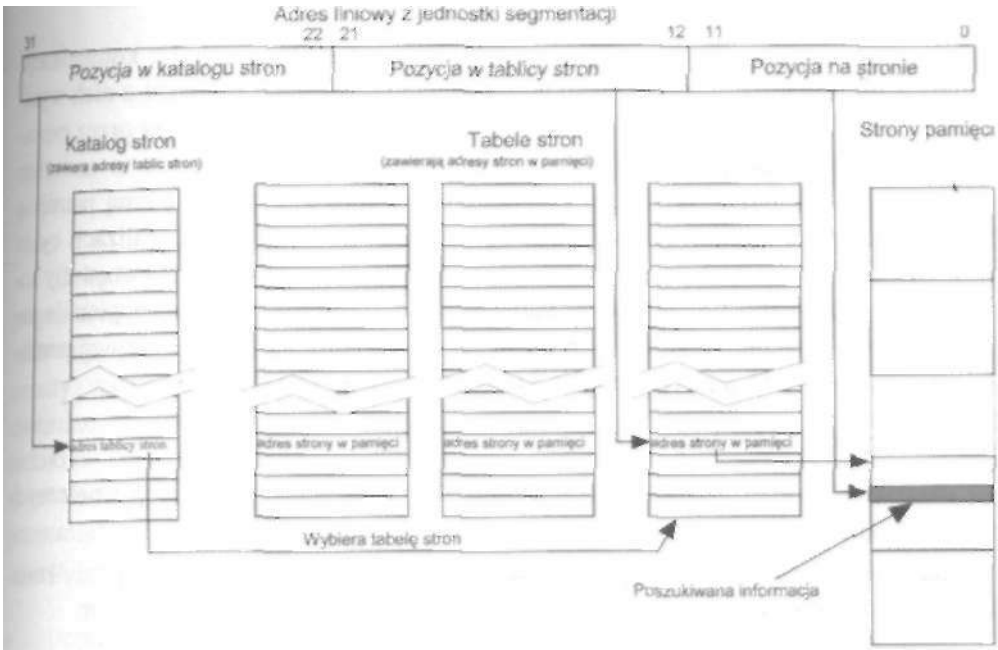
Translacja adresu liniowego na fizyczny w przypadku stron 4-kilobajtowych przebiega w następujących etapach:

» 10-bitowa część adresu liniowego AL31+AL22 wybiera jedną z 1024 pozycji w katalogu stron (ang. *page directory)*. Adres początkowy katalogu stron jest automatycznie ładowany do rejestru CR3.

Zawartość wybranej pozycji w katalogu stron wskazuje na jedną z 1024 tabel stron. Bity adresu liniowego AL 21/AL12 wybierają jedną z 1024 pozycji w wybranej tabeli stron. Wybrana pozycja w tabeli stron podaje adres początkowy 4-kilobajtowej strony, w której znajduje się poszukiwana informacja.

y Bity adresu liniowego ALll-i-ALO stanowią przesunięcie w wybranej stronie wskazujące na poszukiwaną informację.

Opisane etapy translacji adresu liniowego na fizyczny zilustrowane są na rysunku*^



Rysunek 4.10. Generacja adresu w trybie stronicowania

Jeżeli poszukiwanej strony nie ma w pamięci operacyjnej, w rejestrze CR2 jest umieszczany adres liniowy brakującej strony i generowany wyjątek 14 - błąd strony (ang. *page fault)*. Program obsługi tego wyjątku powinien spowodować wczytanie brakującej strony z dysku i zmodyfikowanie odpowiedniej pozycji w tabeli stron.

Posługiwanie się przy translacji adresu liniowego na fizyczny wyłącznie danymi z pamięci (katalogi i tabele stron) prowadziłoby do znacznego zmniejszenia szybkości działania systemu. W celu przeciwdziałania takiej sytuacji, podobnie jak dla pamięci wirtualnej, wprowadzono w procesorze pomocniczą pamięć typu cache przechowującą

zawartość 32 ostatnio używanych tablic stron. Pamięć ta oznaczana jest jako TLB (ang. *translation lookaside buffer).* Ponieważ jest to pamięć asocjacyjna, można ja szybko przeszukać (koncepcję działania pamięci asocjacyjnej omawiamy w podrozdziale poświęconym pamięci cache). W przypadku odnalezienia pozycji odpowiadającej danej tablicy stron nie jest generowany dostęp do pamięci w celu odczytania odpowiedniego adresu fizycznego. Zamiast tego adres (20 starszych bitów) jest podawany przez TLB. 12 młodszych bitów to, przypominamy, przesunięcie w ramach strony.

W przypadku używania stron 4-megabajtowych pozycja w katalogu stron określa bezpośrednio adres bazowy poszukiwanej strony. Do tej wartości dodawane jest przesunięcie określone przez bity 21-s-0 adresu linowego, w wyniku czego uzyskiwany jest fizyczny adres poszukiwanej informacji.

4.4.2. Procesor Intel 80486

Wąskimi gardłami współpracy procesora 80386 z systemem były: komunikacja z koprocesorem arytmetycznym i współpraca z pamięcią cache. Pierwsze z nich wynika z faktu, że zarówno instrukcje, jak i dane dla koprocesora pobierane są przez procesor główny, który następnie przekazuje je do koprocesora. Zmniejsza to znacząco szybkość pracy systemu. W drugim przypadku komunikacja z zewnętrzną pamięcią cache, mimo braku stanów oczekiwania, ograniczona jest szybkością realizacji cyklu magistrali. Obie te wady usunięto w procesorze 80486, wprowadzając wewnętrzny koprocesor arytmetyczny i wewnętrzną pamięć cache. Tryby pracy oraz magistrala danych i adresowa nie uległy zmianie. Poniżej przedstawiamy zasadnicze różnice pomiędzy procesorem 80486 a jego poprzednikiem, procesorem 80386.

4.4.2.1. Schemat blokowy

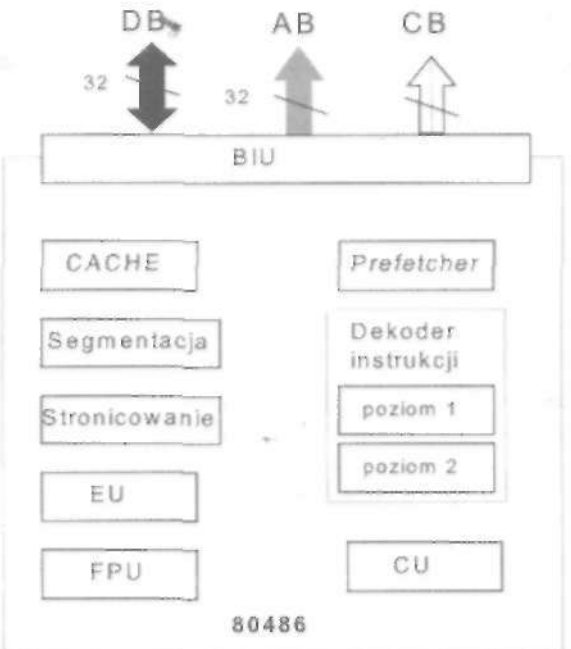
Podstawowe bloki procesora 80486 oraz szerokość jego magistral zewnętrznych I przedstawia rysunek 4.1 l.

Jak widzimy, szerokość magistrali adresowej i magistrali danych nie uległy zmianie. Natomiast schemat blokowy tego procesora zawiera dwa nowe bloki. Są to:

- blok arytmetyki zmiennoprzecinkowej (czyli koprocesor arytmetyczny) oznaczany jako NPU (ang. *numeric processor unit)* lub FPU (ang. *floating-point unit),*
- blok 8 kB pamięci cache wraz ze sterownikiem.

Zmiany te usuwają wymienione we wstępie wąskie gardła systemów z procesorem 80386.

Ponadto rozbudowie uległy bloki BIU oraz blok dekodowania instrukcji. Pierwszy z nich składa się z następujących części: buforów adresowych, buforów zapisu, buforów danych, układów logicznych sterowania szerokością magistrali, układów sterowania operacjami w trybie seryjnym (burst) i układów generacji parzystości.



Rysunek 4.11. Schemat blokowy procesora 80486

Dekodowanie instrukcji zostało podzielone na dwa etapy co, wbrew pozorom, przyspiesza realizację instrukcji (proszę pomyśleć o produkcji taśmowej, gdzie wszystkie etapy produkcji przedmiotu trwają minutę, z wyjątkiem jednego, który trwa dwie minuty. Jeden przedmiot będzie produkowany co dwie minuty. Jeżeli jednak etap dwuminutowy rozbijemy na dwa jednominutowe, dodając przy tym jedno gniazdo technologiczne, to będziemy produkować na minutę jeden przedmiot!). Pierwszy etap lo ustalenie rodzaju operacji i trybu adresowania. W drugim etapie wyliczane jest przesunięcie (AE) i ewentualnie przygotowywane są argumenty natychmiastowe.

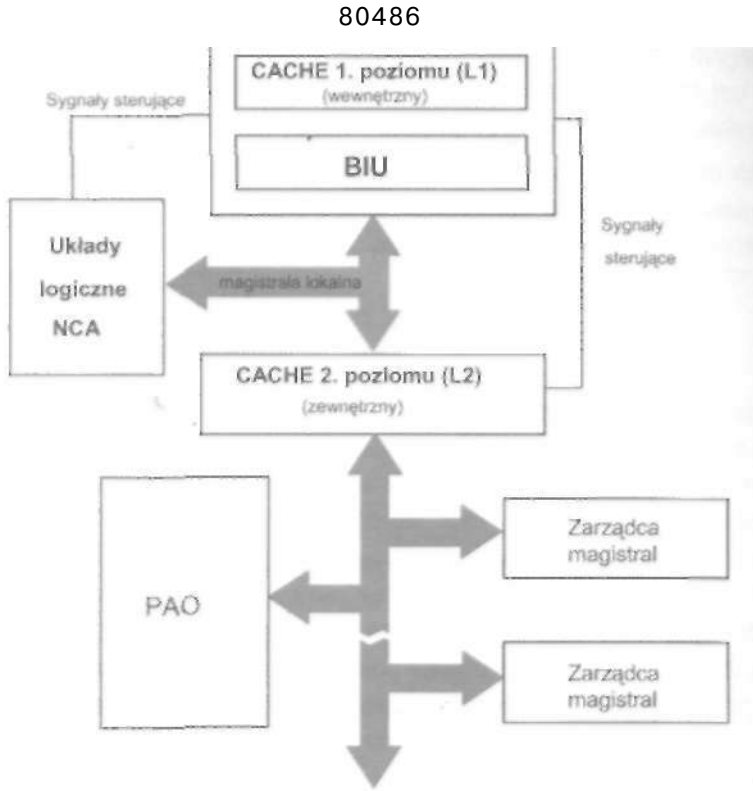
Praca procesora przebiega w trybie potokowym i składa się z pięciu etapów:

- pobrania kodu instrukcji,
- pierwszego etapu dekodowania instrukcji,
- drugiego etapu dekodowania instrukcji,
- wykonania,
- zapisu do rejestrów.

Oprócz używania wewnętrznej pamięci cache procesor 80486 może także współpracować z zewnętrzną pamięcią cache. Kolejny podrozdział krótko opisuje zagadnienia związane z pracą pamięci cache w systemie z procesorem 80486.

4.4.2.2. Pamięć cache

Strukturę pamięci cache dla systemu z procesorem 80486 przedstawia rysunek 4.12.



Rysunek 4.12. Pamięć cache w systemie z procesorem 80486

Architektura przedstawionej na rysunku zewnętrznej pamięci cache jest typu I Look-through. Występuje w systemach znacznie częściej, chociaż możliwa jest także i architektura Look-aside. Zalety i wady obu rozwiązań przedstawiono w punkcie 3.7.1. Pamięć cache zawartą wewnątrz procesora nazywa się często pamięcią cache poziomu i pierwszego lub LI (ang. *level 1),* natomiast zewnętrzną cache pamięcią cache drugiego poziomu lub L2. W nowszych procesorach terminologia ta nieco się skomplikowała. O oznaczeniu poziomu pamięci cache (LI, L2, L3...) decyduje kolejność poszukiwania informacji, natomiast nie jest istotne, gdzie pamięć ta fizycznie się mieści.

Bardzo ważny jest fakt, że informacja w pamięci cache LI znajduje się jednocześnie w pamięci L2. Wynika stąd, że wyraźne efekty stosowania pamięci cache L2 występują dopiero, gdy jest znacząco większa od cache LI. Dopiero wówczas istnieje większa szansa trafienia dla cache L2 w przypadku chybienia dla cache LI.

Na rysunku zamieszczono też blok układów logicznych NCA (ang. *non-cachabk acces).* Układy te dotyczą także następnych procesorów. Związane są z występowa-

rjam w systemie obszarów pamięci, którft^nie mogą być odwzorowywane w pamięci cache, gdyż np. nie jest możliwe zapewnienie zgodności tych obszarów z pamięcią cache. Przykładem takiego obszaru może być dwuportowa pamięć RAM karty sieciowej. Zapis do tej pamięci jest możliwy zarówno przez system (magistralą systemową), jak i przez sieć (bez użycia magistrali systemowej). W tym drugim przypadku system nie jest w stanie wykryć, które komórki zostały zmodyfikowane (nie ma możliwości 'śledzenia magistrali sieci), co z kolei nie pozwala utrzymać spójności pamięci głównej i cache.

W celu uniknięcia powyższej sytuacji projektant systemu musi przewidzieć układy wykrywające operacje odczytu obszarów, które nie mogą być przechowywane w pamięci cache, i przeciwdziałać ich załadowaniu do niej. Muszą to być układy dekodujące określone zakresy adresów. Sterowniki pamięci cache mają wejście sterujące, które umożliwia zablokowanie zapisu tej informacji do pamięci cache. Istnieją dwie grupy rozwiązań:

1, W systemie występują predefiniowane zakresy adresów, które nie mogą być przechowywane w pamięci cache. Układy, które przykładowo używają wspomnianych wyżej pamięci dwuportowych, muszą być konfigurowane tak, aby korzystały z tych zakresów adresów.
2. W systemie występuje programowalny dekoder adresów. Zakresy adresów, które nie mogą być przechowywane w pamięci cache, są wprowadzane w trakcie inicjacji systemu po restarcie.

Wewnętrzna pamięć cache w systemie z procesorem 486 jest czteroblokową pamięcią asocjacyjną. Każdy blok o pojemności 2 kB zawiera 128 16-bajtowych linii. 4-gigabajtowa pamięć główna jest widziana przez cache jako 2097152 2-kilobajtowe bloki. Szczegółowy opis współpracy wewnętrznej i zewnętrznej pamięci cache z pamięcią główną jest dość obszerny i wykracza poza ramy tego podręcznika. Można go znaleźć na przykład w pozycjach [8] lub [9].

Ostatnim problemem związanym z pamięcią cache jest zapewnienie możliwości testowania pamięci przez procesor. W trakcie realizacji takiego testu sterownik cache może przechwytywać odczyty pamięci. Efektem tego jest odczyt z pamięci cache zamiast, jak zakładamy, z pamięci głównej. W celu umożliwienia wykonania testu sterowniki cache zwykle mają wejście sterujące oznaczone jako FLUSH#. Powoduje wyzerowanie wszystkich bitów ważności w katalogu pamięci cache. Bity te są utrzymywane w stanie 0 tak długo, jak długo utrzymywany jest sygnał aktywny na wejściu FLUSH#. Układ sterujący tym wejściem jest zwykle umieszczany w przestrzeni adresowej układów wejścia/wyjścia.

4.4.2.3. Magistrala sterująca

Zestaw linii magistrali sterującej obejmuje 38 sygnałów połączonych w następujące grupy:

- określenie rodzaju cyklu magistrali,
- sterowanie cyklem magistrali,
- sterowanie cyklem magistrali w typie burst,
- sygnały przerwań,
- sterowanie dostępem do magistrali,
- sterowanie pracą wewnętrznej pamięci cache,
- sygnalizacja błędów koprocesora,
- sterowanie rozmiarem przesłań na magistrali danych,
- sterowanie poborem mocy (technologia SL),
- sygnały testujące oraz zegarowe.

4.4.2.4. Rejestry dostępne programowo

Zestaw rejestrów dostępnych programowo dla procesora 80486 nieznacznie różni się od zestawu procesora 80386. Zwiększona została do pięciu liczba rejestrów testujących TR.

4.5. Procesor Pentium™

Wstęp

Rodzina procesorów Pentium obejmuje kilka dość znacznie różniących się rozwiązań. Pierwsza wersja tego procesora, oznaczana później jako P5, a nazywana po prostu Pentium, nie obsługiwała pracy wieloprocesorowej i nie zawierała technologii MMX. Miała rozdzielone wewnętrzne pamięci cache danych i programu, każda o rozmiarze 8 kB. Wersja P54C pozwalała na pracę dwuprocesorowa. Procesor P55C zawierał już technologię MMX i miał zwiększony rozmiar pamięci cache - 2 x 16 kB.

Następnym etapem rozwoju tej rodziny było opracowanie rdzenia P6 w kilku różniących się wersjach. Początek tej gałęzi dało opracowanie procesora Pentium Pro (gniazdo Socket 8), którego rdzeń nosił oznaczenie kodowe Klamath. Pentium Pro nie miał zaimplementowanej technologii MMX. Do tego rdzenia były też zaliczane wczesne wersje Pentium II (Slot 1) zawierające technologię MMX.

Kolejnym etapem rozwoju rdzenia P6 był rdzeń nazywany Deschutes, z którym związane były późniejsze wersje procesora Pentium II (Slotl) oraz procesory Celeron Covington (bez pamięci cache L2), Celeron Mendocino (inaczej Celeron A, 128 kB cache, Socket 370) i pierwsze wersje procesorów Xeon (kilka rozmiarów pamięci

cache L2, praca 4-procesorowa, Slot 2). Wreszcie trzecią wersją rdzenia P6 był rdzeń Katmai, do którego należały procesory Pentium III, Pentium III Coppermine i kolejne wersje procesorów Xeon. Ostatnim typem rdzenia był rdzeń Pentium 4.

Począwszy od Pentium II, od strony handlowej firma Intel opracowywała trzy odmiany danego procesora. Oprócz wersji podstawowej była opracowywana wersja tańsza nosząca nazwę Celeron, przeznaczona dla tańszego sprzętu powszechnego ifżytku, oraz wersja wzbogacona nosząca miano Xeon, do zastosowań w wydajnych komputerach, przykładowo pracujących jako serwery.

4.5.1. Procesor Pentium - rdzeń P5

Zwiększanie szybkos'ci zegara-taktującego nowe procesory oraz wprowadzane w nich rozwiązania, takie jak wstępne pobieranie instrukcji mające na celu przyspieszenie pracy systemu, spowodowały pojawienie się kolejnych ograniczeń. Istnienie jednej pamięci cache wspólnej dla programu i danych powodowało podczas pracy potokowej konflikty dostępu. Kolejne problemy dotyczą prefetchera. W przypadku rozgałęzień w programie pracował nieefektywnie. Przyspieszenie procesu pobierania instrukcji wymaga z kolei przyspieszenia ich wykonywania. Dodatkowo, współczesne systemy wymagają często użycia kilku procesorów, do czego dotychczasowe procesory nie były przystosowane. Nowe rozwiązania wprowadzone w procesorach Pentium pozwoliły w znacznej mierze znieść te ograniczenia.

4.5.1.1. Podstawowe własności procesora Pentium

Poniżej podajemy podstawowe własności i cechy procesora Pentium. Dotyczą podstawowej wersji tego procesora. Kolejne, często znacznie zmienione wersje omówione zostały w dalszej części rozdziału. Większość z wymienionych tu cech jest następnie bardziej szczegółowo dyskutowana w kolejnych podpunktach tego rozdziału. Po podaniu własności procesora Pentium w końcowej części bieżącego podpunktu krótko wyjaśniono też znaczenie wybranych cech tego procesora dla jego możliwości.

Podstawowe cechy i własności procesora Pentium:

> 64-bitowa magistrala danych i 32-bitowa magistrala adresowa.

> Praca w trzech trybach:
  - trybie rzeczywistym,
  - chronionym trybie wirtualnym,
  - trybie wirtualnym 8086.

> Sprzętowe mechanizmy ułatwiające projektowanie systemów operacyjnych obsługujące:
  - pamięć wirtualną,

- pracę wielozadaniową,
- ochronę zasobów.

> Architektura superskalarna:

- praca potokowa,
- dwa potoki przetwarzania instrukcji stałoprzecinkowych.

"r Przewidywanie realizacji rozgałęzień programu.

> Segmentacja i stronicowanie pamięci.

> Wewnętrzna jednostka arytmetyki zmiennoprzecinkowej pracująca w trybie potokowym.

r Dwie wewnętrzne rozdzielone pamięci podręcznej cache (architektura harwardzka):

1. pamięć cache dla danych (ang. *data cache),*
2. pamięć cache dla kodu programu (ang. *codę cache, instruction cache).*

y Możliwos'ć współpracy z pamięcią cache drugiego poziomu.

Procesor Pentium, podobnie jak procesory 80386 i 80486, może pracować w trzech trybach. W trybie rzeczywistym jest w pełni zgodny ze swoimi poprzednikami. W chronionym trybie wirtualnym ujawnia swoje pełne możliwości, oferując między innymi wiele mechanizmów przydatnych dla projektantów systemów operacyjnych. Trzeci tryb - wirtualny 8086, umożliwia realizację programów pisanych dla trybu rzeczywistego w środowisku wielozadaniowym. Bardziej szczegółowo zagadnienia pamięci wirtualnej, tryby pracy: chroniony, V 86, oraz mechanizmy wspomagające pracę wielozadaniową zostały omówione kolejno w podpunktach 4.5.1.6, 4.5.1.7 i 4.5.1.8.

Procesor Pentium realizuje instrukcje w sposób potokowy, przy czym w określonych warunkach mogą być równolegle przetwarzane dwie instrukcje. Umożliwia to realizację średnio do dwóch instrukcji na jeden takt zegara, co czyni Pentium procesorem superskalarnym (procesor superskalamy to taki, który potrafi wykonywać średnio więcej niż jedną instrukcję na jeden takt zegarowy). Fazy potoku są takie same jak dla procesorów 386 i 486, aczkolwiek operacje w nich wykonywane nieco się różnią. Potok realizacji instrukcji procesora Pentium opisujemy w punkcie 4.5.1.11.

Włączenie koprocesora arytmetycznego oraz pamięci cache jako wewnętrznych elementów procesora zapewnia szybszy dostęp do nich, a co za tym idzie, powoduje szybsze przetwarzanie informacji przez procesor.

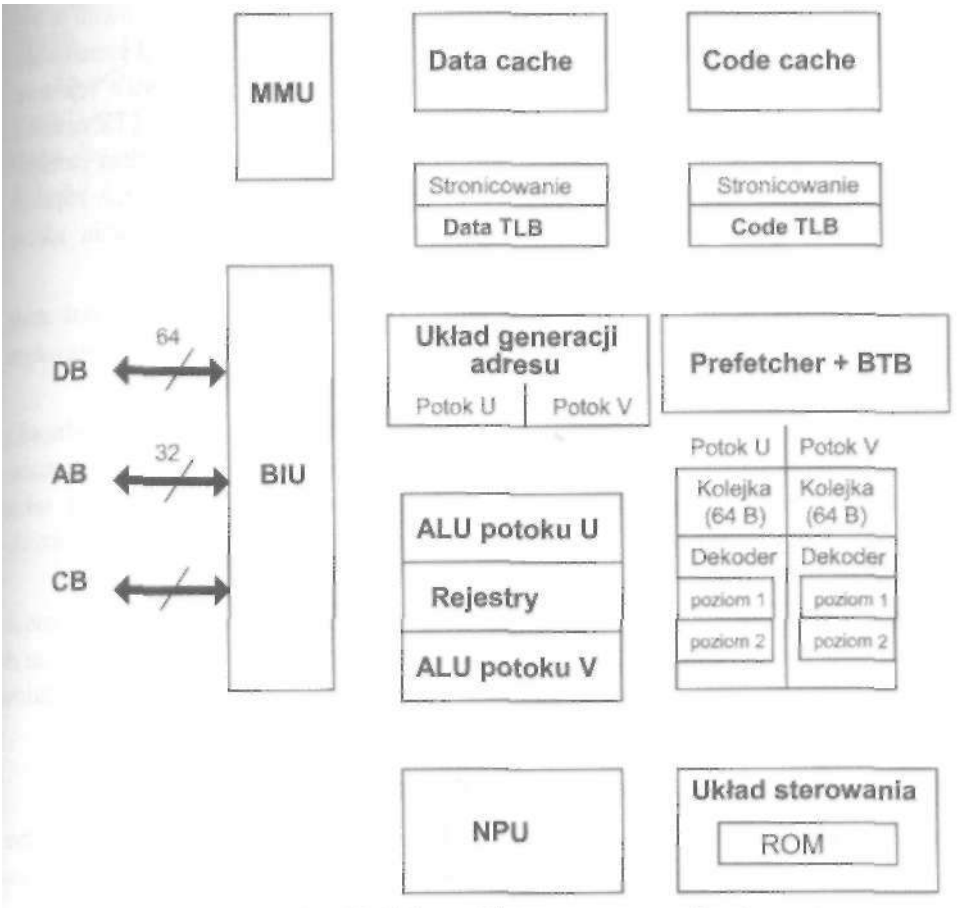
Zastosowanie rozdzielonych wewnętrznych pamięci cache, osobnej dla programu i osobnej dla danych, pozwala uniknąć wielu konfliktów podczas pracy potokowej. Dokładniejsze wyjaśnienia zawarte są w punkcie 4.5.1.11.

Kolejne przyspieszenie pracy procesora Pentium uzyskano dzięki zastosowaniu w bloku wstępnego pobierania instrukcji układu przewidywania realizacji rozgałęzień. Układ ten opisany jest w punkcie 4.5.1.12.

4.5.1.2. Schemat blokowy procesora Pentium

Schemat blokowy procesora Pentium przedstawia rysunek 4.13. Poszczególne bloki związane są z realizacją własności wymienionych w poprzednim punkcie.

Występują w nim dwa bloki pamięci cache. Pierwszy z nich, data cache, przeznaczony jest do przechowywania danych i wyników działania programu. Drugi, code cache. zawiera kody instrukcji wykonywanego programu. Zastosowanie rozdzielonych pamięci danych i programu umożliwia jednoczesne pobieranie kodu instrukcji i zapis/odczyt danych. Zmniejsza to częstotliwość kolizji zasobów podczas pracy potokowej (patrz punkt 4.5.1.11).



Rysunek 4.13. Schemat blokowy procesora Pentium

Z pracą potokową związany jest blok prefetchera wraz z układem BTB. Prefetcher, zwany inaczej układem wstępnego pobierania instrukcji, ma za zadanie wcześniejsze pobieranie kodów instrukcji programu i umieszczanie ich w kolejce rozkazów. Współpracuje z nim układ przewidywania realizacji rozgałęzień, którego częścią I jest bufor rozgałęzień (ang. *Branch Target Bujfer - BTB).* Pięciofazowa praca potokowa procesora Pentium jest możliwa między innymi dzięki istnieniu dwustopniowych dekoderów instrukcji.

Instrukcje mogą być przetwarzane w dwóch potokach, nazywanych U i V, dlatego też istnieją dwie 32-bitowe jednostki arytmetyczno-logiczne.

Układ dekodowania instrukcji zawiera także dwa dekodery, osobny dla potoku U i osobny dla potoku V. Współpracują z kolejkami rozkazów wypełnianymi przez układ prefetchera. Sposób obsługi tych kolejek opisany jest w punkcie 4.5.1.12. Kody instrukcji dekodowane są dwustopniowo.

Układ generacji adresu służy do generowania adresu fizycznego zarówno w trybie rzeczywistym, jak i chronionym. W trybie rzeczywistym adres ten jest generowany identycznie jak dla procesora 8086, za pomocą takiego samego zestawu rejestrów segmentowych, W trybie chronionym układ ten współpracuje z jednostką zarządzania pamięcią MMU, generując na podstawie adresu wirtualnego fizyczny adres pamięci. Zwracamy uwagę, że zawartość rejestrów segmentowych w tym trybie ma zupełnie inne znaczenie. Pojęcie pamięci wirtualnej i prosty przykład generowania adresu fizycznego w trybie chronionym przedstawione są w punkcie 3.6.3

Układy stronicowania wraz z układami TLB (ang. *Translation Look-aside Bujfer)* umożliwiają efektywną obsługę pamięci w trybie stronicowania (opisanym w punkcie 4.4.1.3).

Blok wewnętrznego koprocesora arytmetycznego NPU (zwanego też jednostką arytmetyki zmiennoprzecinkowej FPU) realizuje wszelkie operacje arytmetyki zmiennoprzecinkowej. Rozkazy operacji zmiennoprzecinkowych wykonywane są także potokowo, w ośmiu fazach, przy czym pierwszych pięć faz jest wspólnych z instrukcjami stałoprzecinkowymi.

Pozostałe bloki to jednostka sterowania magistralami BIU zapewniająca komunikację procesora z otoczeniem, blok rejestrów współpracujących z jednostkami arytmetyczno-logicznymi oraz układ sterowania kierujący pracą wszystkich układów procesora.

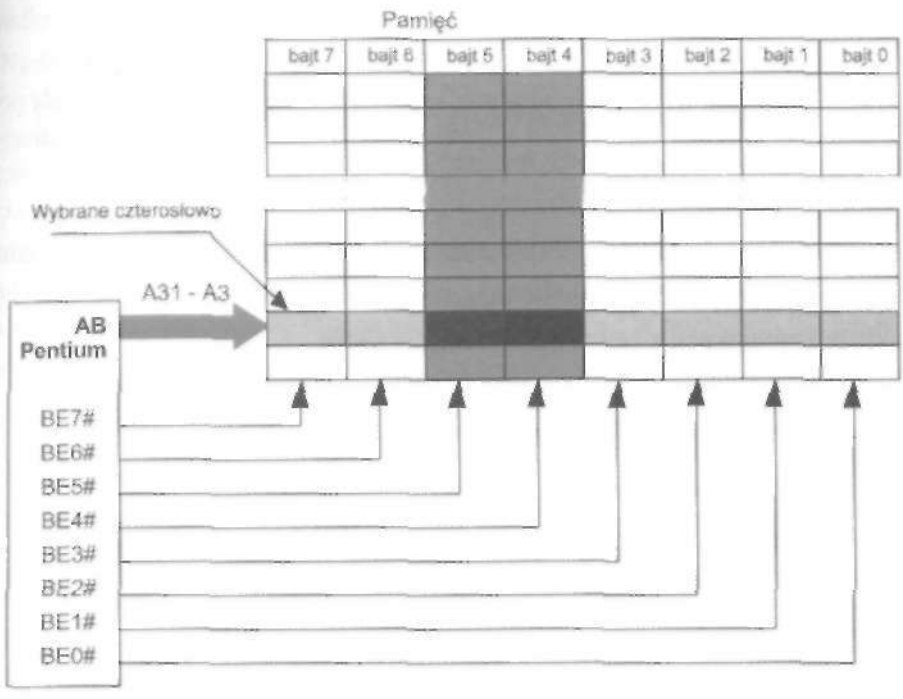
4.5.1.3. Magistrale zewnętrzne procesora Pentium

Procesor Pentium ma 64-bitową magistralę danych, 32-bitową magistralę adresową (fizycznie występuje na niej 37 linii, co wyjaśniamy w kolejnym akapicie) oraz 67 linii magistrali sterującej.

Magistralę danych można traktować jako zestaw ośmiu ścieżek bajtowych (8-bitowych). Można nią przesyłać dane 8-, 16-, 24-, 32- i 64-bitowe. Przesłanie może następować po dowolnym zestawie ścieżek, pod warunkiem że ścieżki te przylegają do siebie. Przesłanie 24-bitowe będzie występować w przypadku przesyłania dwusłowa (słowa 32-bitowego) położonego na granicy czterosłowa. Wówczas dwusłowo zostanie przesłane w dwóch cyklach, przykładowo w jednym 24 bity, natomiast w drugim pozostałych 8 bitów. Unikanie tego typu sytuacji należy do zadań programisty.

Do magistrali adresowej należą wyprowadzenia A31÷A3 oraz BE7#÷BE0#. Wewnątrz procesora generowany jest 32-bitowy adres A31÷A0. Na zewnątrz wyprowadzane są jednak tylko bity A31÷A3. Wybierają określone czterosłowo (8 bajtów). Pozostałe bity adresu oraz wielkość przesyłanej wartości (bajt, dwusłowo – ang. *double word* lub *dword* – lub czterosłowo – ang. *quad word* lub *qword*) decydują o tym, które sygnały BEx# zostaną uaktywnione. Powoduje to uaktywnienie określonych bajtów czterosłowa i przesłanie ich odpowiadającymi im ścieżkami magistrali danych.

Sposób adresowania pamięci za pomocą linii należących do magistrali adresowej przedstawia rysunek 4.14. Przykładowo na rysunku nastąpi przesłanie 16-bitowe ścieżkami danych 5 i 4, bajtów numer 5 i 4 czterosłowa wybranego za pomocą adresu A31÷A3. Oznacza to przesłanie słowa (16 bitów).



Rysunek 4.14. Fizyczne adresowanie pamięci przez procesor Pentium

W tabeli 4.4 podane są przykładowe kombinacje wartości sygnałów BEx# i odpowiadające im rodzaje przesłań.

Tabela 4.4. Rodzaj przesłania w zależności od wartości linii BEx#

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	Rodzaj przesłania
1	1	1	1	1	1	1	1	nie występuje
1	1	1	1	1	1	1	0	8-bitowe
1	1	0	1	1	1	1	1	8-bitowe
1	1	1	1	1	0	1	1	8-bitowe
1	1	1	1	1	0	1	0	nie występuje
1	1	0	0	1	1	1	1	16-bitowe
1	1	1	1	1	0	0	0	24-bitowe
0	0	0	0	0	0	0	0	64-bitowe
1	1	1	1	0	0	0	0	32-bitowe

4.5.1.4. Blok sterowania magistralami (BIU)

W bloku sterowania magistralami możemy wyróżnić następujące układy:

- bufory magistrali danych (transcivery),
- układy wejściowe i wyjściowe magistrali adresowej,
- bufory zapisu,
- układy sterowania rodzajem cyklu magistrali (standardowy lub seryjny - burst),
- sygnały sterowania dostępem do magistral,
- układy komunikacji z zewnętrzną pamięcią cache,
- sygnały komunikacji z wewnętrzną pamięcią cache,
- układy generacji i kontroli parzystości.

4.5.1.5. Część wykonawcza

Część wykonawcza zawiera dwie 32-bitowe jednostki aryunetyczno-logiczne i zestaw współpracujących z nią rejestrów przedstawiony na rysunku 4.15. Tworzą następujące grupy:

- Rejestry ogólnego przeznaczenia EAX, EBX, ECX, EDX, **EBP, EDI,** ESI, ESP. Są to rejestry 32-bitowe, jednak każdy z nich zawiera rejestry AX, BX, CX, DX itd., będące rejestrami 16-bitowymi. Każdy z nich może być z kolei używany ja-

ko dwa oddzielne rejestry 8-bitowe. Noszą wówczas przykładowo oznaczenia AH. AL, BH, BL itd. Każdy z wymienionych rejestrów może zawierać dane, na których wykonujemy obliczenia (czyli operandy), oraz wyniki obliczeń. Ponadto poszczególne rejestry pełnią dodatkowe funkcje. EAX jest akumulatorem - pośredniczy na przykład w wymianie informacji z układami wejścia/wyjścia. Rejestr EBX (ang. *base register)* może być używany jako rejestr bazowy w adresowaniu pośrednim, natomiast rejestr ECX (ang. *count register)* może pełnić rolę licznika w instrukcjach pętli.

- Rejestr EBP zwany wskaźnikiem bazy (ang. *base pointer)* oprócz przechowywania danych i wyników umożliwia operacje na stosie bez zmiany zawartości rejestru ESP (porównaj podrozdział 3.3.2.4). Jest to na przykład wykorzystywane do przekazywania przez stos argumentów do funkcji w języku C czy w Pascalu.
- Rejestry ESI i EDI pełnią dodatkowe funkcje przy operacjach na łańcuchach danych. Rejestr ESI (ang. *source index)* zawiera adres źródła, a EDI (ang. *destination index)* zawiera adres docelowy dla danych przy operacjach łańcuchowych.
- ESP (ang. *stack pointer)* jest wskaźnikiem stosu, którego zadania opisano w punkcie 3.3.2.4.

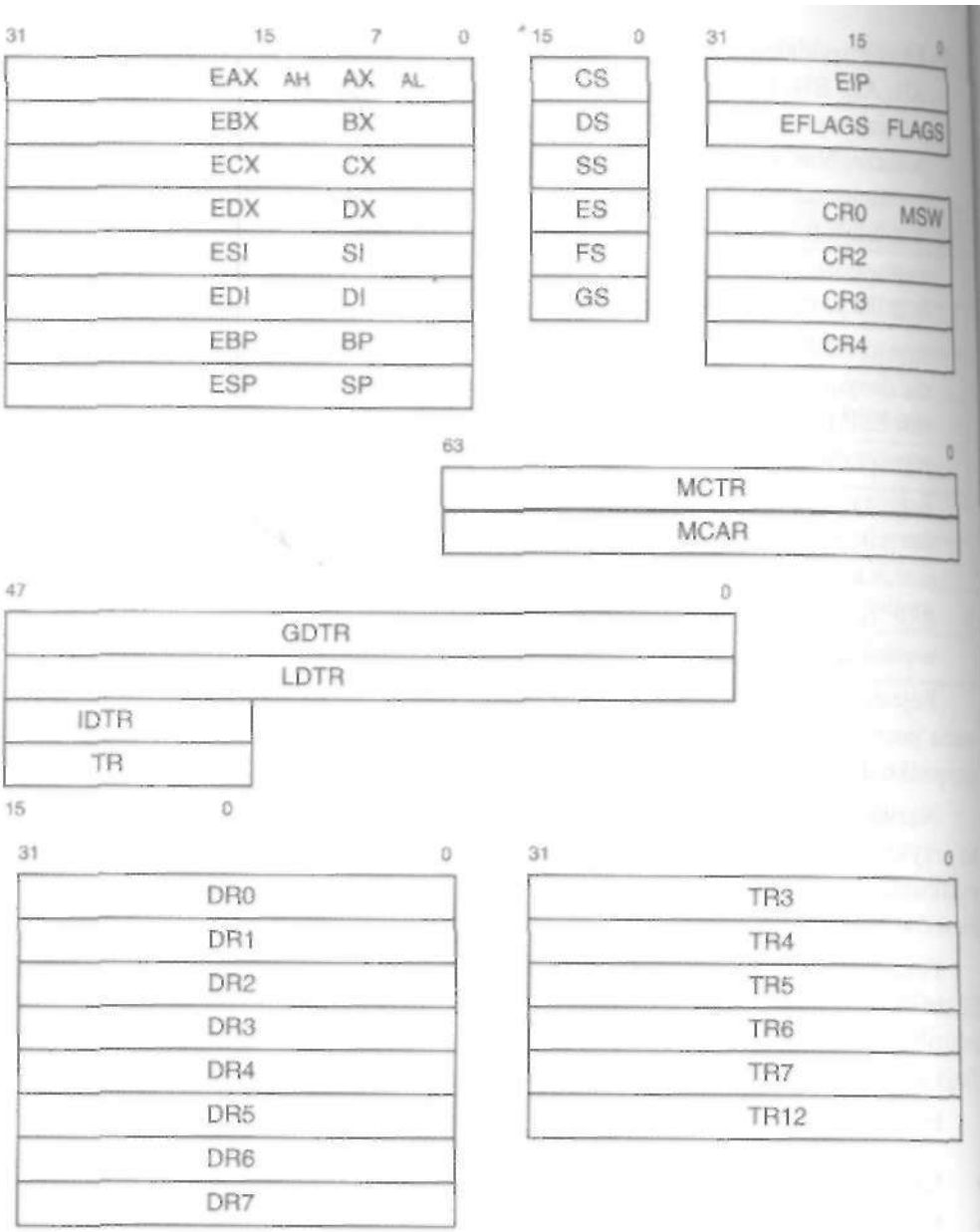
Rejestry EAX, EBX, ECX, EDX, EBP, ESP, ESI, EDI mogą służyć do adresowania pośredniego. Nie dotyczy to jednak ich odpowiedników 16-bitowych. W tym przypadku do adresowania pośledniego służą jedynie rejestry BX, BP, SI i DL

Nazwa rejestru poprzedzona literą E oznacza odwołanie do rejestru 32-bitowego. \a przykład EAX jest nazwą 32-bitowego akumulatora. Użycie nazwy AX powoduje wykonanie operacji na rejestrze 16-bitowym, a nazwy AL lub AH odnoszą się do rejestrów 8-bitowych.

Ponadto z jednostką arytmetyczno-logiczną współpracuje rejestr flagowy EFLAGS. Zawiera zestaw siedemnastu flag, który możemy podzielić na flagi stanu, kontrolne i systemowe. Poniżej podajemy znaczenie wybranych flag kontrolnych i flagi stanu. Pełny zestaw flag można znaleźć na przykład w pozycjach [9] lub [4].

Flagi stanu:

- CF (ang. *carryflag)* - przeniesienie/pożyczka
- PF (ang. *parityflag)* - parzystość
- AF (ang. *auxiliary carry flag)* - pomocnicze przeniesienie/pożyczka z 4. bitu (dla kodu BCD)
- ZF (ang. *zero flag)* - flaga wyniku zerowego
- SF (ang. *signflag)*- flaga znaku
- OF (ang. *overflow flag)* - flaga przepełnienia (przekroczenie zakresu w kodzie **U2).**



Rysunek 4.15. Rejestry procesora Pentium dostępne programowo

Flagi kontrolne:

- **TF** (ang. *trap flag*) – flaga pracy krokowej (pułapka)
- **IF** (ang. *interrupt enable flag*) – flaga zezwolenia na przyjmowanie przerwań (INTR)
- **DF** (ang. *direction flag*) – flaga kierunku wykonywania operacji łańcuchowych.

Rejestry CRO, CR2-S-CR4 są rejestrami sterującymi pracą określonych układów I mcesora (na przykład trybem pracy procesora, sposobem pracy pamięci cache, włączaniem bądź wyłączaniem stronicowania) lub uczestniczą w realizacji określonych ooeracji (na przykład przy realizacji stronicowania pamięci).

Rejestry DRx są rejestrami uruchomieniowymi (ang. *Debug Registers)*. Umieszczane są w nich adresy pułapek oraz ich status. Rejestry TRx wspomagają testowanie nrócesora. Rejestry TR6 i TR7 służą do testowania układu TLB, natomiast rejestry TR3+TR5 są używane do testowania wewnętrznej pamięci cache.

Rejestry MC AR i MCTR służą do obsługi błędnych cykli magistrali. Rejestr MCAR (ang- *Machinę Check Address Register)* zawiera adres nieudanej operacji, a rejestr MCTR (ang. *Machinę Check Type Register)* jej typ.

W skład części wykonawczej wchodzą też dwie jednostki arytmetyczno-logiczne. Podobnie jak potoki, oznaczone są literami U i V. Wykonują operacje logiczne oraz staloprzecinkowe operacje arytmetyczne dla potoków U i V. Jednostka arytmetyczno-logiczna U może zakończyć wykonywanie instrukcji przed jednostką V, ale nie na odwrót.

4.5.1.6. Pamięć wirtualna w procesorze Pentium

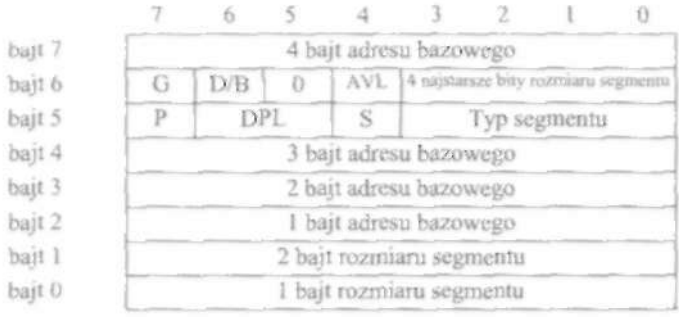
W procesorze Pentium, podobnie jak i w innych procesorach lej rodziny, począwszy od 80286, w trybie chronionym zmienia się interpretacja zawartości rejestrów segmentowych. Zawartość odpowiedniego rejestru segmentowego jest selektorem wybierającym odpowiednią pozycję w tablicach deskryptorów. Dwa z szesnastu bitów w tym rejestrze, oznaczane jako RPL, określają poziom ochrony zadania żądającego dostępu do segmentu. Pozostałych 14 bitów jest używanych do wyboru żądanego f deskryptora. Bit Tl decyduje, czy zostanie użyta tablica deskryptorów globalnych GDT, czy tablica deskryptorów lokalnych LDT, pozostałych 13 bitów jest indeksem do określonej tablicy deskryptorów, czyli wybiera określoną jej pozycję. 14 bitów umożliwia wybór 2^{14}= 16384 różnych deskryptorów segmentów, a więc program ma dostęp do 16 K segmentów. Deskryptor wskazywany przez dany selektor jest używany między innymi do pobrania adresu bazowego zaadresowanego segmentu oraz do sprawdzenia jego rozmiaru. Rozmiar segmentu jest limitowany wielkością przesunięcia (adresu efektywnego) i wynosi 2^{32} = 4 GB. Wynika to także z zawartości określonych pól deskryptora. Maksymalny rozmiar segmentu zapisywany jest na 20 bitach, przy czym jego wielkość może być podawana w bajtach lub w liczbie stron 4-kilobajtowych. O tym, która z możliwości jest używana, informuje bit G (ang. *granularity)*. G=0 oznacza rozmiar w bajtach, G=1 - w stronach 4-kilobajtowych. Przy G=0 maksymalny rozmiar segmentu wynosi 1 MB (2^{20} x 1 B = 1 M x 1 B = 1 MB), a przy G = 1 - 4 GB (2^{20} x 4 k B = 1 M x 4 k B = 4 GB). Mnożąc maksymalną wielkość segmentu przez ich liczbę, otrzymujemy maksymalną wielkość przestrzeni wirtualnej pamięci, jaką może dysponować pojedynczy program. Wynosi ona 16 K x 4 GB = 64 TB. Prze-

strzeń ta dzielona jest na dwie części: przestrzeń globalną wspólną dla wielu zadaj i przestrzeli lokalną należącą do danego zadania.

Format selektora wybierającego określony deskryptor oraz format deskryptorj pokazane są na rysunku 4.16. Nie będziemy dokładnie omawiać poszczególnych pft zwrócimy jednak uwagę na najistotniejsze:

- bit P jest bitem obecnos'ci segmentu,w pamięci,
- pola RPL i DPL związane są z określaniem poziomów ochrony poszczególnych obszarów pamięci,
- bit TI decyduje, czy zostanie użyta tablica deskryptorów globalnych, czy lokalnych
- indeks selektora wybiera konkretny deskryptor w tablicy deskryptorów,
- adres bazowy jest adresem początku segmentu w pamięci.

Rysunek 4.16a. Format rejestru segmentowego jako selektora

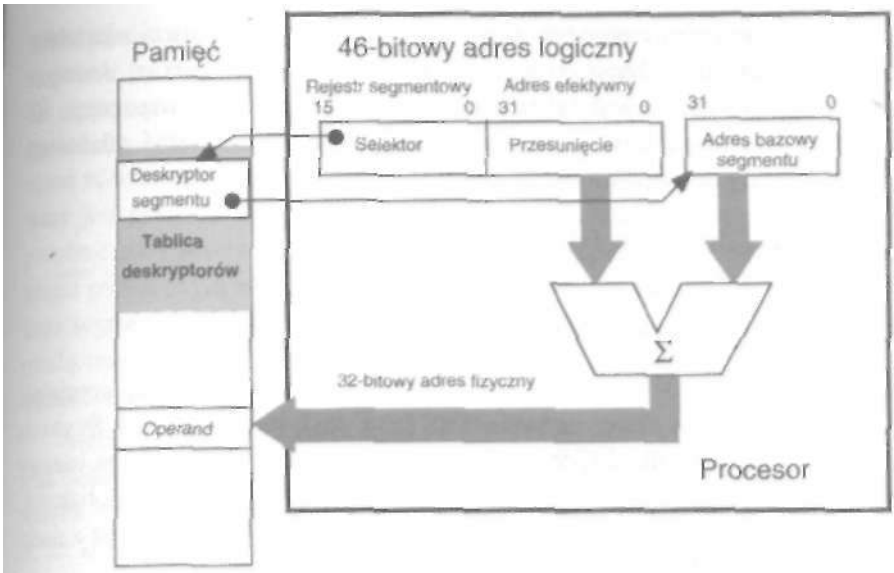


Rysunek 4.16b. Ogólny format deskryptora segmentu

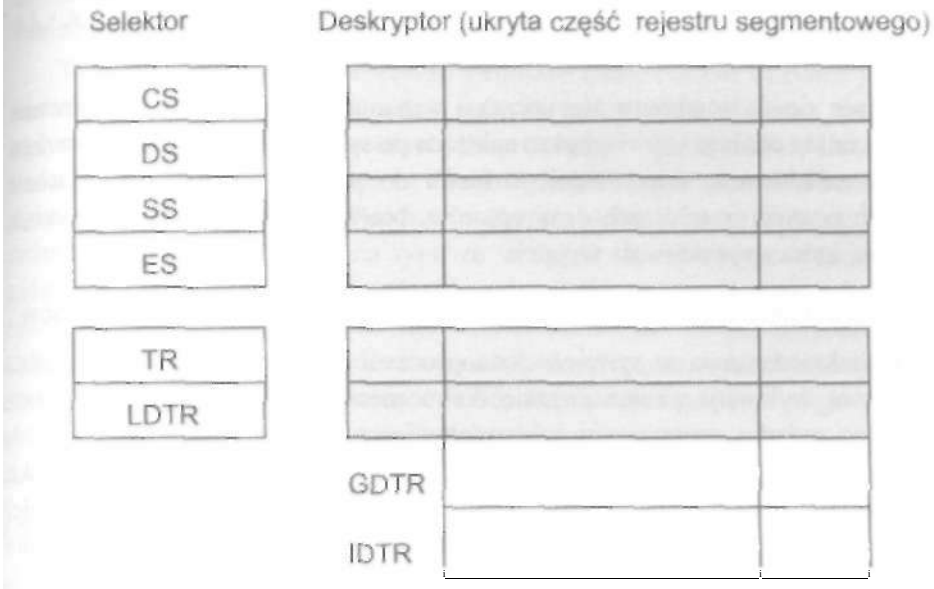
Układ dokonujący translacji adresu logicznego (czyli wirtualnego; na adres fi. zyczny pokazany jest na rysunku 4.17.

Adres fizyczny obliczany jest jako suma adresu bazowego odczytanego z odpowiedniej pozycji tablicy deskryptorów i wartości adresu efektywnego. Sposób wyliczenia wartości adresu efektywnego jest taki sam jak w procesorze 8086.

W celu przyspieszenia dostępu do deskryptorów segmentów aktualnie używanych przez program są one automatycznie umieszczane (w momencie załadowania selektora do rejestru segmentowego) w rejestrach niedostępnych programowo, znajdujących się w MMU, tworzących swego rodzaju pamięć cache. Na rysunku 4.18 pokażującym rejestry procesora związane z adresowaniem wirtualnym rejestry te zostały wycieniowane.



Rysunek 4.17. Układ translacji adresu wirtualnego na fizyczny



Rysunek 4.18. Rejestry związane z adresowaniem w trybie wirtualnym

W pamięci znajdują się trzy rodzaje tablic deskryptorów. Pierwsze dwie dotyczą programu i danych. Jest to po pierwsze tak zwana tablica deskryptorów globalnych GDT (ang. *global descriptor table),* obsługująca na przykład bloki pamięci wspólne dla kilku programów. Istnieje tylko jedna tablica deskryptorów globalnych. Drugi rodzaj to tablice deskryptorów lokalnych LDT (ang. *[ocal descriptor table),* obsługu-

jące bloki pamięci przydzielone pojedynczym programom. System tworzy taką tablicę dla każdego realizowanego zadania. Trzecia tablica - IDT (ang. *intenupt descripior labie),* obsługuje przerwania i jest także tablicą pojedynczą. Przy rozpoczęciu lub kontynuacji danego zadania adresy początkowe tych tablic muszą zostać załadowane przez system operacyjny do odpowiednich rejestrów:

- adres tablicy GDT do rejestru GDTR,
- selektor tablicy LDT do rejestru LDTR,
- adres tablicy IDT do rejestru 1DTR.

W przypadku kontynuacji zadania wartości te są odczytywane ze specjalnej struktury zwanej **segmentem stanu zadania TSS** (ang. *task stare segment).* Struktura taka jest tworzona przez SO dla każdego realizowanego zadania, a jej adres bazowy przechowywany w odpowiednim deskryptorze w tablicy deskryptorów globalnych. Selektor wybierający ten deskryptor dla aktualnie realizowanego zadania jest załadowany do rejestru zadania TR.

Deskryptory segmentów zawierają takie informacje, jak wielkość segmentu, jego adres bazowy, wymagany poziom uprzywilejowania i pewne atrybuty, między innymi bit obecności. Dokładny opis formatu różnego rodzaju deskryptorów oraz schematy translacji adresu wirtualnego na fizyczny można znaleźć w pozycji [11].

Odwołanie się do segmentu nieobecnego w pamięci powoduje wygenerowanie wyjątku. Procedura obsługi tego wyjątku, należąca do systemu operacyjnego, powinna spowodować załadowanie odpowiedniego bloku do pamięci oraz zmodyfikowanie odpowiednich pozycji w tablicach deskryptorów. Następnie ponownie wykonywana jest instrukcja, która spowodowała wyjątek.

4.5.1.7. Mechanizmy wspomagania pracy wielozadaniowej i ochrony zasobów

Praca wielozadaniowa w systemie jednoprocesorowym polega na przełączaniu zadań. Procesor wykonuje pewne zadanie do momentu upływu określonego czasu przydzielonego zadaniu, momentu, w którym chwilowo brak informacji potrzebnej do kontynuowania zadania, lub do momentu żądania zmiany zadania przez użytkownika. Wówczas stan aktualnie wykonywanego zadania jest zapamiętywany w odpowiednich strukturach systemowych i procesor przechodzi do wykonywania innego zadania. Ponadto w sytuacji wykonywania kilku zadań jednocześnie (w sensie przełączania się pomiędzy zadaniami) istnieje potrzeba kontrolowania dostępu poszczególnych zadań zarówno do informacji, jak i innych zasobów systemu. Nie można na przykład dopuścić, by jeden program zapisywał swoje dane w obszarze pamięci przydzielonej innemu programowi.

Podobnie jak poprzednio, sytuację taką można by obsługiwać programowo, natomiast procesor Pentium, podobnie jak procesory, począwszy od 80286, oferuje

wspomaganie sprzętowe tej obsługi. Stan przerywanego zadania przechowywany jest w pamięci w systemowych segmentach stanu zadania TSS (ang. *task state segment)*. W segmentach tych przechowywane są wszystkie wartości rejestrów procesora potrzebne do kontynuowania zadania. Deskryptory segmentów stanu zadania umieszczone są w tablicy GDT. Ponadto procesor ma rejestr zadania TR, do którego załadowany jest selektor wybierający TSS aktualnie wykonywanego zadania. Rejestr ten, podobnie jak rejestry LDTR czy segmentowe, współpracuje z niewidocznym dla programu podręcznym rejestrem deskryptora zawierającym deskryptor aktualnie używanego segmentu zadania. Rejestr TR jest automatycznie przeładowywany w wypadku przełączenia zadania. Istnieją też rozkazy, które umożliwiają programową zmianę zawartości tego rejestru.

Mechanizmy ochrony informacji są związane z zawartością pól deskryptorów oraz selektorów określających poziomy ochrony. Jeżeli wartość poziomu ochrony podana w selektorze jest wyższa lub równa poziomowi ochrony deskryptora segmentu, do którego realizowany jest dostęp, to wówczas cykl magistrali jest realizowany. W przeciwnym wypadku zgłaszany jest wyjątek. Dokładny opis mechanizmów ochrony wykracza daleko poza zakres tej pracy. Można go znaleźć np. w pozycji [11].

4.5.1.8. Tryb wirtualny 8086 (V86)

Trzecim trybem pracy procesora Pentium, traktowanym czasami jako odmiana wirtualnego trybu chronionego, jest tryb wirtualny 8086. Tworzy środowisko dla wykonywania zadań przygotowanych dla procesora 8086 w wielozadaniowym środowisku trybu chronionego procesora Pentium. Zadania procesora 8086 mogą więc być wykonywane przez procesor Pentium w dwóch trybach, rzeczywistym i wirtualnych zadań 8086. Różnica polega na tym, że w trybie rzeczywistym wykonujemy jedno zadanie, podczas gdy tryb wirtualnych zadań 8086 można stosować do określonych zadań wykonywanych w środowisku wielozadaniowym. Po przełączeniu się do tego trybu w celu wykonania konkretnego zadania procesor ten zachowuje się wobec określonego programu jak procesor 8086, lecz po powrocie z niego pracuje dalej w trybie chronionym, umożliwiając pracę wielozadaniową oraz dostęp do całej pamięci operacyjnej.

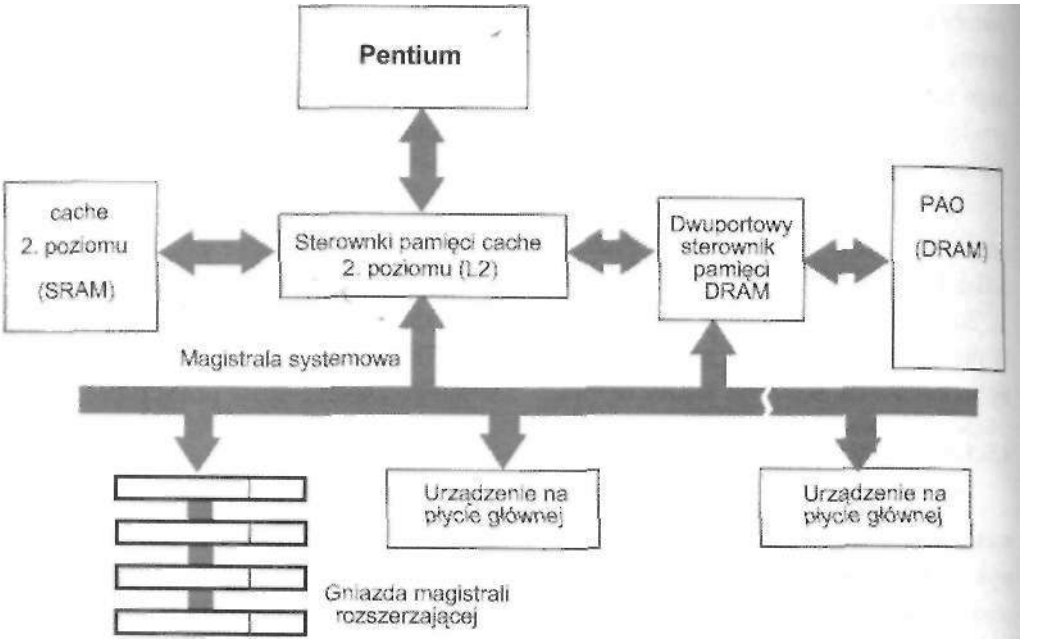
4.5.1.9. Pamięć cache w procesorze Pentium

W systemach z procesorem Pentium możliwe jest stosowanie zarówno wewnętrznej, jak i zewnętrznej pamięci cache.

Jak już wcześniej powiedziano, wewnątrz procesora Pentium zastosowano dwie pamięci cache, każdą o pojemności 8 kB. Jedna z nich przeznaczona jest do przechowywania kodów instrukcji i nosi nazwę codę cache lub I-cache. W drugiej przechowywane są dane i wyniki przetwarzania informacji i nasi ona nazwę data cache. Obie pamięci są dwublokowymi pamięciami asocjacyjnymi (patrz punkt 3.7.4). Pamięć

Code-cache ściśle współpracuje z układem prefetchera. Długość linii w każdej z pamięci wynosi 32 bajty, dlatego przy 64-bitowej magistrali danych może być ona wypełniona w czterech cyklach dostępu (w trybie burst).

Sposób podłączenia zewnętrznej pamięci cache w systemie z procesorem Pentium pokazuje rysunek 4.19.



Rysunek 4.19. Pamięć cache L2 w systemie z procesorem Pentium

Pamięć cache L2 może stosować zarówno politykę Write-back, jak i Write-through. Niezależnie od polityki musi być zapewniona zgodność (w momencie użycia) zawartości obu pamięci cache i pamięci głównej. Sytuacja staje się jeszcze bardziej skomplikowana w przypadku systemów wieloprocesorowych. W systemach takich stosuje się model zachowania zgodności oznaczany jako MESI, który wykorzystywany jest również w procesorze Pentium. Nazwa pochodzi od pierwszych liter czterech stanów, w których może znajdować się linia pamięci cache: Modified - zmieniona; Exclusive -jedyna (dosł. wyłączna); Shared - wspólna; Invalid - nieważna. Opis modelu MESI również wykracza poza ramy tej książki. Można go znaleźć w pozycji [9].

4.5.1.10. Restart procesora Pentium

Jednym z wejść magistrali sterującej mikroprocesora Pentium jest RESET. Aktywny sygnał na tym wejściu powoduje wpisanie wartości początkowych do określonych rejestrów procesora i rozpoczęcie wykonywania programu od określonego, zawsze tego samego miejsca pamięci. W przypadku mikroprocesorów rodziny 80x86

nazwa sygnału RESET jest o tyle niefortunna, że sugeruje wpis do rejestrów wartości zerowych, co nie w każdym przypadku jest prawdą. Restart procesora Pentium powoduje wpisanie do rejestrów wartości początkowych podanych w tabeli 4.5. Do pozostałych rejestrów wpisywane są wartości zerowe (CR2, CR3, CR4, SS, DS, ES, GS, FS, EAX, EBX, ECX, ESI, EDI, EBP, ESP, DR0-DR3, TR12) lub ich stan jest niezdefiniowany.

Tabela 4.5. Zawartość rejestrów po restarcie procesora

Nazwa rejestru	Wartość początkowa
EFLAGS	00000002h
EIP	0000FFF0h
CS	F000h
CR0	60000010h
EDX	000005xxh
DR6	FFFF0FF0h
DR7	00000400h
cache danych i kodu	nieważne

Z wartości wpisanych do rejestru CRO wynika, że procesor Pentium lozpuczyna pracę w trybie rzeczywistym.

Jedną z bardzo ważnych konsekwencji takiego ustalenia wartości początkowych wpisywanych do rejestrów jest adres miejsca w pamięci, z którego mikroprocesor pobierze pierwszą instrukcję do wykonania (czyli miejsca, od którego rozpocznie pracę). Zgodnie z podaną regułą obliczania adresu wyniesie on:

```
CS =  F000h        F0000h
IP =  FFF0h      +FFF0h
                 --------
                 AF = FFFF0h
```

Oznacza to, ze procesor pobierze pierwszą instrukcję z komórki pamięci odległej o 16 bajtów od końca pierwszego megabajtu pamięci (adres FFFFFh - 1 MB, FFFFOh - start pracy procesora). Fakt ten warto zapamiętać, gdyż będzie miał duże znaczenie dla architektury komputerów typu IBM PC.

Oprócz wejścia RESET procesor Pentium ma wejście oznaczone jako INIT. Wejście to zapewnia kompatybilność procesora Pentium z procesorem 80286. Procesor 80286 mógł wyjść z trybu wirtualnego jedynie przez restart. Własność ta jest

wykorzystywana przez część programów napisanych dla procesorów 80286, na przykład dla tak zwanych DOS extenders. Procesor wchodził np. w tryb chroniony w celu umożliwienia dostępu do pamięci Extended, po czym wracał do trybu rzeczywistego, odtwarzając stan swoich rejestrów. W procesorze Pentium taka operacja jest możliwa przy użyciu wejścia INIT, które można traktować jako częściowy restart. Pozostawia on niezmienione wartości w obu wewnętrznych pamięciach cache, buforach zapisu B1U, rejestrach NPU. Nie są zmieniane także wartości bitów CD i NW w rejestrze CRO, decydujące o stanie pamięci cache.

4.5.1.11. Praca potokowa

W procesorze Pentium instrukcje wykonywane są **potokowo. Praca potokowa** (ang. *pipelining)* jest rozwinięciem koncepcji prefetchingu. Polega na równoległym wykonywaniu kilku faz realizacji rozkazu. W procesorze Pentium instrukcje realizowane są w pięciu fazach:

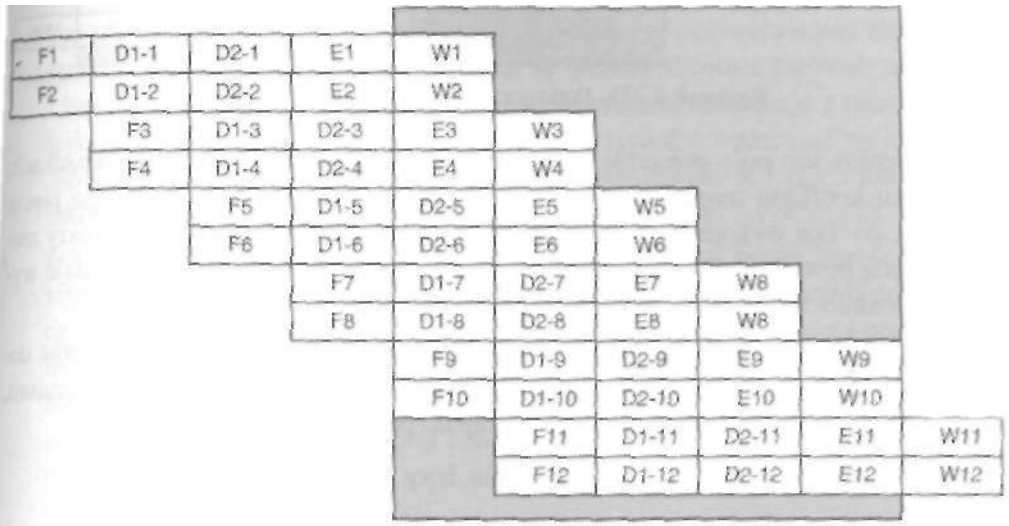
- pobranie kodu instrukcji - F,
- pierwszy etap dekodowania instrukcji - D1,
- drugi etap dekodowania instrukcji - D2,
- wykonanie - E,
- zapis do rejestrów - W.

W fazie F wczytywany jest z pamięci kod instrukcji. W pierwszym etapie dekodowania (D1) instrukcji ustalany jest rodzaj operacji oraz tryb adresowania. W etapie D2 obliczany jest adres efektywny argumentów, ewentualnie przygotowywane są argumenty natychmiastowe. W fazie E realizowany jest (jeśli jest konieczny) dostęp do pamięci i wykonywanażądana operacja. W ostatniej fazie rezultaty operacji zapisywane są w określonym rejestrze (jeżeli wymaga tego instrukcja).

Ponieważ w procesorze Pentium istnieją układy przeznaczone do realizacji dwóch potoków, w sprzyjających warunkach możliwe jest wykonanie w pięciu taktach dziesięciu instrukcji (pięciu w potoku U i pięciu w potoku V), co daje realizację dwóch instrukcji na jeden takt. Oznacza to, że Pentium jest procesorem superskalarnym (czyli wykonującym więcej niż jedną instrukcję na takt). Opisaną sytuacją ilustruje rysunek 4.20. W zacieniowanym obszarze w ciągu pięciu taktów zegara została zakończona realizacja dziesięciu instrukcji.

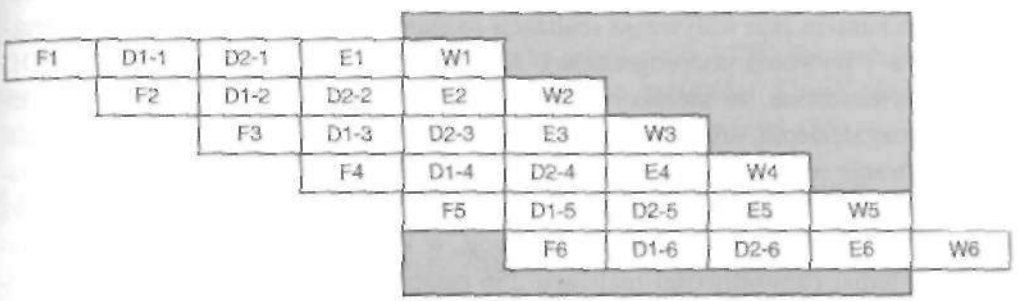
Oczywiście sytuacja taka nie zawsze jest możliwa. Przykładowo, jeżeli rozkaz z potoku V musiałby korzystać z wyników rozkazu przetwarzanego w kolejce U, równoległe przetwarzanie tych rozkazów nie byłoby możliwe. Dlatego też w trakcie pierwszej fazy dekodowania rozkazów (D1) sprawdzane jest także, czy rozkazy z potoku U i V mogą być przetwarzane równocześnie, czyli czy tworzą parę. Jeżeli tak, są przekazywane równocześnie do dalszej realizacji (fazy D2 i następne). Jeżeli

instrukcje nie tworzą pary, realizacja instrukcji z potoku V jest wstrzymywana i następnie przekazywana do realizacji do potoku U. Zasady łączenia instrukcji w pary podane są >a przykład w pozycjach [9] lub 119]. Wykorzystanie tych zasad pozwala ootymalizować programy pisane dla procesorów Pentium.



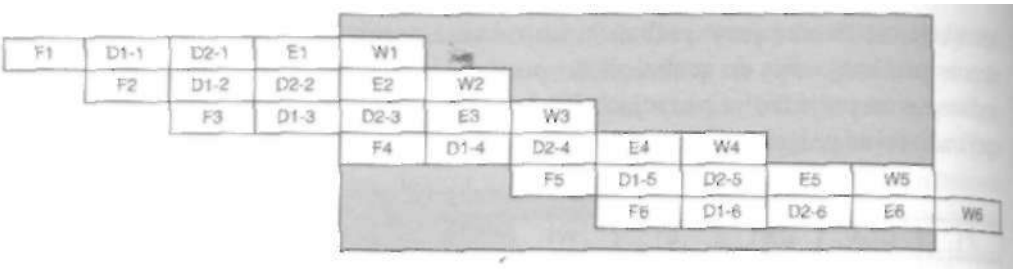
Rysunek 4.20. Praca potokowa w procesorze Pentium

Ponadto realizacja poszczególnych faz w pracy potokowej jest ograniczona dostępem do zasobów systemu, przede wszystkim do magistral. Wyjaśnimy to na przykładzie, zakładając dla uproszczenia, że wykonywany jest jeden potok (ciąg) instrukcji.



Rysunek 4.21a. Teoretyczna realizacja prefetchingu

Teoretycznie można program realizować tak, jak to pokazano na rysunku 4.21a. W rzeczywistości konflikt dostępu do magistral spowoduje wolniejszą realizację instrukcji. Jeżeli na przykład faza El wymaga dostępu do magistrali (zapis lub odczyt pamięci), to faza F4 musi zostać opóźniona o co najmniej jeden takt (czyli zakładamy, że E2 nie używa magistral). Podobnie będzie z fazą F5 w stosunku do fazy E3. Opisana sytuacja spowoduje pracę potokową pokazaną na rysunku 4.2Ib.



Rysunek 4.2Ib. Praktyczna realizacja prefetchingu

Zacieniowane pole pokazuje szybkość realizacji instrukcji w obu przypadkach. Przy braku konfliktu magistral (rys. 4.6a) w taktach kończona jest realizacja pięciu instrukcji, co daje wykonanie jednej instrukcji na jeden takt zegarowy. Konflikty magistrali będą powodować wykonanie pięciu instrukcji w siedmiu taktach, co daje wykonanie s'rednio 5/7 instrukcji na 1 takt zegarowy.

Zwracamy uwagę, że konflikt ten nie wystąpi, jeżeli pamięć programu oraz danych i obsługujące je magistrale są rozdzielone. Taką sytuację mamy w Pentium, jeżeli dostęp następuje do pamięci cache, a nie do pamięci głównej.

4.5.1.12. Przewidywanie rozgałęzień

Kolejnym problemem występującym przy realizacji pracy potokowej jest występowanie rozgałęzień w programie. Jeżeli przy realizacji prefetchingu zakładamy liniowe pobieranie instrukcji (z komórek pamięci leżących jedna obok drugiej), to w przypadku napotkania rozgałęzienia programu (np. instrukcji skoku) tworzona kolejka musi zostać wyczyszczona, gdyż zawiera błędne instrukcje. W celu poprawy skuteczności wstępnego pobierania instrukcji (prefetchingu) w procesorze Pentium wprowadzono mechanizm przewidywania realizacji rozgałęzień (czyli wykonywania instrukcji skoków i wywołań podprogramów). Jeżeli w pierwszej fazie dekodowania (Dl) zostanie stwierdzone, że którakolwiek z dwóch instrukcji z potoku U lub V jest instrukcją rozgałęzienia, uruchamiane są układy przewidywania realizacji rozgałęzienia. Przewidywanie realizacji rozgałęzienia dokonywane jest na podstawie historii realizacji rozgałęzień przechowywanej w buforze rozgałęzień BTB, BTB jest czteroblokową pamięcią asocjacyjną zawierającą 256 pozycji, w których przechowywana jest informacja na temat częstotliwości realizacji 256 ostatnio napotykanych rozgałęzień programu wraz z ich adresami docelowymi (w przypadku realizacji).

W trakcie realizacji programu prefetcher napełnia jedną z dwóch kolejek (ang. *prefetch buffers)* instrukcjami pobieranymi z pamięci codę cache (w przypadku trafienia) bądź z pamięci głównej za pośrednictwem BIU (chybienie). Kolejka używana w danym momencie nazywana jest kolejką aktywną. Jeżeli do pierwszego stopnia dekoderów trafia instrukcja rozgałęzienia, układy przewidywania rozgałęzień podejmują decyzję, czy rozgałęzienie będzie realizowane, czy też nie.

- Jeżeli przewidywana jest realizacja rozgałęzienia, prefetcher przełącza się na drugą kolejkę, rozpoczynając jej wypełnianie instrukcjami pobranymi spod adresu docelowego rozgałęzienia. Jeżeli w fazie realizacji instrukcja rozgałęzienia jest realizowana (przewidywanie było prawidłowe), instrukcje w potoku bezpośrednio po instrukcji rozgałęzienia są prawidłowe i realizacja programu przebiega bez przerw. Jeżeli rozgałęzienie nie jest realizowane (przewidywanie błędne), instrukcje w potoku oraz bieżącej kolejce są nieprawidłowe i zarówno potok, jak i kolejka muszą zostać wyczyszczone. Realizowane są instrukcje z pierwszej kolejki, gdyż są instrukcjami prawidłowymi w przypadku braku realizacji rozgałęzienia. Program jest realizowany z niewielkim opóźnieniem (trzech lub czterech taktów zegara) potrzebnym do przejścia instrukcji z kolejki przez potok.
- Jeżeli przewidywany jest brak realizacji rozgałęzienia, prefetcher kontynuuje wypełnianie pierwszej kolejki kolejnymi instrukcjami następującymi po instrukcji skoku. W przypadku prawidłowego przewidywania potok zawiera prawidłowe instrukcje i program realizowany jest bez opóźnień. Jeżeli przewidywanie jest błędne, czyli rozgałęzienie będzie realizowane, zarówno potok, jak i kolejka zawierają nieprawidłowe instrukcje i muszą zostać wyczyszczone. Prefetcher rozpoczyna pobieranie instrukcji spod adresu docelowego rozgałęzienia. Występuje opóźnienie w realizacji instrukcji.

W każdym przypadku po dojściu instrukcji rozgałęzienia do fazy realizacji modyfikowana jest odpowiednio zawartość BTB.

4.6. Procesory RISC

Poniżej przedstawiamy podstawowe cechy procesorów RISC. Prezentujemy własności tych procesorów w tym momencie, ponieważ część idei w nich stosowanych zaimplementowano w kolejnych wersjach procesorów rodziny Intel x86.

4.6.1. Podstawowe przesłanki budowy procesorów RISC

Skrót RISC pochodzi od ***Reduced Instruction Set Computer,*** co odpowiada polskiemu terminowi procesor ze zredukowaną listą rozkazów. Opisuje to jedną z podstawowych cech tych procesorów, której geneza jest następująca. W niektórych ośrodkach badawczych przeprowadzono badania statystyczne, w trakcie których testowano, jak często poszczególne instrukcje maszynowe z listy rozkazów (czyli rozkazy procesora) są używane do realizacji szerokiej klasy programów. Wyniki były zarówno nieco zaskakujące, jak i interesujące, a wykrytą prawidłowość nazwano później „regułą **80/20".** Stwierdzono mianowicie, że **80%** wykonywanego programu jest realizowane za pomocą **20%** instrukcji z listy rozkazów procesorów. Nie oznacza to oczywiście, że pozostałe instrukcje (a jest ich 80%) nie są w ogóle używane, ale że są używane

sporadycznie/bardzo rzadko. Jednak od samej reguły 80/20 znacznie ciekawsze były wnioski i zalecenia dla konstruktorów procesorów, które na tej podstawie opracowano. Otóż postawiono hipotezę, że należy usunąć z listy rozkazów wszystkie instrukcje, które są używane sporadycznie, a które da się zastąpić złożeniem pewnej liczby instrukcji pozostawionych w liście rozkazów. Inaczej mówiąc, w liście rozkazów pozostawiamy tylko instrukcje proste i często używane (pochodzące z 20-procentowej części listy rozkazów) oraz te, których nie da się zastąpić sekwencją instrukcji prostych. Natomiast efektem takiego uproszczenia (skrócenia) listy rozkazów ma być znaczny wzrost szybkości ich wykonywania, co w efekcie ma prowadzić do wzrostu szybkości realizacji programu. Jest to całkowicie realne zarówno dzięki uproszczeniu układu sterowania procesora, jak i możliwości potokowej realizacji instrukcji, które to cechy z kolei wiążą się z prostotą rozkazów.

Szybkość realizacji programu możemy wyrazić prostym i oczywistym wzorem:

$$\text{Czas wykonania} = \frac{\text{liczba instrukcji}}{\text{na program}} \times \frac{\text{liczba taktów zegara}}{\text{na instrukcję}} \times \frac{\text{czas wykonania}}{\text{jednego taktu}}$$

W przypadku wykonywania programu przez procesor RISC pewnemu wzrostowi może ulec liczba instrukcji potrzebnych do realizacji programu, musimy bowiem pojedyncze instrukcje usunięte z list rozkazów zastąpić sekwencją instrukcji prostych. Wzrost ten jednak nie będzie duży, gdyż należy pamiętać, że usunięte instrukcje były używane w realizacji programów bardzo rzadko (co gwarantują przeprowadzone badania statystyczne i reguła 80/20). Ulegną natomiast zmniejszeniu liczba taktów zegara potrzebnych do wykonania jednej instrukcji i czas trwania jednego taktu (co jest z kolei spowodowane wzrostem częstotliwości zegara taktującego pracę procesora). W efekcie dwa ostatnie czynniki spowodują zdecydowany wzrost szybkości wykonywania programu.

W procesorach RISC oprócz redukcji liczby instrukcji o zwiększeniu szybkości ich wykonywania decyduje wiele dodatkowych rozwiązań i czynników, które opisujemy w następnym podpunkcie. Stanowi on jednocześnie zestawienie podstawowych cech procesorów RISC.

4.6.2. Podstawowe cechy procesorów RISC

Poniżej wymieniamy podstawowe cechy procesorów RISC. Do poszczególnych punktów dołączamy komentarz wyjaśniający wpływ konkretnych zastosowanych rozwiązań na szybkość pracy procesora.

Architektura klasycznych procesorów RISC powinna mieć następujące cechy:

- Lista rozkazów zawiera stosunkowo niewielką liczbę prostych rozkazów, za to bardzo szybko wykonywanych. Cechę tę wyjaśniono w poprzednim podrozdziale.

Wszystkie rozkazy wykonywane potokowo, przy czym każdy etap potoku jest realizowany w takim samym czasie, najlepiej w ciągu jednego taktu zegara.

Potokowa realizacja instrukcji jest pomysłem, który w produkcji przedmiotów zastosowano już dwa wieki temu (czyli w XIX wieku), polega bowiem na zastosowaniu swoistej taśmy produkcyjnej do realizacji rozkazów. Zastanówmy się nad przykładem produkcji przedmiotu. Załóżmy, że jakiś przedmiot jest wykonywany przez pięć minut, a do jego wyprodukowania potrzeba wykonania pięciu różnych czynności. Jeżeli przedmioty tego typu będą produkowane przez jednego robotnika, to co pięć minut dostarczy nam kolejny gotowy wyrób, przy czym będzie musiał umieć wykonywać wszystkie pięć czynności (musi więc być robotnikiem wysoko kwalifikowanym). W celu przyspieszenia produkcji można oczywiście zatrudnić pięciu takich robotników i wówczas w wyniku ich pracy, średnio biorąc, będziemy otrzymywać jeden gotowy produkt na minutę. Istnieje jednak jeszcze lepsze rozwiązanie. Jeżeli produkcję przedmiotu można podzielić na pięć czynności, to możemy zatrudnić pięciu robotników, z których każdy umie wykonywać tylko jedną czynność (jest więc albo nisko kwalifikowany - i płacimy mniej, albo tę czynność wykonuje lepiej i szybciej niż jego uniwersalny kolega). Robotników tych ustawiamy jeden przy drugim, tworząc taśmę produkcyjną. Każdy z nich wykonuje swoją czynność, podając następnie półprodukt współpracownikowi. Czas produkcji jednego przedmiotu nie ulegnie oczywiście zmianie, ale tutaj też, średnio rzecz biorąc, uzyskamy jeden produkt na minutę. Analogia ta doskonale wyjaśnia ideę potokowej realizacji rozkazów (patrz str. 172). Jest na tyle ścisła, że można z niej wyciągnąć kolejne wnioski. Załóżmy, że produkcję przedmiotu podzielimy tak, że jedna z czynności zajmuje dwie minuty. Jak często będziemy teraz otrzymywać gotowy wyrób? Niestety co dwie minuty. Taśma produkcyjna jest tak szybka, jak najwolniejsze jej ogniwo. Wszystkie etapy produkcji taśmowej powinny więc trwać tyle samo czasu. To samo dotyczy potoku realizacji rozkazu. Teza ta wyjaśnia przykładowo, czemu w Pentium w potoku realizacji rozkazu dekodowanie podzielono na dwie rozdzielone fazy.

Jeżeli uda się dodatkowo spełnić warunek, że jeden etap potoku jest realizowany w jednym takcie zegarowym, to uzyskamy (w sprzyjających warunkach, co w praktyce nie zawsze się zdarza) średnio realizację jednej instrukcji na jeden takt zegarowy. Kilka kolejnych zaleceń prowadzących do szybkiej realizacji potoku podamy w kolejnych punktach.

Zwróćmy jeszcze uwagę, że zwiększanie liczby etapów potoku (choć nie należy tu przesadzać) powinno prowadzić do możliwości skrócenia taktu zegarowego (więcej czynności, ale za to krótszych), czyli do podniesienia częstotliwości zegara taktującego procesor, co przy zachowaniu własności realizacji jednego etapu w jednym takcie i realizacji wszystkich etapów potoku równolegle daje średnio szybszą realizację instrukcji.

Kolejne cechy procesorów RISC wynikają z prostego faktu, że rozkazy, których i argumenty znajdują się w rejestrach procesora, są wykonywane zdecydowanie szybciej (patrz hierarchia pamięci - podrozdział 3.6.1) niż te, które wymagają wczytania argumentów z pamięci. W ostatnim przypadku jest bowiem konieczna realizacja przez magistralę cyklu dostępu do pamięci, co nawet przy dostępie do pamięci cache wymaga zdecydowanie dłuższego czasu. Wynikają stąd kolejne cechy procesorów RISC.

> Duża liczba uniwersalnych rejestrów roboczych. Cecha ta wymaga jedynie krótkiego komentarza. W procesorach rodziny Intel x86 (do Pentium włącznie) liczba widocznych rejestrów roboczych wynosi 8 (jeżeli uwzględnimy możliwość użycia ich częs'ci jako oddzielnych rejestrów, to będzie to maksymalnie 12). W procesorach RISC liczba rejestrów roboczych rzędu 32, 128 czy nawet 256 nie jest czymś wyjątkowym. Przy tym są to rejestry uniwersalne, czyli mogą nawzajem się zastępować, nie pełniąc jakiejś szczególnej, wyznaczonej roli. I

r Przy konstruowaniu instrukcji w liście rozkazów procesorów RISC rozdzielono operacje przetwarzania informacji od operacji dostępu do pamięci. Inaczej mówiąc, instrukcje przetwarzające informację operują na argumentach, które znajdują się wyłącznie w rejestrach, a instrukcje, które sięgają do pamięci, nie przetwarzają informacji (czyli nie wykonują żadnych działań arytmetyczno-logicznych). Do komunikacji z pamięcią służy więc grupa specjalnie do tego celu przeznaczonych instrukcji. Rozwiązanie to nazwano architekturą **Load-Store** (od angielskich nazw operacji: *storę* - zachowaj w pamięci, *load* - załaduj z pamięci). Pozawala ono miedzy innymi uniknąć przedłużania realizacji pewnych etapów potoku realizacji rozkazu.

> Rozkazy przetwarzające informację operują na trzech rejestrach. W dwóch z nich znajdują się argumenty, do trzeciego ładowany jest wynik wykonywanej operacji. W klasycznych procesorach instrukcja, której argumenty znajdowały się w rejestrach, wynik zapisywała do jednego z nich, powodując zatarcie (utratę) jednego z argumentów. Potrzeba ponownego użycia tego argumentu wymaga wczytania go ponownie z pamięci, czyli wykonania czasochłonnej operacji. Rozwiązanie z trzema rejestrami pozwala tego uniknąć.

Kolejna cecha procesorów **RISC** wynika z chęci zapewnienia realizacji wszystkich etapów potoku w jednym takcie zegarowym.

> Wszystkie operacje przetwarzania informacji (operacje arytmetyczno-logiczne) realizowane są sprzętowo. Wymóg, aby etap przetwarzania informacji był wykonywany w jednym takcie zegarowym, eliminuje praktycznie realizację tego typu operacji za pomocą mikroprogramu (co w normalnych mikroprocesorach miało często miejsce). Prowadzi to co prawda do wzrostu stopnia komplikacji układów

realizujących te operacje, ale przy postępie technologii i wzroście stopnia scalenia to niewielka cena wzrostu szybkości, którą warto zapłacić.

- Stałą długość i format kodu rozkazu. Takie rozwiązanie ułatwia dekodowanie kodu rozkazu (określone informacje są w określonych polach kodu) i przyspiesza je.
- Obecność pamięci cache. Jest w zasadzie oczywista. Umożliwia szybki dostęp zarówno do danych, jak i kodu programu. Częstym rozwiązaniem będzie tu architektura harwardzka (rozdzielona pamięć danych i pamięć programu).

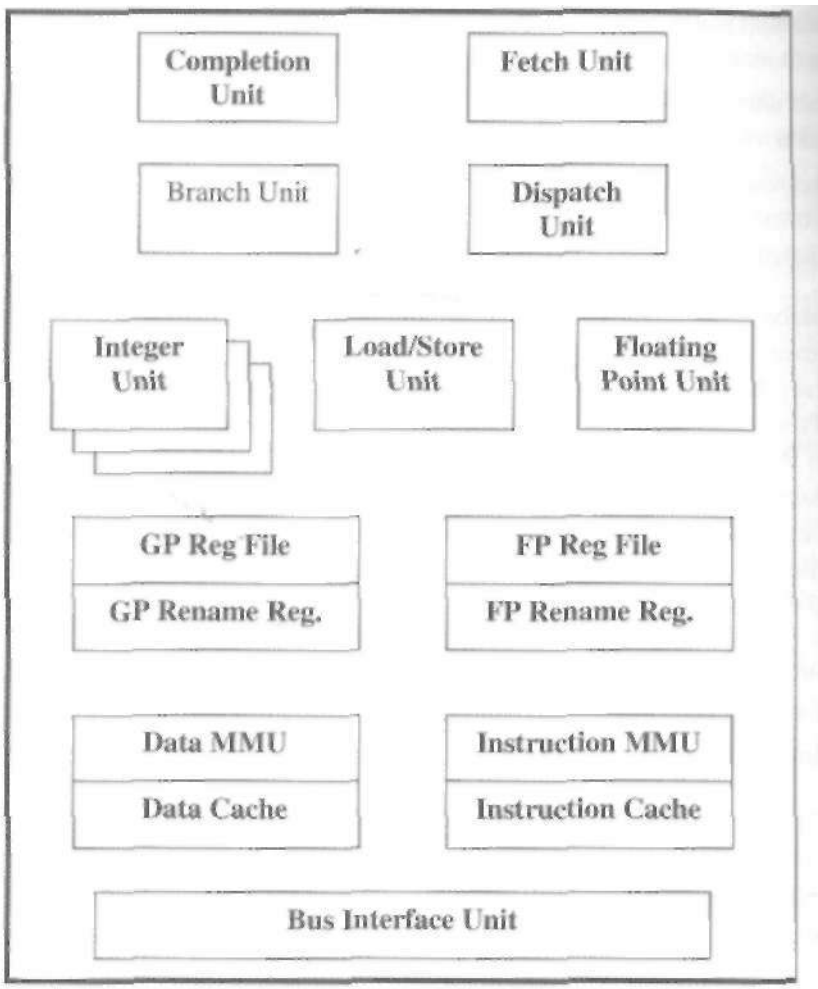
Podane cechy stanowią klasyczną architekturę procesorów RISC. Zostały nieco rozszerzone w opracowaniach firmy IBM (architektura Power), a następnie przy współpracy firm Motorola i Macintosh, tworząc tak zwaną architekturę PowerPC™. Słowo Power jest akronimem pochodzącym od angielskich słów: *Performance Optimization With Enhanced Riscs*. Nazwa wskazuje, że do klasycznych cech procesorów RISC dołączono dodatkowe właściwości poprawiające wydajność tych procesorów. Cechy te jedynie wymienimy, gdyż ich omówienie (zwłaszcza niektórych z nich) wymagałoby rozbudowanego wykładu i dlatego wykracza poza ramy książki. Podstawowymi cechami dodanymi w architekturze Power PC™ są:

1. Możliwość realizacji rozkazów złożonych.
2. Eliminacja tak zwanych opóźnionych skoków.
3. Pełna zgodność ze standardem zapisu zmiennoprzecinkowego IEEE Standard 754.

Czytelników zainteresowanych dokładniejszym omówieniem tych cech odsyłamy do pozycji [17].

Przykładowy schemat blokowy jednego z procesorów rodziny Power PC realizujących omawianą architekturę - Power PC 604, przedstawia rysunek 4.22.

Power PC 604 jest procesorem 32-bitowym o dużej mocy obliczeniowej. Należy zwrócić uwagę na dużą liczbę układów wykonawczych (w sumie sześć jednostek wykonawczych, w tym trzy jednostki stałoprzecinkowe), co powoduje, że Power PC 604 jest procesorem superskalarnym (przetwarza więcej niż jedną instrukcję na takt). Ponadto procesor ten zawiera zestaw 128 rejestrów. Nie będziemy dokładniej omawiać schematu blokowego tego procesora, natomiast prosimy porównać go z później prezentowanymi schematami kolejnych procesorów rodziny x86. Pozwoli to dostrzec ich podobieństwa do procesorów RISC.



Rysunek 4.22. Schemat blokowy procesora Power PC 604

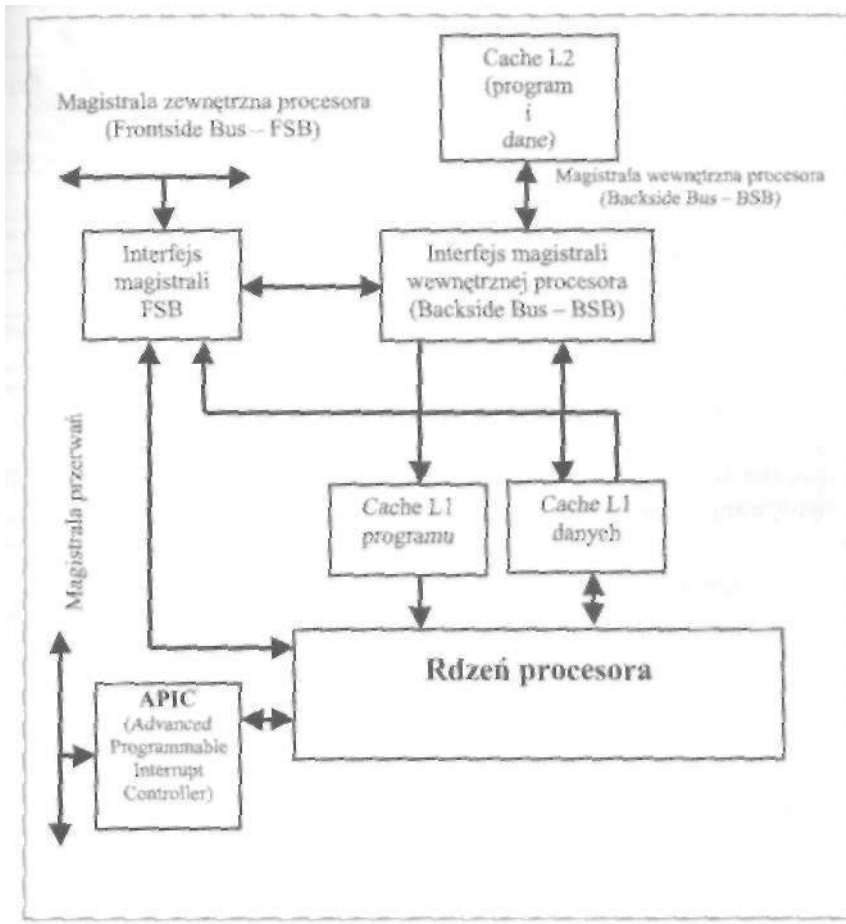
4.7. Pentium Pro™

Pentium Pro™ to wersja procesora Pentium optymalizowana pod kątem obsługi oprogramowania 32-bitowego oraz pracy w systemach wieloprocesorowych (serwery). Projekt tego procesora dal początek architekturze rdzenia zwanej P6. Zaimplementowano w niej rozwiązania stosowane w procesorach RISC, nie rezygnując jednak 7. kompatybilności wstecz z poprzednimi procesorami tej rodziny. Wymagało to ukrycia pewnych operacji. Przykładowo nie można było usunąć istniejących instrukcji z listy instrukcji mikroprocesora. Zamiast tego zmieniono sposób ich realizacji. Instrukcje architektury IA 32, do Pentium włącznie, miały zmienną długość. Począwszy

od Pentium Pro, instrukcje te są konwertowane na ciąg prostych instrukcji podobnych do instrukcji procesorów RISC, o stałej długości, zwanych mikrooperacjami. Część wykonawcza realizuje właśnie mikrooperacje.

Innym rozwiązaniem zapożyczonym od procesorów RISC jest duża liczba rejestrów roboczych. Jest to realizowane za pomocą tak zwanych rejestrów zamienników. Oba rozwiązania opisano bardziej szczegółowo poniżej.

Uproszczony schemat blokowy procesora Pentium Pro™ przedstawia rysunek 4.23.



Rysunek 4.23. Uproszczony schemat blokowy procesora Pentium Pro™

Zwracamy uwagę na istnienie dwóch dróg (magistral) komunikacji procesora z pamięciami. Pierwsza zapewnia komunikację z pamięcią zewnętrzną (pamięcią operacyjną komputera) oraz innymi urządzeniami zewnętrznymi i została później nazwana ***Frontside Bus*** i oznaczona skrótem FSB. Druga zapewnia komunikację z cache L2 i nosiła później nazwę *Backside* ***Bus,*** w skrócie BSB. Magistrale te mogą

pracować równolegle (niezależnie) i dlatego istotnie przyspieszają pracę procesora Rozwiązanie to zostało później, dla procesora Pentium IT, nazwane *Dual Independent Bus* (niezależną podwójną magistralą) i oznaczane skrótem **DIB.**

Innym istotnym rozwiązaniem jest zastosowanie lokalnego zaawansowanego kontrolera przerwań (ang. *APIC - Advanced Programmable Interrupt Controlkr),* Kontroler ten zapewnia obsługę przerwań w systemach wieloprocesorowych, komunikując się z lokalnymi kontrolerami przerwań innych procesorów za pomocą dedykowanej, trójsygnałowej magistrali.

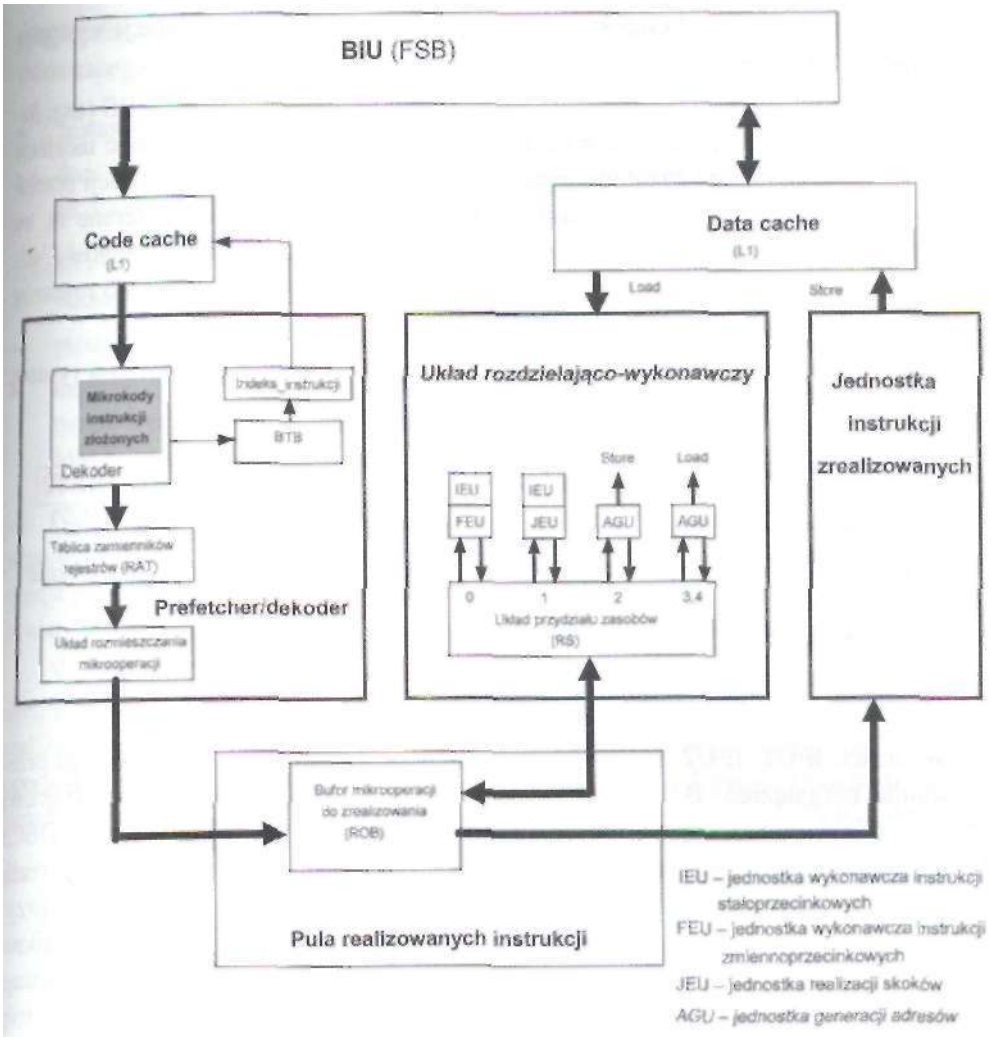
Podstawowe własności Pentium Pro™ to:

- mikroarchitektura dynamicznej realizacji instrukcji (ang. *Dynamie Execution Microarchitecture),*
- realizacja potokowa instrukcji podzielona na 11 faz,
- zintegrowana pamięć cache L2 o pojemności 1 MB lub 2 MB,
- zintegrowany interfejs magistrali,
- realizacja instrukcji optymalizowana dla aplikacji 32-bitowych,
- przystosowanie do pracy wieloprocesorowej (4 procesory).

4.7.1. Dynamiczna realizacja instrukcji

Dynamiczna realizacja instrukcji stanowi dalsze rozwinięcie idei pracy potokowej i przewidywanie realizacji rozgałęzień programu. Pozwala uniknąć części przestojów układów procesora występujących w tradycyjnym sposobie realizacji rozkazów jeden po drugim (pomijamy tu możliwość równoległego wykonywania rozkazów przy większej liczbie potoków). Efekt taki jest uzyskiwany przez wcześniejsze pobieranie rozkazów, a następnie na podstawie przewidywania kolejności realizacji instrukcji (i rozgałęzień programu) optymalny dobór kolejności wykonywania instrukcji na poziomie mikrooperacji. Kolejność ta dobierana jest między innymi tak, aby w miarę możliwości uniknąć wszelkich konfliktów zasobów oraz oczekiwania układów procesora przy realizacji instrukcji. Prowadzi to zwykle do wykonywania mikrooperacji i instrukcji w kolejności niezgodnej z rzeczywistą kolejnością ich wykonywania w programie. Następnym więc koniecznym krokiem jest przywrócenie ich właściwej kolejności po ich realizacji.

Układy związane z dynamiczną realizacją instrukcji pokazane są schematycznie na rysunku 4.24. Kody instrukcji po pobraniu z pamięci codę cache trafiają do układów trzech równoległych dekoderów. O kolejności pobierania kodów instrukcji decyduje indeks instrukcji Next_IP, na którego zawartość ma wpływ układ przewidywania rozgałęzień BTB. Wynikiem dekodowania są mikrooperacje realizujące dany rozkaz (w przypadku złożonych instrukcji pochodzą one z układu mikrokodów instrukcji złożonych MIS - ang. *Microcode Instruction Sequencer).*



Rysunek 4.24. Schemat blokowy Pentium Pro – układy dynamicznej realizacji instrukcji

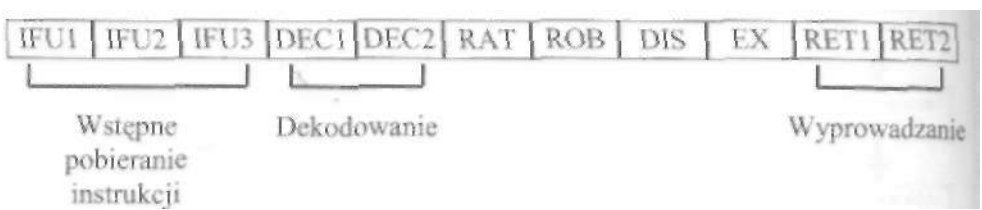
W pewnych przypadkach, na przykład przy chybieniu dla pamięci cache, może być realizowane do 20-30 mikrooperacji wyprzedzających instrukcję, która spowodowała chybienie. Wśród tych instrukcji występuje zwykle kilka instrukcji rozgałęzień, co oznacza kilka wariantów dalszej realizacji programu. Wymaga to znacznie większej liczby rejestrów niż liczba rejestrów dostępnych programowo. Rejestry te są przydzielane przez układ prefetchera/dekodera, a sposób przydziału umieszczany w tablicy zamienników rejestrów RAT (ang. *Register Alias Table).*

Po przydzieleniu rejestrów do mikrooperacji dodawana jest przez układ rozmieszczania mikrooperacji informacja statusowa pozwalająca na prawidłowe ustalenie ich ostatecznej kolejności. Następnie mikrooperacje przekazywane są do puli realizo-

wanych instrukcji. Na tym kończy się cześ'ć potoku, w której mikrooperaeje występują w oryginalnej kolejności.

Dalsza realizacja przebiega następująco. Układ przydziału zasobów RS (ang. fo. I *servation Station)* pobiera mikrooperaeje w takiej kolejności, aby zapewnić możliwie największe wykorzystanie zasobów jednostki wykonawczej. Wyniki realizacji przekazywane są z powrotem do puli instrukcji realizowanych. Stamtąd pobierane są, we właściwej kolejności, do jednostki instrukcji zrealizowanych RU (ang. *Retire Unii),* ' Jednostka ta powoduje też przepisanie wyników z rejestrów zamiennych do rejestrów ' procesora dostępnych programowo.

Potok wykonywania mikrooperacji realizujących instrukcję przedstawia rysunek 4.25.



Rysunek 4.25. Potok realizacji instrukcji w Pentium Pro™

W fazach IFU1, IFU2, IFU3 układ prefetchera współpracujący z układem przewidywania rozgałęzień BTB pobiera kody instrukcji wykonywanego programu i umieszcza je w pamięci cache programu (codę cache LI). Następnie w fazach DECl i DEC2 instrukcje są dekodowane, czyli konwertowane na ciąg mikrooperacji realizujących daną instrukcję. Kolejne fazy to realizacja mikrooperacji. W fazie RAT przydzielane są rzeczywistym rejestrom rejestry zamienniki. Dalej, w fazie ROB mikrooperaeje przekazywane są do bufora mikrooperacji do zrealizowania ROB. Stamtąd w fazie DIS są przekazywane do wykonania, przy czym sekwencja ich wykonywania może być inna niż oryginalna kolejność w programie. W fazie EX realizuje się wykonanie danej mikrooperacji. Ostatnie dwie fazy RET1 i RET2 przywracają właściwą kolejność operacji i ładują wyniki z rejestrów zamienników do rejestrów oryginałów.

Procesor Pentium Pro montowano w prostokątnym gnieździe ZIF oznaczanym jako Socket 8.

4.8. Pentium MMX

Kolejna wersja Pentium to MMX, będące modyfikacją Pentium „zwykłego" (rdzeń P5). Doceniając rozwój multimediów, Intel zmodyfikował architekturę i listę rozkazów tego procesora w ten sposób, aby można było skuteczniej realizować aplikacje multimedialne. Ponadto zmodyfikowano architekturę tego procesora w celu

osiągnięcia większej szybkości przetwarzania. Dodatkowe cechy Pentium MMX (oznaczanego często jako P55C) to:

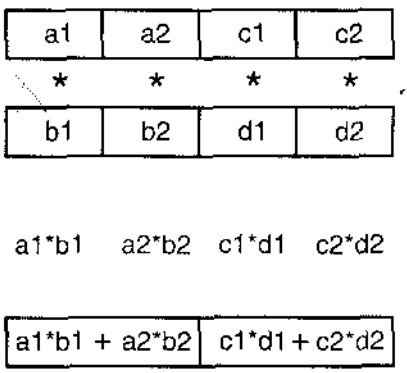
- zestaw instrukcji MMX realizujący rozkazy typu SIMD (patrz niżej),
- dwie 16-kilobajtowe wewnętrzne pamięci cache (jedna dla kodu programu i jedna dla danych),
- ulepszony układ przewidywania rozgałęzień,
- udoskonalona praca potokowa,
- praca potokowa w trybie MMX,
- możliwość wykonania do dwóch instrukcji na takt. W określonych warunkach mogą być wykonywane dwie instrukcje całkowitoliczbowe, dwie instrukcje MMX lub jedna instrukcja calkowitoliczbowa i jedna instrukcja MMX.

Należy podkreślić, że Pentium MMX wymaga dwóch napięć zasilających, osobnego dla rdzenia procesora i osobnego dla jego układów wejścia/wyjścia. Wyprowadzenia dla tych napięć zostały odpowiednio pogrupowane (w tak zwane wyspy) w celu ułatwienia ewentualnego unowocześnienia (ang. *upgrade)* dla tych procesorów.

W Pentium MMX dodano do listy rozkazów ich grupę ułatwiającą obsługę urządzeń multimedialnych. Podczas obsługi tych urządzeń stwierdzono, że znaczna część realizacji programów multimedialnych polega na powtarzaniu tych samych prostych operacji na dużej ilości dość krótkich danych (na przykład piksel obrazu w określonym trybie można opisać jednym bajtem). Dokonano więc następujących zmian:

1. Dodano 8 nowych rejestrów MM0+MM7 o długości 64 bitów. Umożliwia to wykonywanie operacji na większych porcjach informacji. Adresy tych rejestrów pokrywają się z adresami rejestrów koprocesora arytmetycznego, dlatego przed ich użyciem dla rozkazów MMX wymagane jest ich zapamiętanie.
2. Wprowadzono nowe typy danych, tak zwane dane spakowane. Pozwalają traktować zawartość 64-bitowych rejestrów jako spakowane bajty (8x8 b), spakowane słowa (4x16 b), spakowane dwusłowa (2x32 b) lub pojedyncze czterosłowo. Dane spakowane możemy interpretować jako kilka danych umieszczonych w jednym argumencie (64-bitowym).
3. Wprowadzono rozkazy wykonujące równolegle tę samą operację na danych spakowanych. Są to operacje typu SIMD (ang. *Single Instruction Multiple Data).*
4. Wprowadzono tak zwaną arytmetykę nasycenia. W przypadku pakowania danych lub wykonywania działań na danych spakowanych może wystąpić przekroczenie zakresu, którego nie możemy zasygnalizować. Przyjęto, że wówczas wynik ma maksymalną (np. dla dodawania) lub minimalną (np. dla odejmowania) dla określonego typu danych wartość (np. dla bajtu FPh łub 0). Dla wideo może to być interpretowane jako maksymalne lub minimalne nasycenie obrazu, stąd nazwa.

5. Wprowadzono operacje łączące dwa działania, mnożenie i dodawanie. Ułatwiało realizację operacji na wektorach i macierzach. Przykładowo wykonanie operacji pokazanej poniżej jest równoważne z obliczeniem dwóch iloczynów skalarnych wektorów a i b oraz c i d.



$$\overline{a} * \overline{b} = c = a1b1 + a2b2$$

Uwzględniając warianty poszczególnych operacji, mamy do dyspozycji 57 nowych instrukcji pozwalających znacznie szybciej realizować wiele operacji multimedialnych. Dostępny zestaw instrukcji oraz sposób konstrukcji mnemonika dla tych operacji opisany jest np. w pozycji [10].

Pentium MMX, podobnie jak zwykłe procesory Pentium, montowane są w gnieździe ZIF Socket 7.

4.9. Pentium II

Kolejna wersja procesora Pentium - 11, łączy w sobie rozwiązania rdzenia F6 zastosowane w Pentium Pro z technologią MMX. Poprawiono w nim też obsługę aplikacji 16-bitowych, która była słabą stroną Pentium Pro. Niektóre z rozwiązań idą w kierunku obniżenia kosztów produkcji procesora, powodując jednak zmniejszenie jego osiągów. Ponieważ jednym z problemów przy produkcji Pentium Pro była zintegrowana pamięć cache drugiego poziomu, w Pentium II zastosowano rozwiązanie kompromisowe. Rdzeń procesora wraz z układami wejścia/wyjścia stanowią osobne struktury umieszczone na wspólnej płytce drukowanej. Płytka ta zapewnia też kontakt z płytą główną w postaci listwy na krawędzi płytki oznaczanej SEC (ang. *Single Edge Contact).* Płytka z procesorem montowana jest na płycie głównej w złączu krawędziowym oznaczanym przez Intel jako Slot 1. Jądro procesora komunikuje się z pamięcią cache L2 za pośrednictwem specjalizowanej magistrali, pozwalając odciążyć główną magistralę procesora. Architektura taka nosi nazwę Dual Independent Bus.

Podstawowe własności procesora Pentium II można podsumować w następujących punktach:

- mikroarchitektura dynamicznej realizacji instrukcji,
- dwie rozdzielone magistrale, osobna dla pamięci cache L2 i osobna magistrala zewnętrzna (Dual Independent Bus),
- zwiększona pojemność pamięci cache LI - 2 x 16 kB,
- technologia MMX,
- udoskonalony system zarządzania poborem mocy
- zintegrowana 512-kilobajtowa pamięć cache L2,
- możliwość pracy w systemach dwuprocesorowych.

4.9.1. Celeron

Jak wspomniano, Celeron jest tanią wersją procesora Pentium II. Obniżenie ceny osiągnięto w sposób prosty, lecz drastyczny, usuwając z płytki procesora pamięć cache L2 (wersje 266 i 300) lub, w późniejszych wersjach (300A i 333), ograniczając jego rozmiar do 128 kB. Pozwoliło to rzeczywiście obniżyć cenę oraz zmniejszyć pobór mocy (procesor nie wymaga radiatora), lecz obniżyło szybkość jego działania. Pozostałe rozwiązania architektury i możliwości są takie jak dla Pentium II.

4.9.2. Ścieżki rozwoju Pentium II

Oryginalne Pentium II oparte było na rdzeniu o kodowej nazwie Klamath i miało magistralę FSB taktowaną częstotliwością 66 MHz. Magistrala BSB pracowała z połową częstotliwości taktowania rdzenia procesora. Kolejną wersją rdzenia był Deschutes współpracujący z magistralą FSB o częstotliwości 100 MHz. Rdzeń dla wersji procesora dla komputerów mobilnych nosił nazwę Tonga.

Ścieżka Celerona rozpoczynała się od rdzenia o kodowej nazwie Covington. Była to wersja całkowicie pozbawiona pamięci cache L2. Jego następcą był rdzeń Mendocino ze 128 kB pamięci cache. Wreszcie ostatnia wersja nosiła nazwę Dixon, miała 256 kB pamięci cache L2 i magistralę BSB pracującą z pełną szybkością zegara procesora. Wszystkie układy Celerona były przeznaczone do pracy w systemach jednoprocesorowych. Celeron montowany był w gnieździe Slotl, a jego ostatnie wersje w gnieździe Socket 370.

Wersje Xeon miały cache L2 o rozmiarach 512 kB, 1 MB lub 2 MB, magistrala BSB pracowała z pełną szybkością rdzenia procesora i był on przeznaczony (w zależności od wersji) do pracy w systemach dwu lub czteroprocesorowych. Częstotliwość FSB wynosiła 100 MHz. Xeony były montowane w gnieździe Slot 2.

4.10. Pentium III

Następnym opracowaniem Intela (maj 1999) był procesor Pentium 111 (przygotowywany pod roboczą nazwą Katmai). Zasadnicza architektura tego procesora jest 32-bitowa, jednak wprowadzono w nim wiele istotnych zmian, głównie z myślą o grafice trójwymiarowej i multimediach. Rozszerzono zestaw rozkazów technologii MMX (do 69). Usunięto też istotną wadę technologii MMX, a dotyczącą nowych rejestrów wprowadzonych dla tej technologii. W Pentium II pokrywały się one z rejestrami koprocesora, w Pentium III są niezależnymi rejestrami.

Kolejną istotną zmianą jest rozszerzenie rozkazów typu SIMD na rozkazy zmiennoprzecinkowe. Przypominamy, że rozkazy SIMD wykonują operacje na danych spakowanych. Oznacza to wykonanie przez jeden rozkaz tej samej operacji na kilku argumentach równocześnie (równolegle). Rozkazy te mają wiele zastosowań, często związanych z multimediami, między innymi przy obróbce grafiki 3D (np. pozycjonowaniu trójkątów modelujących bryły). Zestaw nowych rozkazów Pentium III oznaczany jest często jako SSE (ang. *Streaming SIMD Extensions),* znany pod wcześniejszą nazwą roboczą KNI (ang. *Katmai New Instructions).* Ponadto dla instrukcji zmiennoprzecinkowych wprowadzono osiem nowych, 128-bitowych rejestrów XMM[7:0] oraz dodatkowy rejestr sterujący dla instrukcji MMX, MXCSR. Podsumowując, w Pentium III mamy 50 nowych instrukcji zmiennoprzecinkowych SIMD i 12 nowych instrukcji MMX.

W procesorze Pentium III usprawniono też współpracę procesora z pamięciami. Wprowadzono między innymi osiem nowych instrukcji buforowania danych, które mogą być wykorzystywane przy realizacji kompresji wideo oraz obsłudze grafiki 3D. Nowe rozkazy w połączeniu ze wzrostem wydajności obliczeniowej umożliwiają między innymi programową realizację kompresji MPEG-2 pełnoekranowego obrazu w czasie rzeczywistym.

Kolejną, bardzo konkretną, ale też i kontrowersyjną zmianą było wprowadzenie numeru pozwalającego programowo zidentyfikować konkretny egzemplarz procesora.

Procesor Pentium III dysponuje 36-bitową magistralą adresową, co pozwala zaadresować pamięć o pojemności 64 GB (przy założeniu bajtowej organizacji pamięci).

Począwszy od wersji rdzenia o nazwie kodowej Coppermine, cache L2 była wykonywana w chipie procesora i nosiła angielską nazwę **Advanced Transfer Cache** (ATC).

Podobnie jak Pentium II, Pentium III instalowany był początkowo w gnieździe Slot 1, a późniejsze wersje także w gniazdach typu socket.

Projekt tego procesora rozpoczął się od rdzenia o kodowej nazwie Katmai i byl wykonany w technologii 0,25u.m. Wersja rdzenia dla Xeona nosiła nazwę Tanner.

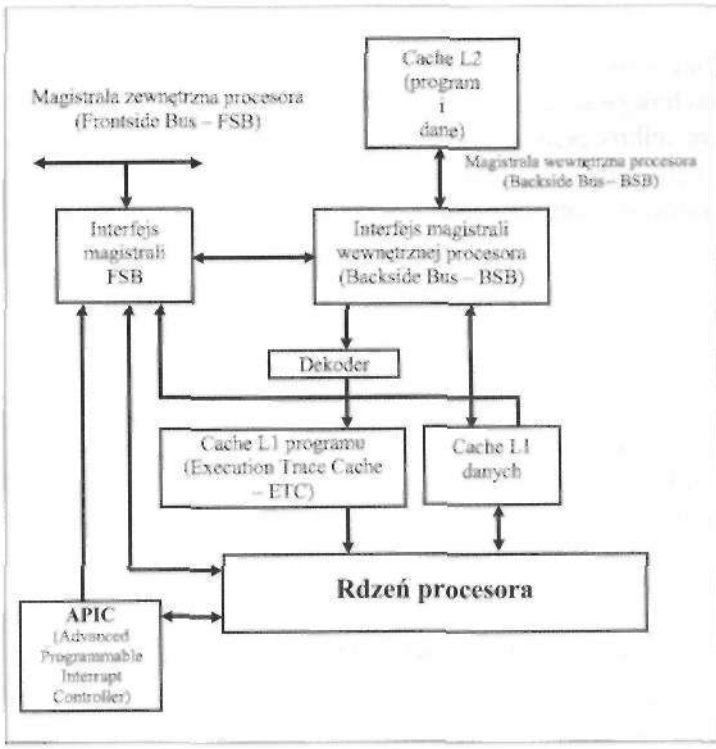
Kolejny rdzeń, wykonany w technologii 0,18(im, dla Pentium III i Celerona nosił nazwę Coppermine, dla Xeona zaś - Cascades.

Rdzeń Pentium III i Celerona wykonany w technologii 0,13 \xm nosił nazwę Tualatin, a rdzeń dla procesorów do komputerów mobilnych - Geyserville.

4.11. Pentium 4

Najnowszym rozwiązaniem procesora noszącego nazwę Pentium jest Pentium 4. Rdzeń tego procesora opracowywano pod kodową nazwą Willamette. Wprowadzono w nim dalsze rozwinięcie architektury dynamicznej realizacji instrukcji zwane mikroarchitekturą NetBurst.

Uproszczony schemat blokowy procesora Pentium 4 przedstawia rysunek 4.26.



Rysunek 4.26. Uproszczony schemat blokowy Pentium 4

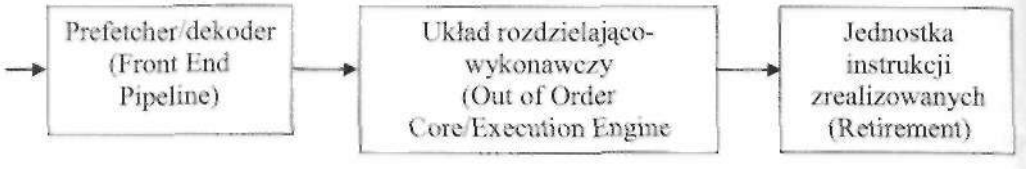
W stosunku do wcześniejszych rozwiązań bardzo istotną różnicą jest pojawienie się pamięci cache pierwszego poziomu (LI) dla programu, o angielskiej nazwie Execution Tracę Cache - ETC. Podobnie jak w poprzednich procesorach, począwszy od Pentium Pro, instrukcje architektury IA 32 wykonywane przez procesor, dekoder konwertuje na ciąg mikrooperacji (patrz rozdział 4.7 o Pentium Pro). Tym razem jednak efekt dekodowania jest przechowywany w specjalnej pamięci cache zwanej ETC.

Podstawowe elementy architektury NetBurst to:

- Jednostka wykonawcza Rapid Execution Engine. W jej skład wchodzą między innymi dwie jednostki arytmetyczno-logiczne ALU oraz dwie jednostki adresowe AGU. Układy te pracują z podwojoną częstotliwością zegara procesora. Większość instrukcji stałoprzecinkowych, w tym wszystkie instrukcje proste wykonywane są ze zwiększoną szybkością. Instrukcje złożone wykonywane są przez osobną ALU ze zwykłą częstotliwością zegara procesora.
- Zmieniony podsystem pamięci cache. Podstawową różnicą jest zastosowanie pamięci cache, w której umieszczane są mikrooperacje zdekodowanych instrukcji. Ciągi takich mikrooperacji otrzymały angielską nazwę „traces" i stąd pamięć ta nosi miano Execution Tracę Cache (była też używana nazwa Tracę Cache), co można przetłumaczyć jako pamięć s'cieżek realizacji instrukcji. W wielu pozycjach literatury używana jest nazwa „pamięć ze śledzeniem instrukcji". Pamięć ta może przechowywać około. 12 000 mikrooperacji. Zastosowanie pamięci przechowującej mikrooperacje realizujące instrukcje przyspiesza wykonanie wielokrotnych pętli, gdyż eliminuje powtórne dekodowanie tych samych rozkazów. Ponadto zarówno pamięci cache LI, jak i L2 pracują z pełną szybkością zegara procesora.
- Ulepszona architektura dynamicznej realizacji instrukcji oraz zmieniony potok. Potok w Pentium 4 jest 20-stopniowy (patrz rysunek 4.26). W związku z rozbudową potoku oraz realizacją mikrooperacji w zmienionej kolejności (do 126) ulepszone zostały też układy przewidywania realizacji rozgałęzień.
- Rozszerzono zestaw instrukcji SIMD, które tworzą obecnie zestaw 144 instrukcji o nazwie SSE 2, a w nowszych wersjach procesora Pentium 4 realizujących technologię Hyper-Threading zestaw SSE 3, wykonujący także rozkazy SIMD dla instrukcji zmiennoprzecinkowych.

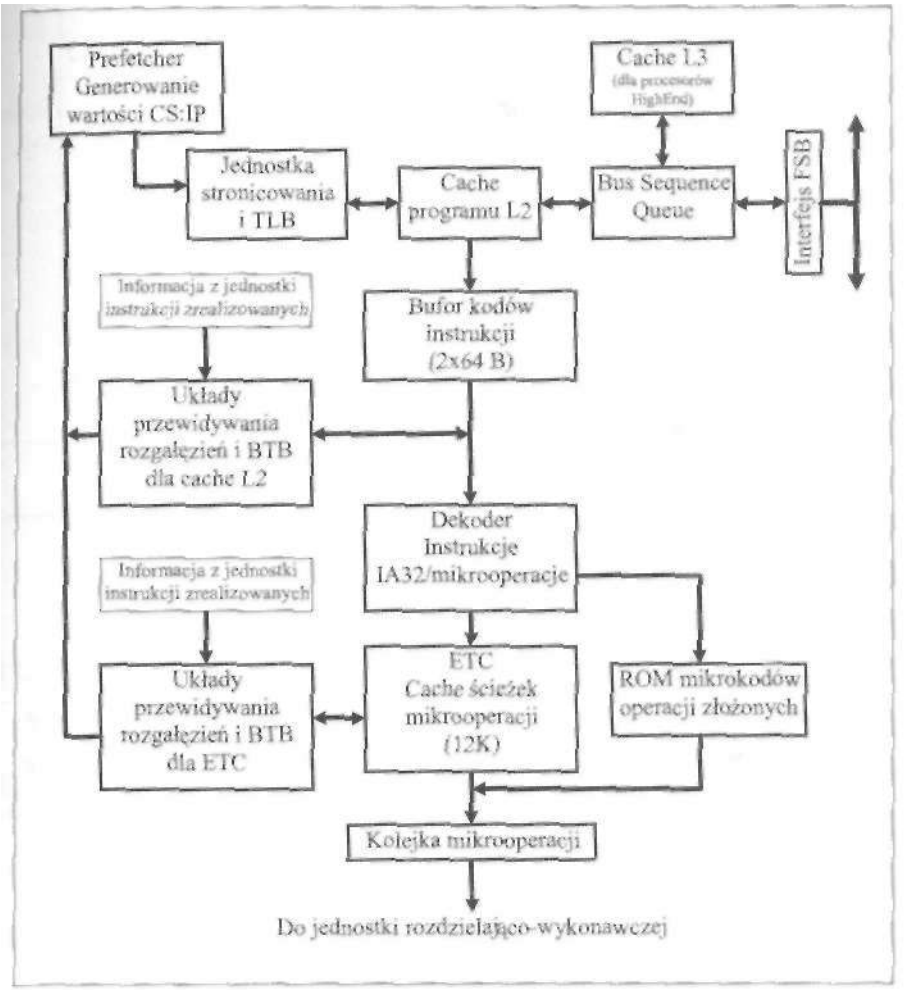
Magistrala adresowa Pentium 4 ma szerokość 36 bitów, co pozwala na zaadresowanie fizycznej pamięci o pojemności 64 GB.

Układy realizujące potok instrukcji Pentium 4 można podzielić na trzy bloki pokazane na rysunku 4.27.



Rysunek 4.27. Bloki realizujące potok instrukcji w Pentium 4

Układy związane z realizacją pierwszej części potoku, w której instrukcje przetwarzane są w takiej kolejności, w jakiej występują w programie, pokazano na rysunku4.28.

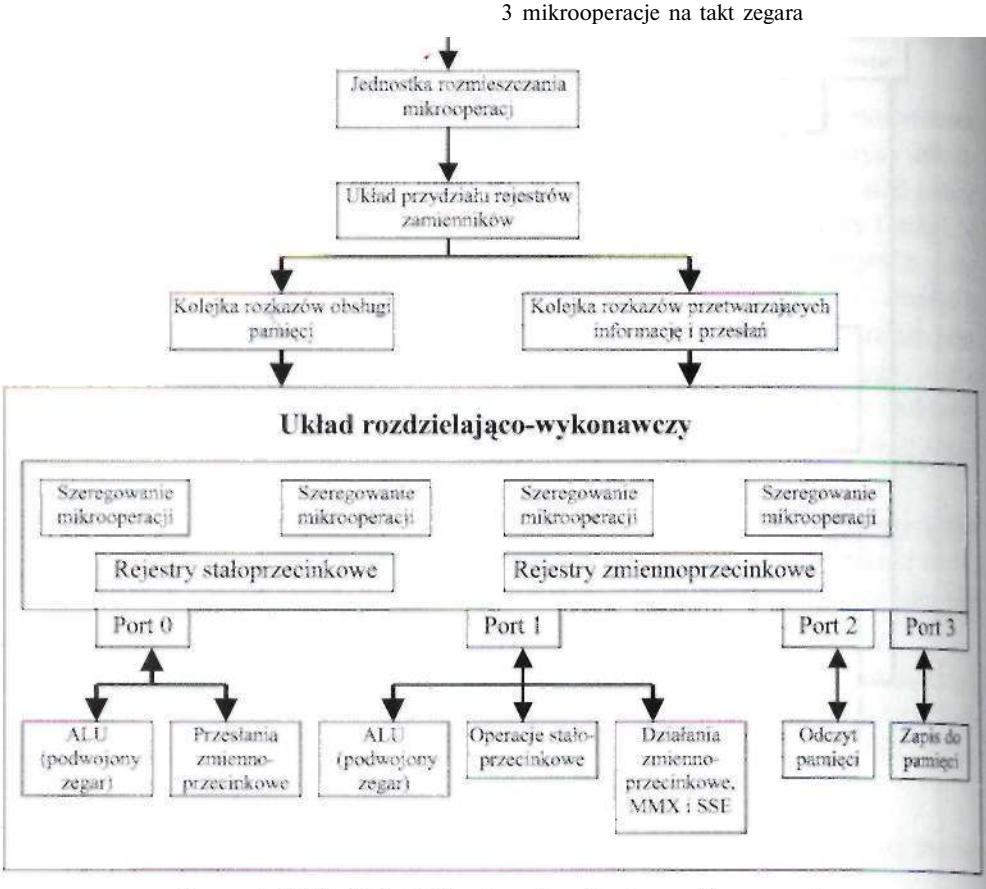


Rysunek 4.28. Układy Pentium 4 realizujące pierwszą część potoku instrukcji (Front End Pipeline)

Rysunek 4.29a przedstawia układy realizujące mikrooperacje, których potok realizacji pokazany jest na rysunku 4.29b, przy czym kolejność ich realizowania może być zmieniona (ang. *Out of Oder Executioń)*. Taka realizacja mikrooperacji nosi angielską nazwę *Speculative Execution* i ma na celu zapewnienie najlepszego wykorzystania zasobów wykonawczych mikroprocesora, a tym samym w miarę możliwości unikania ich przestojów. Rozwiązanie takie zastosowano po raz pierwszy w proce-

sorze Pentium Pro, co opisano w podrozdziale 4.7. Układ rozdzielająco-wykonawczy pobiera do realizacji trzy mikrooperacje na jeden takt zegara.

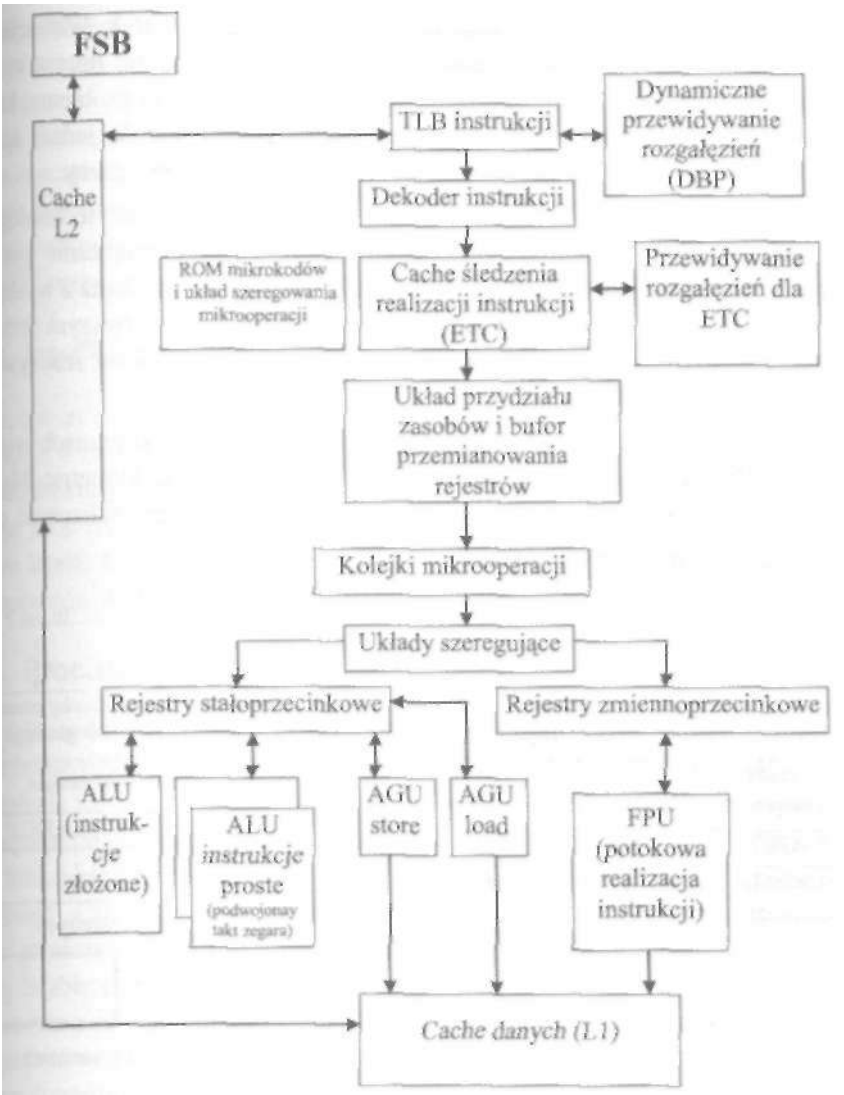
Fazy potoku realizującego mikrooperacje są pokazane na rysunku 4.2%. Intel nie dokumentuje działań, które są realizowane w każdej z faz.



Rysunek 4.29a. Układy Pentium 4 realizujące mikrooperacje

Rysunek 4.29b. Potok realizacji mikrooperacji w Pentium 4

Schemat blokowy Pentium 4, będący podsumowaniem naszych rozważań, przedstawia rysunek 4.30.



Rysunek 4.30. Schemat blokowy procesora Pentium 4

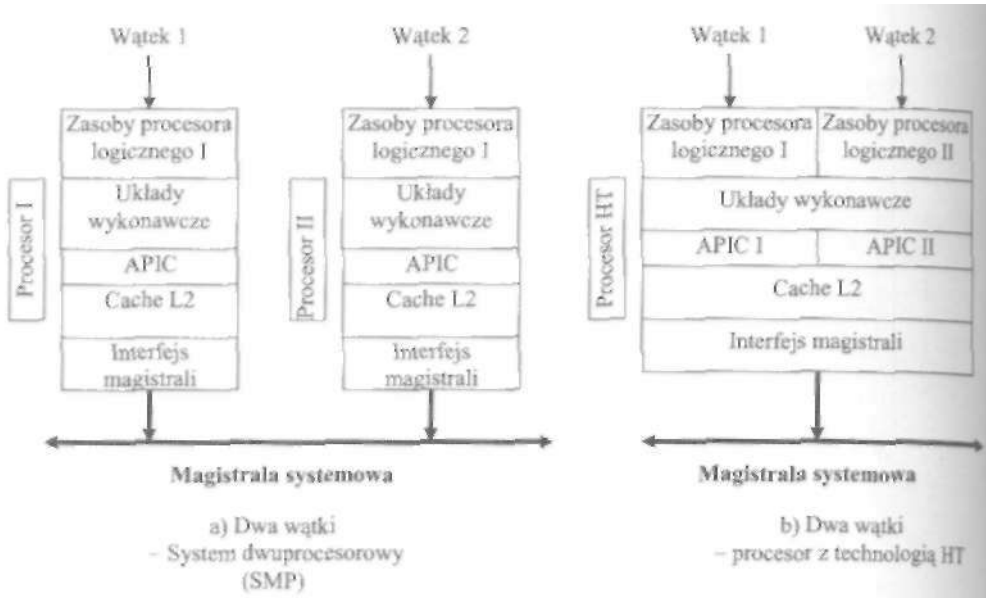
4,11.1. Technologia Hyper-Threading

Kolejnym rozwiązaniem poprawiającym wydajność procesorów rodziny IA 32 było zastosowanie w części procesorów Pentium 4 tak zwanej technologii Hyper-Threading, którą w skrócie oznaczamy HT. Poprawia ona realizację wielowątkowych aplikacji i systemów operacyjnych lub realizację aplikacji jednowątkowych w środowisku wielozadaniowym. Krótko omówimy, z czym wiąże się ta technologia.

Angielski termin *thread* odpowiada polskiemu terminowi **wątek.** Wątek jest wydzielonym fragmentem kodu, który może być wykonywany w dużej mierze niezależnie od pozostałych (istnieją pewne zależności sprawiające czasami problemy, które są odpowiednio rozwiązywane). Wykonywane wątki mogą należeć do jednej aplikacji (programu), wykonując różne podzadania, lub realizować różne programy.

W przypadku systemu jednoprocesorowego z procesorem bez technologii HT może on realizować dwa wątki naprzemiennie, przy czym przełączanie pomiędzy wątkami może wynikać bądź z upływu określonego odcinka czasu, bądź z wystąpienia określonego zdarzenia (więcej na ten temat w rozdziale SO). W pierwszym przypadku symulowane jest równoległe wykonywanie wątków, które jednak w rzeczywistości realizowane są naprzemiennie.

Rzeczywista równoległa realizacja wątków możliwa jest w systemach wieloprocesorowych MPS (Multiprocessor System lub zbliżony termin Symetrie Multiprocessing - SMP). Każdy z fizycznie obecnych procesorów może realizować osobny wątek. Sytuację taką przedstawia rysunek 4.31a.



Rysunek 4.31. Sposoby realizacji wątków

Innym rozwiązaniem umożliwiającym prawie równoległą realizację wątków jest technologia HT. Procesor, w którym zastosowano technologię HT, umożliwia realizację dwóch lub więcej wątków naprzemiennie przez wspólne układy wykonawcze procesora, przy czym każdy z wątków dysponuje własnym zestawem zasobów, takich jak rejestry ogólnego przeznaczenia, segmentowe, sterujące czy kontrolery przerwań (APIC - *Advanced Progammable Iterrupt Controller).* Inaczej mówiąc, w obrębie jednego fizycznego procesora tworzone są dwa (lub więcej) procesory logiczne z okre-

mi zasobami, korzystające z wspólnych układów wykonawczych. Rozwiązanie u zdecydowanie poprawia wykorzystanie tych układów. Przypominamy, że celem glosowania w rdzeniu P6 i następnych mikroarchitektury dynamicznej realizacji instrukcji (i związanej z nią realizacji mikrooperacji w zmienionej kolejności - *speculative execution)* było zapewnier

procesora. Realizacja dwóch lub więcej wątków pozwala elastyczniej wykorzystać dostępne wolne układy wykonawcze. Rozwiązanie, w którym dwa wątki prze-
-------- są przez procesor z technologią HT, symbolicznie obrazuje rysunek 4.3 Ib.

111.2. Intel® Extended Memory 64 Technology (Intel® EM64T)

Począwszy od procesora Intel® Pentium 4 Xeon z magistralą systemową 800 MHz (patrz tabela 4.1) (rdzeń Nocona DP), Intel wprowadził do swoich procesorów rozsze-a architektury IA 32 do 64-bitów opracowane przez AMD. Intel nazywał to roż-nie jako IA 32e lub Intel® Extended Memory 64 Technology, oznaczane też owo Intel® EM64T. Rozwiązanie to jest opisane krótko w punkcie 4.13 dotyczą-cym procesorów AMD.

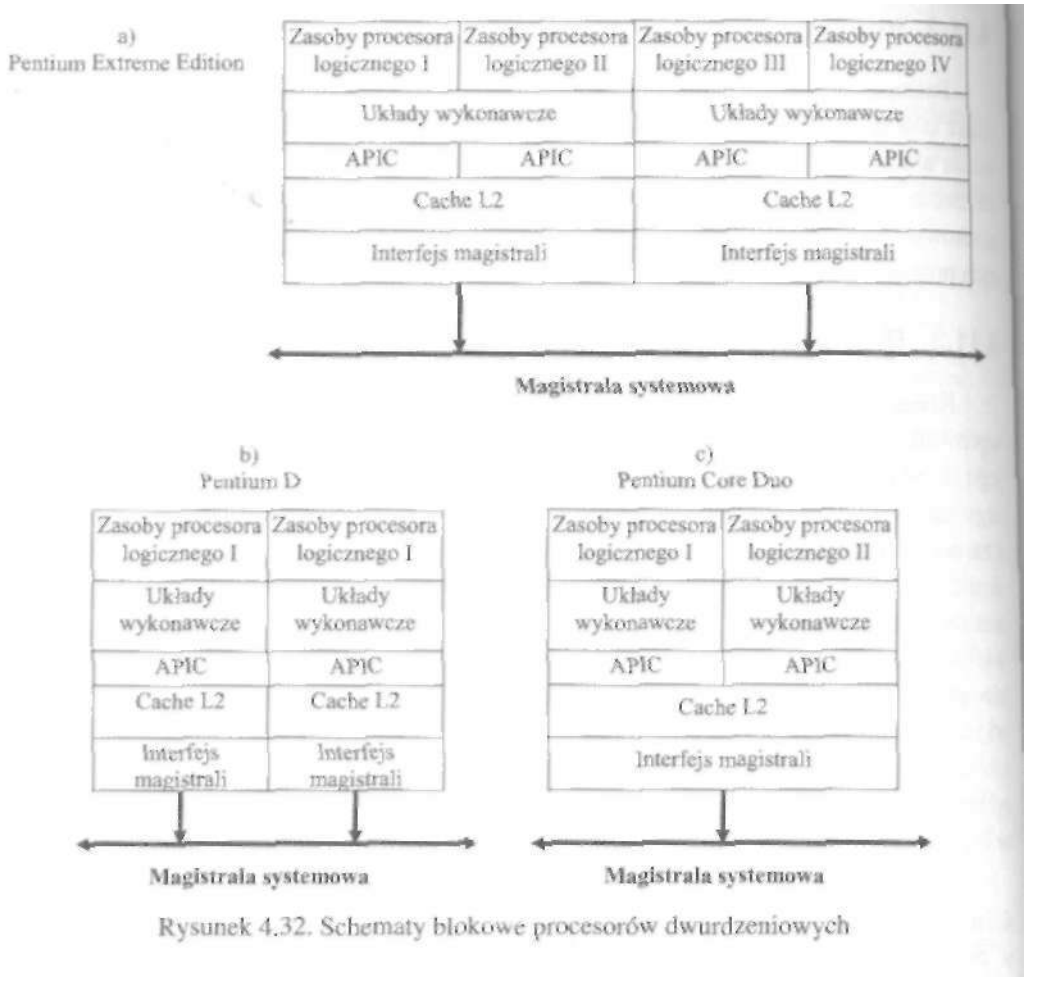
111.3. Procesory dwurdzeniowe

Kolejnym krokiem w rozwoju architektury procesorów, dość oczywistym, jest uprowadzenie procesorów dwurdzeniowych. Moc obliczeniowa procesora i szybkos'ć jego działania może być zwiększana przez zwiększanie częstotliwości zegara taktują-cego operacje procesora. Sposób ten ma jednak ograniczenia wynikające z praw fizyki i zasad funkcjonowania układów elektronicznych i nie może być prowadzony w nie-skończoność. Częstotliwości taktowania współczesnych procesorów zbliżają się do] wartości granicznych. W takiej sytuacji innym rozwiązaniem, występującym w coraz większej liczbie miejsc, równoległe wykonywanie operacji powodujące przyspiesze-nie przetwarzania informacji przez jednoczesne jej przetwarzanie przez dwa lub więcej układów. Rozwiązanie to pojawiało się już wcześniej. Proces zwany prefe-Khingiem (wstępnym pobieraniem instrukcji) czy architektura harwardzka pamięci to tylko nieliczne przykłady. Od dawna rozwijana jest też dziedzina przetwarzania rów-noległego informacji.

Obecnie technikę tę wykorzystuje się w konstrukcji architektury procesorów ro-dziny IA 32 (pomysł ten spożytkowano też w procesorze Itanium, o którym piszemy w punkcie 4.12) i nazywa się technologią wielordzeniową (ang. *Multi-Core Techno-*|j$'). Procesorami, w których ją zastosowano, są Pentium 4 Extreme Edition, Pentium D, i Pentium* M* z architekturą Intel* Core™ Duo. W następnych punktach krótko omawiamy rozwiązania używane w tych procesorach.

4.11.3.1. Pentium 4 Extreme Edition

Pierwszym procesorem rodziny IA 32, w którym zastosowano technologię Multi-Core, byl Pentium 4 Extreme Edition. Procesor ten zawiera dwa rdzenie, przy czym każdy z nich realizuje technologię HT. Oznacza to, że procesor ten zapewnia cztery procesory logiczne w jednym układzie (dwa procesory logiczne w każdym z dwóch rdzeni. Odmiana tego procesora **Dual** Core **Intel** Xeon realizuje wielordzeniową technologię HT i może pracować, w systemach wieloprocesorowych. Uproszczony schemafblokowy procesora Pentium 4 Extreme Edition przedstawia rysunek 4.32a.



Rysunek 4.32. Schematy blokowe procesorów dwurdzeniowych

4.11.3.2. Pentium D

Następnym procesorem wykorzystującym technologię wielordzeniową jest Pentium D. Procesor ten zawiera dwa rdzenie, jednak nie obsługuje technologii HT. Oznacza to obecnos'ć dwóch procesorów logicznych w jednym układzie, przy czym każdy z procesorów ma własne układy wykonawcze (de facto są to dwa w znacznej mierze

niezależne rdzenie). Uproszczony schemat blokowy tego procesora przedstawia rysunek 4.32b.

4.11.3.3. Intel® Core™ Duo

Kolejnym rozwiązaniem stosującym technologię wielordzeniową jest Intel* Core™ Microarchitecture, w procesorach oznaczana jako Intel"' Core™ Duo, zapewniająca między innymi stosunkowo niski pobór mocy i dlatego wykorzystywana przede wszystkim w procesorach do komputerów mobilnych, choć mająca także zastosowanie w procesorach dla komputerów typu desktop i serwerów. W architekturze tych procesorów wprowadzono nowe rozwiązania zapewniające nie tylko niższy pobór mocy, ale także efektywniejszą pracę wielowątkową z wykorzystaniem obu rdzeni. Rozwiązania zastosowane w Intel* Core™ Microarchitecture to:

. **Intel® Wide Dynamie Execution.** Rozwiązanie to pozwala realizować cztery mikrooperacje na cykl, ulepszono też w nim przewidywanie rozgałęzień oraz zwiększono pojemność bufora kodów instrukcji. Ponadto zastosowano łączenie często występujących par instrukcji w jedną całość (na etapie dekodowania) i realizację tak powstałej struktury jako jednej instrukcji. Proces ten nosi angielską nazwę *Macro-Fusion.*

- **Intel® Intelligent Power Capability.** Procesor zawierający to rozwiązanie ma układy logiczne, które włączają tylko te podsystemy logiczne procesora, które są konieczne do realizacji aktualnie wykonywanych operacji. Pozwala to zdecydowanie obniżyć pobór mocy procesora.
- **Intel® Advanced Smart Cache.** Poprawia wykorzystanie pamięci cache L2 przez oba rdzenie. W przypadku mniejszego zapotrzebowania dla jednego rdzenia drugi może korzystać z większego obszaru cache, gdyż jest on przydzielany dynamicznie, co z kolei zwiększa szansę trafienia.
- **Intel® Smart Memory Access.** Jednym z problemów występujących przy spekulatywnym wykonywaniu instrukcji w zmienionej kolejności jest zapewnienie dostępu do danych z jak najkrótszym czasem oczekiwania. W metodzie Smart Memory Access stosuje się specjalne algorytmy umożliwiające przewidywanie, które lokacje pamięci będą aktualnie potrzebne, co w konsekwencji pozwala na wcześniejsze ich przygotowanie. W przypadku błędnego przewidywania, co zdarza się stosunkowo rzadko, dostęp jest powtarzany.
- **Intel® Advanced Digital Media Boost.** Ostatnie z wymienianych rozwiązań związane jest z przetwarzaniem aplikacji multimedialnych - instrukcji SSE. W klasycznej architekturze 128-bitowe instrukcje przetwarzane były w dwóch taktach zegara, 64 bity na takt. W architekturze Intel® Core instrukcje SSE operujące na 128 bitach przetwarzane są w jednym takcie.

Uproszczony schemat blokowy procesora z rozwiązaniem Intel® Core™ Duo przedstawia rysunek 4.32c. Proszę zwrócić uwagę na jedną z różnic pomiędzy technologią HT a rozwiązaniem Intel* Core™ Duo. W drugim rozwiązaniu każdy z procesorów logicznych dysponuje własnymi układami wykonawczymi. Wspólne są natomiast cache L2 i interfejs magistrali. Część rozwiązań Intel* Core™ Microarchitecture (opisywanych wczes'niej) dotyczy właśnie poprawy współpracy z tymi elementami.

4.11.4. Centrino Mobile Technology

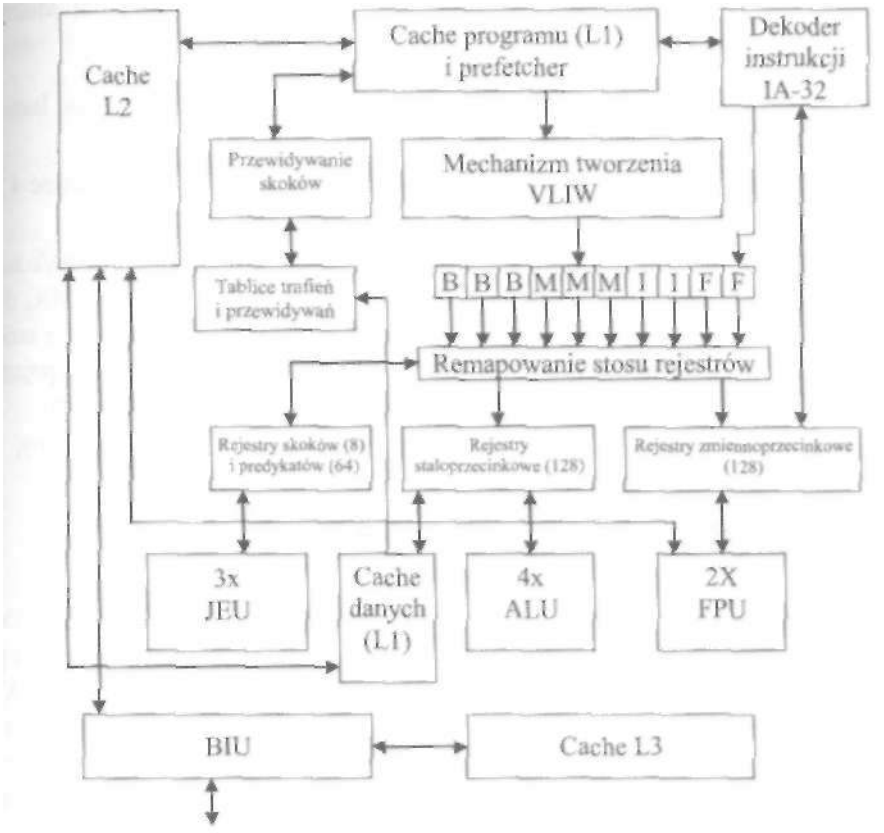
Jednym z zauważalnych trendów rynkowych w technice komputerowej jest wyraźny wzrost popularności komputerów mobilnych (laptop, notebook). W celu zmniejszenia poboru energii przez tego typu komputery firma Intel wprowadziła rozwiązanie będące połączeniem współpracy trzech elementów: procesora, chipsetu i bezprzewodowej karty sieciowej (ang. *Wireless LAN).* Współpraca tych elementów jest optymalizowana pod kątem zmniejszenia zużycia energii i nosi właśnie nazw **technologii mobilnej Centrino** - *Centrino Mobile Technology.* Została po raz pierwszy zastosowana dla procesora Pentium M (kodowa nazwa Banias) w połączeniu z chipsetem Intel® 855, nosiła nazwę kodową Carmel i doczekała się już kolejnych wersji. Wersja druga o kodowej nazwie Sonoma używała procesora Pentium M o kodowej nazwie Dothan i chipsetu Mobile Intel© 915M Express. W roku 2006 opracowano trzecią generację Centrino o nazwie Napa, używającą procesorów Intel Core o kodowej nazwie Yonah, w wersjach Intel Core Solo lub Intel Core Duo i chipsetu Intel Mobile 945 Express. Od tego momentu zaczęto używać terminu Centrino Duo.

4.12. Procesor Itanium

Procesor Itanium, rozwijany pod nazwą kodową Merced, to pierwszy 64-bitowy procesor rodziny x86. Przy konstruowaniu tego procesora zachowano kompatybilność z wcześniejszymi procesorami, choć kod 32-bitowy jest wykonywany dosyć wolno w stosunku do możliwości tego procesora. Należy jednak podkreślić, że Itanium nie jest procesorem należącym do architektury IA 32. To procesor 64-bitowy, a jego architekturę Intel określa skrótem EPIC (skrót wyjaśniono poniżej). Schemat blokowy procesora Itanium przedstawia rysunek 4.33.

W budowie tego procesora wykorzystano dwa interesujące rozwiązania. Pierwsze z nich wiąże się z techniką realizacji instrukcji określanej angielskim skrótem VLIW - *Very Long Instruction Word.* Technika ta polega na łączeniu (wewnątrz procesora) kodów instrukcji w długie słowa („paczki") zawierające kody trzech instrukcji (41 bitów każdy). W Itanium technika ta została dodatkowo wzbogacona przez dodanie do tak powstałego słowa 5-bitowego pola szablonu określającego typ i sposób wykonania rozkazu oraz czy instrukcje następnej paczki mogą być wykonywane równolegle

z paczką bieżącą. Rozwiązanie to umożliwia znacznie szybszy przydział układów wykonawczych realizujących poszczególne operacje i nosi skrótową nazwę EPIC - *faplicitly Paralłel lnstruction Computing,* co przetłumaczono na język polski jako przetwarzanie jawnie równoległe. Technologie VLIW i EPIC wymagają optymalizacji kodu na poziomie kompilacji.



Rysunek 4.33. Schemat blokowy procesora Itanium

Kolejnymi rozwiązaniami zapewniającymi wysoką wydajność tego procesora są:

- bardzo duża liczba rejestrów roboczych (128 rejestrów stałoprzecinkowych 46-bitowych, 128 rejestrów zmiennoprzecinkowych 80-bitowych i 72 rejestry przewidywania i realizacji skoków), które tworzą stos rejestrów;

duża liczba jednostek wykonawczych (11:4 jednostki stałoprzecinkowe, 2 jednostki zmiennoprzecinkowe, 2 jednostki realizujące operacje Load-Store, 3 jednostki przewidywania rozgałęzień) oraz ulepszona technika spekulatywnej realizacji rozkazów. W odróżnieniu od klasycznego przewidywania rozgałęzień wykonywane są obydwie przewidywane sekwencje rozkazów (co jest możliwe przy

dużej liczbie rejestrów oraz układów wykonawczych), a w stosownym momencie wybierana jest właściwa sekwencja;

- duża liczba pamięci cache zapewniająca dopływ danych i kodów możliwie bez opóźnień. Cache LI ma rozmiar 32 kB, L2 - 96 kB, a L3 - 2 lub 4 MB;
- ulepszona architektura jednostek zmiennoprzecinkowych, w których zastosowano układy przetwarzające oznaczone jako FMAC (Floating point Multiply Accumulate).

Blok dekodera instrukcji IA-32 zapewnia kompatybilność procesora Itanium I z wcześniejszymi procesorami rodziny x86.

Itanium dysponuje 42-bitową magistralą adresową pozwalającą zaadresować 4 TB I fizycznej pamięci.

Ostatnia wersja Itanium - Itanium 2, ma zwiększoną liczbę jednostek wykonawczych do 21 (sześć stałoprzecinkowych jednostek ALU, sześć jednostek MMX, dwie jednostki Load-Store, trzy jednostki przewidywania rozgałęzień, dwie jednostki zmiennoprzecinkowe o zwiększonej precyzji i dwie jednostki zmiennoprzecinkowe o pojedynczej precyzji). Zwiększono też liczbę portów układu rozdzielającego do jedenastu.

Itanium 2 ma 50-bitową magistralę adresową pozwalającą zaadresować 1 PB.

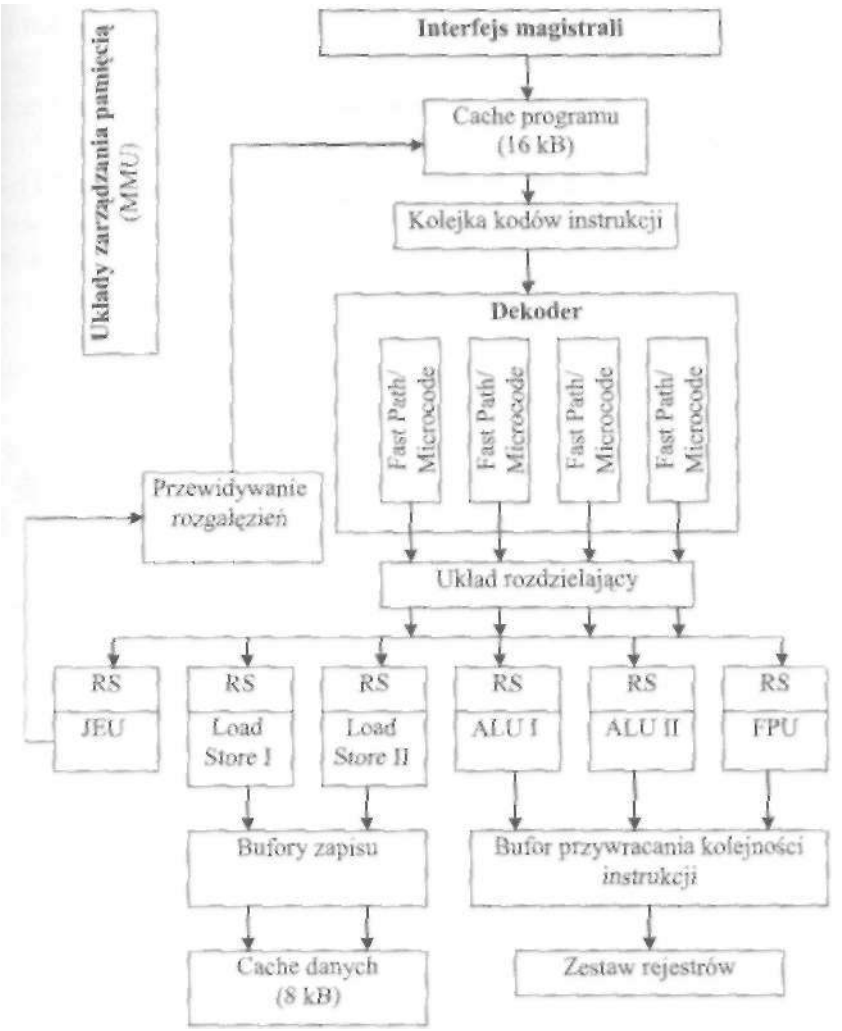
4.13. Przegląd procesorów firmy AMD

Poniżej przedstawiamy przegląd procesorów firmy AMD (Analog Micro Devices). Do modelu 486 włącznie procesory produkowane przez AMD były kopiami procesorów Intela (na zasadzie licencji). Począwszy od procesora AMD5_K86, firma AMD rozpoczęła opracowywanie własnych konstrukcji konkurujących z rozwiązaniami firmy Intel. Bardziej szczegółowo opiszemy jedynie jedną z nich, w pozostałych przypadkach ograniczając się do podania podstawowych, ogólnych cech tych procesorów. Obecnie firma AMD stanowi bardzo silną konkurencję dla Intela, osiągając w dziedzinie wydajności procesorów wyniki porównywalne, a często i lepsze.

Pierwszą samodzielną (i udaną) konstrukcją był procesor AMD5_K86, znany także jako K5. Schemat blokowy tego procesora przedstawia rysunek 4.34.

Podstawowe własności tego procesora to:

- tłumaczenie instrukcji x86 na mikrooperacje,
- spekulatywne wykonywanie instrukcji (w zmienionej kolejności),
- dwie kolejki operacji stałoprzecinkowych,
- przemianowywanie rejestrów (rejestry zamienniki),
- dynamiczne przewidywanie realizacji rozgałęzień,
- przyspieszanie przekazywania danych (ang. *data forwarding and bypassing).*



Kysunek 4.34. Schemat blokowy procesora AMD5K.86

Jak widzimy, w procesorze tym zastosowano tłumaczenie instrukcji na mikrooperacje typu RISC oraz spekulatywne ich wykonywanie w zmienionej kolejności w celu jak najlepszego wykorzystania jednostek wykonawczych. Pomysł ten dla procesorów do PC pojawił się zresztą po raz pierwszy jeszcze wcześniej, w procesorach 6xB6 firmy Cirix.

Następcą procesora K5 był K6. Jego architektura procesora opierała się na architekturze procesora Nx685 opracowanego przez firmę NexGen (wykupioną przez AMD). Podobnie jak poprzednio, jądro procesora miało architekturę RISC. Procesor miał bardzo wydajny dekoder produkujący cztery mikrooperacje na jeden takt zegara, sześć jednostek wykonawczych, wydajny układ przewidywania rozgałęzień (śledzący

realizację 8192 instrukcji rozgałęzień) i pamięć cache 2 x 32 kB (program i dane). Realizował też rozkazy MMX.

Procesor K6 produkowany był w różnych odmianach, różniących się częstotliwością zegara czy też częstotliwością taktowania magistral FSB i BSB.

Kolejnym procesorem rodziny AMD był Athlon, określany przez AMD jako procesor siódmej generacji. Procesor ten (podobnie jak Pentium II) miał początkowo pamięć cache L2 o pojemności 512 kB'wykonaną jako oddzielna struktura półprzewodnikowa. Dość szybko pojawiła się jednak wersja nazywana Thunderbird, w której pamięć cache L2 zintegrowana była ze strukturą procesora. Jej pojemność została zmniejszona do 256 kB, jednak jej magistrala pracowała z pełną częstotliwością zegara procesora.

Athlon zawierał dziewięć jednostek wykonawczych (trzy jednostki stałoprzecinkowe, trzy zmiennoprzecinkowe, trzy jednostki generowania adresów).

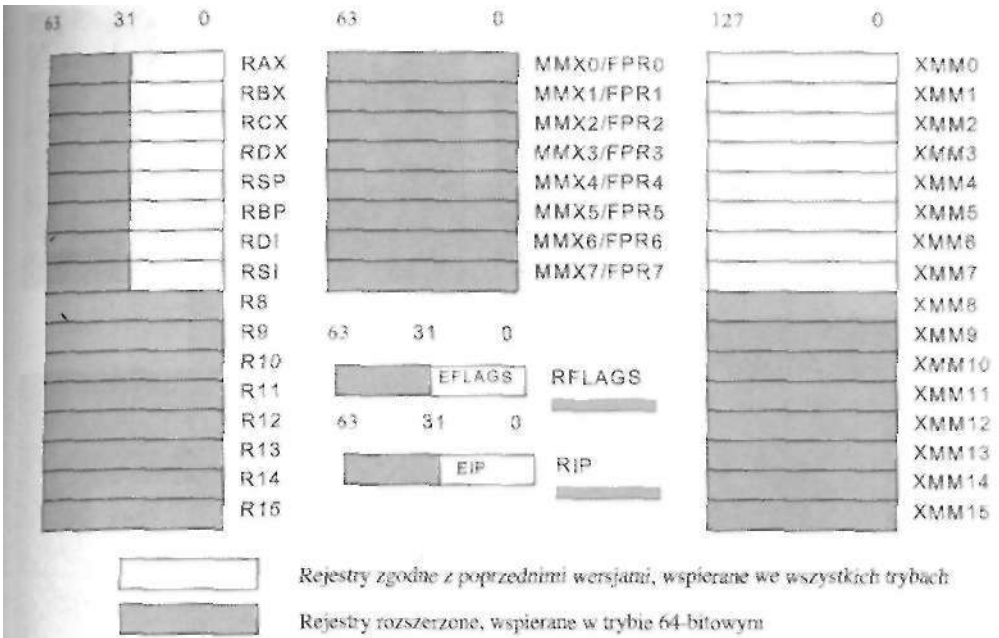
Athlon produkowano w wielu odmianach. Wersja Duron była na przykład tanią wersją tego procesora, ze zmniejszoną do 64 kB pamięcią cache. Interesujące rozwiązanie stanowił też Athlon XP, w którym zmieniono sposób współpracy pamięci cache poziomów LI i L2.

Najnowszym opracowaniem AMD jest rdzeń procesora oznaczany jako K8 lub Hammer, będący pierwszym 64-bitowym procesorem tej firmy. W oparciu o ten rdzeń produkowane są procesory o nazwach Opteron, Athlon64 FX, Athlon64 i Sempron. Pierwszy z nich przeznaczony jest do stacji serwerowych i wysoko wydajnych stacji roboczych, drugi i trzeci dla stacji roboczych lub wydajnych komputerów osobistych, i wreszcie ostatni dla tanich komputerów osobistych. Procesory te są konkurencją dla 64-bitowych procesorów Intela Itanium. Rozwiązania w nich zastosowane znacznie różnią się od wprowadzanych przez Intel, i biorąc pod uwagę reakcję rynku, wydają się trafione. Architektura ta nosi nazwę AMD64.

W rozwiązaniu tym rozszerzono długość rejestrów roboczych oraz dodano osiem nowych rejestrów ogólnego przeznaczenia (GPR - ang. *General Purpose Registers)* i osiem 128-bitowych rejestrów XMM. Procesor może pracować w jednym z dwóch podstawowych trybów:

- trybie dziedziczonym *(Legaty Mode),* w którym pracuje zgodnie z dotychczasowymi trybami rodziny x86: rzeczywistym, chronionym i V86,
- trybie rozszerzonym *(Long Mode),* przeznaczonym dla nowych 64-bitowych systemów operacyjnych. Ma dwa podtryby: tryb zgodności *(Compatpbility Modę),* w którym stare programy nic wymagają rekompilacji, i 64-bitowy, w którym procesor wykorzystuje pełne możliwości związane z 64-bitowymi rozszerzeniami.

Zestaw rejestrów roboczych procesorów zgodnych z architekturą AMD64 pokazano na rysunku 4.35.



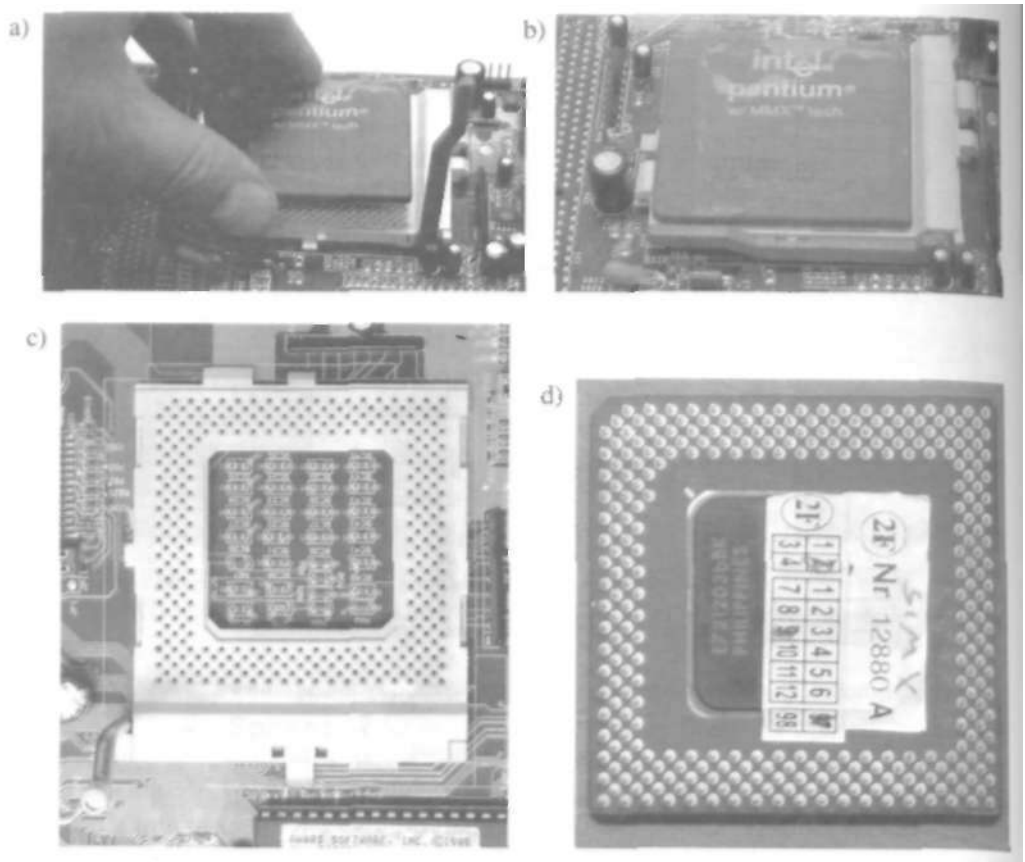
Rysunek 4.35. Rejestry robocze procesorów zgodnych z architekturą AMD64

Intel wprowadził w swoich procesorach nowy tryb pracy nazywany IA-32e, będący fragmentem rozwiązania oznaczanego jako Intel® Extended Memory 64 Technology (Intel® EM64T) i zapewniający kompatybilność z rozwiązaniami stosowanymi w procesorach AMD.

Dokładniejszy opis rodziny procesorów firmy AMD wykracza poza zakres tej książki. Informacje te można znaleźć na przykład w pozycji [6J.

Praktyka

W punkcie tym prezentujemy wybrane gniazda procesorów, zwracając uwagę na sposób ich montażu. Prezentacja wszystkich rodzajów gniazd nie ma sensu i zajęłaby wiele miejsca. Nie pokazujemy też gniazd, które obecnie mają znaczenie jedynie historyczne. Rozpoczynamy od gniazd określonych mianem ZIF (ang. *Zero Insert Force* - wkładanie bez użycia siły). Konstrukcja tego typu gniazda jest prosta: gniazdo ma dźwigienkę, która w jednym położeniu pozwala swobodnie wkładać i wyjmować procesor, w drugim zaś procesor jest zamocowany w gnieździe przez kontakty, które po zamknięciu dźwigienki zaciskają się na jego wyprowadzeniach. Przykładem tego typu gniazda jest Socket 7 pokazane na rysunkach 4.36 a do c.



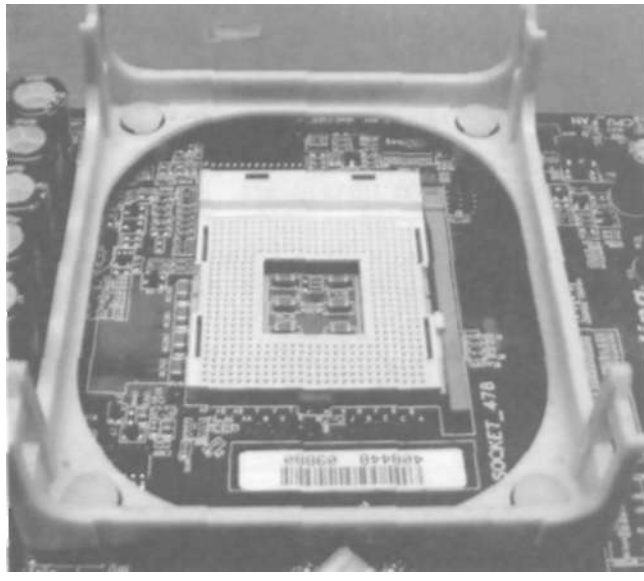
Rysunek 4.36. Gniazdo ZIF (a do c) i widok procesora od spodu (d)

Na rysunku a dźwigienka gniazda jest podniesiona i procesor można swobodnie włożyć do gniazda (lub wyjąć). Oczywiście obowiązuje tu precyzja i delikatność, dzięki czemu unikniemy przykładowo pogięcia wyprowadzeń procesora. Na rysunku b dźwigienka jest opuszczona i procesor zamocowany w złączu. Dźwigienka w niektórych przypadkach przed podniesieniem wymaga lekkiego odgięcia w bok w celu zdjęcia jej z zaczepu.

Na rysunkach c i d pokazano jeszcze jedną ważną rzecz. Proszę zwrócić uwagę, że zarówno w gnieździe (rysunek c), jak i w procesorze (którego widok od spodu jest przedstawiony na rysunku d) jeden z rogów różni się od pozostałych. Na nim to znajdują się odpowiednio kontakt i nóżka numer i. Róg procesora, przy którym mieści się ta nóżka, jest ścięty. Na zdjęciu widać też dodatkowe oznaczenie na spodzie procesora w postaci dodatkowej kreseczki. Nie jest to jednak standard. Wkładając procesor do gniazda, należy o tym pamiętać. Próba włożenia w innym położeniu zakończy się niepowodzeniem (i ewentualnym pogięciem styków procesora). W starszych płytach głównych możliwe było włożenie procesora 80486 niepoprawnie i groziło to nawet uszkodzeniem płyty.

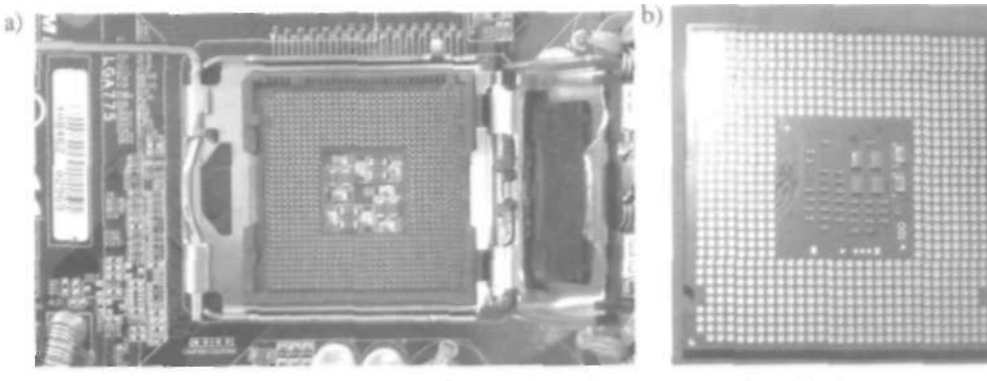
Kolejne gniazdo na rysunku 4.37 to Socket 478 przeznaczone do procesorów Intel Pentium 4, Celeron, Celeron D lub Intel Pentium 4 Extreme Edition. Poza liczbą wyprowadzeń (478) gniazdo to nie różni się zbytnio kształtem od pokazanego poprzednio.

Socket AM2



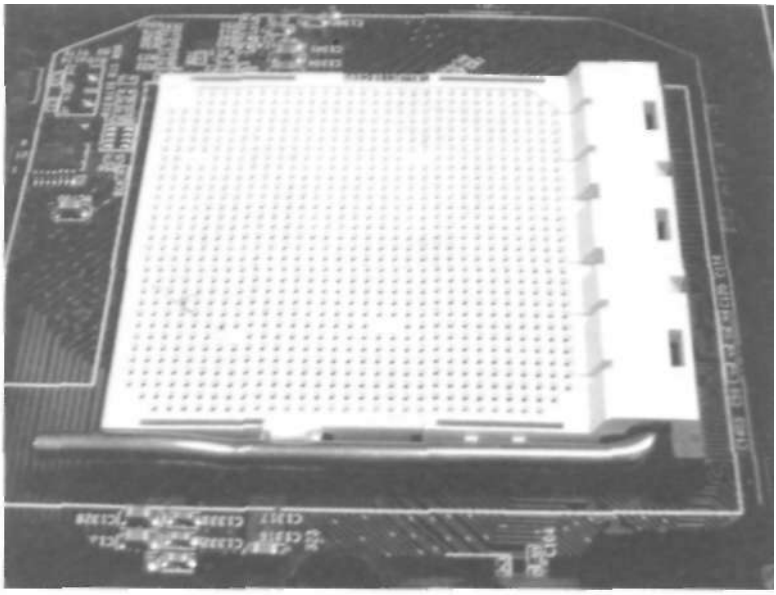
Rysunek 4.37. Gniazdo mPG 478B

Kolejne gniazdo procesorów firmy Intel - LGA 775, nazywane też Socket T lub Socket 775, przedstawione na rysunku 4.38, różni się od poprzednich tym, że zamiast otworów z kontaktami ma blaszki, a w procesorze w miejsce pinów wprowadzono pola dotykowe. Rozwiązanie nosi angielską nazwę *Land Grid Array* - stąd skrót LGA. Na rysunku 4.38a pokazujemy gniazdo, a na rysunku 4.38b spodnią stronę procesora.



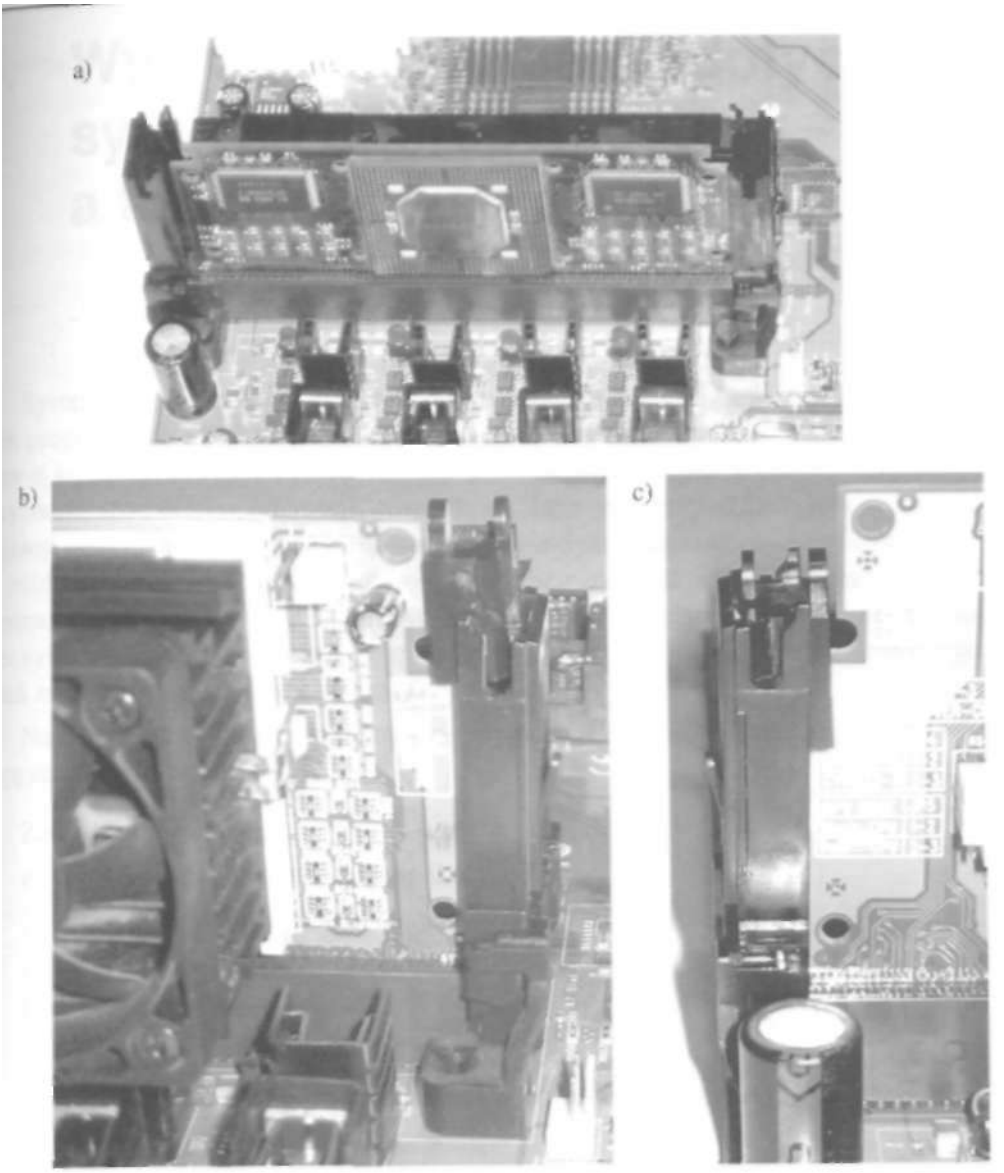
Rysunek 4.38. Gniazdo LGA 775 (a) i widok procesora od spodu (b)

Ostatnie z prezentowanych gniazd typu ZIF to gniazdo procesora AMD Socket AM2. Pokazujemy je na rysunku 4.39. Gniazdo to jest przeznaczone na przykład do procesorów AMD Athlon 64 FX-62, AMD Athlon 64 4000+ czy też AMD Sempron 3800+. Gniazdo jest typu PGA (Pin Grid Array), co oznacza, że procesor ma bolce a gniazdo otwory.



Rysunek 4.39. Gniazdo Socket AM2

Kolejne gniazdo to pewne cofniecie się w historii. Rysunek 4.40a przedstawia złącze Slot 1 z zamontowanym procesorem (bez radiatora). Złącza te można w komputerach spotkać coraz rzadziej. Po wsunięciu płytki procesora w pionowe prowadnice i wciśnięciu w złącze krawędziowe procesor należy zabezpieczyć zatrzaskami. Rysunek 4.40b przedstawia zatrzask niedopięty, rysunek 4.40c zapięty.



Rysunek 4.40. Gniazdo Slot 1

Wybrane zagadnienia dotyczące systemu operacyjnego a funkcjonowanie komputera

System operacyjny jest zestawem oprogramowania sterującego wykonaniem innych programów na komputerze oraz działającym jako interfejs pomiędzy użytkownikiem a komputerem. System operacyjny powinien zapewniać wydajne, bezpieczne wygodne środowisko realizacji programów. Jako program nadzorujący realizację aplikacji system operacyjny jest w pewnym sensie dystrybutorem zasobów komputera. W rozdziale tym spróbujemy krótko przeanalizować, jak system operacyjny jest wspierany przez rozwiązania sprzętowe występujące w procesorach i innych elementach systemu. Z tego też względu spojrzymy na system operacyjny od strony jego zadań związanych z realizacją aplikacji.

Najistotniejsze zadania systemu operacyjnego związane z uruchamianiem i obsługą aplikacji (procesu) to:

1) Zarządzanie pamięcią:
   a) przydział pamięci dla kodu aplikacji,
   b) przydział pamięci dla danych (środowiska) aplikacji,
   c) ochrona poszczególnych obszarów pamięci.
2) Przydział czasu procesora:
   a) rozpoczęcie wykonywania aplikacji,
   b) przełączanie pomiędzy aplikacjami w systemie wielozadaniowym,
      i) na żądanie użytkownika,
      ii) praca z wywłaszczaniem,
      iii) zakończenie aplikacji,
   c) praca wielowątkowa.
3) Obsługa pamięci:
   a) stronicowanie pamięci,
   b) obsługa pamięci wirtualnej.
4) Obsługa wejścia/wyjścia:
   a) obsługa systemów plików,

b) izolowanie warstwy sprzętu od bezpośredniego dostępu przez aplikacje w środowisku wielozadaniowym,

c) kolejkowanie zadań,

d) obsługa przerwań.

Poniżej krótko opisujemy poszczególne zadania, starając się podkreślić ich wspieranie przez hardware.

Podstawowe zadania systemu operacyjnego związane z obsługą procesów to:

1) Załadowanie kodu aplikacji do pamięci operacyjnej. Zadanie to można podzielić na kilka części:

   a) wyszukanie w fizycznej pamięci wystarczająco dużego bloku mogącego pomieścić kod aplikacji,

   b) przetransmitowanie pliku wykonywalnego zawierającego kod z miejsca przechowywania w pamięci masowej do wybranego bloku pamięci operacyjnej.

   Kolejnym krokiem jest uruchomienie i obsługa aplikacji. Zakładamy tu, że mamy do czynienia z systemem operacyjnym wielozadaniowym.

2) Przydział obszaru pamięci dla aplikacji, w którym będą przechowywane dane i wyniki. W środowisku wielozadaniowym wiąże się to z ochroną obszarów pamięci przed niewłaściwym dostępem, na przykład innej, błędnie działającej aplikacji.

3j Przydział czasu procesora:

   a) rozpoczęcie wykonywania aplikacji, co jest równoznaczne z wykonaniem przez procesor skoku do pierwszej instrukcji kodu aplikacji umieszczonej w pamięci operacyjnej,

   b) przełączanie pomiędzy aplikacjami lub wątkami.

System operacyjny wielozadaniowy cechuje się możliwością przełączania pomiędzy zadaniami załadowanymi do pamięci operacyjnej. Przełączanie to może wynikać z dwóch powodów:

- żądania użytkownika,
- upływu określonego, granicznego (maksymalnego) odcinka czasu przydzielonego aplikacji.

Pierwszy przypadek wynika z działania operatora komputera (na przykład kliknął ikonę programu na pasku zadań). Drugi przypadek ma miejsce w przypadku systemów operacyjnych pracujących z wywłaszczaniem. W systemie takim wyznaczany jest określony czas, po którym sterowanie musi wrócić do systemu operacyjnego lub inaczej mówiąc, po określonym czasie procesor musi przerwać wykonywanie aplikacji

i powrócić do kodu systemu operacyjnego. Zapobiega to możliwości zawieszenia systemu operacyjnego przez błędnie działającą aplikację.

W przypadku przełączania pomiędzy aplikacjami (w systemie wielozadaniowym) istnieje konieczność zapamiętania stanu aplikacji, która zostaje zawieszona, w celu jej późniejszej kontynuacji (dokładnie od miejsca i stanu, w jakim została przerwana). Jest to zadanie systemu operacyjnego, który musi zapisać do określonych, utworzonych przez siebie struktur wszelkie potrzebne dane (stan rejestrów procesora, wygląd ekranu itp.). Współczesne procesory ułatwiają jego wykonanie - patrz punkt 4.5.1.7.

Kolejnym problemem związanym z przydziałem czasu procesora jest obsługa aplikacji wielowątkowych. Aplikacje takie budowane są w znacznej mierze z niezależnych fragmentów kodu realizującego określone zadania, zwanych wątkami. Podział kodu aplikacji na wątki pozwala przełączać czas procesora pomiędzy różnymi wątkami, umożliwiając na przykład wyeliminowanie braku kontaktu użytkownika z aplikacją w przypadku oczekiwania na wykonanie długotrwałej operacji. Operację taką realizuje się jako oddzielny wątek, co pozwala na wykonywanie go na przemian z wątkiem, który zapewnia kontakt z użytkownikiem. Innym zastosowaniem wątków jest wykonywanie operacji w tle. Przełączanie wątków można realizować w sytuacji, gdy któryś z nich musi oczekiwać na pewien zasób systemu. Jedno z usprawnień realizacji aplikacji wielowątkowych stanowi oczywiście technologia Hyper-Threading.

Jednym z ważnych zadań jest stronicowanie pamięci. Ma znaczenie z kilku przyczyn. Pozwala przeadresowywać pamięć w celu uzyskania większych ciągłych bloków. Ułatwia też obsługę pamięci wirtualnej. I wreszcie jest niezbędne do realizacji pamięci asocjacyjnej, co z kolei wiąże się z obsługą pamięci cache. Mechanizm stronicowania wbudowany jest w procesory x86. Mają one także mechanizmy przyspieszające ten proces (TLB - 4.4.1.3).

Procesory tej rodziny ułatwiają też realizację obsługi pamięci wirtualnej. Mechanizm translacji adresu wirtualnego na fizyczny również wbudowano w procesory tej rodziny. Mają dodatkowo rejestry przyspieszające obsługę tej pamięci (patrz punkt 4.5.1.6).

Z pracą wielozadaniową, czyli obecnością w pamięci operacyjnej więcej niż jednej aplikacji, wiąże się zadanie ochrony poszczególnych obszarów pamięci. Ma przeciwdziałać błędnym zapisom mogącym zniszczyć dane lub kod innej aplikacji. Szczególnie starannie należy oczywiście chronić obszary pamięci używane przez system operacyjny. Procesory wspomagają tę ochronę (pola poziomów uprzywilejowania w deskryptorach, wyjątek błędu ochrony pamięci - patrz 6.2.1.1).

Wyraźne wsparcie sprzętowe ma też obsługa przerwań - układy AP1C, co jest szczególnie istotne w przypadku systemów wieloprocesorowych. W systemach wieloprocesorowych skomplikowane staje się też zapewnienie spójności pamięci głównej z pamięciami cache. Wiele ułatwia tu protokół MESI,

16. Płyty główne

6.1. Koncepcja budowy PC - drugie przybliżenie

Krótkiego wyjaśnienia wymaga koncepcja architektury komputerów typu IBM PC. Założono w niej modułową budowę komputera. Podstawowym elementem systemu jest tak zwana płyta główna lub płyta matka (ang. *main board* lub *mather-board)*. Powinna zawierać **podstawowe** układy potrzebne do pracy systemu, a więc CPU, podstawowe układy wejścia/wyjścia oraz układy logiczne koordynujące oraz inicjujące pracę tych układów. Ponadto założono, że:

- budowa lub inaczej konfiguracja sprzętowa powinna być możliwie elastyczna. Oznacza to możliwość dostosowywania tej konfiguracji do naszych wymagań i możliwości finansowych;
- i płyty różnych producentów powinny z punktu widzenia systemu operacyjnego zachowywać się identycznie.

Pierwsze założenie zrealizowano, umieszczając znaczną część układów i urządzeń, szczególnie rzadziej spotykanych czy specjalizowanych, na tak zwanych karlach, czyli płytkach elektronicznych montowanych w specjalnie do tego celu przeznaczonych gniazdach. Noszą nazwę gniazd magistrali rozszerzającej (w żargonie są to sloty) i są umieszczone na płycie głównej. Mogą być wykonane w różnych standardach. Ponadto, w zależności od sposobu rozwiązania konstrukcji płyty głównej, producenci oferują możliwość rozbudowy pamięci, zmiany typu procesora, szybkości jego zegara itp. Wymaga to **konfiguracji** (sprzętowej bądź programowej) płyty.

Druga kwestia została rozwiązana przez umieszczenie na płycie głównej pamięci ROM zawierającej **BIOS,** czyli **podstawowy system obsługi wejścia/wyjścia** (ang. [*Basic Input Output System)*. BIOS jest integralną częścią danej płyty i nie może być wymieniany pomiędzy różnymi płytami. Pełni dwojaką rolę.

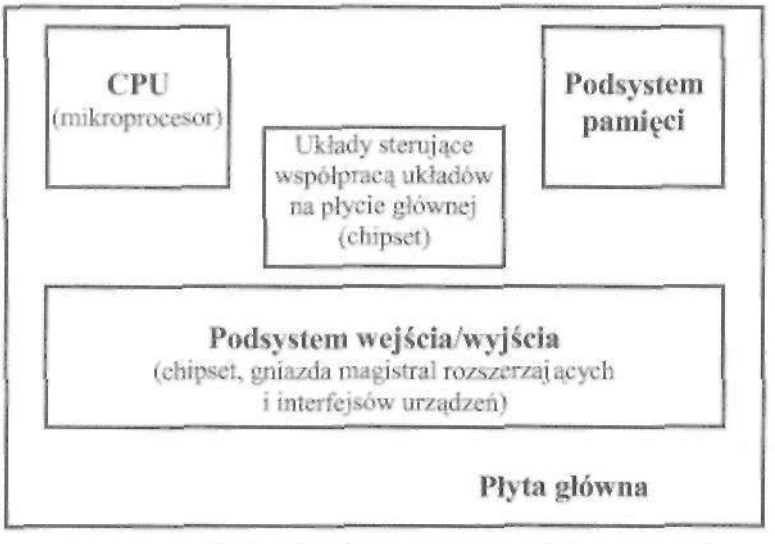
Po pierwsze likwiduje, z punktu widzenia systemu operacyjnego, różnice pomiędzy układowymi rozwiązaniami płyty. Po drugie (co jest zresztą związane z pierw| szym punktem), oferuje procedury obsługi standardowych układów i urządzeń wej-
ścia/wyjścia, z których może korzystać zarówno system operacyjny, jak i programista [(są dostępne jako tak zwane przerwania BIOS-u). Nieco dokładniej funkcje BIOS-u omówimy w dalszej części rozdziału.

Zastosowanie gniazd magistrali rozszerzającej pozwala użytkownikowi zarówno wybierać rodzaj używanych urządzeń (przykładowo karty telewizyjne, karty do digitalizacji obrazów i dźwięku, sieciowe itd.), jak i model danego urządzenia (na przykład prostą kartę graficzną lub wyrafinowaną kartę graficzną). Dzięki temu osiągamy wspomnianą elastyczność budowy komputera. Część urządzeń, popularnych bądź przyjętych jako standardowe, jest obecnie umieszczana bezpośrednio na płytach głównych. Jednak i tu mamy zwykle możliwości zmiany przez wyłączenie obsługi danego urządzenia w BIOS-ie i zainstalowania w jego miejsce urządzenia na karcie.

Schemat blokowy płyty głównej pokazano na rysunku 6.1. Proszę zwrócić uwagę, że pokrywa się on w dużej mierze ze schematem blokowym systemu mikroprocesorowego (bo takim systemem jest właśnie między innymi komputer).

Blok CPU to głównie mikroprocesor, zawierający także koprocesor arytmetyczny. We współczesnych płytach blok ten nieco się rozmywa, gdyż mogą się pojawiać specjalizowane procesory, np. procesor karty graficznej. Mikroprocesor umieszczany jest na współczesnych płytach głównych w gnieździe umożliwiającym łatwą jego wymianę.

Podobnie jest z podsystemem pamięci. Pamięć główna to przede wszystkim różnego rodzaju pamięci DRAM. Do podsystemu tego należy jednak również pamięć ROM, a także pamięci cache będące najczęściej elementem procesora. Obecnie układy tworzące pamięć główną są montowane na modułach umieszczanych w specjalnie do tego celu przeznaczonych gniazdach.



Rysunek 6.1. Schemat blokowy współczesnej płyty głównej

Podsystem wejścia/wyjścia zapewnia komunikację CPU i podsystemu pamięci z urządzeniami peryferyjnymi przez różnego rodzaju interfejsy i wspomagające je układy wejścia/wyjścia, przykładowo układ przerwań. Komunikację tę zapewniają magistrale (na przykład PCI-X, PCI Express, USB lub określone interfejsy EIDE, SATA i tak dalej). Układy sterujące tych magistral i interfejsów w większości są elementem chipsetów.

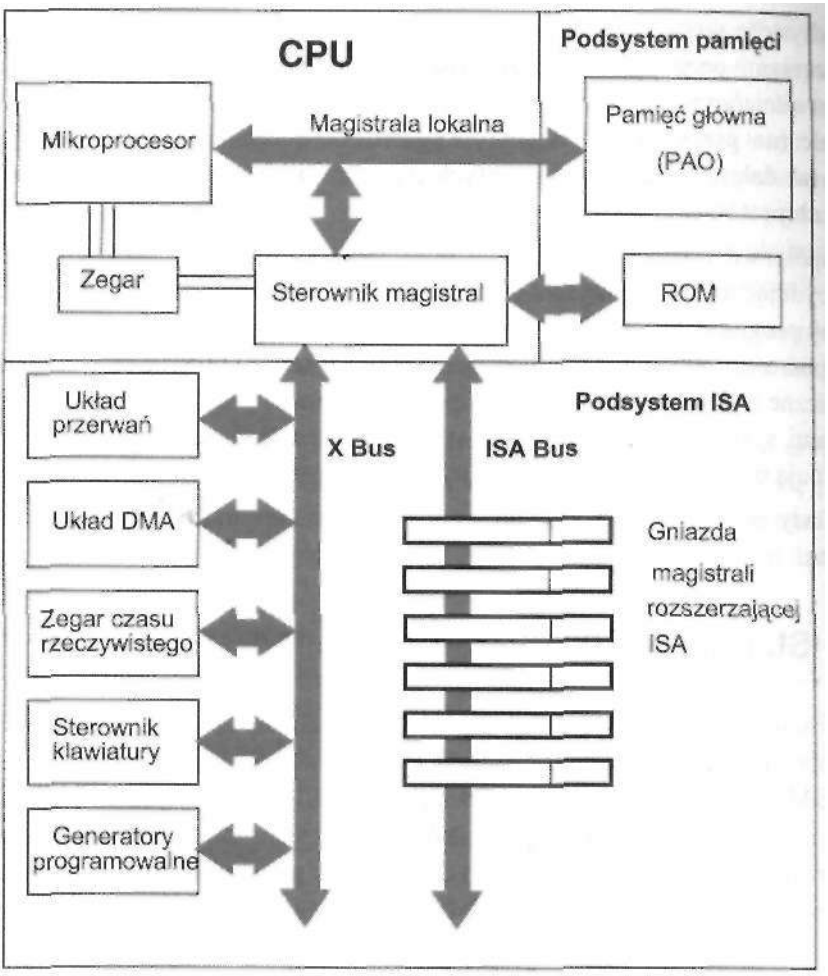
Współpraca wszystkich urządzeń płyty głównej musi być koordynowana. Układem decydującym o operacjach wykonywanych przez system jest CPU realizująca określony program. Jednak mikroprocesor, będący centralnym układem CPU, nie steruje bezpośrednio pracą pozostałych elementów systemu. Zadanie to wykonują układy elektroniczne zawarte w chipsetach. Zapewniają komunikację procesora z pozostałymi elementami systemu oraz, zgodnie z sygnałami nadchodzącymi z procesora, sterują i koordynują działaniem pozostałych elementów systemu.

Układy sterujące płyty głównej są jej niewymiennym elementem, co jest rzeczą oczywistą.

6.2. Standard ISA

Rozwój standardu ISA (ang. *Industrial Standard Archheciure)* rozpoczął się wraz z opracowaniem komputerów IBM PC. Podstawą obecnego standardu są komputery IBM PC AT. Należy zwrócić uwagę, że standard ten dotyczy nie tylko specyfikacji złącza magistrali rozszerzającej (taki pogląd jest bardzo częstym błędem), lecz także pewnych elementów występujących na płycie głównej. Wyjaśnione zostało to na rysunku 6.2, który pokazuje podział płyty głównej na podstawowe podsystemy.

Jak widać, podział układów płyty głównej na podsystemy jest zgodny z podziałem systemu mikroprocesorowego na bloki. Blok CPU tworzą procesor, zegar oraz sterownik magistral. Podsystem pamięci to przede wszystkim pamięć główna (pamięć operacyjna) zbudowana z pamięci DRAM oraz pamięć ROM (zawierająca BIOS). Do podsystemu tego można zaliczyć także pamięć cache, choć w przypadku płyt z procesorami 80468 i późniejszymi nie jest to już takie oczywiste. Blokowi układów wejścia/wyjścia odpowiada na naszym schemacie podsystem ISA. Układy DMA, przerwań, sterownik klawiatury i złącza magistrali rozszerzającej w sposób oczywisty należą do układów wejścia/wyjścia. Dwa pozostałe bloki, zegar czasu rzeczywistego i zespół generatorów programowanych (timerów), związane są z odmierzaniem czasu, co w pewnym sensie jest dostarczaniem informacji do systemu. Oba układy odgrywają w systemie ważną rolę.



Rysunek 6.2. Schemat blokowy płyty głównej systemu ISA

Elementy pierwszych dwóch podsystemów (procesory i pamięci) przedstawione zostały w rozdziałach 4. i 2. Omawianie architektury płyt głównych komputerów typu IBM PC rozpoczniemy więc od podsystemu ISA.

6.2.1. Podsystem ISA

W skład podsystemu ISA wchodzą następujące układy:

- układ przerwań zbudowany w oparciu o dwa sterowniki przerwań 8259A,
- układ DMA zbudowany przy użyciu dwóch sterowników DMA 8237A,
- sterownik klawiatury będący mikrokontrolerem, na przykład 8042,
- zegar czasu rzeczywistego będący odpowiednikiem układu 146818,

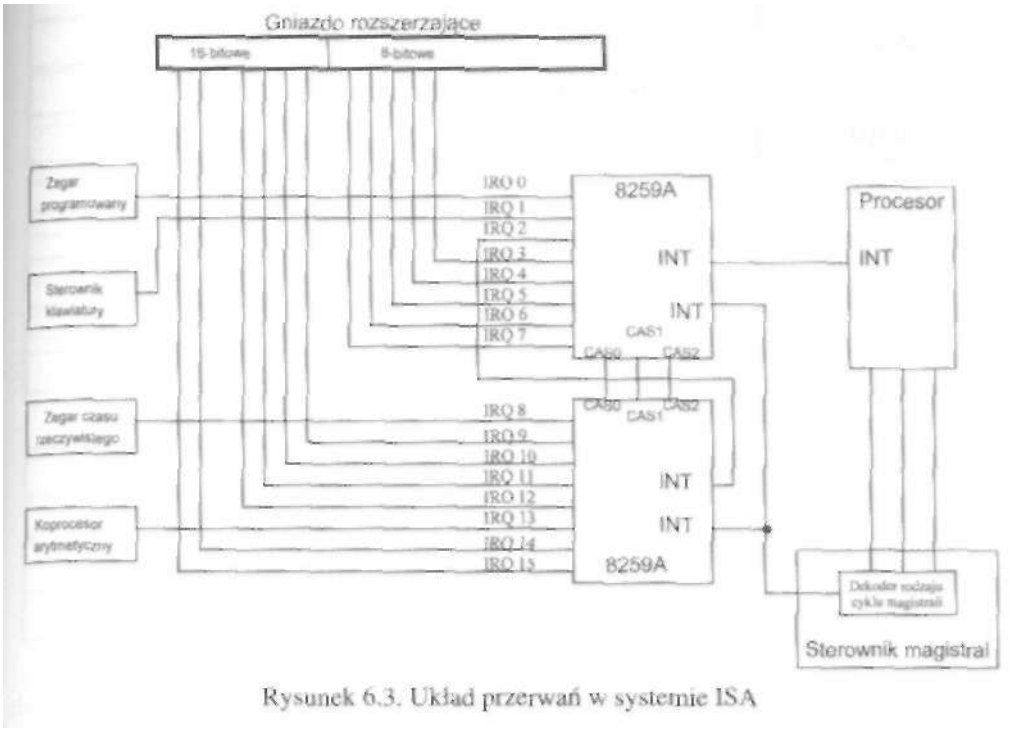
- układ trzech generatorów programowanych - układ 8254,
- gniazda magistrali rozszerzającej w systemie ISA.

W starszych wersjach płyt głównych wymienione układy były fizycznymi układami scalonymi. W nowszych płytach głównych znalazły miejsce w chipsetach, czyli układach scalonych wielkiej skali integracji. Należy jednak podkreślić, że zawierają funkcjonalne odpowiedniki w pełni zgodne z pierwowzorami wykonanymi w postaci pojedynczych układów scalonych.

6.2.1.1- Układ przerwań

Układ przerwań zbudowany jest za pomocą dwóch sterowników przerwań 8259A połączonych kaskadowo. Sposób połączenia tych sterowników oraz przyporządkowanie poszczególnych wejść przerwań sprzętowych urządzeniom standardowo umieszczonym na płycie głównej pokazuje rysunek 6.3.

Sygnały przerwań o numerach 0, 1, 8 i 13 nie zostały wyprowadzone na gniazda magistrali rozszerzającej. Odpowiadające im urządzenia znajdują się standardowo na płycie głównej.



Rysunek 6.3. Układ przerwań w systemie ISA

Pełny zestaw standardowych przyporządkowań przerwań sprzętowych urządzeniom znajdującym się na kartach rozszerzających bądź płycie głównej przedstawia tabela 6.1. W tabeli podano też pozycje w tablicy wektorów przerwań odpowiadające

danemu przerwaniu sprzętowemu. Pojedyncza pozycja w tablicy wektorów przerwań zawiera adres początku programu obsługi przerwania odpowiadającego danej pozycji Każda pozycja w tablicy wektorów przerwań ma więc rozmiar 4 bajtów. Podział tablicy wektorów przerwań podajemy w dalszej części rozdziału.

Skrót LPT oznacza port równoległy, a COM - port szeregowy. Przerwanie IRQ8 jest podłączone do wyjścia ALARM układu zegara czasu rzeczywistego. Linia IRQ13 stanowi wejście sygnału ERROR koprocesora arytmetycznego. Słowo „wolne" oznacza, że dane przerwanie jest dostępne dla kart ISA niemających standardowo przyporządkowanych przerwań (przykładowo karty sieciowe czy muzyczne).

Tabela 6.1. Przydział przerwań sprzętowych w systemie ISA

Numer przerwania sprzętowego	Nazwa urządzenia	Numer pozycji w tablicy wektorów przerwań
IRQ0	Licznik 0	08h
IRQ1	Sterownik klawiatury	09h
IRQ2	Wejście kaskadowe sterownika Slave	0Ah
IRQ8	Zegar czasu rzeczywistego	70h
IRQ9	Wywołanie przerwania IRQ2	71h
IRQ10	Wolne	72h
IRQ11	Wolne	73h
IRQ12	Wolne	74h
IRQ13	Koprocesor arytmetyczny	75h
IRQ14	Sterownik dysku twardego	76h
IRQ15	Wolne	77h
IRQ3	COM2	0Bh
IRQ4	COM1	0Ch
IRQ5	LPT2	0Dh
IRQ6	Sterownik dysków elastycznych	0Eh
IRQ7	LPT1	0Fh

Przypominamy, że zawartość tablicy wektorów przerwań decyduje, jak zostanie obsłużone dane przerwanie, gdyż podaje ona adres początku programu (procedury) obsługi tego przerwania (co zostało opisane w podrozdziale 3.5.1.3). Tablica wektorów przerwań inicjowana jest przez BIOS w trakcie startu komputera. Zmiana sposobu

obsługi przerwania polega na załadowaniu nowego programu obsługi przerwania pod określonym adresem i wpisaniu tego adresu do tablicy wektorów przerwań. Dzieje się tak na przykład, gdy instalujemy niestandardowy sterownik (ang. *driver)* urządzenia obsługiwanego przez BIOS.

Podział tablicy wektorów przerwań przedstawia tabela 6.2.

Tabela 6.2. Podział tablicy wektorów przerwań

Numer pozycji	Rodzaj przerwania
0÷0Fh	Przerwania sprzętowe (XT+AT)
10÷1Fh	Przerwania BIOS
20÷33h	Przerwania DOS
34÷3Eh	Emulacja koprocesora arytmetycznego
40÷42h	Przerwania BIOS
43	Tablica znaków EGA, MCGA, VGA
44÷4Ah	Przerwania BIOS
50÷57h	Zarezerwowane dla systemów pracujących w trybie chronionym
5A÷5Ch	Przerwania sieciowe
60÷66	Przerwania użytkownika
67	Sterownik LIM-EMS
70÷77	Przerwania sprzętowe (AT)
78÷Ffh	Przerwania użytkownika

W tablicy wektorów przerwań pozycje 5, 8+OEh i lOh są używane także prze; wyjątki, czyli wewnętrzne przerwania zgłaszane przez procesor. Wyjątki zdefinio wane są w podrozdziale 3.5.1.3. Na kolejnej stronie podajemy tabelę wyjątków proce sora 80386 wraz z odpowiadającymi im numerami pozycji w tablicy wektorów prze rwań. Tabela obowiązuje także dla następnych procesorów rodziny x86.

Połączenie w jednej pozycji dwóch przyczyn zgłoszenia przerwania jest możli we, wymaga jednak odpowiedniej konstrukcji programu obsługującego to przerwanie Program ten musi mieć możliwość stwierdzenia, co było przyczyną zgłoszenia prze rwania.

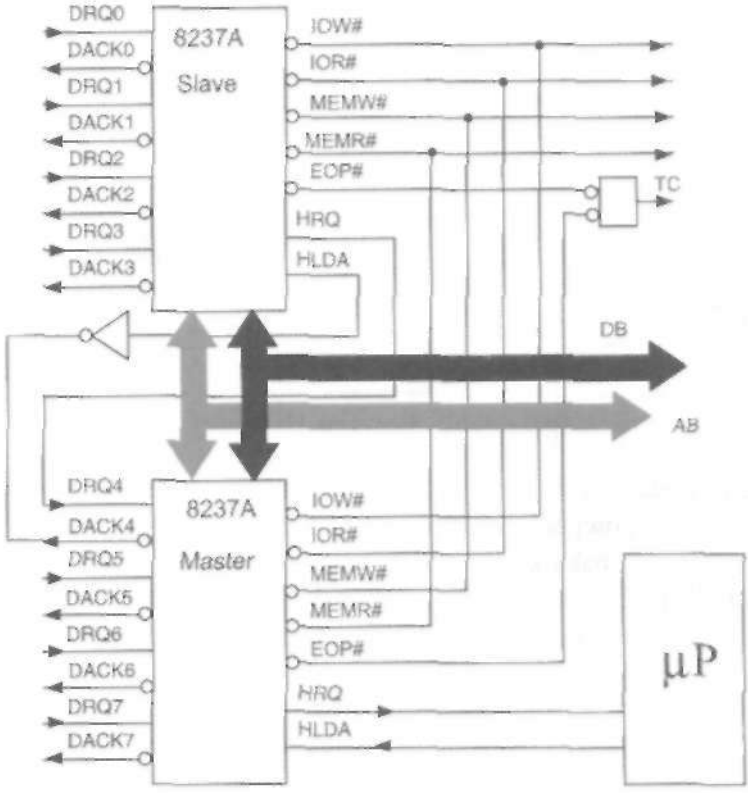
Tabela 6.3. Wyjątki procesora 80386

Nazwa wyjątku	Numer pozycji w tablicy wektorów przerwań
Dzielenie przez zero	0
Praca krokowa lub pułapka	1
NMI	2
Pułapka	3
Przepełnienie	4
Przekroczenie wartości argumentu dla instrukcji BOUND	5
Niedozwolony kod	6
Brak koprocesora	7
Podwójny błąd	8
Przekroczenie adresu w segmencie przez koprocesor	9
Nieprawidłowa zawartość TSS	0Ah
Brak segmentu w PAO	0Bh
Błąd segmentu stosu	0Ch
Ogólny błąd ochrony	0Dh
Błąd strony	0Eh
Zarezerwowane przez Intel	0Fh
Błąd koprocesora	10h
Zarezerwowane przez Intel	11÷1Fh

6.2.1.2. Układ DMA

Układ DMA zawiera dwa sterowniki przerwań 8237A połączone kaskadowo. Każdy sterownik 8237A może obsługiwać cztery kanały DMA. Sygnały DRQ I DACK jednego kanału zostały użyte do połączenia kaskadowego obu sterowników. Pozwala to na obsługę siedmiu kanałów DMA. Sposób połączenia sterowników pokazany jest na rysunku 6.3.

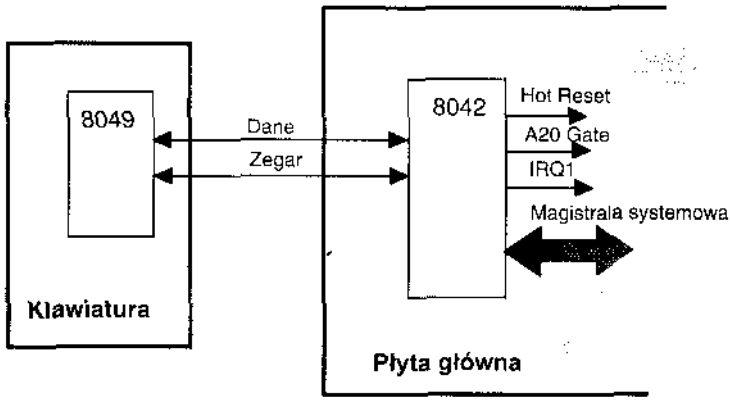
Sterownik master obsługuje transmisje 16-bitowe (kanały 5/7) natomiast sterownik slave transmisje 8-bitowe (kanały 0/3). Numer kanału DMA jest często jednym z parametrów, który musimy podać podczas konfigurowania kart czy innych urządzeń. Oczywiście wybrany kanał nie może być używany przez inne urządzenie.



Rysunek 6.4. Układ DMA

6.2.1.3. Sterownik klawiatury

Sterownik klawiatury wykonano przy użyciu układu mikrokontrolera, czyli jednoukładowego komputera. Najczęściej stosowanym układem jest mikrokontroler Intel 8042. Komunikuje się przez port szeregowy z mikrokontrolerem pracującym w klawiaturze. Sposób działania klawiatury przedstawiono w drugiej części książki poświęconej urządzeniom peryferyjnym. Mikrokontroler klawiatury przesyła do mikrokontrolera zainstalowanego na płycie głównej **kody** (numery) klawiszy (ang. *Scan Code),* przy czym rozróżniany jest kod naciśnięcia klawisz (ang. *make Code)* oraz kod zwolnienia klawisza (ang. *break Code).* Jest to istotne w celu ustalenia kolejności naciskania i zwalniania klawiszy, co z kolei ma znaczenie przy interpretowaniu kombinacji klawiszy. Informacja odbierana przez mikrokontroler 8042 jest obrabiana i interpretowana przez programowy sterownik klawiatury (ang. *keyboard handler)* uruchamiany zgłoszeniem przerwania sprzętowego IRQ1. Sposób podłączenia sterownika klawiatury do systemu pokazano na rysunku 6.5.

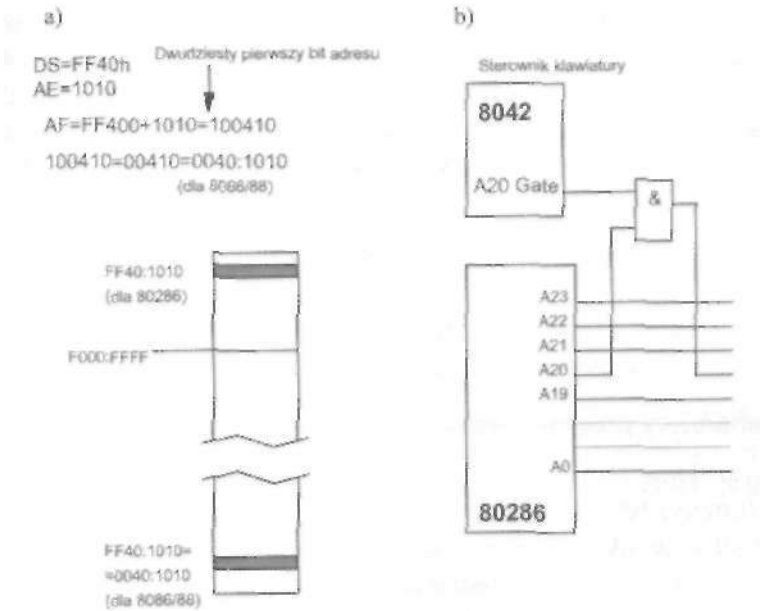


Rysunek 6.5. Sterownik klawiatury

Po wykryciu naciśnięcia bądź też zwolnienia klawisza mikrokontroler klawiatury przesyła łączem szeregowym jego kod (numer) do mikrokontrolera na płycie głównej. Po odebraniu pełnego znaku zgłaszane jest przerwanie IRQ1 powodujące uruchomienie przerwania INT9 będącego programem obsługi klawiatury. Przez system sterownik klawiatury (mikrokontroler) jest widziany jako układ wejścia/wyjścia o adresach 60h i 64h. Znak jest odczytywany spod adresu 60h (wynika to z zachowania kompatybilności z komputerami XT). Następnie program obsługujący klawiaturę przyporządkowuje mu zgodnie ze swoją wewnętrzną tablicą kodującą kod znaku. Kolejne napływające numery klawiszy i kody znaków umieszczane są w 32-bajtowym buforze pamięci (zwykle pod adresem 0040:001 Eh). W buforze mieści się 16 znaków (numer + kod), jednak z powodu sposobu organizacji bufora możemy w nim przechowywać maksymalnie 15 znaków. W przypadku zbyt szybkiego napływania znaków lub nieodczytywania ich przez aplikację sygnalizowane jest przepełnienie bufora (sygnał dźwiękowy). Dokładny opis sposobu obsługi bufora można znaleźć na przykład w pozycji [6],

Oto znaczenie pozostałych sygnałów. Sygnał A20 Gate jest używany w celu zapewnienia kompatybilności podczas realizowania programów wykonywanych w trybie rzeczywistym, napisanych dla procesorów 8086/88, a wykonywanych na procesorach 80286 lub późniejszych. Dla procesorów 8086/88 przy określonej konfiguracji wartości następowało tak zwane zawinięcie adresu, pokazane przykładowo na rysunku 6.6a. Było używane np. do wywołania danych umieszczonych pod niskimi adresami.

Natomiast w procesorach 80286 zjawisko to nie występuje, gdyż pojawiające się i przeniesienie ustawia linię A20, co powoduje wygenerowanie adresu w zakresie 64 kB powyżej pierwszego megabajtu. W celu zablokowania linii A20 i zapewnienia wspomnianej kompatybilności wykorzystywany jest układ pokazany na rysunku 6.6b. Umożliwia zablokowanie sygnałem A20 Gate wartości 1 pojawiającej się na linii A20.



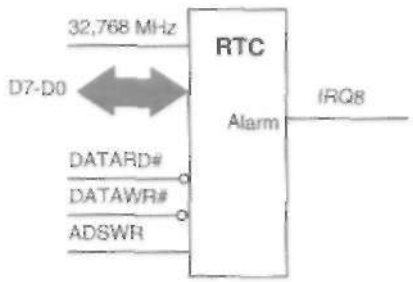
Rysunek 6.6. Znaczenie linii A20 Gate

Linia Hot Reset była używana w systemach z procesorem 80286 do spowodowania przejścia procesora z trybu chronionego w tryb rzeczywisty przez zgłoszenie tak zwanego gorącego restartu. Szczegółowy opis tego procesu znajduje się w pozycji [3].

6.2.1.4. Zegar czasu rzeczywistego

Począwszy od modelu AT, na płytach głównych ISA montowany jest układ Motoroli MC 146818 lub jego odpowiedniki (na przykład układ Dallas). Układ ten pełni dwie funkcje:

- jest zegarem czasu rzeczywistego (ang. *RTC - Real Time Clock),* czyli przechowuje informacje o dacie i godzinie;
- przechowuje w niewielkiej pamięci RAM dane dotyczące parametrów sprzętu zainstalowanego w systemie.



Rysunek 6.7. Zegar czasu rzeczywistego

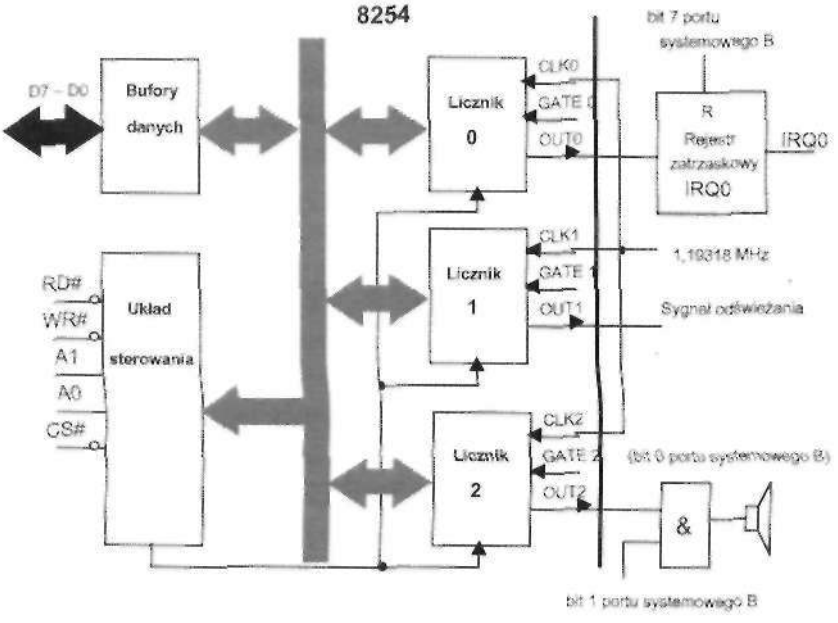
Pamięć RAM układu ma łącznie 64 B pod adresami od 00 do 3Fh. Adresy od 00 do 09 dotyczą godziny i daty; zawierają między innymi tak zwany czas alarmu, powodujący wygenerowanie przerwania sprzętowego IRQ8. Adresy 0A do0Dh dotyczą rejestrów sterujących pracą zegara. Adresy od OE do 3F zawierają między innymi informację o rodzaju napędów dyskowych, ilości pamięci, a także sumę kontrolną dla przechowywanych wartości.

Sygnały używane do komunikacji z zegarem czasu rzeczywistego przedstawione są na rysunku 6.7.

Przerwaniu IRQ8 odpowiada pozycja 70h w tablicy wektorów przerwań. Zalecaną metodą sterowania pracą zegara czasu rzeczywistego jest użycie przerwania INT lAh.

6.2.1.5. Generatory programowalne

Kolejnym elementem systemu ISA są trzy programowalne generatory interwałów czasowych (timery) będące elementem układu 8254 lub jego funkcjonalnego odpowiednika. Dokładniej układ 8254 zawiera trzy programowalne dzielniki częstotliwości napędzane zewnętrznym przebiegiem o częstotliwości l, 19318 MHz. Przebieg ten jest uzyskiwany z multiwibratora zawartego w podsystemie ISA, wytwarzającego przebieg I oznaczony przez OSC przeznaczony dla kart rozszerzeń. Przebieg ten ma częstotliwość 14,31818 MHz, która po podzieleniu przez 12 przez układ dzielnika na płycie głównej daje częstotliwość 1,19318 MHz. Schemat blokowy układu 8254 oraz standardowe zastosowania poszczególnych timerów przedstawia rysunek 6.8.



Rysunek 6.8. Liczniki programowalne w systemie ISA

Timer O nazywany jest timerem systemowym. W trakcie inicjalizacji (po restarcie) wpisywana jest do niego wartość dzielnika OFFFFh, co daje na wyjściu częstotliwość 18,21 Hz (impuls co 54,9 ms). Timer ten jest często używany do odmierzania czasu w systemie, przykładowo do odmierzania czasu wyłączenia silnika napędu dyskietek czy też w programach testujących możliwości komputera.

Timer 1 jest źródłem sygnału odświeżania pamięci, natomiast timer 2 generuje przebiegi podawane na głośnik standardowo instalowany w komputerach IBM PC.

6.2.2. BIOS (Basic Input Output System)

Jednym z niezwykle istotnych rozwiązań zastosowanych w komputerach typu IBM PC jest BIOS, czyli podstawowy system obsługi wejścia/wyjścia. BIOS to w istocie zestaw programów przechowywanych w pamięci ROM na płycie głównej wykonujących kilka bardzo ważnych zadań. Podstawowe zadania BIOS-u to:

- przeprowadzenie po restarcie testów podstawowych układów i urządzeń systemu, zwanych autotestem po włączeniu zasilania - POST (ang. *Power-On SelfTest);*
- inicjalizacja pracy systemu (instrukcje pobierane podczas startu pracy procesora, programowanie układów programowalnych, takich jak sterowniki przerwań czy DMA, wpisanie wartości początkowych do struktur systemowych w pamięci, na przykład inicjacja tablicy wektorów przerwań);
- zapewnienie w postaci programów obsługi przerwań (programowych bądź sprzętowych) procedur obsługi (sterowników, driverów) podstawowych, standardowych urządzeń systemu, zwłaszcza tak zwanych urządzeń IPL (ang. *Initial Program Load)* pozwalających załadować system operacyjny.

Dodatkowo, jak już wcześniej wspominano, zadaniem BIOS-u jest niwelacja, z punktu widzenia systemu operacyjnego, różnic konstrukcyjnych płyt głównych pochodzących od różnych producentów.

BIOS jest umieszczony w pamięci nieulotnej ROM (w nowszych rozwiązaniach flash EEPROM lub NOVRAM) w zakresie wysokich adresów, przy końcu pierwszego megabajtu pamięci (patrz też następny podrozdział - mapa pamięci), zajmując ostatnich 128 kB tego obszaru. Pozycja ta wynika między innymi ze sposobu restartu procesorów rodziny x86. Jak pamiętamy, pierwsza instrukcja po restarcie jest pobierana z komórki pamięci oddalonej o 16 bajtów od końca pierwszego megabajtu pamięci. Instrukcją tą jest skok do procedur POST. Poniżej omówimy krótko podstawowe zadania BIOS-u.

6.2.2.1. Procedura POST

Po restarcie systemu, niezależnie od jego przyczyny, procesor rozpoczyna wykonywanie instrukcji od adresu pamięci F000:FFF0h (patrz podrozdział 4.2.3). Pod

adresem tym znajduje się kod instrukcji skoku do procedury POST. Jej zadaniem jest przetestowanie oraz inicjalizacja podstawowych układów płyty głównej. Kolejno testowane są: procesor, zawartos'ć pamięci ROM (w zasadzie poprawność jej odczytu gdyż jej zawartość nie powinna ulegać zmianom). Po pomyślnym wykonaniu tych testów następuje test pamięci RAM, a następnie testowane i inicjowane są układy programowalne płyty głównej. Inicjowane są struktury systemowe, takie jak tablica wektorów przerwań czy też obszar zmiennych BIOS-u. Następnie testowane są standardowe urządzenia peryferyjne - klawiatura, stacje dysków, karta grafiki. Na etapie ostatnim sprawdzana jest obecność BIOS-ów na kartach (patrz podrozdział 6.2.2.4). Następnie sterowanie przekazywane jest do procedury ładującej system operacyjny (ang. *Bootstrap Loader),* dostępnej także jako przerwanie INT19h.

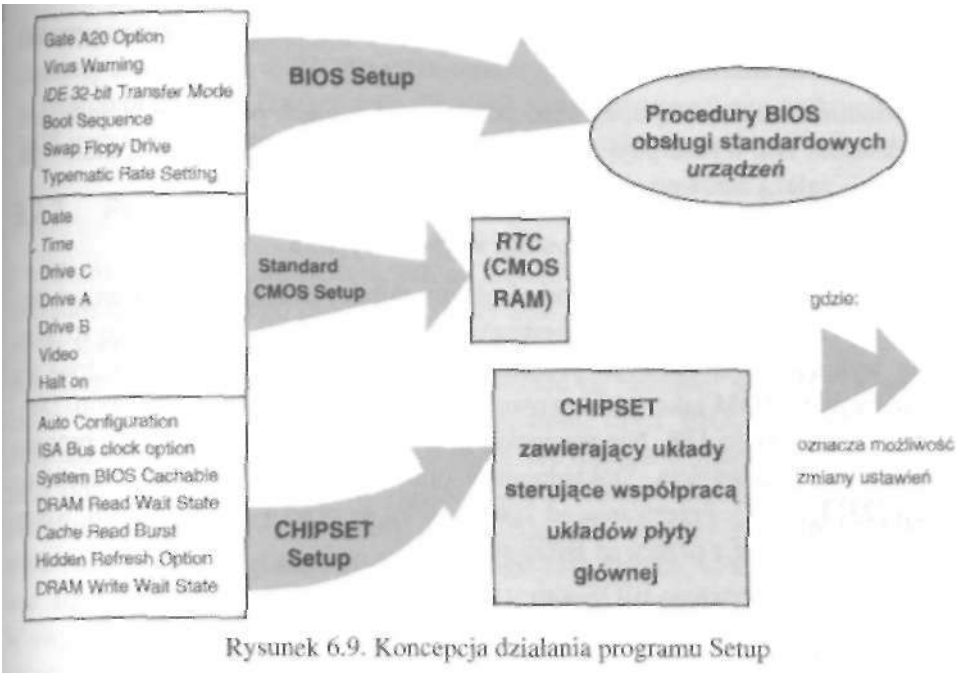
Stwierdzenie błędu podczas któregokolwiek z testów sygnalizuje (jeżeli to możliwe) odpowiedni komunikat na ekranie oraz sygnał dźwiękowy. Sposób zgłaszania błędów zależy od konkretnego BIOS-u. Ponadto wykrycie błędu na poziomie podstawowych układów powoduje wstrzymanie dalszych testów. Numery kolejnych testów wysyłane są przez rejestr AL na port wejścia/wyjścia o adresie 80h. Dlatego w przypadku błędu z zawartości tego portu można odczytać numer testu, który wykrył błąd.

6.2.2.2. BIOS Setup

W systemie występuje wiele układów wymagających zaprogramowania sposobu pracy, a więc wpisania pewnych warunków początkowych, słów sterujących itd. Zaprogramowanie tych układów po restarcie należy do procedur zawartych w BIOS-ie. Oprócz tego rozwiązania płyt głównych poszczególnych producentów umożliwiają wybór dodatkowych możliwości decydujących o sposobie pracy systemu. Przykładem może być określenie liczby stanów oczekiwania przy dostępie do pamięci czy włączenie lub wyłączenie opcji „shadow BIOS". Wyboru tych opcji dokonujemy za pomocą j programu zwanego Setupem (dosł. ang. *setup - ustawienia)* będącego także częścią j BIOS-u, a uruchamianego na nasze życzenie w trakcie restartu komputera. Dokonanie przez nas określonego wyboru zmienia sposób programowania, a więc i pracy układów płyty głównej. Koncepcję Setupu przedstawia schematycznie rysunek 6.9.

Ustawienia dokonywane w Setupie dotyczą oczywiście nie tylko układów wchodzących w skład systemu ISA, lecz wszystkich układów znajdujących się na płycie głównej (co też uwzględniono na rysunku). Generalnie opcje Setupu możemy podzielić na kilka bloków. Rodzaj i liczba bloków zależą oczywiście od konkretnej wersji BIOS-u. Podstawowe ustawienia spotykane w większości BIOS-ów to:

- Standard CMOS Setup - dotyczy parametrów zapisywanych w pamięci konfiguracji zegara czasu rzeczywistego;
- BIOS Features Setup - zmienia własności procedur BIOS-u wywoływanych przerwaniami sprzętowymi bądź programowymi;



Rysunek 6.9. Koncepcja działania programu Setup

. CHIPSET Features Setup - zmienia sposób pracy układów zawartych w chipsecie sterujących pracą układów płyty głównej;

- PCI Configuration Setup - ustawia opcje dotyczące sposobu pracy magistrali PCI;
- Power Management Setup - zarządza oszczędzaniem mocy.

Więcej przykładów pozycji Setupu można znaleźć w dodatku A.

Informacje na temat roli i zadań przykładowych chipsetów podajemy w podrozdziale 6.3.

Na rysunku 6.9 podane są wybrane ustawienia oraz elementy systemu, na jakie wpływają. Przykładowo, zmiana opcji Boot Sequence w BIOS Features Setup z A,C na C,A powoduje zmianę sposobu obsługi przerwania programowego INT19h. Procedura ta, nazywana po angielsku *Bootstrap Loader,* próbuje załadować do pamięci system operacyjny. Wspomniana zmiana spowoduje, że będzie on szukany najpierw na dysku C, a dopiero później, w wypadku niepowodzenia, na dysku A.

Zmiana daty w Standard CMOS Setup spowoduje po prostu wpisanie odpowiednich wartości do zegara czasu rzeczywistego. Zmiana opcji Cache Read Burst w CHIPSET Features Setup, np. z 3-2-2-2 na 2-1-1-1, spowoduje zmianę sposobu pracy sterownika pamięci (który jest umieszczony w jednym z chipsetów). Zmiana ta spowoduje szybszy dostęp do pamięci w trybie burst przez wstawienie mniejszej liczby stanów oczekiwania. Oczywiście w przypadku zbyt wolnych pamięci zmiana

taka może spowodować nieprawidłowe działanie systemu. Należy wówczas powrócić do poprzedniego ustawienia.

Przykładowe, najczęściej spotykane pozycje w Setupach podajemy w dodatku A na końcu książki. Nie wyczerpują one oczywiście wszystkich możliwości, które jak wspomniano - zależą zarówno od producenta, jak i od wersji BIOS-u.

6.2.2.3. Podstawowe procedury obsługi wejścia/wyjścia

W rozdziale dotyczącym przerwań pisaliśmy między innymi o przerwaniach programowych. Przerwania te, których przyczyną jest wykonanie określonego rozkazu powodują wykonanie określonych procedur. Część tych procedur umieszczona jest właśnie w pamięci ROM jako podstawowe procedury wejścia/wyjścia BIOS-u. Dotyczy to przerwań od 0 do 1Fh. Nie wszystkie przerwania z wymienionych są procedurami obsługi wejścia/wyjścia. Dokładniejszy opis ich działania można znaleźć np. w pozycji [23] lub [24]. Przerwania od 20h do OFFh są przerwaniami DOS-u. Korzystają w znacznej mierze z przerwań BIOS-u.

Przykładowe przerwania BIOS-u to:

- INTlOh - obsługa ekranu,
- INT13h - obsługa dysków twardych,

INT17h - obsługa drukarki.

Należy zdać sobie sprawę, że część urządzeń nie może być obsługiwana przez I BIOS. Są to urządzenia niestandardowe bądź nowe, nieznane BlOS-owi (umieszczenie dużej liczby sterowników w BlOS-ie jest zresztą niemożliwe z powodu ograniczonej ilości miejsca). Urządzenia takie wymagają albo własnego BIOS-u (patrz rozdział następny), albo sterowników zawartych w systemie operacyjnym lub dostarczonych przez producenta. W ostatnim przypadku sterowniki te musimy zainstalować. Zapewniają obsługę nowych urządzeń lub zastępują istniejące w BlOS-ie starsze procedury obsługi.

6.2.2.4. BlOS na kartach

System ISA zapewnia możliwość obsługi niestandardowych układów i urządzeń instalowanych w postaci kart w gniazdach magistrali rozszerzającej przez umieszczenie na tych kartach ich własnego BIOS-u. BIOS ten jest nazywany BlOS-em na kartach (ang. *device ROM).* Zadaniem jego jest, podobnie jak BIOS-u na płycie głównej, przeprowadzenie testu karty oraz zapewnienie potrzebnych jej programów obsługi.

Dla BIOS-ów na kartach zarezerwowano adresy pamięci od COOOOh do DFFFFh. W trakcie procedury POST obszar ten jest przeszukiwany w celu wykrycia obecności BIOS-ów na kartach. Przeszukiwanie rozpoczyna się od adresu COOOOh. Jeżeli pod tym adresem zostanie wykryta sygnatura 55AAh, oznacza to wykrycie BIOS-u karty.

i Odczytywana jest wówczas jego długość, sprawdzana suma kontrolna, a następnie uru-
I chamiany test zawarty w BIOS-ie. Następnie sprawdzana jest kolejna lokacja pamięci po wykrytym BIOS-ie (znamy jego długość) w celu ewentualnego wykrycia kolejnego BIOS-u na karcie. Postępowanie trwa aż do końca podanego zakresu adresów pamięci.

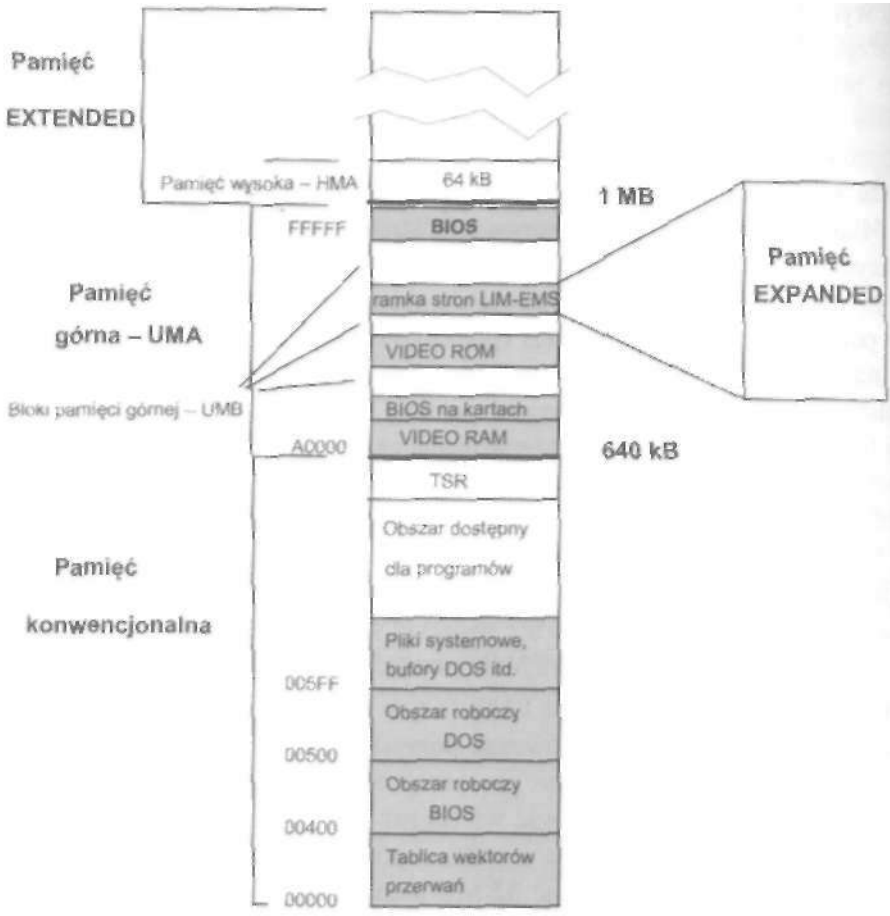
6.2.3. Przestrzeń adresowa pamięci i układów wejścia/wyjścia

Jednym z rozwiązań rzutujących na architekturę komputerów IBM PC była wartość adresu, spod którego procesor rodziny x86 po restarcie pobiera pierwszą instrukcję do wykonania. Adres ten wynosi FFFFOh i jest oddalony o 16 bajtów od końca pierwszego MB pamięci. Wymusza to położenie co najmniej początku, a praktycznie całości programu inicjującego pracę komputera, czyli BIOS-u. Następne decyzje związane z powyższym faktem, a również decydujące o mapie pamięci i sposobie funkcjonowania systemu operacyjnego zostały podjęte przez konstruktorów IBM. pierwsze komputery osobiste IBM PC i IBM XT używały procesorów 8088 bądź 8086. Miały, jak pamiętamy, 20-bitową magistralę danych, która pozwalała zaadresować fizyczną pamięć o pojemności 1 MB (przy założeniu organizacji bajtowej). W owym czasie 1 MB pamięci wydawał się ogromnym obszarem. Pamięć tę postanowiono podzielić, przyznając 640 kB (początkowo 512 kB) dla programów i ich danych, natomiast 384 kB pamięci zarezerwowano do specjalnych zadań. Pierwszy z wymienionych obszarów nosi nazwę pamięci konwencjonalnej, drugi zaś pamięci górnej (inne nazwy to pamięć zarezerwowana czy bloki pamięci górnej - ang. *Upper Memory Blocks)*. W pamięci górnej zarezerwowano miejsce dla BIOS-u i innych obszarów istotnych dla obsługi systemu. Umieszczono tam VIDEO BIOS, VIDEO RAM (czyli obszar pamięci zawierający treść obrazu wyświetlanego na monitorze) współpracujący z kartą graficzną czy też ramkę pamięci stronicowanej (EXPANDED lub inaczej LIM-EMS). Obszary te nie tworzyły spójnego obszaru, lecz dzieliły obszar pamięci górnej na bloki (stąd nazwa bloki pamięci górnej).

Bardzo szybko okazało się, że 1 MB (a tym bardziej 640 kB) pamięci nie jesi tak dużym obszarem. Pojawił się wówczas pomysł rozbudowy pamięci dołączanej na kartach rozszerzających. Ponieważ procesory 8086/88 nie potrafiły fizycznie, bez dodatkowych zabiegów, zaadresować więcej niż 1 MB pamięci, wprowadzono odpowiednie sterowniki oraz utworzono w pamięci górnej tak zwaną ramkę pamięci stronicowanej, co pozwoliło zaadresować 32 MB pamięci. Pamięć tę nazywano LIM-EMS (od nazw firm-twórców standardu) lub pamięcią EXPANDED. Inna możliwość rozszerzenia pamięci pojawiła się wraz z nowymi procesorami mającymi szerszą niż 20 bitów magistralę adresową. Pamięć powyżej 1 MB adresowaną przez te procesory nazywa się EXTENDED lub XMS. Wykorzystanie jej przez system operacyjny DOS było niewielkie, gdyż obszar pamięci górnej tworzył nieciągłość, której DOS nie potrafił ominąć, przynajmniej jeśli chodzi o ładowanie do pamięci operacyjnej programów. Jedynie do bloków pamięci górnej oraz obszaru pierwszych 64 kB powyżej

pierwszego megabajtu (związanego z linią adresową A20 - patrz podrozdział 6.2.1.3) możliwe było ładowanie części systemu operacyjnego DOS oraz sterowników urządzeń. Potrzebne są jednak do tego sterowniki pamięci, np. HIMEM.SYS i EMM386.EXE firmy Microsoft.

Dopiero pojawienie się możliwości stronicowania oraz systemu operacyjnego (lub nakładki na system operacyjny) Windows pozwoliło w pełni wykorzystać rozbudowaną pamięć EXTENDED. Natomiast pamięć EXPANDED potrzebną niektórym programom tworzy się obecnie przy wykorzystaniu fragmentu pamięci EXTENDED,



Rysunek 6.10. Mapa pamięci komputera IBM PC z systemem DOS

Mapę pamięci wynikłą z opisanych faktów przedstawia rysunek 6.10. Zwracamy uwagę, że w celu zapewnienia czytelnos'ci rysunku nie są zachowane proporcje wielkości poszczególnych obszarów.

Mapę zakresów adresów przyporządkowanych poszczególnym układom wejścia/wyjścia przedstawiono w tabeli 6.4.

Tabela 6.4. Przestrzeń adresowa układów wejścia/wyjścia

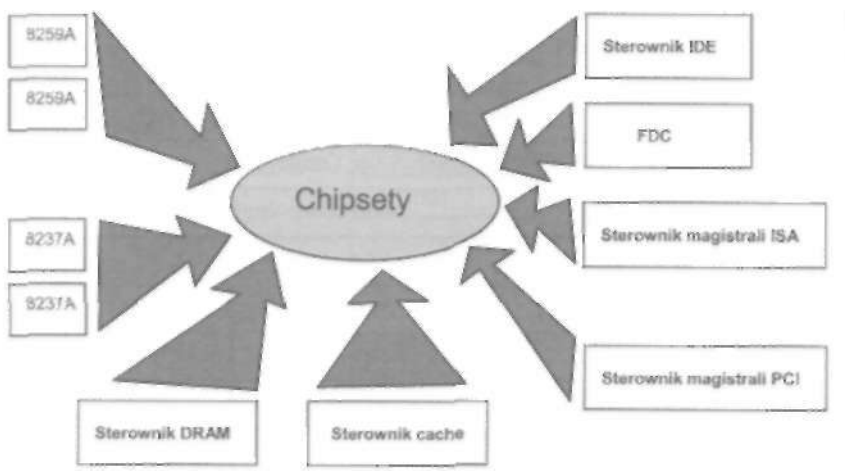
Adres	Nazwa układu
000 – 01F	Sterownik DMA nr 1
020 – 03F	Sterownik przerwań nr 1
040 – 05F	Generatory programowalne
060 – 06F	Sterownik klawiatury
070 – 07F	*Zegar czasu rzeczywistego*
080 – 08F	Rejestr stron DMA
0A0 – 0BF	Sterownik przerwań nr 2
0C0 – 0DF	Sterownik DMA nr 2
0F1	Reset koprocesora arytmetycznego
0F8 – 0FF	Koprocesor arytmetyczny
1F0 – 1F8	Sterownik dysków twardych
200 – 207	Game-port
278 – 27F	Port równoległy nr 2
2F8 – 2FF	*Port szeregowy nr 2*
378 – 37F	Port równoległy nr 1
3B4 – 3BA	Sterownik VGA (mono)
3C0 – 3DA	Karta VGA
3F0 – 3F7	Sterownik dysków elastycznych
3F8 – 3FF	Port szeregowy nr 1

Szczegółową mapę tych portów można znaleźć np. w pozycji [3].

6.3. Chipsety

Współpraca poszczególnych elementów systemu, takich jak jednostka centralna (CPU), podstawowa pamięć operacyjna, pamięć cache, układy wejścia/wyjścia, wymaga dodatkowych układów logicznych koordynujących ich działanie. Zadaniem tych układów jest między innymi dekodowanie adresów, wytwarzanie sygnałów taktują-

cych i sterujących pracą poszczególnych układów, przeprowadzanie arbitrażu i tym podobne. We wczesnych rozwiązaniach płyt głównych układy te były budowane przy użyciu cyfrowych układów funkcjonalnych (średniej skali integracji), co skutkowało dużą liczbą tych układów na płycie głównej. Wraz z rozwojem technologii coraz więcej tych układów trafia do układów wielkiej skali integracji zwanych chipsetami (właściwie do chipsetu; chipset oznacza po angielsku zestaw chipów - układów scalonych, jednak tradycyjnie, choć błędnie, używa się liczby mnogiej - chipsety), zmniejszając w ten sposób liczbę układów scalonych na płycie głównej. Niejednokrotnie układy wewnątrz chipsetów są ścisłymi odpowiednikami funkcjonalnymi układów, które wcześniej występowały jako pojedyncze układy scalone (tak było np. z układami sterownika przerwań 8259A, sterownika DMA 8237A czy sterownika klawiatury), Dodatkowo, wraz z rozwojem nowych standardów, pojawiają się rozwiązania nowych chipsetów obsługujące te standardy. Przykładem może być tu standard PCI czy też możliwość konfigurowania programowego płyt głównych. Koncepcję przekształcania funkcjonalnych układów scalonych występujących na płytach głównych w chipsety przedstawia rysunek 6.11.



Rysunek 6.11. Przekształcanie pojedynczych układów funkcjonalnych w chipsety

Podstawowymi grupami układów występujących w chipsetach są:

- interfejs procesora,
- kontroler pamięci dynamicznych i pamięci cache
- układy wymagane przez standard ISA (np. kontroler przerwań i DMA, kor klawiatury itd.)
- kontrolery magistral i interfejsów (n.p. PCI Express, SATA, IDE itp.),
- system zarządzania poborem mocy,
- inne układy związane z wejściem/wyjściem.

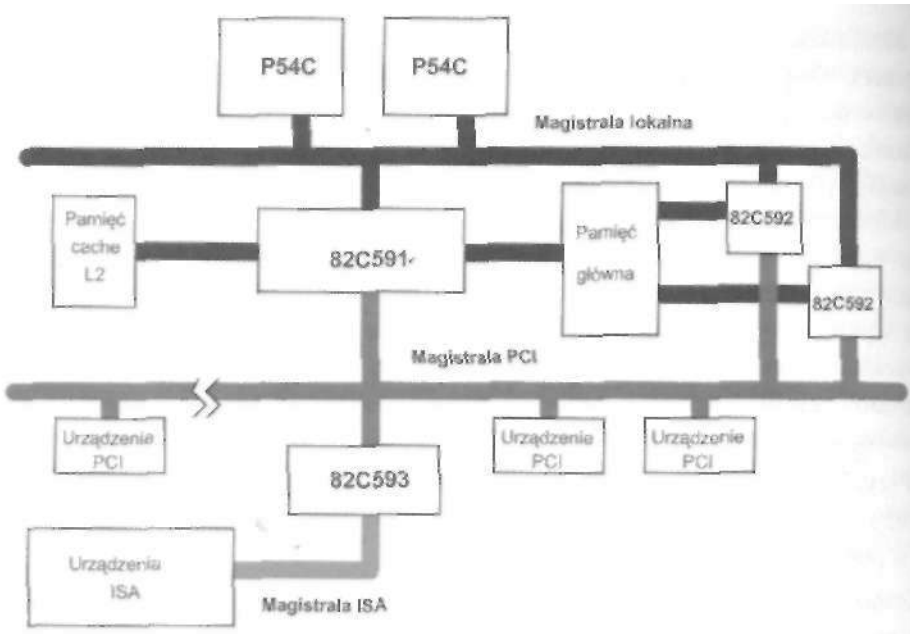
Możliwości chipsetów w znacznej mierze decydują o możliwościach danego komputera. Chipsety dokładnie opisano w ich dokumentacji technicznej (nie jest jednak łatwo dostępna, cześć można uzyskać przez Internet). Dostęp do niej nie jest jednak konieczny. Znakomite źródło informacji o tych możliwościach stanowią dokumentacja płyty głównej i pozycje zawarte w Setupie. Wystarczy dokładnie zapoznać się z dokumentacją komputera (szczególnie płyty głównej) i nim samym (Setup niesie bardzo dużo informacji). Dokładne ich przestudiowanie pozwala, pod warunkiem posiadania i rozumienia pewnych informacji, poznać możliwości komputera i odpowiednio je wykorzystywać. Możliwości zmian dotyczą zarówno ustawień w Setupie, jak i (ewentualnie, choć coraz rzadziej) ustawień zworek czy mikroprzełączników na płycie głównej. Większość z tych zmian powoduje modyfikację sposobu działania chipsetów, a co za tym idzie, sposobu działania samego komputera.

Przykładowe pozycje występujące w Setupach podajemy w dodatku A. Pojęcia i terminy występujące w nim zostały wyjaśnione (taka jest przynajmniej nadzieja autora) w poszczególnych rozdziałach książki.

Poniżej przedstawiamy przykładowe zestawy chipsetów firmy Intel. Pierwszy z nich prezentujemy dla porównania z późniejszymi rozwiązaniami. Składa się z czterech układów scalonych: VL82C591 - sterownik systemu, interfejs procesor-PCI, VL82C592 x 2 - bufory danych, i VL82C593 - interfejs PCI - ISA. Zestaw ten obsługuje procesory Pentium do wersji P54C 66 MHz włącznie, przy czym w systemie mogą pracować dwa takie procesory (opcja ta była jednak rzadko używana). Schemat blokowy systemu zawierającego te chipsety przedstawia rysunek 6.12. Porównanie tego zestawu chipsetów z prezentowanym później zestawem 440BX oraz 845 i 975 pozwala dostrzec kierunek zmian, jakie zachodzą w układach stosowanych do produkcji płyt głównych.

Podstawowe elementy i cechy tego zestawu to:

- interfejs procesora,
- interfejs magistrali PCI,
- arbiter magistrali PCI,
- sterownik pamięci cache L2 (strategia Write-through),
- sterownik pamięci DRAM,
- obsługa SMM (zarządzanie poborem mocy) i Shadow RAM,
- interfejs PCI-ISA,
- układy standardu ISA,
- obsługa opóźnionych zapisów do pamięci.



Rysunek 6.12, Schemat blokowy systemu z chipsetami VL82C59x

Drugim przykładowym zestawem chipsetów jest Intel® 440BX składający się z dwóch elementów: 82443BX - interfejs procesor-PCI, kontroler pamięci dynamicznych i kontroler magistral PCI i AGP, oraz 82371EB - interfejs PCI-ISA. Układy te zaczęto później nazywać odpowiednio mostkiem północnym i mostkiem południowym. Zestaw może obsługiwać płyty z dwoma procesorami Pentium II (100/166MHz).

Konfigurację systemu z zestawem chipsetów 440BX przedstawia rysunek 6.13.

Na rysunku tym zaznaczono też przepustowość poszczególnych magistral łączących układy tworzące jednostkę centralną komputera.

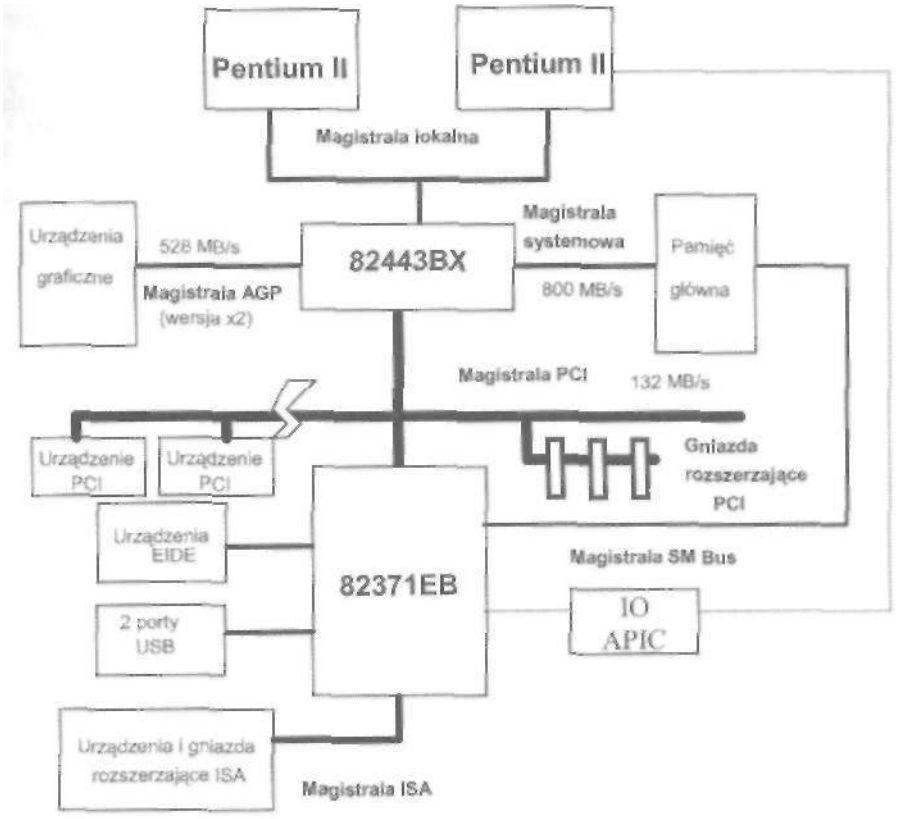
Elementy i własnos'ci zestawu to:

- interfejs procesor-PCI,
- magistrala pamięci 100 MHz (obsługiwane jest też 66 MHz),
  kontroler pamięci DRAM (EDO i SDRAM),
- obsługa autodetekcji typu pamięci,
- obsługa dwóch procesorów Pentium II zgodna z protokołem SMP (ang. *Symetrie Multiprocessor Protocol),*
- obsługa magistrali PCI w wersji 2.1. (4 złącza),
- arbiter magistrali PCI,
- obsługa SMM (zarządzanie poborem mocy),
- kontroler magistrali AGP (ang. *Accelerated Graphics Port),* wersje xl i x2,

- obsługa opóźnionych zapisów,

obsługa równoległych transmisji pomiędzy procesorem, PCI i AGP a pamięcią główną,

- zintegrowany kontroler EIDE (Ultra DMA/33),
- zintegrowany kontroler interfejsu USB (ang. *Universal Serial Bus) - 2* porty,
- , obsługa magistrali ISA, kontroler dysków elastycznych, portów szeregowych i równoległych,
- kontroler przerwań pracy dwuprocesorowej (I/O APIC).



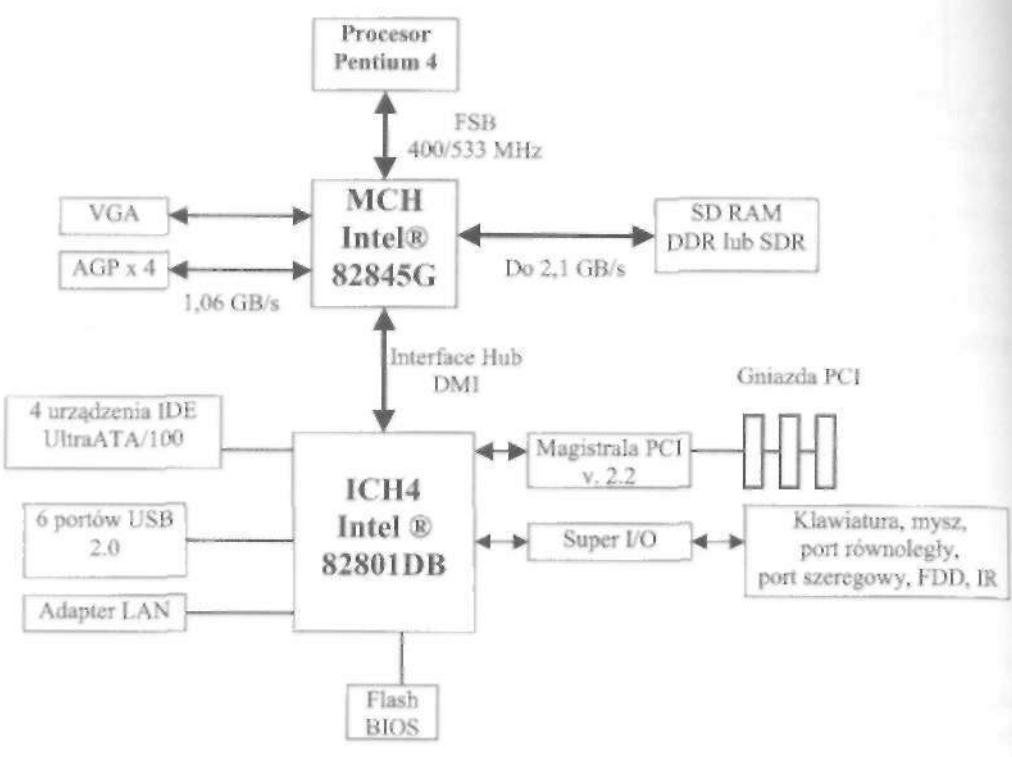
Rysunek 6.13. Schemat blokowy systemu z chipsetami Intel®440BX

Kolejny przykład to zestaw Intel® 845G chipset. Składa się z dwóch układów: układu Intel® 82845G oznaczanego jako GMCH (ang. *Graphics and Memory Controller Hub)* oraz układu Intel® 82801DB (ang *I/O Controller Hub)* - ICH4. Zwracamy uwagę na kolejną zmianę terminologii, mającą jednak pewne uzasadnienie. Pierwszy z układów, komunikujący się z procesorem i pamięcią, nosi nazwę *Memory Controller* ***Hub*** i na schematach jest oznaczany skrótem **MCH.** Drugi z układów, obsługujący komunikację z większością układów wejścia/wyjścia (z wyjątkiem podsystemu gra-

fiki, który zawsze traktowany był w specjalny sposób) nosi nazwę *110 Controlłer Hub* i jest oznaczany skrótem **ICH** (zwykle z numerem, na przykład ICH4, oznaczającym kolejną generację tych układów). Interfejs pomiędzy MCH i ICH oznaczany jest skrótem **DMI** od nazwy *Direct Media Interface* lub po prostu nazwą *Hub Interface*.

Podstawowe własności układu 82845G to:

- obsługa Pentium 4 z technologią Hyper-Threading,
- przeznaczony dla systemów jednoprocesorowych,
- częstotliwość magistrali procesora 400 lub 533 MHz,
- obsługa magistrali AGP x 4,
- zintegrowany adapter graficzny 2D/3D,
- obsługa pamięci SDR 133 lub DDR 200/266 SDRAM z pojedynczym kanałem,
- maksymalny rozmiar pamięci - 2 GB,
- DMI pracuje z częstotliwością 66 MHz (266 MB/s).



Rysunek 6.14. Schemat blokowy systemu z chipsetami Intel® 845G

Własności układu 82801DB są następujące:

. obsługa magistrali PCI (33 MHz) w wersji 2.2,
, interfejs IDE ATA 100,
. obsługa 6 portów USB 2.0 (480 Mb/s),
- obsługa elementów systemu ISA: 2 kontrolery przerwań 8259A, 2 kontrolery DMA 8237A, zegar czasu rzeczywistego zgodny z układem MC 143818, układ generatorów programowalnych 8254,
- zintegrowany adapter sieciowy,
- wbudowany kodek audio (AC'97) i modem.

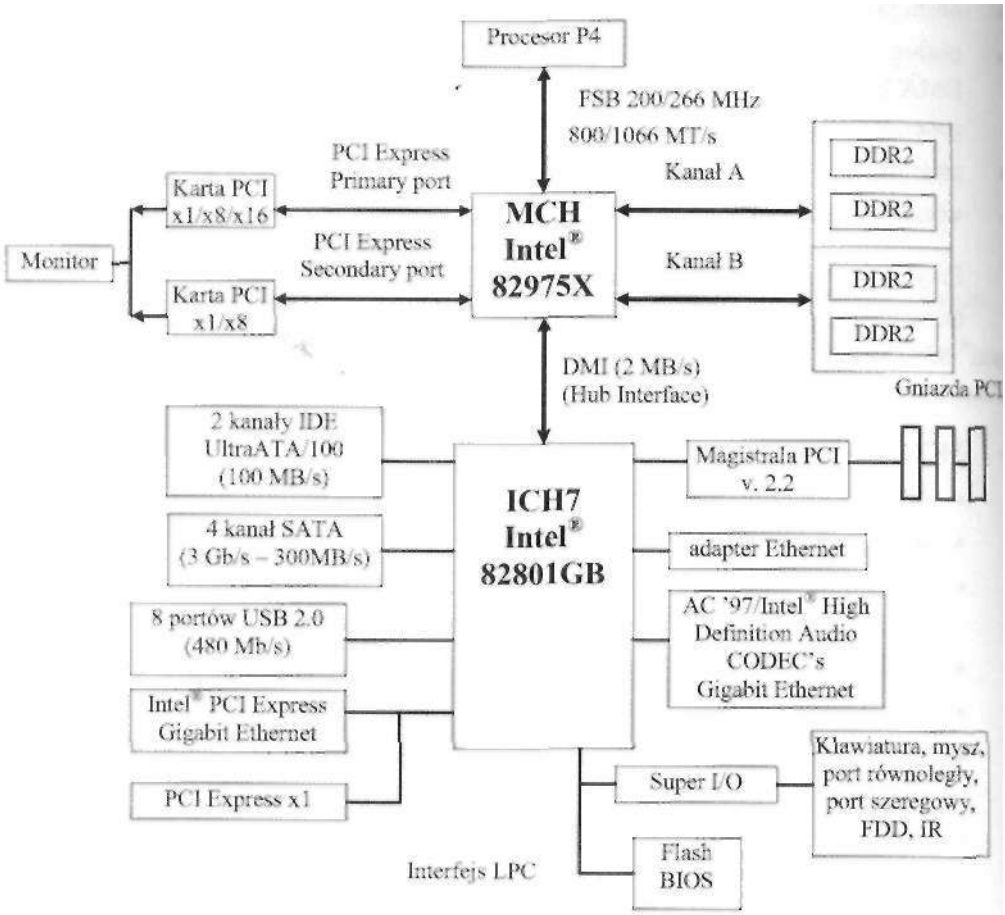
Przykładowy system z chipsetem 845 przedstawia rysunek 6.14.

Ostatnim przykładem jest chipset Intel© 975X Express składający się z układów Intel 82975X MCH oraz układu Intel* 82801GB ICH7. Podstawowe własności tego zestawu to:

> MCH:
  - obsługa Pentium 4, Pentium D i Pentium Extreme Edition,
  - obsługa 2-kanałowej pamięci DDR2 SDRAM 533 i 667 MHz,
  - magistrala procesora o częstotliwości 200/233 MHz i przepływności (quad pumped) 800/1066 MT/s *(mega-transfers per second),*
  - obsługa technologii Hyper-threading,
  - maksymalna wielkość pamięci - 8 GB,
  - 16-kanałowy port PCI Express dla urządzeń graficznych, obsługujący także tryb PCI i AGP,
  - przepustowość DMI 2MB/s (1 MB/s w każdym kierunku),
  - zaawansowana obsługa przerwań.
> ICH:
  - obsługa PCI Express wersja 1.0a,
  - obsługa PCI wersja 2.3,
  - zawiera układy standardu ISA (2 kontrolery przerwań 8259A, 2 kontrolery DMA 8237A, zegar czasu rzeczywistego zgodny z układem MC 143818, układ generatorów programowalnych 8254) podłączone przez interfejs LPC (patrz następny podpunkt),
  - kontroler SATA - 4 porty,
  - kontroler IDE Ultra ATA 100,
  - 8 portów USB 2.0,

- zintegrowany adapter sieciowy,
- Gigabit Ethernet.

Schemat blokowy systemu z chipsetem 975 przedstawia rysunek 6.15.



Rysunek 6.15. Schemat blokowy systemu z chipsetami Intel® 975X Express

Interfejs Low pin Count (LPC) i układy Super I/O

W podpunkcie tym chcemy krótko opisać rozwiązanie pozwalające na zapewnienie obsługi układów wymaganych przez standard ISA przy jednoczesnym braku magistrali ISA, a co za tym idzie, braku magistrali oznaczanej jako X Bus (patrz rysunek 6.2). Magistrala ta zapewniała komunikację systemu z urządzeniami wymaganymi przez standard ISA.

Celem istnienia magistrali LPC jest zapewnienie obsługi układów wcześniej obsługiwanych przez magistralę ISA/X-Bus przy braku tej ostatniej, co ma miejsce

w nowoczesnych płytach głównych. Interfejs LPC jest przezroczysty dla operacji wejścia/wyjścia (inaczej mówiąc, operacje te nie zauważają, że są obsługiwane przez LPC zamiast przez ISA/X-Bus) i kompatybilny z istniejącymi urządzeniami peryferyjnymi.

Interfejs LPC wymaga jedynie siedmiu obowiązkowych sygnałów (cztery linie multipleksowane dwukierunkowe adresu/danych i trzy dwukierunkowe linie sterujące) oraz zezwala na szes'ć dodatkowych linii pozwalających na obsługę przerwań, DMA oraz zarządzanie poborem mocy. Mimo jedynie czterech linii danych interfejs LPC jest nieco szybszy od magistrali ISA/X-Bus, ponieważ taktowany jest zegarem 33 MHz (w porównaniu z zegarem 8 MHz).

Jedną z głównych zalet LPC, sygnalizowaną zresztą w nazwie, jest niewielka liczba wymaganych linii. Pozwala zaoszczędzić od 30 do 72 linii w porównaniu z magistralą ISA, co jest szczególnie ważne w przypadku komputerów mobilnych.

Magistrala LPC współpracuje zwykle z kontrolerem nazywanym Super 170, wprowadzonym do użytku w początku lat 80. i przeznaczonym do obsługi urządzeń niewymagających dużej przepustowości, takich jak sterownik dysków elastycznych, porty szeregowe (RS 232C), równoległe (Centronics i pochodne), sterownik klawiatury, myszy itp. Kontroler Super I/O jest obecnie zwykle elementem mostka południowego lub układu ICH.

6.4. Standardy magistrali rozszerzającej

Rodzaj magistrali rozszerzającej decyduje między innymi o szybkości przesyłania informacji pomiędzy procesorem lub pamięcią a innymi urządzeniami występującymi w systemie (patrz podrozdziały 3.4 i 3.5 poświęcone układom i operacjom wejścia/wyjścia). Wymagania w stosunku do tej szybkości ciągle rosną. Podamy kilka przykładów. Jednym z ważniejszych jest komunikacja z kartą graficzną. Już wprowadzenie graficznego interfejsu użytkownika (ang. *GUI - Graphical User's Interface),* np. Windows, zwiększyło wymagania co do szybkości komunikacji z tą kartą. Dalszy ogromny wzrost wymagań co do szybkości transmisji wynika z konieczności zapewnienia obsługi telekonferencji, płynnego odtwarzania ruchomych obrazów czy też obsługi gier używających obrazów 3D. Innym przypadkiem wymagającym dużej szybkości transmisji jest przykładowo obsługa szybkiego interfejsu dysków twardych, na przykład SCSI. Kolejnym przykładem może być obsługa szybkich kart sieciowych, np. FDDI czy Gigabit Ethernet.

Podajemy krótką charakterystykę najpopularniejszych standardów magistrali rozszerzającej. Uwagę koncentrujemy szczególnie na działaniu magistrali PCI (bardzo długo stanowiącej standard magistrali rozszerzającej dzięki starannemu i rozwojowemu opracowaniu jej założeń i własności) oraz jej następcy - standardowi PCI Express.

6.4.1. ISA

ISA jest magistralą 16-bitową, taktowaną zegarem około 8 MHz (BCLK-patrz rozdział 6.)- Ponieważ transmisja jednego słowa (2 bajtów), przy założeniu braku stanów oczekiwania, zajmuje dwa cykle zegara, to maksymalna przepustowość tej magistrali wynosi:

$$\frac{8\ \text{MHz} \times 2\ \text{B}}{2\ \text{takty}} = 8\ \text{MB/s}$$

W stosunku do szybkości przesyłania informacji przez współczesne procesory i urządzenia to bardzo wolna transmisja, a więc jedno z wąskich gardeł przetwarzania informacji.

Dodatkową wadę magistrali rozszerzającej ISA stanowi brak mechanizmów wspierających autokonfiguracje (o czym piszemy w dalszej części rozdziału). Cechy te (wady) powodują, że od kilku lat magistrala ISA nie jest implementowana na płytach głównych komputerów IBM/PC. Część magistrali ISA oznaczana jako X-Bus, służąca do komunikacji z elementami systemu ISA, została obecnie zastąpiona interfejsem (magistralą) LPC (piszemy o nim w podpunkcie 6.3).

6.4.2. EISA

Pewną próbą unowocześnienia standardu ISA było wprowadzenie jego rozszerzenia oznaczanego jako EISA (ang. *Enhanced ISA)*. Nie zmieniając szybkości zegara taktującego magistralę, zwiększono jej szerokość do 4 bajtów. W trybie burst, gdy jedno czterosłowo transmitowane jest w jednym takcie zegara, maksymalna przepustowość tej magistrali wynosi 32 MB/s. Magistrala ta zawierała pewne elementy autokonfiguracji. Podobnie jak ISA, magistrala nie jest obecnie implementowana na płytach głównych.

6.4.3. VESA Local Bus

Magistrala VESA Local Bus (oznaczana w skrócie V-LB lub VL-Bus) jest tak zwaną *magistralą lokalną*. Magistrala lokalna to taka, która korzysta bezpośrednio z sygnałów sterujących procesora, bez żadnej ich translacji na inny zestaw sygnałów. \ Magistrala VL-Bus dobrze spełnia tę definicję.

W najprostszej wersji magistrala VL-Bus jest zestawem części niebuforowanych sygnałów procesora 80486, do których dołączono dodatkowe sygnały związane z przejęciem zarządzania magistralą i przerwaniami. Oznaczana jest wówczas jako typ A. Może obsługiwać jedno urządzenie umieszczone bezpośrednio na płycie głównej, W wersji nieco bardziej skomplikowanej magistrala VL-Bus obsługuje do trzech gniazd rozszerzających znajdujących się na płycie głównej. To typ B magistrali

VL-Bus. Sygnały procesora są wówczas buforowane. Oznacza to, że dołączenie kilku urządzeń magistrala lokalna procesora odbiera jako pojedyncze obciążenie.

Magistrala VL-Bus jest magistralą 32-bitową. Szybkość wynosi 105 MB/s przy zegarze taktującym o częstotliwości 33 MHz (1 takt na wpisanie adresu i 4 takty na transmisję czterech dwusłów w trybie burst).

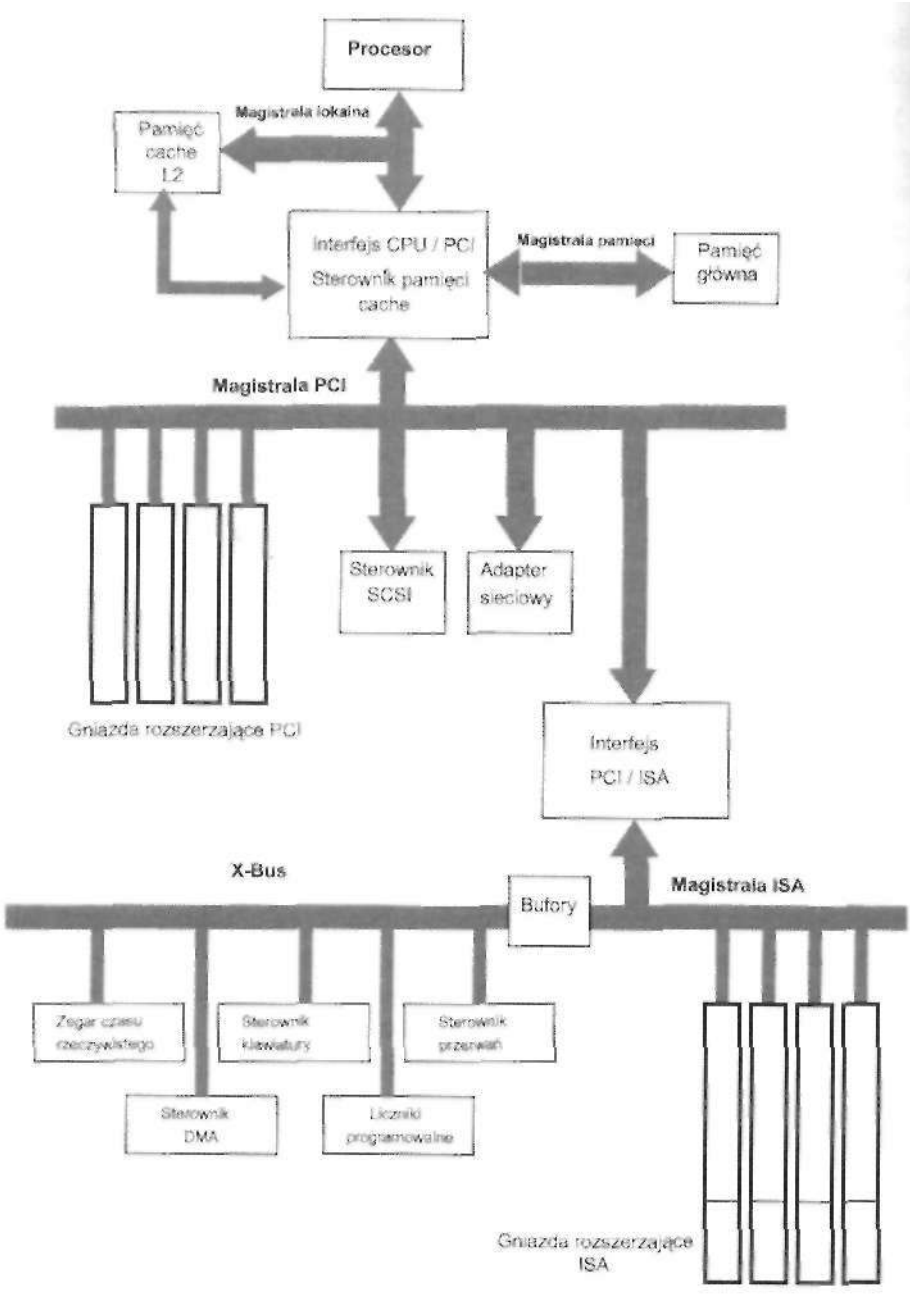
Zaletami magistrali VL-Bus są jej niski koszt i prostota. Wady to niewielka liczba gniazd rozszerzających oraz brak możliwości równoległego wykonywania operacji na magistrali lokalnej procesora i magistrali rozszerzającej. Druga z wymienionych wad wynika z faktu, że do obsługi obu magistral używane są te same sygnały (buforowanie zwiększa jedynie obciążalność linii). Jedną z wymienianych wad stanowi fakt, że jest jednak związana z konkretnym typem procesora (80486). Zmiana procesora wymagałaby zmiany konstrukcji kart lub stosowania układu zapewniającego translację „nowych" sygnałów na „stare". Takie samo jednak stwierdzenie, jak zobaczymy, dotyczy PCI.

6.4.4. PCI

PCI nie jest w istocie magistralą lokalną, chociaż nosi taką nazwę. W rzeczywistości dysponuje własnym, dobrze zdefiniowanym zestawem sygnałów, różnym od sygnałów magistrali lokalnej procesora. Z procesorem i pamięcią cache magistrala PCI komunikuje się za pośrednictwem specjalnego układu zwanego sterownikiem lub interfejsem PCI (ang. *PCI bridge)*. Schemat blokowy systemu z magistralą PCI przedstawia rysunek 6.16.

Na schemacie tym widać od razu jedną z zalet magistrali PCI. Możliwa jest jednoczesna komunikacja procesora z pamięcią cache (przy użyciu magistrali lokalnej) i jednego z urządzeń dołączonych do magistrali, przykładowo z pamięcią główną lub pamięcią wideo. Inaczej mówiąc, trzech magistral: lokalna procesora, PCI i (w naszym przykładzie) ISA, można używać równocześnie.

Magistrala PCI realizuje wszelkie przesłania w trybie burst, zarówno dla zapisu, jak i dla odczytu. Oznacza to, że w pierwszym cyklu inicjującym przesłanie podawany jest rodzaj operacji i adres początkowy, w następnych cyklach zaś przesyłane jest jedna lub więcej danych, przy czym ich liczba nie jest limitowana. To kolejna zaleta magistrali PCI pozwalająca osiągnąć dużą szybkość przesyłania danych. Częstotliwości zegara magistrali dla wersji 2.1 wynoszą do 66 MHz. Pozwala to osiągnąć w trybie burst maksymalny transfer 264 MB/s dla magistrali 32-bitowej i 528 MB/s dla magistrali 64-bitowej.



Rysunek 6.16. Schemat blokowy systemu z magistralą PCI

Dalsze własności magistrali PCI są następujące. Dzięki wykorzystaniu zjawiska tai odbitych (ang. *Reflected-Wave Switching)* magistrala ta ma bardzo korzystne własności energetyczne (brak terminatorów). Przydział dostępu do magistrali może się odbywać jeszcze w trakcie posiadania dostępu do magistrali przez zarządcę magistrali

I realizującego bieżącą operację (ang. *hidden bus arbitration).* Magistrala PCI zapewnia wszelkie mechanizmy potrzebne do realizacji autokonfiguracji. Układy scalone zapewniające współpracę procesora z magistralą są projektowane pod kątem niezmienności sygnałów magistrali. Inaczej mówiąc, zmiana typu procesora wymaga modyfikacji układu będącego interfejsem pomiędzy procesorem a magistralą, a nie zmian w układach podłączonych do magistrali. Pojedyncza magistrala zapewnia teoretycznie Współpracę do 256 układów funkcjonalnych. Dodatkowo, przy zastosowaniu tak zwanych mostów PCI-PCI możliwa jest współpraca wielu magistral PCI (do 256). Wreszcie standard opracowano z myślą o jego dalszym rozwoju. Dlatego istnieje także jego wersja 64-bitowa (i dane, i adres), a także wersje używające napięcia zasilania zarówno 5 V, jak i 3,3 V.

W praktyce liczba gniazd jest ograniczona, zwykle do 4, 5. Przyczyny tego są natury elektrycznej. Stosowanie metody przełączania z falą odbitą ma zalety energetyczne, jednak płaci się za to szybkością. Metoda ta wymaga niezbyt dużych odległości pomiędzy układami oraz stosunkowo niewielkiej liczby obciążeń (gdyż te ostatnie ze wzrostem liczby pogarszają szybkość narastania zboczy sygnałów). Konsekwencją tego jest właśnie ograniczona liczba gniazd. Pewnym rozwiązaniem, choć rzadko spotykanym, może być stosowanie mostów PCI-to-PCI i użycie kilku magistral PCI w systemie.

Układy współpracujące z magistralą mogą być wykonane w postaci adapterów umieszczonych na płycie głównej (na naszym przykładowym schemacie jest to sterownik SCSI i adapter sieciowy) lub mogą być umieszczane w gniazdach rozszerzających.

6.4.4.1. Zasada działania magistrali PCI

Poniżej przedstawiamy w skrócie zasadę działania magistrali PCI.

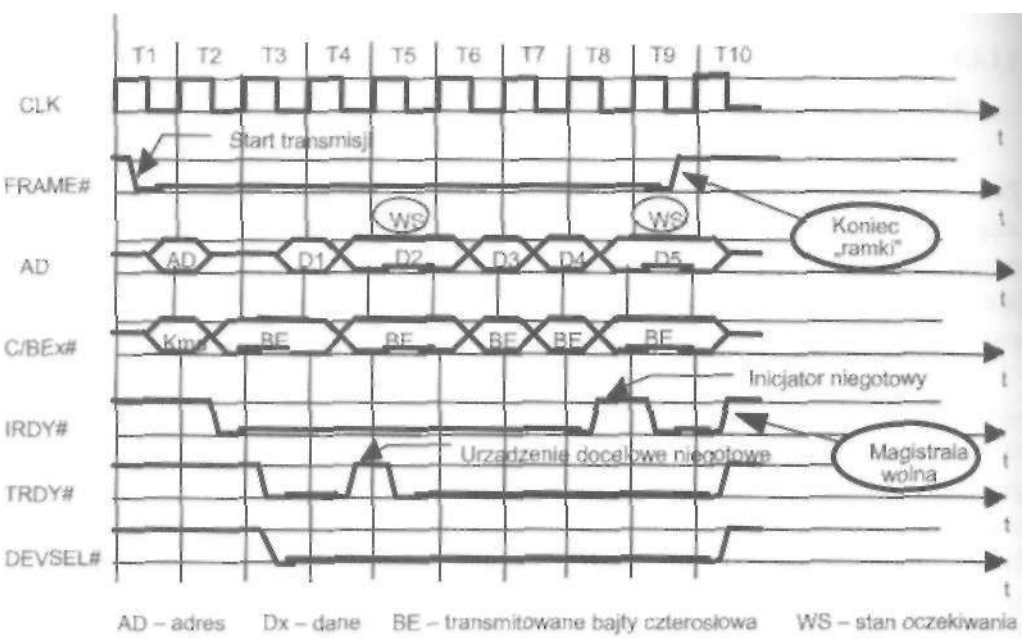
Standard PCI dzieli urządzenia dołączone do magistrali na dwie kategorie. Pierwsza z nich to urządzenia inicjujące transmisję (inicjatory, ang. *initiators),* które stają się zarządcą magistrali. Druga kategoria to urządzenia docelowe (ang. *targets)* mogące transmitować dane. Należy zwrócić uwagę, że podział na urządzenia inicjujące i docelowe nie ma nic wspólnego z kierunkiem transmisji (przykładowo inicjator może zarówno nadawać, jak i odbierać informację).

Sygnały magistrali PCI możemy podzielić na kilka grup. Pierwsza z nich to adres/dane/komendy. Magistrala PCI ma multipleksowaną magistralę danych i adresową AD31/AD0. W fazie adresowania liniami tymi przesyłany jest adres, a w fazie transmisji - dane. Z liniami tymi związane są też cztery linie C/BE3# : HC/BE0#. W fazie adresowania ich stan decyduje o tym, jaka operacja zostanie wykonana. W fazie transmisji danych (podobnie jak w procesorach 386 i późniejszych) decydują one, które bajty czteroslowa będą transmitowane. Z przesyłaniem danych związany jest też sygnał PAR (kontrola parzystości).

Kolejna grupa to sygnały sterujące. Należą do nich sygnały: FRAME#, TRDYI, IRDY#, STOP#, DEVSEL# i IDSEL. Ich rola zostanie wyjaśniona poniżej przy opisie pracy magistrali. Dalsze grupy sygnałów to: systemowe (CLK, RST#), obsługi błędów (PERR#, SERR#), zarządzania dostępem do magistrali (arbitrażu) (REQ#, GNT#), obsługi przerwań (INTA#, INTB#, INTC#, INTD#), sygnały testujące i obsługi pamięci cache. Ostatnia grupa sygnałów dotyczy 64-bitowego rozszerzenia magistrali (AD63-AD32, C/BE7#-C/BE4#, PĄR64, REQ64#, ACK64#).

Transmisję na magistrali PCI przedstawia rysunek 6.17. Ma ona następujący przebieg. Inicjator, któremu w wyniku arbitrażu został przyznany dostęp do magistrali, podaje na linii AD adres początkowy identyfikujący jednocześnie urządzenie docelowe (każde urządzenie ma przyznany pewien zakres adresów), a na linii C/BEx# kod rodzaju operacji.

Inicjator uaktywnia też sygnał FRAME# (ramka). Adres i kod komendy powinny zostać zatrzaśnięte w rejestrach urządzenia docelowego. W odpowiedzi, w określonym czasie, zaadresowane urządzenie docelowe powinno odpowiedzieć uaktywnieniem sygnału DEVSEL# (w przeciwnym wypadku transmisja nie jest realizowana). W kolejnych taktach przesyłane są liniami ADx dane. Przesłanie kolejnej informacji wymaga gotowości zarówno inicjatora, jak i urządzenia docelowego, co jest sygnalizowane aktywnymi poziomami sygnałów IRDY# (inicjator) i TRDY# (urządzenie docelowe), W przypadku braku poziomu aktywnego chociaż jednego z tych dwóch sygnałów wstawiane są stany oczekiwania.



Rysunek 6.17. Przebieg transmisji na magistrali PCI

Należy podkreślić, że w trakcie transmisji przekazywany jest tylko adres początkowy. Generowanie kolejnych adresów należy do urządzenia docelowego.

O zakończeniu transmisji decyduje inicjator. W trakcie transmisji ostatniej informacji ustawia poziom nieaktywny sygnału FRAME# i uaktywnia sygnał IRDY#. Po zakończeniu transmisji ostatniej informacji przejście sygnału IRDY# w stan nieaktywny sygnalizuje zwolnienie magistrali.

'Oczywiście przebiegi na rysunku 6.17 prezentują sytuacje bardzo uproszczoną, lecz oddającą sposób działania magistrali PCI. Przedstawienie jednak wszystkich możliwych przypadków arbitrażu, realizacji i zakończenia transmisji wymagałoby napisania osobnej książki.

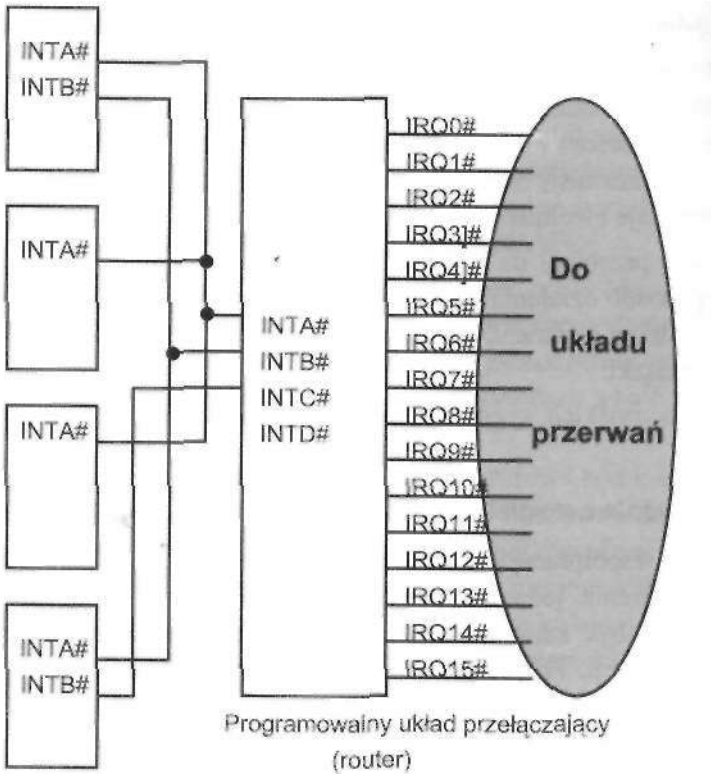
Wartości wszystkich sygnałów sterujących są próbkowane dodatnim zboczem zegara.

6.4.4.2. Przerwania a magistrala PCI

Urządzenia współpracujące z magistralą PCI można podzielić na jedno- i wielofunkcyjne. Urządzenie jednofunkcyjne jest pojedynczym urządzeniem logicznym. Przykładem może być karta graficzna. Urządzenie wielofunkcyjne pełni od strony logicznej kilka funkcji. Przykładem mogłaby tu być karta multi I/O (interfejs szeregowy, równoległy, sterownik dysków elastycznych i dysku twardego).

Magistrala PCI dysponuje czterema liniami zgłoszenia przerwań oznaczonych INTA# + INTD#. Przerwanie jest zgłaszane poziomem (niskim), co umożliwia współdzielenie danej linii zgłoszenia przerwania przez kilka urządzeń. Standard PCI określa, że urządzenie jednofunkcyjne może używać tylko linii zgłoszenia przerwania INTA#. Urządzenia wielofunkcyjne mogą używać wszystkich czterech linii. Natomiast standard PCI nie precyzuje sposobu podłączenia linii zgłoszeń przerwań INTA#H : INTD# do układu przerwań systemu. Linie te mogą być zarówno dowiązane na stałe do określonych wejść sterownika przerwań, jak i na przykład przez programowalny układ przełączający (ang. *programmable router),* pozwalający w sposób programowy zmieniać przyporządkowanie linii przerwań magistrali PCI sygnałom zgłoszeń przerwań układu przerwań systemu. Jedno z wielu możliwych rozwiązań przedstawiono na rysunku 6.18.

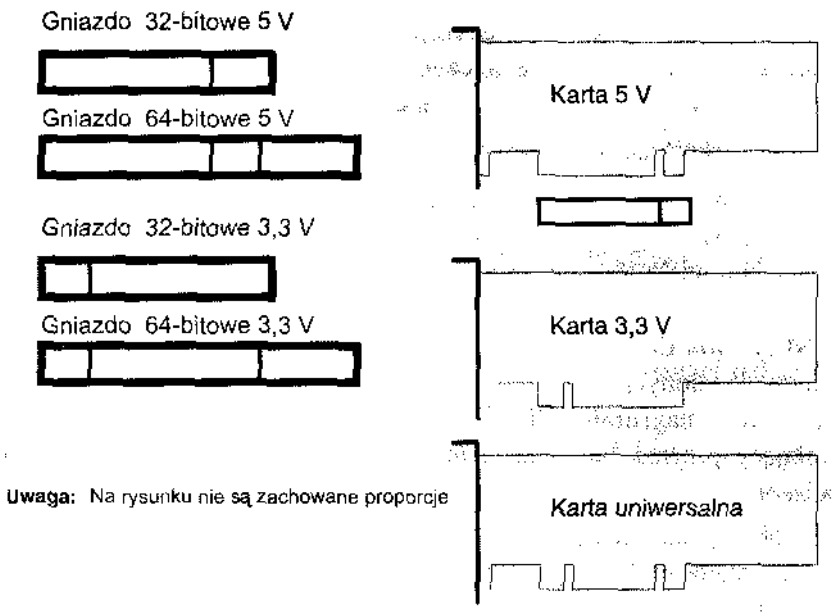
Na rysunku tym widać miedzy innymi, że urządzenia 1 i 3 oraz 2 i 3 używają wspólnych linii do zgłaszania swoich przerwań (są to przerwania współdzielone - ang. *sharable).* Wymaga to oczywiście odpowiedniej konstrukcji programów obsługujących te przerwania, a także istnienia bitów w rejestrach tych urządzeń pozwalających stwierdzić, które z nich (jedno lub obydwa) zgłosiło przerwanie.



Rysunek 6.18. Przykładowy sposób podłączenia sygnałów przerwań magistrali PCI do systemu

6.4.4.3. Wersje elektryczne kart PCI

Urządzenia PCI mogą używać zarówno logiki (i napięcia zasilającego) 5 V, jak i 3,3 V. Typ gniazd rozszerzających pochodzi, podobnie jak dla dodatkowego gniazda VL-Bus, od gniazd zastosowanych dla magistrali Micro Channel. Gniazda dla urządzeń zasilanych napięciem 5 V lub 3,3 V różnią się umieszczeniem w złączu przegrody (klucza) zabezpieczającej przed włożeniem niewłaściwej karty (w miejscu umieszczenia przegrody na złączu krawędziowym karty jest szczelina). Karty mogą być wykonywane jako urządzenia zasilane napięciem 5 V albo 3,3 V lub jako karty uniwersalne, zasilane dowolnym z tych dwóch napięć. Schematyczny wygląd gniazd oraz kart przedstawia rysunek 6.19.



Rysunek 6.19. Wygląd gniazd i kart PCI

6.4.5. Magistrala PCI-X

W momencie opracowywania magistrala PCI była bardzo nowoczesna i szybka (na owe czasy). Dodatkowo przy opracowywaniu jej konstrukcji od początku uwzględniano możliwości jej rozwoju. Dlatego mimo szybkich zmian w technice komputerowej magistrala PCI przez długi czas zapewniała wystarczające osiągi. Jednak z upływem czasu dały znać o sobie pewne ograniczenia rodzące potrzebę nowych rozwiązań. Ograniczeniami tymi były:

- maksymalna częstotliwość taktowania 66 MHz, co wynika ze sposobu obsługi sygnałów sterujących i stosowanej metody fali odbitej,
- zmniejszenie rzeczywistej przepływności magistrali z powodu możliwości wstawiania stanów oczekiwania dla wolnych urządzeń, zarówno przez inicjator, jak i urządzenie docelowe,
- brak informacji o liczbie transmitowanej informacji, czego rezultatem było nieefektywne zarządzanie buforami transmisji,
- nieefektywna obsługa opóźnionych transakcji (na przykład w przypadku powtarzania czy też wznawiania transmisji),
- mało efektywna obsługa przerwań (używających w większości przypadków jednej linii zgłoszenia przerwania).

Kolejną wersją magistrali rozszerzającej będącą rozwinięciem magistrali PCI, w której starano się usunąć te ograniczenia, jest magistrala oznaczana PCI-X (ang, *PCI e.Xtension)*. Umożliwiono w niej zwiększenie szybkos'ć zegara taktującego i wprowadzono wiele ulepszeń poprawiających szybkość transmisji. Ulepszenia te to:

- Wprowadzenie dodatkowej fazy w protokole transmisji magistrali PCI-X, zwanej fazą atrybutów (ang. *attribute phase),w* której między innymi przekazywana jest informacja o wielkości transmitowanego bloku czy też możliwości rezygnacji ze sprawdzania spójności pamięci głównej z pamięcią cache, na przykład dla obszarów NCA *(Non Cachable Areas* - patrz rozdział 3.7). Atrybuty pozwalają też na obsługę określonych transakcji w pierwszej kolejności, co nie było możliwe w przypadku magistrali PCI. Ponadto w atrybutach wprowadzono identyfikator określający przynależność danej transakcji do transmisji pomiędzy określonymi urządzeniami.
- Zmieniony sposób obsługi w przypadku konieczności oczekiwania na informację. Transakcja może być rozbita na dwie fazy, żądania i realizacji, pomiędzy którymi kontroler magistrali może obsługiwać inne transakcje. Z kolei w przypadku pojawiania się konieczności oczekiwania na realizacje transmisji (stany oczekiwania) dostęp do magistrali PC1-X przekazywany jest innym urządzeniom. Eliminuje to przestoje magistrali.
- Urządzenia PCI-X muszą obsługiwać architekturę zgłaszania przerwań oznaczaną skrótem MSI (ang. *Message Signaled Inermpt* - przerwanie sygnalizowane komunikatem). W rozwiązaniu tym przerwanie jest zgłaszane przez urządzenie przez przesłanie (w wyniku wykonania cyklu zapisu) komunikatu do MCH (czy też mostka północnego) zawierającego wektor przerwania. Eliminuje to konieczność identyfikacji źródła przerwania (co występowało w przypadku przerwań współdzielonych na PCI - patrz punkt 6.4.4.2) oraz nie wymaga linii sygnału zgłoszenia przerwania.

Magistrala PCI-X może, w zależności od wersji, pracować z zegarem o częstotliwości 66, 133, 266 lub 533 MHz.

6.4.6. Magistrala PCl Express

Kolejnym rozwiązaniem obsługi magistrali rozszerzającej jest standard o nazwie PCI Express. Reprezentuje odmienny sposób komunikacji pomiędzy urządzeniami i jest uważana za magistralę trzeciej generacji (pod taką angielską nazwą - *3GIO - Third Generation l/O architecture* była zresztą opracowywana). Zamierzeniem jej twórców było stworzenie magistrali, która mogłaby zastąpić istniejące magistrale PCI, PCI-X i AGP oraz zapewnić komunikację wszelkich urządzeń wymagających dużego transferu (na przykład komunikację pomiędzy układami MCH a ICH, choć możliwość

ta nie została jeszcze wykorzystana). Przewiduje się zastosowanie tej magistrali zarówno w komputerach typu desktop, jak i w dużych maszynach serwerowych wymagających podłączenia wielu, z reguły szybkich, urządzeń.

Magistrala PCI Express (skrótowo oznaczana PCI-XP) zapewnia bardzo szybkie, wysoko wydajne łącze szeregowe typu point-to-point (punkt-punkt), zapewniające komunikację w dwóch kierunkach (podwójne łącze simpleksowe). Dane nadawane i odbierane mają oddzielne zestawy linii transmisyjnych. Nadajniki i odbiorniki magistrali PCI wykorzystują różnicową metodę transmisji danych (zalety metody różnicowej przedstawione są w drugiej części podręcznika w rozdziale na temat interfejsu SCSI). Informacja pomiędzy urządzeniami przesyłana jest w postaci pakietów zawierających oprócz danych dodatkowe informacje.

Fizyczne połączenie pomiędzy dwoma urządzeniami PCI Express nosi nazwą **łącza** (ang. *Link).* Łącze może składać się z x1, x2, x4, x8, x12, x16, lub x32 połączeń point-to-point zwanych s'cieżkami (ang. *Lanes).* Pojedyncza ścieżka składa się z czterech linii będących dwoma parami sygnałów transmitujących szeregowo dane w dwóch kierunkach. Częstotliwość zegara wynosi 2 GHz, zatem szybkość transmisji w jednym kierunku to 2,5 Gb/s (gigabitu na sekundę) na jedną ścieżkę, co łącznie (w obu kierunkach) daje transfer 5 Gb/s.

Transmitowane dane są dodatkowo kodowane za pomocą transkodera 8b/10b (do każdych ośmiu bitów dodawane są dwa bity nadmiarowe). Tworzy to 25-procentowy nadmiar przesyłanej informacji. Operacja ta jest jednak niezbędna. Pomiędzy urządzeniami na magistrali PCI Express nie jest przesyłany przebieg zegarowy. Przebieg ten musi zostać odtworzony przez urządzenie odbierające. Jest to realizowane przy użyciu układów **pętli sprzężenia fazowego** (ang. ***PLL*** - *Phase Locked Loop).* Układy te wymagają z kolei odpowiedniej gęstości zboczy (zmian 0—>1 i 1—>0) w transmitowanych pakietach (podobny problem opisujemy w drugiej części podręcznika, omawiając metody kodowania informacji dla dysków twardych). Gęstość tę zapewnia właśnie kodowanie 8b/l Ob.

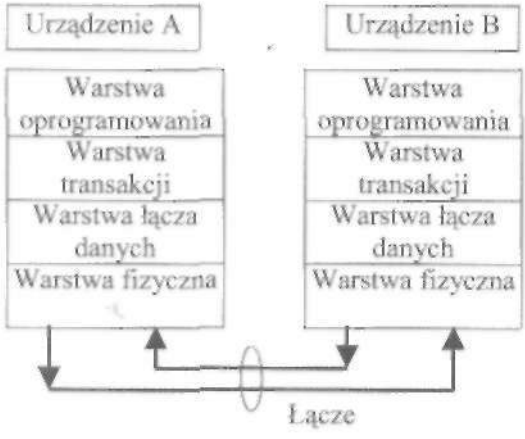
Uwzględniając szybkość transmisji pojedynczej ścieżki oraz kodowanie 8b/10b (czyli 25-procentowy nadmiar), podajemy w tabeli 6.5 transfer łączy magistrali PCI Express x1, x2, x4, x8, x12, x16, i x32 (w gigabajtach na sekundę). Wynika on z prostego wyliczenia. Szybkość transmisji w jednym kierunku mnożymy przez dwa (transmisja w obu kierunkach), następnie przez liczbę ścieżek i wreszcie dzielimy przez dziesięć (bo zamiast 1 bajtu, czyli 8 bitów, przesyłamy 10 - kodowanie 8b/10b), otrzymując wynik w bajtach na sekundę. Na przykład dla magistrali x4 mamy

$$(2 \text{ x } 2{,}5 \text{ GHz x } 4)/10 = 2 \text{ GB/s}$$

Rozwiązania stosowane dla magistrali PCI Express są w znacznej mierze podobne do rozwiązań sieciowych. Dlatego, podobnie jak w sieciowym modelu OSI, w obsłudze transmisji na magistrali PCI Express wyróżniono warstwy definiujące działania realizujące transmisję. Warstwy te pokazano na rysunku 6.20.

Tabela 6.5. Transfery różnych wersji złącza PCI Express

Szerokość łącza PCI Express	x1	x2	x4	x8	x12	x16	x32
Całkowity transfer w GB/s	0,5	1	2	4	6	8	16



Rysunek 6.20. Model warstwowy magistrali PCI Express

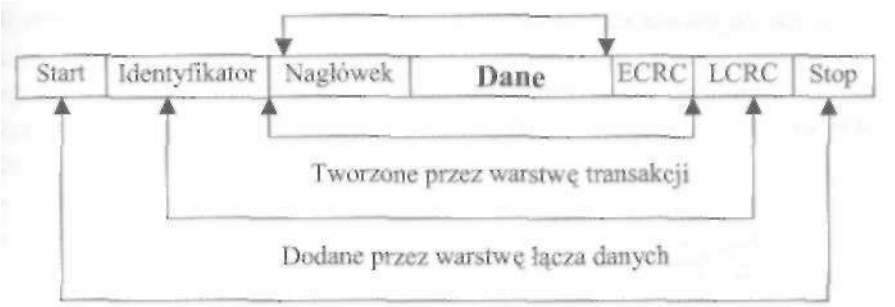
Warstwa oprogramowania generuje żądania transmisji. Dodatkowym jej zadaniem jest zapewnienie kompatybilności z magistralą PCI. Warstwa transmisji w wyniku żądania transmisji generuje serię realizujących ją pakietów. Warstwa łącza danych zapewnia pewność i spójność transmisji przez dodanie identyfikatorów oraz bajtów kontrolnych. Tak przygotowane pakiety są następnie transmitowane przez warstwę łącza składającą się z układów nadawczych, odbiorczych i medium transmisyjnego (okablowania).

Ponadto warstwa fizyczna, przykładowo nadajnika, odpowiedzialna jest za następujące działania:

- dodanie znaku Start oraz Stop;
- w przypadku łącza z wieloma ścieżkami przesyłanie pakietów do kolejnych ścieżek;
- eliminowanie powtarzających się sekwencji przez pseudolosowe przestawianie kolejności bitów (układ realizujący to nazywamy skramblerem). Proces ten zmniejsza zakłócenia elektromagnetyczne (EMI);
- kodowanie 8b/l0b;
- zamianę równoległej postaci informacji na szeregową.

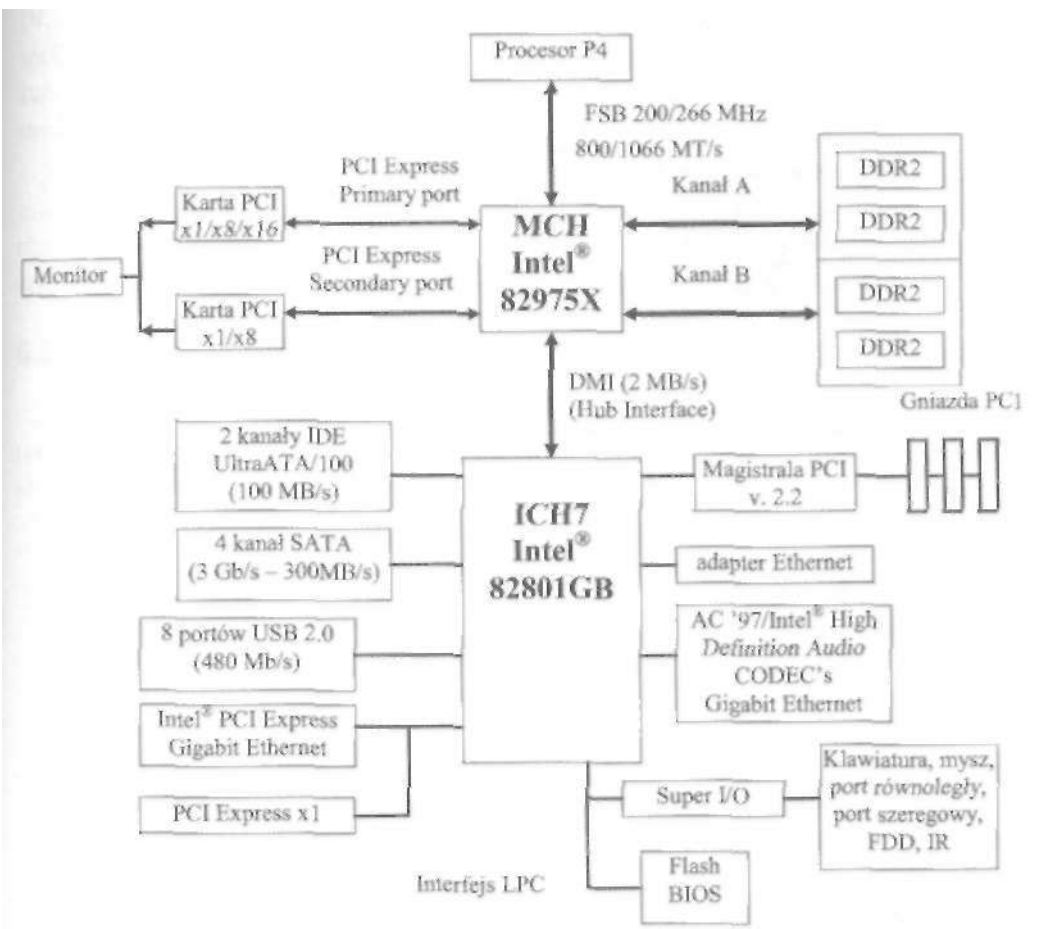
W odbiorniku realizowane są odwrotne procesy.

Udział poszczególnych warstw w tworzeniu pakietu transmitowanego przez łącze pokazuje rysunek 6.21. Skróty ECRC i LCRC oznaczają bajty zabezpieczające dodane odpowiednio przez warstwę transakcji i warstwę łącza danych.



Rysunek 6.21. Struktura pakietu magistrali PCI Express

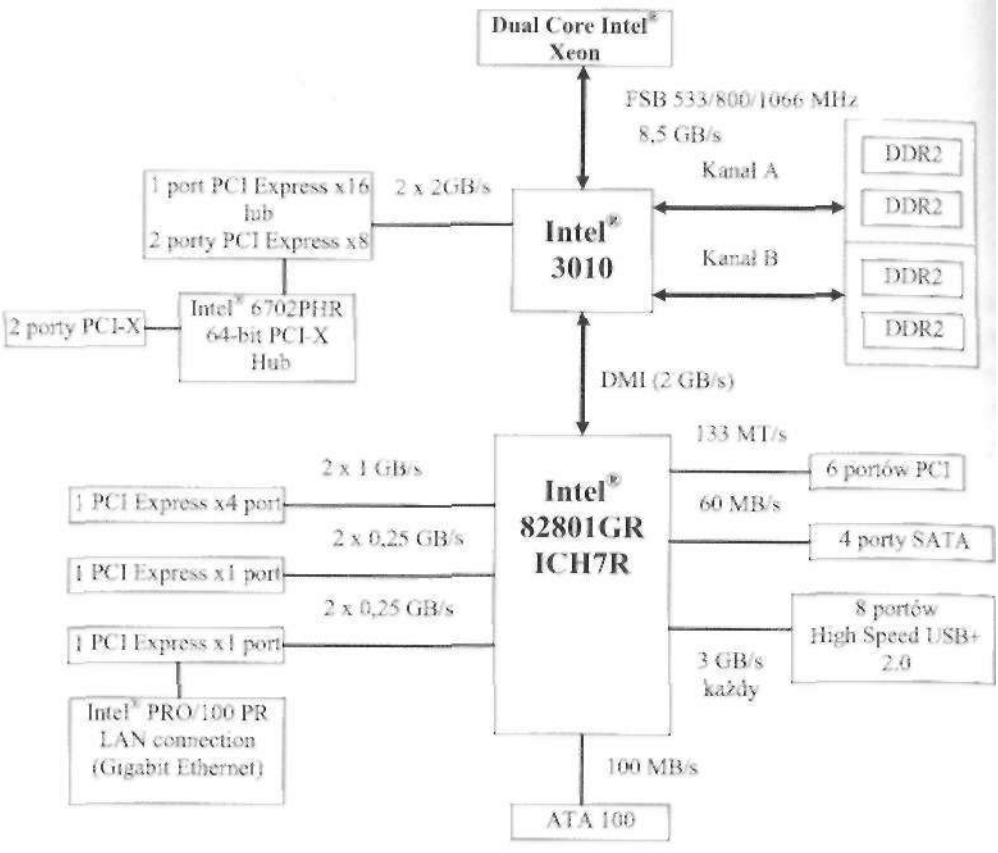
Prosty system zawierający magistralę PCI Express przedstawia rysunek 6.22.



Rysunek 6.22. Przykład prostego systemu z magistralą PCI Express

W systemie tym zwraca uwagę możliwość zainstalowania dwóch kart graficznych, czego nie zapewnia magistrala AGP. Rozwiązanie pozwalające na równoległe przetwarzanie tego samego obrazu nosi w nomenklaturze nVidii nazwę SLI - Scalable Link Interface (podobne rozwiązanie ATI nazywa się CrossFire) (zobacz też rysunek 6.27). Więcej o kartach graficznych piszemy w drugiej części podręcznika. Karty graficzne są obecnie głównym rodzajem urządzeń instalowanych w gniazdach PCI Express, jako że właśnie głównie one wymagają transferów, które może zapewnić tylko magistrala PCI Express. Innymi przykładami zastosowań magistrali PCI Express są szybkie interfejsy sieciowe.

System z magistralą PCI Express wykorzystuje się także w stacjach serwerowych, co pokazano na rysunku 6.23.



Rysunek 6.23. Przykład bardziej rozbudowanego systemu z magistralą PCI Express - chipset 3010

Z ciekawszych rozwiązań, które oferuje, warto wspomnieć o obsłudze dla pamięci korekcji błędów ECC (ang. *Error Correction Code)* podnoszącej bezpieczeństwo przechowywanych danych w pamięci. Innymi rozwiązaniami, występującymi zresztą

zresztą także i w innych chipsetach, są obsługa macierzy dyskowych RAID (tryb 0, 1, 5 i 10) nosząca nazwę **Intel Matrix Storage Technology** oraz rozwiązanie zwane **Intel Active Management Technology** (Intel® AMT). To ostatnie ułatwia proces zdalnego diagnozowania i usuwania usterek sprzętowych, gromadzenia informacji o sprzęcie i oprogramowaniu oraz zdalne zarządzanie oprogramowaniem. Wymienione funkcje szczególnie przydają się przy zarządzaniu serwerami.

5. Koncepcja działania urządzeń standardu Plug and Play

Podczas instalacji nowego sprzętu w komputerze problemy sprawiała konieczność jego konfigurowania. Polegała na wyborze (ustawieniu) za pomocą zworek lub mikroprzełączników parametrów, takich jak numer przerwania sprzętowego, kanału DMA czy też adresów, których będzie używał. Dodatkowo, nowo zainstalowany sprzęt wymaga sterowników programowych (driverów) do jego obsługi. Koncepcja kart standardu Pług and Play pozwala zautomatyzować ten proces, zwalniając tym samym użytkownika z koniecznos'ci jego wykonania.

Realizacja standardu Pług and Play wymaga spełnienia określonych warunków zarówno przez karty, jak i przez płyty główne (złącza magistrali rozszerzającej, BIOS, chipsety) oraz system operacyjny. Postaramy się krótko przedstawić te wymagania oraz odnies'ć je do podstawowych standardów gniazd rozszerzających i kart.

6.5.1. Zasada działania i wymagania standardu Pług and Play

Podstawowe założenia dotyczące działania urządzeń i systemu spełniającego standard Pług and Play (w dalszej części książki będziemy używać powszechnego skrótu nazwy tego standardu - PnP) są następujące:

- W przypadku zainstalowania nowego sprzętu w systemie po włączeniu zasilania system stwierdzi ten fakt, a następnie automatycznie skonfiguruje nowe urządzenie, przydzielając potrzebne mu zasoby w sposób niepowodujący konfliktu z innymi, już zainstalowanymi urządzeniami. Dotyczy to także urządzeń instalowanych w trakcie pracy systemu (ang. *hot insertion* lub w żargonie *installation on thefly).*
- W przypadku usunięcia urządzenia z systemu ponownie rozpozna on ten fakt i zwolni zasoby systemu przydzielone usuniętemu urządzeniu. Dotyczy to także usunięcia urządzenia w trakcie pracy systemu (ang. *hot removal).*

W celu realizacji wymienionych zadań zarówno system, jak i urządzenia muszą spełniać określone wymagania. Urządzenia (np. karty rozszerzające) muszą zapewniać:

- istnienie mechanizmu detekcji obecności karty,
- identyfikacje rodzaju urządzenia oraz jego producenta,
- źródło informacji o zasobach wymaganych przez urządzenie,
- możliwość konfigurowania programowego (wybór ustawień przez zapis do określonych rejestrów konfiguracyjnych).

Wymagania w stosunku do systemu są następujące:

- Powinna istnieć nieulotna pamięć konfiguracji i przydziału zasobów dla urządzeń.
- Musi istnieć program obsługujący wykrywanie obecności i autokonfiguracje urządzeń.

W ostatnim przypadku wymienione oprogramowanie może stanowić w całości element BIOS-u (PnP BIOS) lub być podzielone pomiędzy BIOS a system operacyjny. Ustawienia konfiguracji na poziomie BIOS-u są zapamiętywane w nieulotnej pamięci oznaczanej najczęściej skrótem ESCD (ang. *Extended System Configuration Data)* (patrz też dodatek o programie BIOS Setup). Ustawienia dokonywane przez system operacyjny przechowywane są na dysku twardym.

Ponadto, w celu umożliwienia stosowania tego samego oprogramowania w urządzeniach różnych producentów konieczny jest standard określający między innymi:

- mechanizm detekcji urządzeń,
- adresy rejestrów przechowujących listę wymaganych zasobów i format tej listy,
- adresy rejestrów konfiguracyjnych urządzenia,
- adresy pamięci w systemie, gdzie zapisana jest konfiguracja urządzeń.

Te wymagania spełnia standard PnP.

Start systemu PnP przebiega następująco. Po włączeniu zasilania inicjowane są i działają urządzenia niezbędne do rozpoczęcia pracy systemu (mogą to być zarówno urządzenia PnP, jak i zwykłe). Przykłady tych urządzeń to klawiatura, karta graficzna wraz z monitorem oraz urządzenie umożliwiające załadowanie systemu operacyjnego (ang. *IPL device, Initial Program Load device),* np. stacja dysków elastycznych, dysk twardy, CD-ROM, Boot-PROM. Za inicjację tych urządzeń odpowiedzialny jest BIOS.

Następnie system powinien przeszukać wszystkie magistrale w celu stwierdzenia obecności określonego sprzętu i wykrycia ewentualnych zmian. Proces ten wymaga istnienia programu przeszukującego (ang. *bus enumerator)* dla każdego rodzaju magistrali. Po wykryciu nowego urządzenia system powinien odczytać jego rodzaj, wytwórcę oraz zasoby wymagane do poprawnej pracy urządzenia (adresy, przerwania, kanały DMA, sterowniki). Informacje te są przechowywane, zgodnie ze standardem,

w określonym miejscu. Po odczytaniu tych informacji program konfigurujący powinien przydzielić urządzeniu potrzebne mu zasoby w sposób niepowodujący konfliktu z innymi urządzeniami. Jednocześnie przydział tych zasobów jest zapisywany do nieulotnej pamięci konfiguracji i przydziału zasobów w systemie. Informacja z tej pamięci jest wykorzystywana zarówno przy kolejnym starcie systemu, jak i przy niepowodującym konfliktu przydzielaniu zasobów urządzeniom.

W trakcie pierwszego startu systemu żadne urządzenie PnP nie jest jeszcze skonfigurowane, dlatego opisana procedura dotyczy wszystkich urządzeń. W trakcie kolejnych restartów systemu wynik przeszukiwania może dać następujące rezultaty:

- Brak zmian w stosunku do poprzedniego startu. W takim wypadku ustawienia zapisane w pamięci konfiguracji w systemie przepisywane są do zainstalowanych urządzeń.
- Wykryte zostało nowe urządzenie. Ten przypadek powinien spowodować powtórzenie opisanej procedury. Innymi słowy, program powinien przydzielić zasoby urządzeniu w sposób niepowodujący konfliktu z innymi urządzeniami oraz zmodyfikować pamięć przydziału zasobów w systemie.
- Urządzenie zostało zdezinstalowane (usunięte z systemu). Powinno to spowodować zwolnienie zasobów przydzielonych usuniętemu urządzeniu przez odpowiednie zmodyfikowanie pamięci przydziału zasobów.

Podobne do opisanych działania powinny być podjęte w przypadku urządzeń instalowanych i dezinstalowanych w trakcie pracy systemu (np. kart w złączach PCMCIA czy urządzeń w stacjach dokujących).

6.5.2. Standard PnP a rodzaj magistrali rozszerzającej

W podpunkcie tym przyjrzymy się dwóm standardom gniazd rozszerzających - ISA i PCI.

6.5.2.1. ISA

W momencie tworzenia standardu ISA nie zakładano możliwości automatyzacji procesu konfiguracji instalowanych urządzeń. Dlatego też karty standardu ISA powstałe przed opracowaniem standardu PnP nie zapewniają praktycznie żadnych mechanizmów wymaganych przez standard PnP. W szczególności nie istnieje mechanizm detekcji obecności karty. Także dla nowych kart, ze względu na zestaw sygnałów w złączu rozszerzającym, ich konfigurowanie jest dość skomplikowane.

Każda karta ma pod określonymi adresami trzy porty konfiguracyjne, za pomocą których oprogramowanie konfiguracyjne komunikuje się z kartami. Proces konfiguracji przebiega w następujących etapach:

1. Do portu konfiguracyjnego kart wpisywana jest określona sekwencja znaków powodująca ich przejście w stan oczekiwania.
2. Następuje seria odczytów, która powoduje przejście jednej wybranej karty w tak zwany stan izolacji.
3. Oprogramowanie konfiguracyjne nadaje karcie niepowtarzalny numer CSN (ang, *Card Select Number)*. Po jego nadaniu karta przechodzi w stan konfiguracji.
4. W stanie konfiguracji oprogramowanie systemowe odczytuje listę wymaganych zasobów karty. Następnie karta jest wprowadzana w stan uśpienia.

Podane czynności są powtarzane dla wszystkich kart ISA obecnych w systemie. Następnie wykonywane są dla wszystkich konfigurowanych kart dwie kolejne czynności:

5. Każda z kart jest wybierana za pomocą CSN i wprowadzana w stan konfiguracji. W tym stanie do rejestrów konfiguracyjnych karty wpisywane są wartości zapewniające jej bezkonfliktową pracę.
6. Karta jest wprowadzana w stan aktywny.

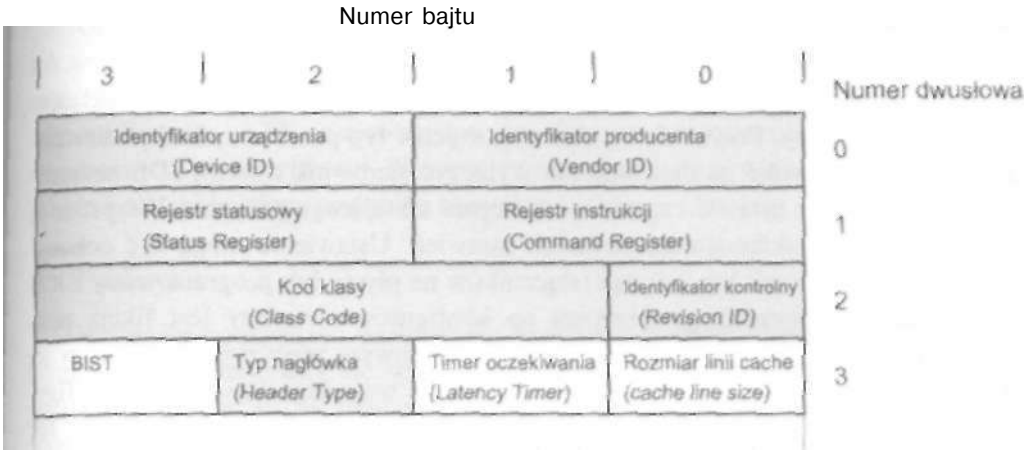
6.5.2.2. PCI

Założenia standardu PCI zostały opracowane z uwzględnieniem wspierania standardu PnP. Istnieją więc wszystkie niezbędne mechanizmy potrzebne do automatycznego skonfigurowania kart w systemie. Mechanizm detekcji obecności karty zapewniają linie magistrali PCI oznaczone PRSNT1# i PRSNT2#. W przypadku braku karty w złączu wartość obu sygnałów wynosi 1. Jeżeli choć jeden z tych sygnałów ma wartość 0, świadczy to o obecności karty w złączu.

Każde urządzenie PnP dysponuje przestrzenią adresową do konfiguracji wynoszącą 256 bajtów. Pierwsze 64 bajty ma określone, zdefiniowane znaczenie. Reszta pozostaje do dyspozycji projektantów kart. Rysunek 6.24 pokazuje przykładowo pierwszych 16 bajtów rejestrów konfiguracyjnych. Na rysunku zacieniowano pola, które muszą wystąpić dla każdej karty.

Przebieg autokonfiguracji jest następujący:

1. W pierwszym etapie autokonfiguracji należy stwierdzić, jakie urządzenia są zainstalowane w systemie i jakich wymagają zasobów. Pewien problem stanowi to, że w komputerach IBM PC oprócz magistrali PCI współistnieje magistrala ISA. Karty niebędące kartami PnP nie dają możliwości stwierdzenia ich wymagań co do zasobów. Możliwe są następujące rozwiązania:
   A. Program inicjujący wykonuje serię zapisów i odczytów w zakresie adresów przydzielonych urządzeniom ISA w celu wykrycia ich obecności. Proces ten da wiarygodne rezultaty jedynie dla kart ze standardowo przyporządkowanymi numerami przerwań i kanałów DMA.



Rysunek 6.24. Zawartość pierwszych 16 bajtów rejestrów konfiguracyjnych urządzeń FC1

B. Uruchomiony zostaje program, który pozwoli użytkownikowi wprowadzić zasoby przydzielone zainstalowanym urządzeniem ISA. Istnieje też możliwość wyboru typu urządzenia z podanej przez program listy (dotyczy to popularnych urządzeń).

Po określeniu zasobów przydzielonych urządzeniom ISA przeszukiwana jest magistrala PCI. Następuje to przez odczytanie identyfikatora producenta zainstalowanego sprzętu. Odczytywane są jego rejestry konfiguracyjne w celu określenia wymaganych zasobów systemu.

2. Program konfiguracyjny przydziela w sposób niepowodujący konfliktów żądane zasoby, a następnie programuje zgodnie z dokonanym wyborem rejestry konfiguracyjne urządzeń PCI.
3. Po przydzieleniu urządzeniom potrzebnych im zasobów oprogramowanie konfiguracyjne wpisuje określoną sekwencję do rejestru komend urządzenia w celu zezwolenia na jego działanie.

Następnie (w razie potrzeby) system operacyjny ładuje sterowniki zainstalowanych urządzeń. Do sterowników przekazywane są wartości zasobów przydzielonych urządzeniom (np. numery przerwań) potrzebne do prawidłowego działania sterowników. Na tym kończy się proces autokonfiguracji.

Następczynie magistrali PCI - PCI-X i PCI Express, są z nią w pełni zgodne co do wymagań PnP. Mechanizmy autokonfiguracji magistrali PCI Express zostały oczywiście znacznie rozbudowane. Ich opis wykracza jednak poza zakres tej książki. Można go znaleźć, sięgając do pozycji [5].

6.6. Konfigurowanie płyt głównych

Praktycznie każda płyta ma możliwość zmiany zainstalowanej na niej układów lub sposobu ich pracy. Przykładowo możemy zmienić typ procesora, liczbę zainstalowanej pamięci, zezwolić na działanie lub wyłączyć sterownik dysków IDE zintegrowany z płytą czy też ustawić częstotliwość zegara taktującego procesor. Wszystkie te zmiany wiążą się z dokonaniem pewnych ustawień. Ustawienia mogą być dokonywane za pomocą zworek lub mikroprzełączników na płycie lub programowo w BIOS Setup. Podanie uniwersalnego przepisu na konfigurowanie płyty jest fikcją poza jednym zaleceniem. Sposób konfiguracji płyty jest zawsze precyzyjnie opisany w jej dokumentacji (ang. *User's Manuał)* i przestrzeganie instrukcji dotyczących konfigurowania płyty daje gwarancję sukcesu (przy założeniu, że płyta jest sprawna). Jak więc widać, do skonfigurowania płyty nie potrzeba tajemnej wiedzy, a jedynie staranności oraz podstawowej znajomości języka angielskiego.

6.7. Formaty płyt głównych

Poniżej podajemy wymiary najczęściej spotykanych płyt ATX. Specyfikacja ATX określa nie tylko wymiary mechaniczne płyt (także rozmieszczenie otworów montażowych), ale także niektóre parametry elektryczne, na przykład rodzaj złącza zasilania czy też wymagania dotyczące zasilacza. Ponadto podaje wiele zaleceń co do rozmieszczenia elementów i gniazd na płycie głównej (przykładowo: łatwo dostępne gniazda modułów pamięci, gniazda interfejsów IDE i FDD blisko prowadnic, procesor blisko zasilacza itp.).

Tabela 6.6 przedstawia wymiary najpopularniejszych formatów płyt ATX.

Tabela 6.6. Wymiary wybranych formatów płyt ATX

Nazwa formatu	Rozmiar w calach	Rozmiar w mm
Flex ATX	9x7,5	229x191
Micro ATX	9,6x9,6	244x244
ATX	12x9,6	305x244

Proszę zwrócić uwagę, że formaty ATX i Micro ATX są tak dobrane, iż wiele rozwiązań chassis umożliwia montaż zarówno jednego, jak i drugiego formatu.

Oprócz formatu ATX możemy spotkać jeszcze format płyt AT. Format ten jednak wyszedł już z użycia. W roku 2004 firma Intel zaproponowała nowy format płyt oznaczany jako BTX (ang. *Balanced Technology eXtended)* usuwający wiele niedociągnięć formatu ATX, zapewniający miedzy innymi znacznie lepsze chłodzenie elementów na płycie głównej. Niestety Intel wycofał się z wspierania tego formatu, który nie zdążył jeszcze przyjąć się na rynku.

Praktyka

W książce podkreślano wielokrotnie modułową budowę komputera PC. W pierwszym rozdziale staraliśmy się spojrzeć na komputer jako na całość i wyróżnić jego podstawowe elementy. W rozdziale drugim prezentujemy wygląd modułów pamięci oraz odpowiadających im gniazd znajdujących się na płycie głównej. W rozdziale czwartym pokazujemy wybrane gniazda procesorów i wygląd ich samych. Oczywiście gniazda procesorów są również umieszczone na płycie głównej.

Poniżej staramy się przedstawić pozostałe fragmenty płyty głównej, na które warto zwrócić uwagę. Po kolei pokazujemy: gniazda magistrali rozszerzającej, wybrane elementy płyty głównej, inne dodatkowe złącza płyty głównej, a także sposób podłączenia do płyty głównej podstawowych urządzeń peryferyjnych, takich jak dysk twardy, napęd DVD-ROM oraz napęd dyskietek elastycznych. Same urządzenia i ich interfejsy opisujemy dokładnie w drugiej części podręcznika.

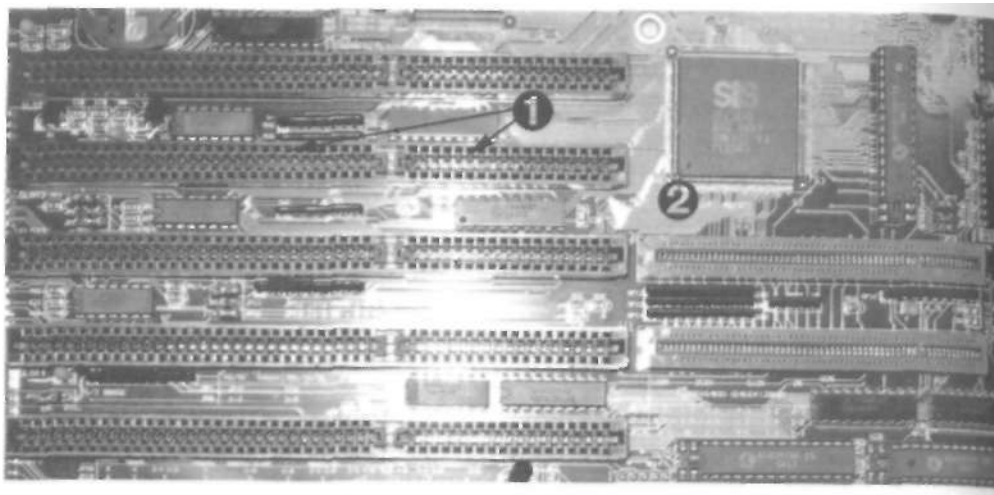
Oddzielnie prezentujemy wygląd płyty głównej komputera typu laptop czy notebook.

W rozdziale siódmym pokazane są złącza zasilania i wygląd zasilacza.

Standardy złącza magistrali rozszerzającej

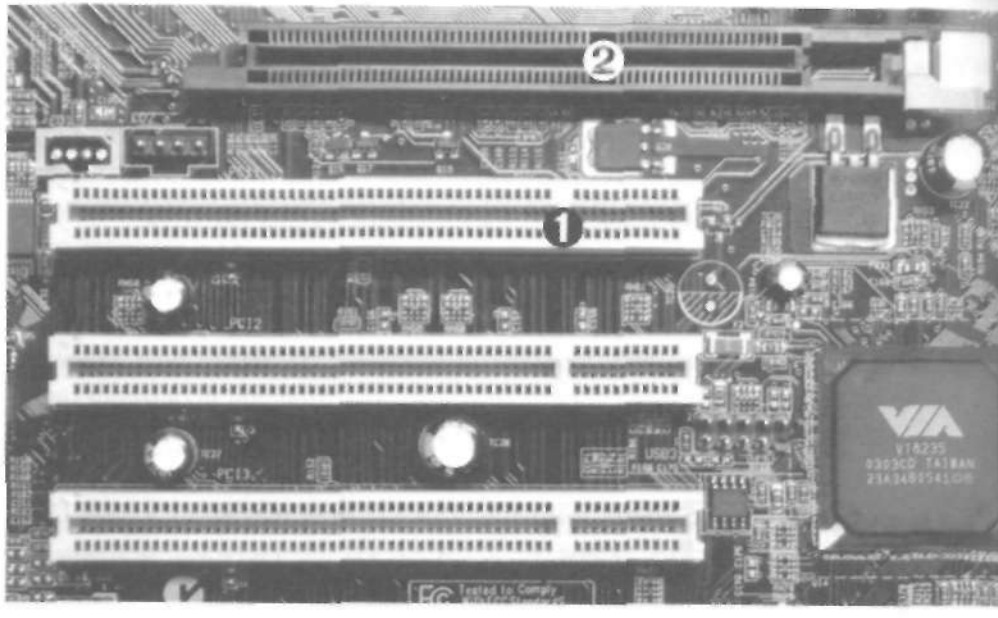
Poniżej przedstawiamy złącza magistrali rozszerzającej różnych standardów. Niektóre z nich mają już znaczenie głównie historyczne. Do złączy tych zaliczamy także złącze magistrali AGP, która choć dedykowana określonemu urządzeniu (adapterowi graficznemu), to jednak pozwala instalować go w postaci karty, więc w tym sensie jest niewątpliwie złączem magistrali rozszerzającej.

Na rysunku 6.25 widać złącza magistrali ISA (1) oraz magistrali VESA Local Bus (2). Złącze magistrali ISA składa się z dwóch części. Pierwsza, dłuższa, jest obecna od początku standardu i jest magistralą 8-bitową. Druga, krótsza, została dodana przy rozszerzaniu standardu ISA do 16 bitów. Obecność magistrali V-LB wiąże się z pojawieniem dodatkowego złącza, umieszczonego w jednej linii ze złączem ISA. Montaż kart w tej magistrali był zadaniem niezbyt wdzięcznym, właśnie z powodu rodzaju złączy.



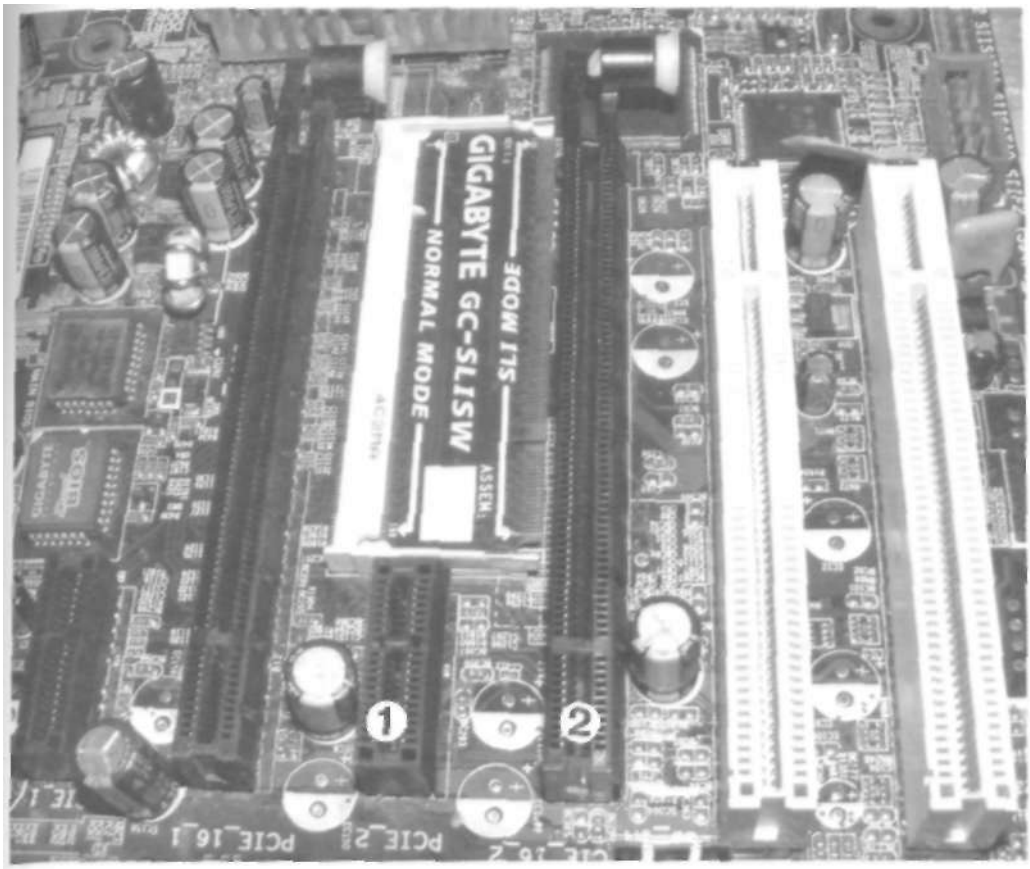
Rysunek 6.25. Złącza magistrali ISA i VESA Local Bus

Kolejny rysunek 6.26 pokazuje złącza magistrali PCI (1) oraz jedną z zaawansowanych wersji magistrali A GP, AGP Pro (2).



Rysunek 6.26. Złącza magistrali PCI i AGP

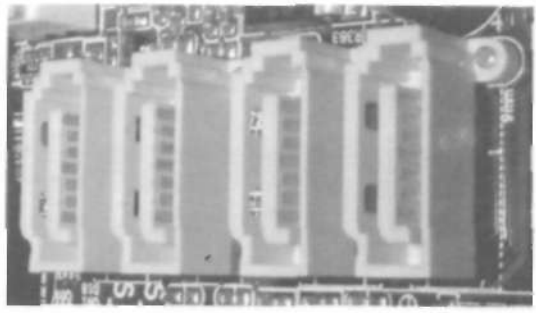
Na rysunku 6.27 widzimy (obok złączy magistrali PCI) złącza magistrali PCI Express, w wersji xl - (1) i x16 - (2). Na zdjęciu widać też złącze mostka SU z zamontowanym mostkiem (patrz strona 278).



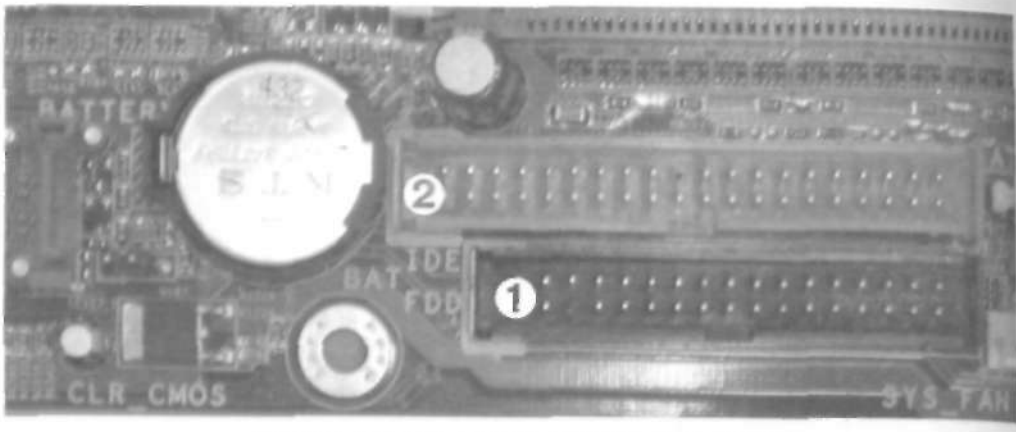
Rysunek 6.27. Złącza magistrali PCI Express

Inne złącza płyty głównej

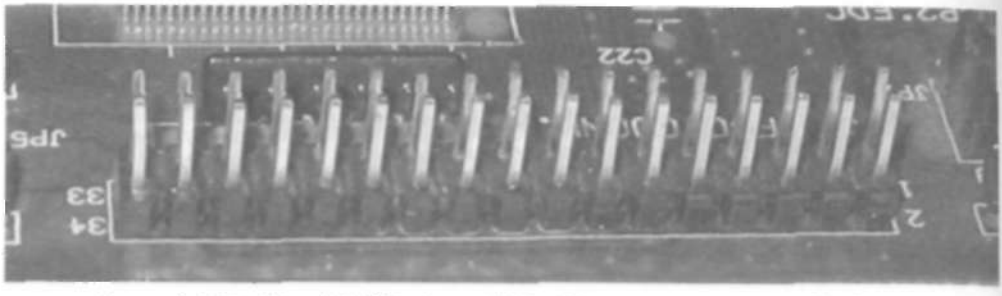
Na rysunku 6.28 pokazano cztery złącza standardu interfejsu dysków twardych Serial ATA (SATA). Przykład podłączenia dysku do takiego złącza przedstawia rysunek 6.43. Rysunek 6.29 pokazuje złącze interfejsu napędu dyskietek elastycznych (1) (FDD - ang. *Floppy Disk Drive)* (34-pinowe złącze typu HDR mające wyprowadzenia w postaci igiełek - rysunek 6.30) i złącze IDE (2) (40-pinowe). Złącze IDE umożliwia podłączenie dysku twardego bądź napędu CD/DvD.



Rysunek 6.28. Złącza interfejsu SATA



Rysunek 6.29. Złącze IDE, FDD i gniazdo baterii

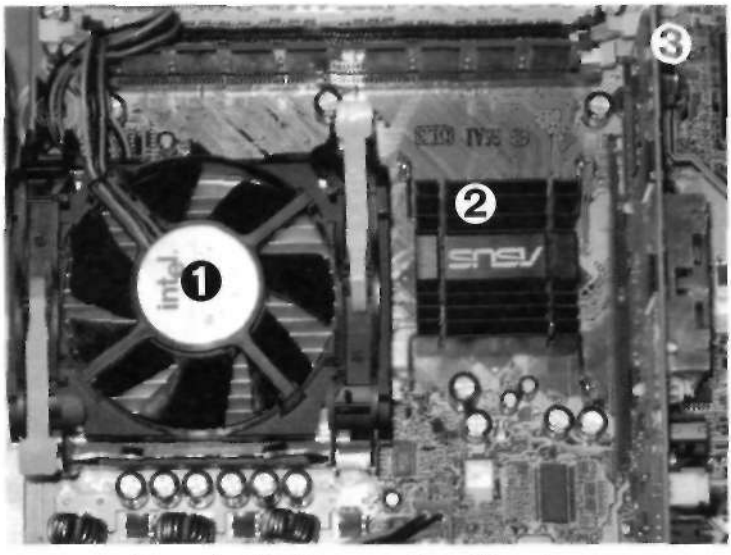


Rysunek 6.30. Piny (igiełki) złącza FDD widoczne w starszym typie złącza

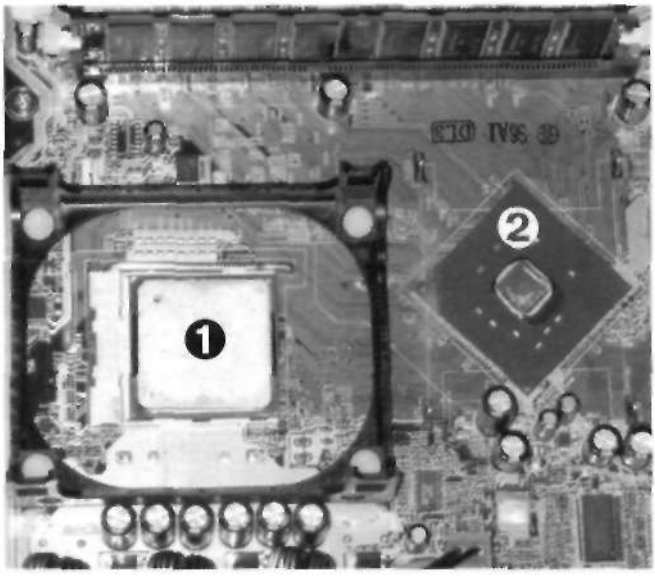
Przykładowe elementy płyty głównej

Na rysunku 6.31 widać wentylator i radiator, pod którym znajduje się procesor (1), radiator jednego z chipsetów (2), moduł pamięci DDR DIMM w złączu oraz kartę (graficzną) (3) w gnieździe magistrali AGP.

Na zdjęciu 6.32 widać procesor (1) oraz chipset (2), z których zostały zdjęte radiatory.



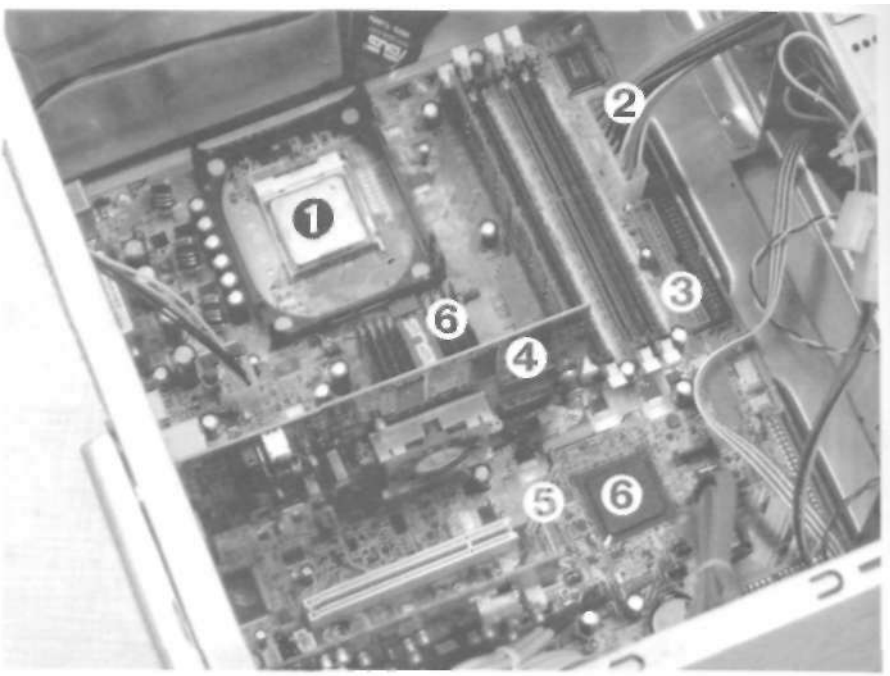
Rysunek 6.31. Elementy płyty głównej cz. I



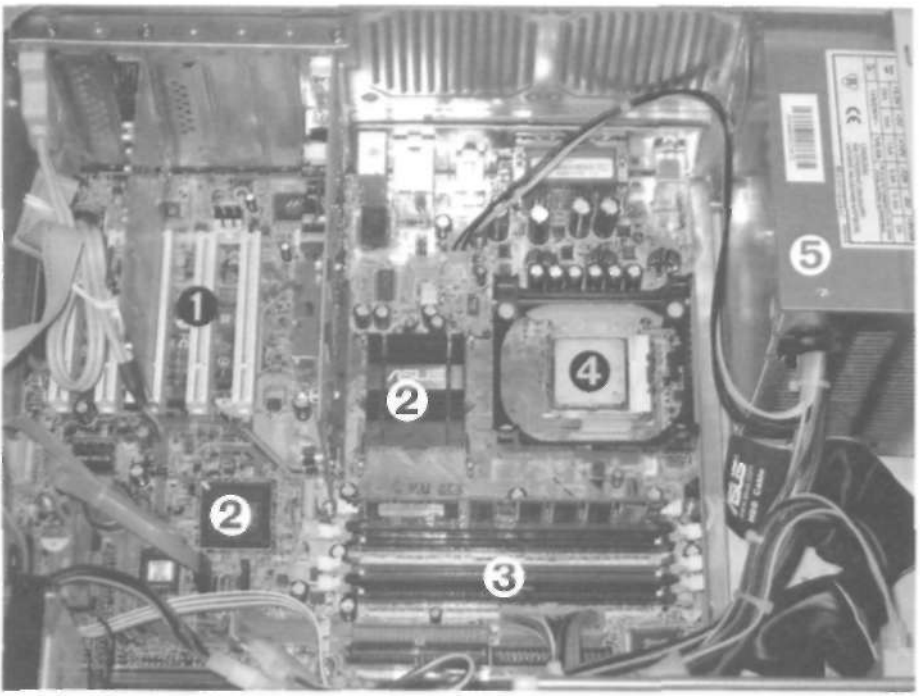
Rysunek 6.32. Elementy płyty głównej cz. II

Na kolejnym rysunku 6.33 pokazujemy procesor (1), złącze zasilania płyty głównej (2), gniazda interfejsu EIDE (3), kartę graficzną (4), na której procesorze zamontowany jest wentylator. Widać także gniazdo PCI (5) oraz chipsety (6).

Na rysunku 6.34 możemy zobaczyć zespół gniazd magistrali rozszerzającej PCI (1), chipsety (2), gniazda modułów pamięci (3) z zamontowanym jednym modułem DIMM, procesor (4) oraz blok zasilacza (5).



Rysunek 6.33. Elementy płyty głównej cz. III



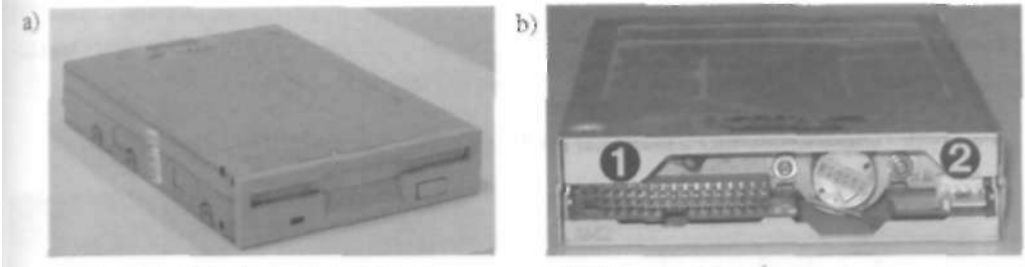
Rysunek 6.34. Elementy płyty głównej cz. IV

Podłączenie wybranych urządzeń peryferyjnych do płyty głównej

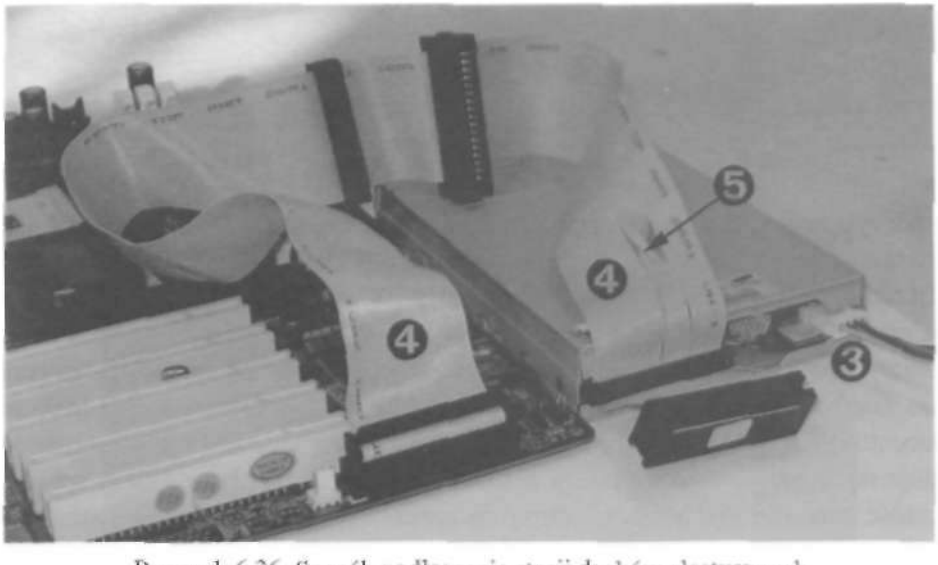
W punkcie tym chcemy pokazać wybrane urządzenia peryferyjne montowane wewnątrz komputera i sposób ich elektrycznego połączenia z komputerem. Jednak ze względu na dużą liczbę elementów i ograniczone miejsce trudno jest pokazać te elementy w miejscu ich zamontowania, połączenia elektryczne postanowiliśmy więc pokazać na stole laboratoryjnym, czyli poza obudową komputera. Miejsce fizycznego umieszczenia wybranych elementów przedstawiamy na rysunku 6.42.

O pokazywanych tu urządzeniach znacznie szerzej piszemy w drugiej części podręcznika.

1. Stacja dysków elastycznych



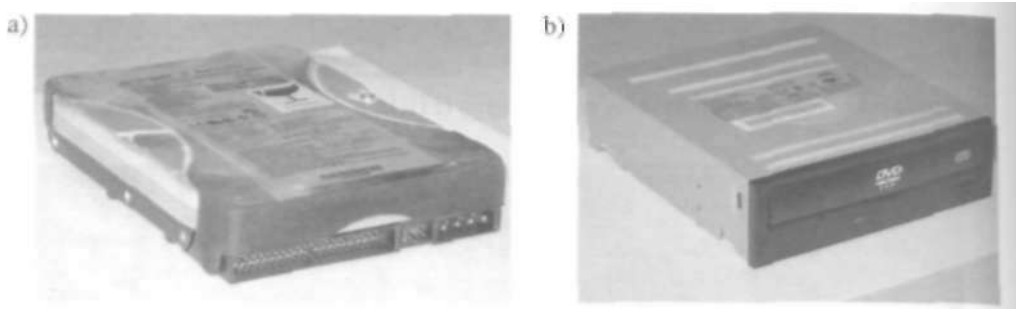
Rysunek 6.35. Widok stacji dysków elastycznych (FDD): a – od czoła, b – od strony złączy



Rysunek 6.36. Sposób podłączenia stacji dysków elastycznych

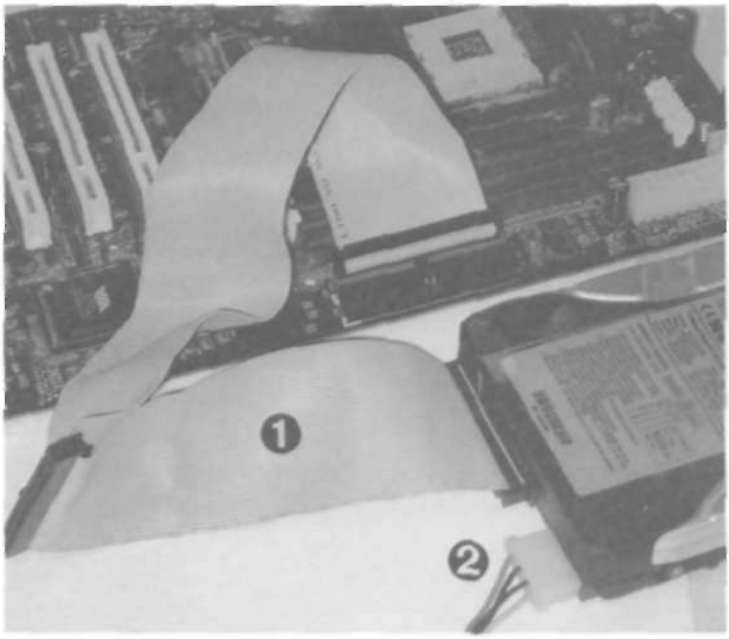
Widok stacji dysków elastycznych pokazany jest na rysunku 6.35 a i b. (1) to złącze interfejsu tych dysków (SA 450), a (2) - złącze zasilania. Napęd dysków należy podłączyć do zasilania (3) - rysunek 6.36, i pasmem interfejsu SA 450 (4) połączyć złącze na napędzie i na płycie głównej (wygląd tego złącza - rysunek 6.29). Pasmo SA 450 jest 34-żyłowe i 34-pinowe są obydwa złącza (dwa rzędy po 17 igiełek). Pasmo ma oznaczoną jedną stronę - jest to oznaczenie żyły numer 1. Złącze na płycie należy sprawdzić: większość producentów oznacza wyprowadzenie 1, 2, 33 lub 34 (rysunek 6.30). W ostateczności prawidłowe położenie można ustalić metodą prób i błędów (dwie możliwości). Przy napędzie żyła ta jest łączona zwykle od strony złącza zasilającego, choć bywają odstępstwa. Jest to jednak łatwe do wykrycia (a pomyłka nic grozi uszkodzeniem). Przy błędnie podłączonym paśmie kontrolka na napędzie dysku pali się bez przerwy. Proszę zwrócić uwagę na obrócenie części żył pasma zwane przeplotem (5). Dysk podłączony po przeplocie (czyli na końcu) traktowany jest jako dysk A, chyba że zmienimy odpowiednią opcję w BIOS Setup.

2. Dysk **twardy IDE i DVD-ROM (ATAPI)**



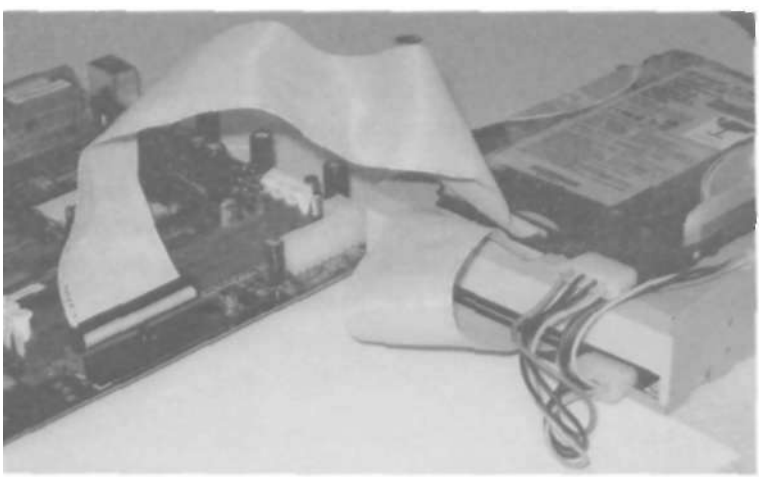
Rysunek 6.37. Widok dysku twardego (a) i napędu D VD od czoła (b)

Na rysunku 6.37 a i b pokazano dysk twardy i napęd DVD. Oba urządzenia obsługiwane są przez interfejs IDE (dla napędów CD/DVD nazywany także ATAPI), Podłączenie dysku twardego pokazuje rysunek 6.38. Pasmo (1) jest 40- lub 80-żyłowe (w zależności od wersji IDE), przy czym wygląd złączy nie ulega zmianie. Ponownie zwracamy uwagę na oznaczenie w paśmie żyły numer 1. Wprawdzie w złączu przewidziano odpowiedni klucz identyfikujący prawidłowe jego położenie, jednak wielu producentów nie korzysta z tej możliwości. Od strony dysku kontakt numer jeden znajduje się zawsze od strony złącza zasilania (2). Złącze na płycie należy sprawdzić: większość producentów oznacza wyprowadzenie 1, 2, 39 lub 40. W ostateczności prawidłowe położenie można ustalić metodą prób i błędów (dwie możliwości).

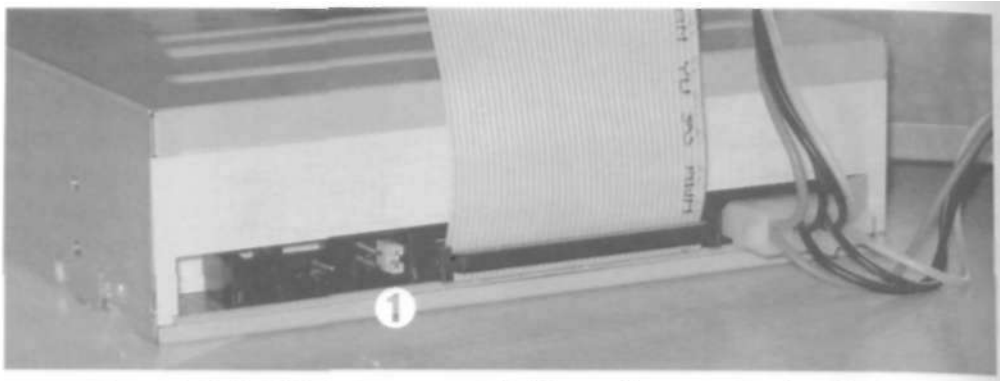


Rysunek 6.38. Sposób podłączenia dysku twardego

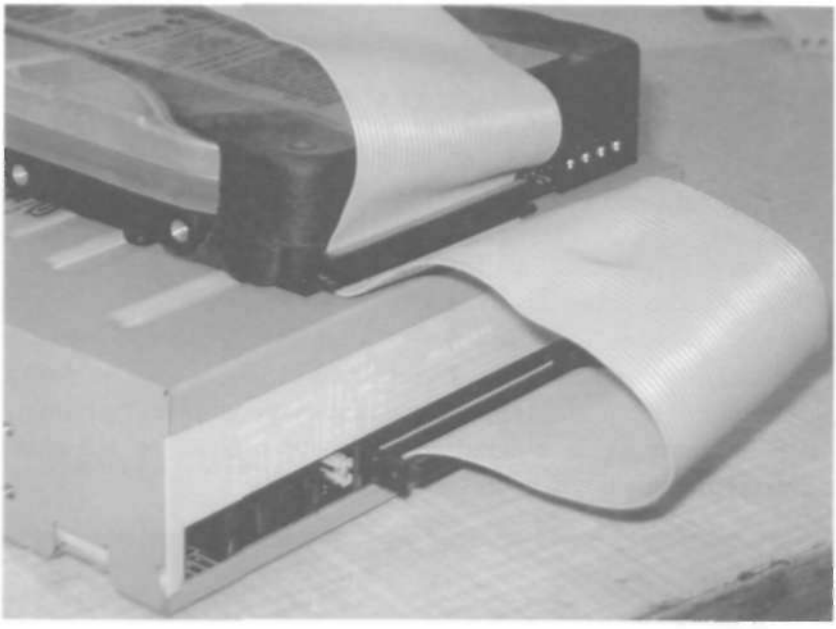
Na rysunku 6.39 pokazano podłączenie dwóch urządzeń: dysku twardego i napędu DVD, za pomocą jednej taśmy. Łączymy je łańcuchowo, dopinając każde do oddzielnego złącza pasma - rysunek 6.41. Urządzenia łączone jedną taśmą muszą mieć odpowiednio ustawione zworki. Zworkę taką widać na zdjęciu 6.40 (1), obok podłączonej taśmy i złącza zasilania. Oczywiście dysponując drugą taśmą i wolnym złączem IDE, możemy każde z ur?.ądzeń podłączyć osobno, co jest rozwiązaniem lepszym. Szerzej piszemy o tym w drugiej części książki.



Rysunek 6.39. Sposób podłączenia dysku twardego i napędu DVD za pomocą jednej taśmy

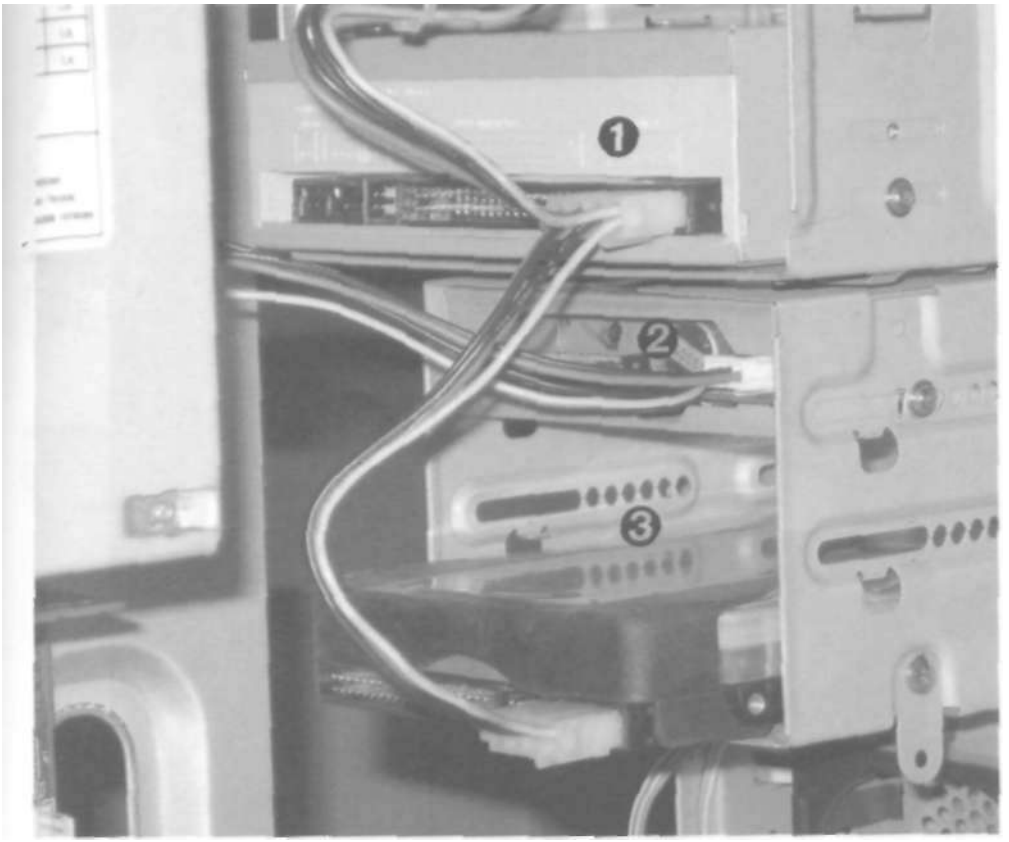


Rysunek 6.40. Widok złączy napędu DVD z podłączoną taśmą i zasilaniem



Rysunek 6.41. Połączenie łańcuchowe dysku twardego i napędu DVD

Na rysunku 6.42 pokazano, jak omówione urządzenia (1 - napęd DVD, 2 - stacja dysków elastycznych i 3 - dysk twardy) mogą być przykładowo zamontowane w odpowiednich prowadnicach komputera. Urządzenia mają dopięte złącza zasilania, natomiast dla czytelności nie są podłączone pasma sygnałowe.



Rysunek 6.42. Napędy dyskowe zamontowane w obudowie komputera

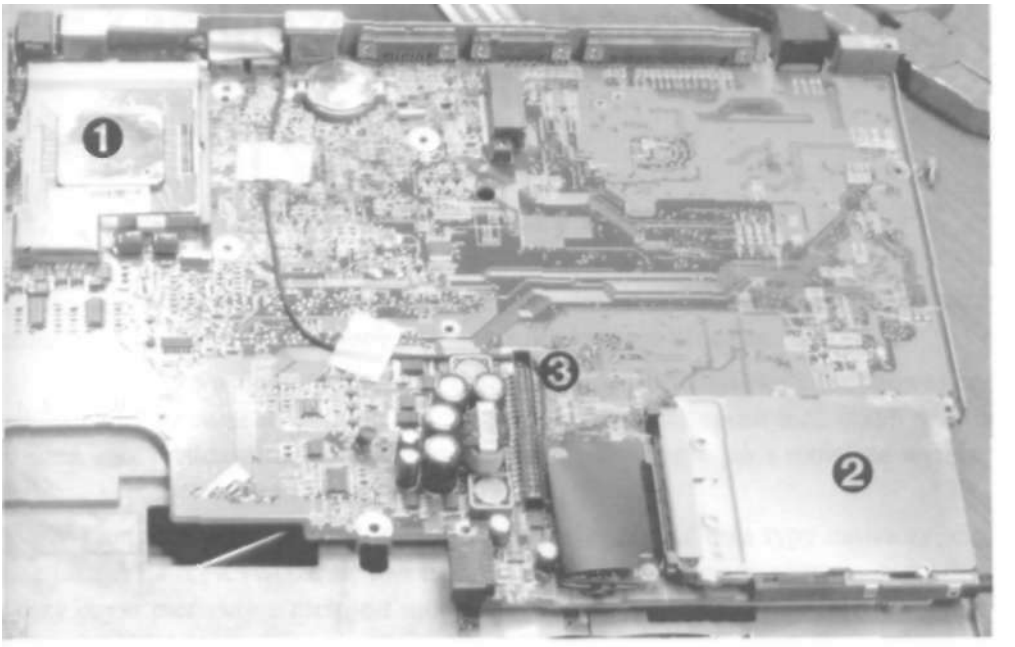
3. Dysk twardy SATA

Ostatnim z urządzeń, które pokazujemy, jest dysk twardy w standardzie SATA. **Na zdjęciu** widać złącze i przewody zasilania (1) oraz przewody sygnałowe (2) dopięte z jednej strony do napędu dysku, a z drugiej strony do złącza kontrolera dysków SATA na płycie głównej. Tym razem połączenia te pokazujemy w miejscu zamontowania dysku, we wnętrzu obudowy komputera.

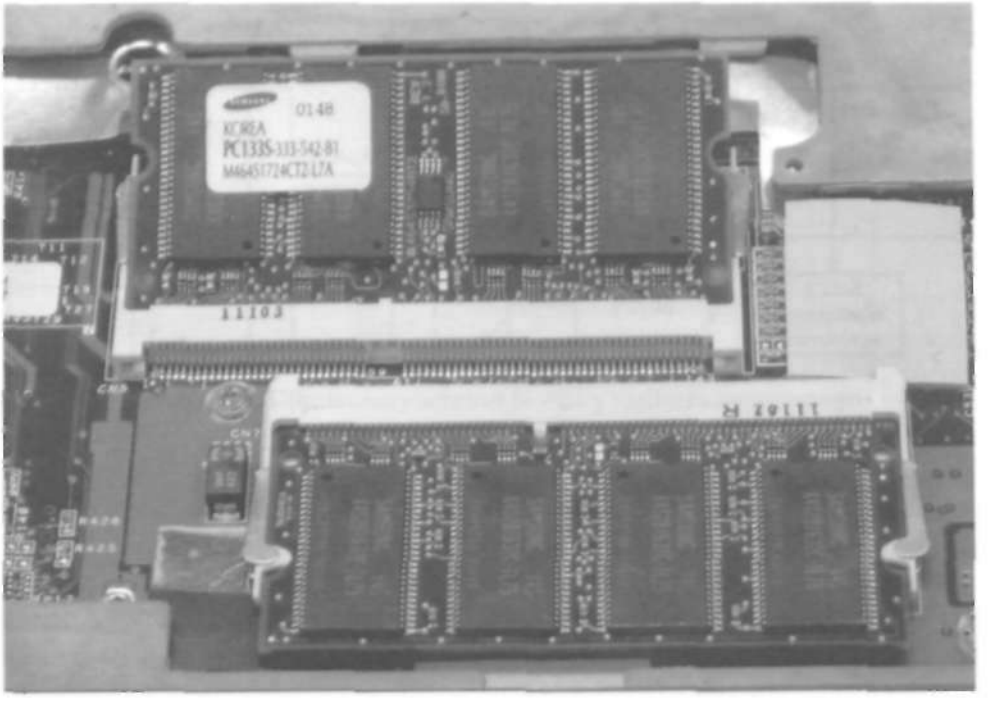
Rysunek 6.43. Sposób podłączenia dysku twardego SATA

Płyta główna komputera typu laptop

Kolejny rysunek 6.44 przedstawia płytę główną komputera typu laptop. Widoczny są między innymi: procesor (1), obudowa złącza kart PCMCIA (2) oraz złącze IDE dysku twardego (3). To ostatnie ma 42 piny zamiast 40, gdyż prowadzi także zasilanie. Natomiast rysunek 6.45 pokazuje moduły pamięci SODIMM zamontowane z drugiej strony płyty głównej.



Rysunek 6.44. Płyta główna komputera typu laptop



Rysunek 6.45. Moduły pamięci SODIMM w gniazdach płyty głównej komputera laptop

7. Zasilacze komputerów IBM/PC

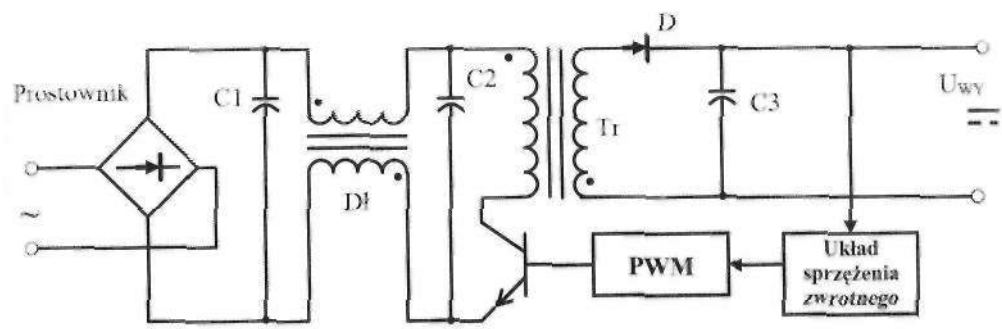
7.1. Zasada działania zasilaczy komputerów IBM/PC

Zadaniem zasilacza jest wytworzenie odpowiednich napięć i zapewnienie utrzymania ich wartości przy określonym poborze prądu przez urządzenia.

Zasilacze, ze względu na zasadę działania mogą być liniowe (analogowe) bądź impulsowe. Ponieważ te drugie cechuje zarówno większa sprawność (czyli większy stosunek mocy oddawanej do obciążenia do mocy traconej), jak i mniejsze wymiary, zasilacze komputerów są z reguły impulsowe.

W komputerach typu IBM/PC można obecnie spotkać dwa typy zasilaczy, określane jako AT i ATX (wiąże się to z nazwami typów płyt, o czym piszemy w rozdziale 6, przy czym pierwszy z nich jest modelem rzadko spotykanym, obecnie już (praktycznie) nieprodukowanym.

Uproszczony schemat blokowy zasilacza impulsowego przedstawiono na rysunku 7.1



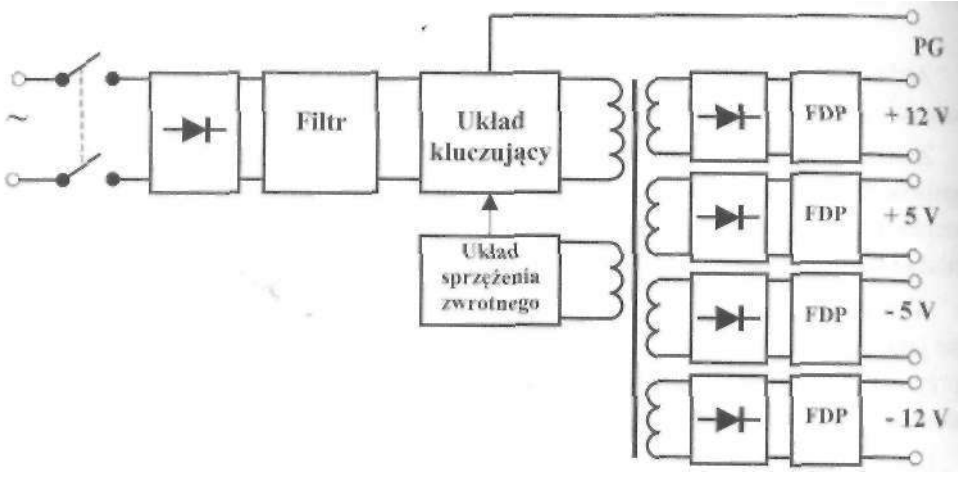
Rysunek 7.1. Uproszczony schemat blokowy zasilacza impulsowego

Jego zasada działania jest następująca.

Napięcie przemienne z sieci energetycznej jest prostowane w prostowniku (zwykle w układzie Graetza), a następnie filtrowane (C1, Dł, C2). Filtr ten jednocześnie zapobiega przedostawaniu się zakłóceń elektromagnetycznych do sieci. Następnie napięcie wyprostowane przetwarzane jest na przebieg zmienny o częstotliwości rzędu kiloherców i o zmiennym współczynniku wypełnienia. Współczynnik ten jest zmieniany przez układ modulacji szerokości impulsów PWM (ang. *Pulse Width Modulation)*. Wytworzony przebieg jest następnie filtrowany, aby otrzymać stałe napięcie

(składową stałą). Wartość tego napięcia zależy od współczynnika wypełnienia impulsów. Układ sprzężenia zwrotnego steruje układem PWM tak, aby przy zmieniającym się obciążeniu lub zmianach napięcia w sieci (w dopuszczalnych granicach) utrzymywać stałą wartość napięcia wyjściowego.

Schemat blokowy zasilacza AT pokazano na rysunku 7.2.

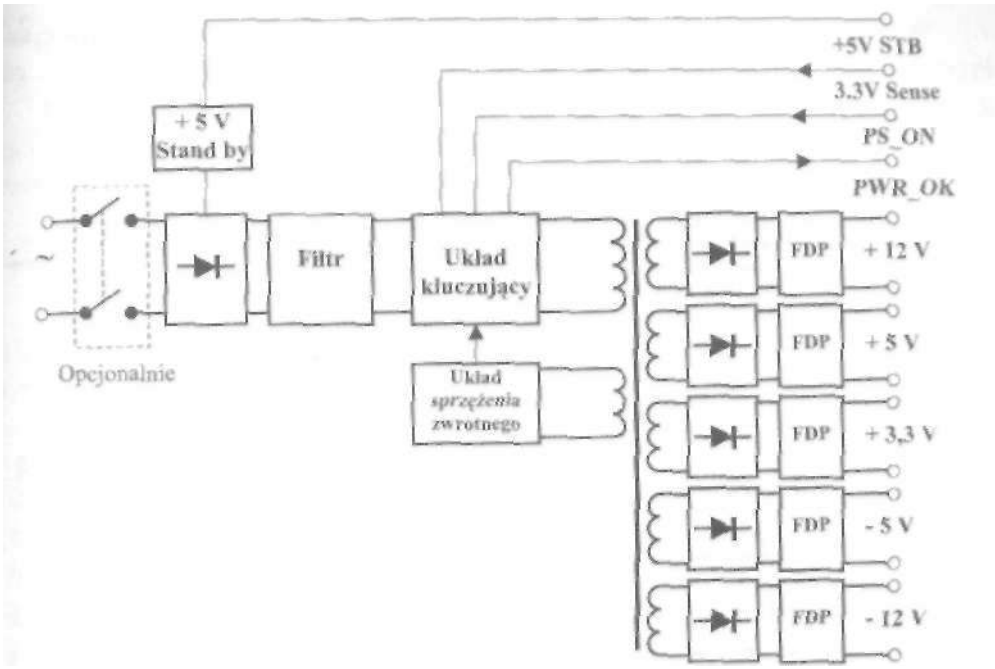


Rysunek 7.2. Schemat blokowy zasilacza AT

Zasilacz AT jest podłączany do płyty głównej za pomocą dwóch wtyków oznaczanych jako P8 i P9, jak widać na rysunku 7.5. Tam podano też znaczenie poszczególnych przewodów. Istotny jest sposób podłączenia tych wtyków do płyty głównej. Są zabezpieczone przed ich odwróceniem o 180°, jednak można zamienić je miejscami. Reguła prawidłowego podłączenia jest prosta. Czarne przewody we wtykach (prowadzące masy) powinny się spotkać w środkowej części złącza zasilającego płytę. Podłączenie odwrotne grozi poważnymi skutkami.

Rysunek 7.3 przedstawia schemat blokowy zasilacza ATX. Na rysunku różnice są niewielkie. Podstawowe różnice pomiędzy zasilaczem AT a ATX:

- nowe napięcie +3,3 V;
- zasilacz jest włączany i wyłączany za pomocą sygnału elektronicznego o poziomach TTL o nazwie PS_ON. Napięcie niskie na tym wyprowadzeniu oznacza zasilacz włączony. Stan wysoki to wyłączenie zasilacza. Istnieje więc możliwość programowego sterowania zasilaczem, na przykład przez system operacyjny;
- obecność napięcia +5V STB (oznaczanego też ST). Napięcie to jest obecne niezależnie od tego, czy zasilacz jest włączony. Pomijamy tu oczywiście przypadek, gdy zasilacz odłączony jest od sieci. Wyłącznik sieciowy pokazany na schemacie jest jednak opcjonalny i nie we wszystkich zasilaczach występuje;

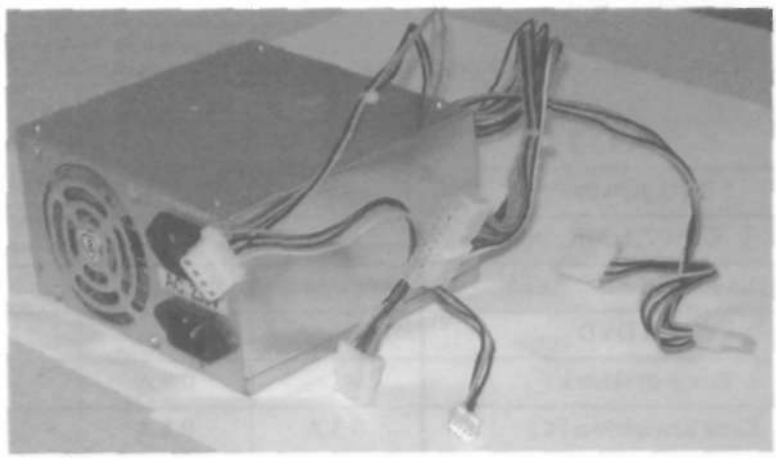


Rysunek 7.3. Schemat blokowy zasilacza ATX

» zmieniono sposób podłączania zasilacza do płyty głównej. Do tego celu siuzy pojedynczy wtyk, pokazany na rysunku 7.8, zabezpieczony przed niepoprawnym podłączeniem odpowiednim kształtem.

»-» Praktyka

Wygląd zasilacza przedstawiono na rysunku 7.4.



Rysunek 7.4. Zasilacz ATX

W tabeli 7.1 podajemy znamionowe wartości napięć wraz z ich tolerancjami. Przypominamy jednak, że zasilacz impulsowy możemy sprawdzać tylko wtedy, gdy jest obciążony!

Tabela 7.1. Znamionowe wartości napięć zasilacza

Napięcie	Tolerancja	Minimum	Maksimum
+5VDC	±5%	+4.75 V	+5.25 V
+12VDC	±5%	+11.40 V	+12.60 V
-5VDC	±10%	-4.5 V	-5.5 V
-12VDC	±10%	-10.8 V	-13.2V
+3.3VDC	±5%	+3.14 V	+3.47 V
+5V SB	±5%	+4.75 V	+5.25 V

Orientacyjne wartości prądu pobierane przez przykładowe elementy systemu komputerowego podano w tablicy 7.2. Aby obliczyć moc, którą będzie pobierał dany element, wartość pobieranego prądu mnożymy przez wartość napięcia. Oczywiście, moc, którą musi dostarczyć zasilacz, musi być równa sumie mocy pobieranych przez wszystkie elementy. Pewnym problemem jest też obciążenie poszczególnych napięć. Generalnie przy wydajnych procesorach i wyrafinowanych kartach graficznych stosuje się obecnie (koniec roku 2006), zasilacze 350, 400 W (ze wskazaniem na te ostatnie). Dla kart graficznych pracujących z technologią SLI zaleca się nawet zasilacze 500 W.

Tabela 7.2. Znamionowe napięcia zasilacza ATX

Element	Pobór prądu dla napięcia		
	+3,5 V	+5 V	+12 V
Athlon XP 3200+			5,1 A
Pentium 4 3,2			10,5 A
Płyta główna	4 A	3 A	
DDR 256 MB	1,2 A		
HDD 120 GB 7200 SATA		0,8 A	0,5 A
Napęd DVD		1,2 A	1,1 A
Stacja dyskietek		0,9 A	
Karta dźwiękowa PCI	0,5 A	0,5 A	
Klawiatura		0,2 A	

7.2. Złącza zasilaczy

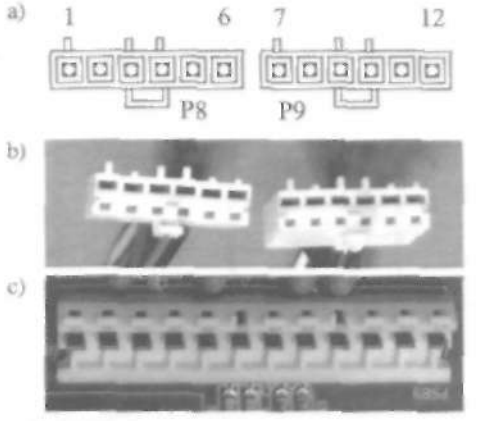
Poniżej przedstawiamy złącza zasilaczy PC. Nieco dokładniej piszemy o najczęściej spotykanych. Dla pozostałych podajemy rozkład wyprowadzeń, napięć i kolory przewodów.

7.2.-1. Złącze AT (AT PowerConnector)

Złącze to od strony zasilacza składa się z dwóch wtyków oznaczanych jako P8 i P9 pokazanych na rysunku 7.5a i b. Na rysunku tym pokazano też złącze od strony płyty głównej - rysunek c. Podłączając wtyki P8 i P9, należy zwrócić uwagę na to, że choć nie można ich odwrócić o 180°, to można je zamienić miejscami (prawe, lewy). Reguła prawidłowego podłączenia jest prosta. Przewody czarne, prowadzące masy, powinny się spotkać w środku złącza. Podłączenie odwrotne jest groźne w skutkach. I jeszcze krótki komentarz do sposobu podłączania tych wtyków, który nie jest zbyt wygodny. Wtyki umieszczamy nad kontaktami nieco ukośnie, tak aby wypustki wtyków weszły w otwory złącza na płycie, i dopiero wtedy ustawiamy wtyk pionowo i nasuwamy go na kontakty. Podczas demontażu postępujemy odwrotnie.

Na rysunku 7.5c pokazano gniazdo zasilania płyty głównej.

Pin	Kolor	Wyjście	Wtyk
1	Pomarańczowy	Power Good (+5 V)	P8
2	Czerwony	+5 V	
3	Żółty	+12 V	
4	Niebieski	–12 V	
5	Czarny	Masa	
6	Czarny	Masa	
7	Czarny	Masa	P9
8	Czarny	Masa	
9	Biały	–5 V	
10	Czerwony	+5 V	
11	Czerwony	+5 V	
12	Czerwony	+5V	

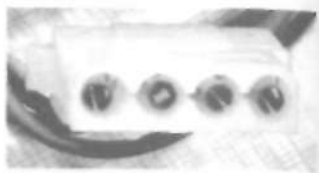


Rysunek 7.5. Złącza płyty głównej zasilacza AT

7.2.2. Złącze urządzeń peryferyjnych

Konstrukcja tego złącza praktycznie uniemożliwia niepoprawne jego zamontowanie. Natomiast czasami kłopot sprawia jego demontaż spowodowany bardzo ciasnym jego osadzeniem. Polecamy wtedy trzymać także część złącza od strony urządzenia w celu uchronienia eo od uszkodzenia i zsuwanie złącza drobnymi ruchami.

Pin	Kolor	Wyjście
1	Żółty	+12 V
2	Czarny	GND
3	Czarny	GND
4	Czerwony	+5 V

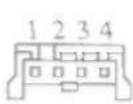


Rysunek 7.6. Złącze urządzeń peryferyjnych

7.2.3. Złącze napędu dyskietek

Złącze to jest zabezpieczone przed niepoprawnym włożeniem grzebieniem w złączu od strony napędu, który powinien zostać wsunięty w prowadnicę we wtyku zasilacza. Zabezpieczenie to jest jednak mało skuteczne, gdyż przy użyciu niewielkiej siły i lekkim wygięciu igiełek złącza napędu jest możliwe błędne podłączenie. Należy zwrócić na to uwagę, gdyż grozi to uszkodzeniem napędu.

Pin	Kolor	Wyjście
1	Żółty	+5 V
2	Czarny	GND
3	Czarny	GND
4	Czerwony	+12 V



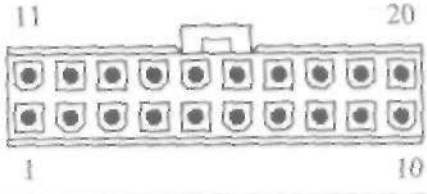
Rysunek 7.7. Złącze napędu dysków elastycznych

7.2.4. Złącze ATX(ATXv1.x Power Connector)

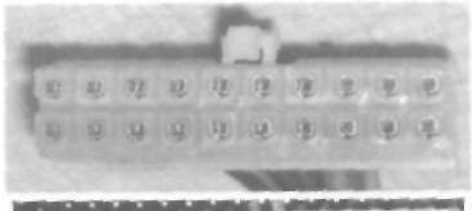
Złącze to jest bardzo dobrze zabezpieczone przed niepoprawnym podłączeniem. Natomiast przy jego odpinaniu należy pamiętać o naciśnięciu zaczepu z boku złącza w celu jego odpięcia. Na rysunku 7.8a pokazano schemat wyprowadzeń, 7.8b to zdjęcie złącza zasilacza i wreszcie 7.8c to gniazdo na płycie głównej.

Pin	Kolor	Wyjście
1	Pomarańczowy	+3.3 V
2	Pomarańczowy	+3.3 V
3	Czarny	Masa
4	Czerwony	+5 V
5	Czarny	Masa
6	Czerwony	+5 V
7	Czarny	Masa
8	Szary	PWR_OK
9	Purpurowy	+5VSB
10	Żółty	+12 V
11	Pomarańczowy	+3.3 V
12	Niebieski	–12 V
13	Czarny	Masa
14	Zielony	PS_ON#
15	Czarny	Masa
16	Czarny	Masa
17	Czarny	Masa
18	Biały	–5 V
19	Czerwony	+5 V
20	Czerwony	+5 V

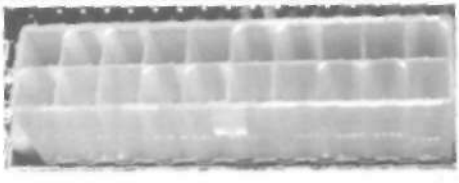
a)



b)



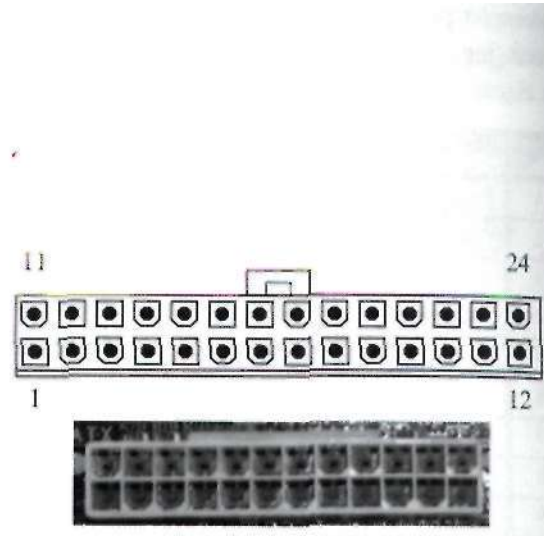
c)



Rysunek 7.8. Złącze ATX, numeracja wyprowadzeń, wtyk zasilacza i gniazdo na płycie głównej

7.2.5. Złącze ATX o podwyższonej mocy (ATX12V v2.x Power Connector)

Pin	Kolor	Wyjście
1	Pomarańczowy	+3.3 V
2	Pomarańczowy	+3.3 V
3	Czarny	Masa
4	Czerwony	+5 V
5	Czarny	Masa
6	Czerwony	+5 V
7	Czarny	Masa
8	Szary	PWR_OK
9	Purpurowy	+5VSB
10	Żółty	+12 V
11	Żółty	+12 V
12	Pomarańczowy	+3.3 V
13	Pomarańczowy	+3.3 V
14	Niebieski	–12 V
15	Czarny	Masa
16	Zielony	PS_ON#
17	Czarny	Masa
18	Czarny	Masa
19	Czarny	Masa
20	Biały	–5 V
21	Czerwony	+5 V
22	Czerwony	+5 V
23	Czerwony	+5 V
24	Czarny	Masa



Rysunek 7.9. Złącze ATX o podwyższonej mocy

7.2.6. Pomocnicze złącze ATX12 (ATX12V vl.x Auxiliary Connector)

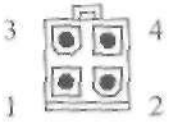
Pin	Kolor	Wyjście
1	Czarny	Masa
2	Czarny	Masa
3	Czarny	Masa
4	Pomarańczowy	+3.3 V
5	Pomarańczowy	+3.3 V
6	Czerwony	+5 V

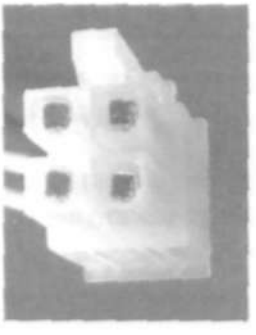
Rysunek 7.10. Pomocnicze złącze ATX

7.2.7. Złącze ATX12 12 V (ATX12V 12V Connector)

Uwagi do tego złącza są identyczne jak w przypadku złącza ATX (ATX v1.x Power Connector).

Pin	Kolor	Wyjście
1	Czarny	Masa
2	Czarny	Masa
3	Żółty	+12 V
4	Żółty	+12 V

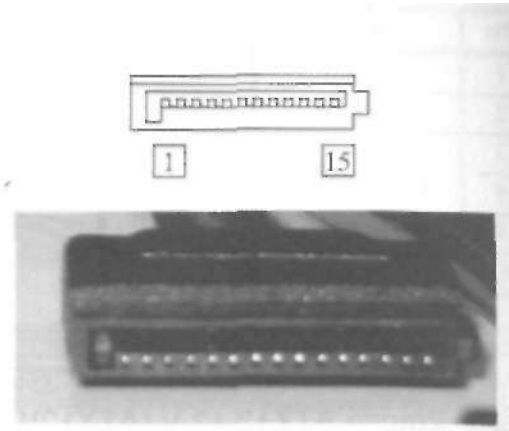




Rysunek 7.11. Dodatkowe złącze ATX 12V

7.2.8. Złącze zasilania Serial ATA (Serial ATA Power Connector)

Pin	Kolor	Wyjście
1	Pomarańczowy	+3.3 V
2	Pomarańczowy	+3.3 V
3	Pomarańczowy	+3.3 V
4	Czarny	Masa
5	Czarny	Masa
6	Czarny	Masa
7	Czerwony	+5 V
8	Czerwony	+5 V
9	Czerwony	+5 V
10	Czarny	Masa
11	Czarny	Masa
12	Czarny	Masa
13	Żółty	+12 V
14	Żółty	+12 V
15	Żółty	+12 V



Rysunek 7.12. Złącze zasilania interfejsu SATA

Dodatek A.
Wybrane pozycje Setup BIOS

W programach Setup możliwości ustawień dzielone są na pewne grupy. Przykładowy podział, który przedstawimy, wygląda następująco:

- ustawienia standardowe (ang. *Standard CMOS Features),*
- ustawienia BIOS-u (ang. *Advanced BIOS Features),*
- ustawienia własności chipsetu (ang. *Advanced chipset Features),*
- ustawienia urządzeń zainstalowanych na płycie głównej (ang. *Internat Peripherals),*
- zarządzanie mocą (ang. *Power Management Features),*
- konfiguracja PCI i Pług and Play (ang. *PNP/PCI Configuration),*
- monitorowanie parametrów systemu (ang. *H/W Monitor),*
- konfiguracja napięć i częstotliwości (ang. *Celi Menu).*

Nazwy pozycji w menu są przykładowe, wzięte z programu Setup prezentowanego poniżej, jednak podobne funkcje, choć czasami pod innymi nazwami, pojawiają się także w innych programach Setup.

Przedstawiamy jedną z wersji programu BIOS Setup firmy American Megatrends Incorporation (AMI). Opisywanie wszystkich opcji i wielu BIOS-ów nie ma sensu i mija się z celem. Chcemy natomiast zwrócić szczególną uwagę na trzy grupy ustawień: opcje ważne, w których błędne ustawienia mogą spowodować kłopoty, opcje, z którymi możemy spokojnie eksperymentować i opcje, w których najlepiej pozostawić wartości domyślne. Zwrócimy też uwagę na podobieństwa pewnych pozycji, nawet jeżeli występują w różnych BIOS-ach w różnych miejscach.

W opisie BIOS-u stosujemy następujące konwencje:

1. Nazwy głównych pozycji są wytłuszczone, natomiast obok, w nawiasach podany jest komentarz ukazujący się w dolnej części ekranu, informujący skrótowo, czego dana opcja dotyczy. W nawiasach podano też jej polskie tłumaczenie.
2. Jeżeli dana opcja ma podpunkty, są rozwijane jako a), b) itd., a te z kolei mogą być rozwijanie jako i), ii), iii) itd.

3. W niektórych przypadkach podajemy komentarz pomocy ukazujący się po wybraniu danej opcji (czasem są bardzo lapidarne). Komentarz ten piszemy kursywą.
4. Jeżeli cytujemy gdzieś możliwe ustawienia, podajemy je w nawiasach prostokątnych, czcionką Arial.
5. W nielicznych przypadkach, zwłaszcza gdy jest dużo pozycji w danej opcji, zamiast angielskiego określenia podajemy opis zawartości w języku polskim.

BIOS

1) **Standard CMOS Feature** (Set time, Date, Hard Disks - ustawianie czasu, daty parametrów dysków...)
   a) Date
   b) Time
   c) Primary IDE Master
      i) parametry i ustawienia trybów
   d) Secondary IDE ... itd.
   e) Floppy A
   f) Halt On [All But Keyboard]

2) **Advanced BIOS Features** (Set Boot Devices, Floppy function ... - wybiera urządzenia ładujące system, funkcjonowanie napędów dyskietek ...)
   a) Quick Booting
   b) Boot Sector Protection
   c) BoottoOS/2
   d) IOAPIC Function; *Include ACPI APIC table pointer to RSDT pointer list*
   e) MPS Table Version
   f) Fool Screen Logo Display
   g) Boot Sequence
      i) I^{sl} Boot Device
      ii) 2nd Boot Device
      iii) 3rd Boot Device
      iv) Boot From Other Device
      v) Hard Disk Drives
      vi) Removable Drives
      vii) CD/DVD Drivers

3) **Advanced Chipset Features** (Set System/DRAM Timing - ustawianie taktowania systemu i pamięci DRAM)
   a) Adjust CAS Latency Modę
   b) Burst Length; *Bursł length can be set to 8 or 4 beats. 64 Bit Dq must use the 4 beats.*
   c) VGA Share Memory Size
   d) Display Device select
   e) TV NTSC/PAL Display Select
4) **Internal Peripherals** (Set Super l/O, USB, IDE Function... - ustawienia urządzeń Super I/O (patrz podrozdział 6.3), USB, IDE,...)
   a) USB Controller
   b) USB Device Legacy Support
   c) Onboard LAN Controller
   d) Onboard LAN Option ROM
   e) Onboard 1394 Controller
   f) HD Audio Azalia Device
   g) IDE Device Configuration
      i) On-Chip IDE Controller - both, primary ....
      ii) PCI IDE BusMaster: Enabled: BlOSuses PCI busmastering for reading/ writting to IDE drivers
   h) I/O Devices Configuration
      i) COM Port
      ii) Parallel Port
      iii) Parallel Porte Mode : Normal, ECP
   i) SATA Device Configuration
      i) OnChip SATA Channel: disabled, single both
      ii) OnChip SATA Type:as IDE, as RAID, as Storage
5) **Power Management Features** (Set Sleep State, Suspend Timer, Wake Up Events ... - ustawia czas przejścia w stan uśpienia, opcje włączania...)
   a) ACPI Function
   b) ACPI Standby State [Sl/POS, S3/SPR, Auto]
   c) Suspend Time Out (Minutę)
   d) Power Button Function [Power Off, SuspendJ

e) Restore On AC Power Loss [Off, On, Last State]

0 Wakeup Event Setup : lista

6) **PNP/PCI Configuration** (Set IRQ?DMA... - ustawia przerwania i kanały DMA)

a) Clear ESCD

b) Primary Graphic's Adapter: opcja dla debugowania

c) PCI Latency Timer: w jednostkach zegara PCI, dla urządzenia PCI

d) IRQ Resource Setup: przydział przerwań dla starych kart - obecnie auto

e) DMA Resources Setup

7) **H/W Monitor** (Monitor Fan Speed, Temperaturę, Voltage... Nadzoruje obroty wiatraka, temperaturę, napięcie...)

a) CPU Shutdown Temperaturę

b) CPU FAN Failure Warning

c) Chassis Intrusion

d) Smart Fan: - disabled, 40°, 50°,...

e) PC Health Status: CPU Temperature, System Temperature, CPU FanSpeed, System Fan Speed, Vcore

8j **Celi Menu** (Set Frequency, Spread SpectrumFunction ... - ustawia częstotliwość, funkcję rozproszonego spektrum,...)

a) Current CPU Clock (tylko wyświetla)

b) Cool'n'Quict

c) Adjust DDR memory Frequency: - auto, Manuał - DDR memory Frequency:

d) Ratio Change : If AUTO,FID/VID will be left at the rated frequency/voltage. If Manuał FID/VID will be set based on User Selection in Setup

e) Auto Disable PCI Clock

f) Spread Spectrum

9) **Load Fail-Safe Defaults** (Load Fail-Safe Defaults values for all the setup questions)

10) **Load Optimized Defaults** (Load Optimized Defaults values for all the setup questions)

11) **Save** & **Exit Setup** (Exit System Setup with saving the changes)

12) **Exit Without Saving** (Exit System Setup without saving the changes)

Pozycja 1) od dawna ma podobny wygląd. Zwracamy jedynie uwagę, że wpisanie w danej pozycji urządzenia opcji None (zamiast n/p. Auto) powoduje, że jest ono ignorowane.

Pozycja 2) zawiera przede wszystkim bardzo ważną opcję f) Boot Sequence. Decyduje ona o kolejności przeszukiwania urządzeń w celu odnalezienia i załadowania

systemu operacyjnego. W pewnych przypadkach zależy nam na przykład, żeby system został załadowany z CD-ROM-u, a nie z dysku twardego. Jednocześnie zwracamy uwagę, że przykładowo ustawienie jako pierwszego urządzenia ładującego system stacji dyskietek i włożenie do niej dyskietki niesystemowej spowoduje komunikat błędu i oczekiwanie na naszą reakcję. Inaczej mówiąc, system nie będzie już szukany na dalszych urządzeniach. Inaczej dzieje się na przykład w przypadku CD-ROM-u.

Niektóre pozycje BIOS-u, na przykład 2e dotycząca tak zwanej tablicy MPS (ang. *Multi-Processor Specification Table),* są słabo dokumentowane i lepiej pozostawić w nich wartości domyślne.

W pozycji 3) możemy poeksperymentować z ustawieniem a) dotyczącym opóźnienia sygnału CAS. Proszę jednak pamiętać, że może to prowadzić do niestabilności systemu. W takim przypadku należy powrócić do ustawień stabilnych.

Opcja b) dla procesorów z 64-bitową magistralą danych musi wynosić cztery cykle. Opcja c) to przydział pamięci dla karty graficznej. Opcję tę często nazywa się Apreture Size.

Pozycja 4) jest dość istotna. Możemy tu między innymi zezwalać na działanie lub wyłączać urządzenia będące elementami płyty głównej. Wyłączenie urządzenia na płycie ma sens, gdy chcemy zamiast niego używać urządzenia tego samego przeznaczenia na karcie (na przykład szybsza karta sieciowa niż adapter na płycie). Możemy tu także ustawić parametry niektórych urządzeń, na przykład tryb pracy portu równoległego - opcja h), iii), czy też sposób wykorzystania dysków SATA i), ii).

W pozycji 5) mamy ustawienia dotyczące opcji oszczędzania energii (na przykład czas przejścia w stan uśpienia oraz ustawienia sposobu włączania i wyłączania systemu).

Pozycja 6) dotyczy ustawień PCI i Pług and Play. Opcja a) Clear ESCD (ang. *Extended System Configuration Data)* powoduje każdorazowe czyszczenie pamięci konfiguracji PnP BIOS-u i przeprowadzanie jej od nowa. Opcja c) decyduje, jak długo urządzenie PCI może transmitować na magistrali. Opcje d) i e) mówią o przydziale przerwań i kanałów DMA dla klasycznych kart ISA i nie są już obecnie użyteczne.

Pozycja 7) pozwala ustawiać wartości progowe takich wielkości, jak temperatura procesora czy obroty jego wiatraka, powodujące określone wymienione reakcje. Opcja e) podaje stan określonych wymienionych wielkości.

Pozycja 8) pozwala między innymi na przetaktowywanie procesora. Jestem jednak zdania, że w zastosowaniach profesjonalnych układy i urządzenia powinny pracować ze swoimi nominalnymi parametrami. Dlatego entuzjastów podkręcania zegarów systemu odsyłam do licznych publikacji w czasopismach. Opisując pozycję 8), warto wspomnieć o dwóch opcjach. Opcja b) Cool'n'Quiet jest technologią opracowaną przez AMD pozwalającą na oszczędzanie energii i cichą pracę komputera przy jego mniejszym obciążeniu. Wymaga współpracy procesora, płyty głównej, BIOS-u

i wentylatora. Opcja f) Spread Spectrum pozwala na zmniejszenie zakłóceń elektromagnetycznych przez niewielkie zmiany częstotliwości zegara. Jeżeli jednak podejrzewamy, że powoduje to niestabilną pracę systemu, opcję tę należy wyłączyć.

Na zakończenie podajemy opis przykładowych opcji BIOS-u z poprzedniego wydania książki. Pozwoli to Czytelnikowi zorientować się w kierunkach zmian oraz w pozycjach, które pozostają niemal niezmienione.

I. Ustawienia BIOS-u:

A. **CPU Internal Cache** - włączenie/wyłączenie pamięci cache L1.

B. **External Cache** - włączenie/wyłączenie pamięci cache L2.

Włączenie zarówno A, jak i B przyspiesza pracę systemu. Wyłączenie ma sens jedynie w razie problemów z działaniem systemu.

C **Quick Power On Self Test** - skrócona (szybka wersja) POST. Włączenie spowoduje szybszy start komputera.

D. **Boot Up Floppy Seek** - sprawdzenie obecności i rodzaju napędu dysków elastycznych. Wyłączenie spowoduje szybszy start komputera.

E. **System BIOS Shadow** - włączenie tej opcji powoduje przepisanie BIOS-u z wolniejszej pamięci ROM (czas dostępu rzędu 180 ns) do szybszej pamięci DRAM (czasy dostępu rzędu 60 ns). Używanie Shadow BIOS-u przyspiesza pracę komputera. Inna nazwa tej opcji to **System BIOS Cacheable.**

F. Video BIOS Shadow -jak wyżej dla BIOS-u wideo.

G. **Security Option** - pozwala wybrać zabezpieczenie hasłem dostępu do komputera (opcja System - hasło podajemy przed załadowaniem systemu operacyjnego) lub dostępu do Setupu (opcja Setup - hasło podajemy przed wejściem do Setupu).

II. Ustawienia własności chipsetu:

A. **Autoconfiguration** - włączenie/wyłączenie autokonfiguracji. W przypadku jej włączenia system dobiera ustawienia w oparciu na własnych danych. Nie muszą to być ustawienia optymalne.

B. ISA Bus (AT Bus) Clock Option - ustawienie szybkości zegara taktującego magistralę rozszerzającą ISA. Podawana zwykle w postaci wzoru, np. CLK/4. Im mniejszy dzielnik, tym szybszy zegar taktujący magistralę ISA. Możliwe są ostrożne eksperymenty. Po każdej zmianie sprawdzamy (przez pewien czas) poprawność działania kart ISA. Inne nazwy: **AT Clock frequency Select, Synchronous AT Clock.**

C. DRAM Read Wait State(s) - sposób odczytu pamięci w trybie burst. Przykładowo pozycja 4-3-3-3 oznacza pierwszy odczyt w 4 taktach zegara (dochodzi faza adresowania), a kolejne 3 odczyty zajmują 3 takty zegara. Oznaczenie

4-3-3-3 jest równoznaczne z oznaczeniem 3T (3 takty) lub 2 WS (2 stany oczekiwania, czyli 2 dodatkowe takty zegara). Im mniejsze liczby, tym szybciej pracuje komputer. Z ustawieniami tymi można swobodnie eksperymentować, sprawdzając za każdym razem stabilność systemu. W przypadku jego wadliwego działania przywracamy poprzednie ustawienia.

D. DRAM Write Wait State(s) -jak wyżej dla zapisu.

Podobne pozycje o tym samym znaczeniu występują dla pamięci cache, burst SRAM itp.

E. **Extemal Cache WB/WT** - sposób utrzymywania zgodnos'ci pamięci cache z pamięcią główną. WB - Write-back, WT - Write-through. Write-back jest szybszy.

F. **Internal Cache WB/WT** -jak wyżej dla wewnętrznej pamięci cache.

G. **IDE PIO Modę** - występuje w systemach ze zintegrowanym sterownikiem IDE dysku twardego. Wybiera tryb obsługi dysku twardego. Ustawiamy najszybszy tryb, w którym potrafi pracować nasz dysk twardy.

H. **Hidden Refresh Option** - włączenie lub wyłączenie opcji ukrytego odświeżania pamięci. Włączenie tej opcji powoduje przyspieszenie pracy komputera. Należy jednak sprawdzić poprawność jego funkcjonowania.

111. Zarządzanie poborem mocy:

A. **Power Management** - wybór trybu oszczędzania energii.

B. **PM Control By APM** - gdy odpowiemy tak (yes), sterowanie poborem mocy przejmuje oprogramowanie APM (ang. *Advanced Power Management)*.

C. **HDD Power Down** - ustawienie czasu, po którym zostanie wyłączony silnik dysku, jeżeli dysk nie jest używany.

Video Off Method - metoda dezaktywowania karty graficznej. V/H Sync+Blank - wyłączenie sygnałów synchronizacji i RGB, Blank Screen - wyłączenie sygnałów RGB (wygaszenie ekranu).

*

Bibliografia

11] Murray Sargent III, Richard L. Shoemaker, *The Personal Computer front the Inside Out,* Addison-Wesley 1995.

[2] Antoni Niederliński, *Mikroprocesory, mikrokomputery, mikrosystemy,* Wydawnictwa Szkolne i Pedagogiczne 1991.

[3] Tom Shanley, Don Anderson, *ISA System Architecture,* Addison-Wesley 1995.

[4] Tom Shanley, *The Unabridged Penttium 4,* Addison-Wesley 2006.

[5] Ravi Budruk, Don Anderson, Tom Shanley, *PCI Express System Architecture,* Addison-Wesley 2005.

]6] Piotr Metzger, *Anatomia PC,* Helion 2006.

[7] Tom Shanley, Don Anderson, *PCI System Architecture,* Addison-Wesley 1995.

[8] Tom Shanley, *80486 System Architecture,* Addison-Wesley 1995.

[9] Don Anderson, Tom Shanley, *Pentium Processor System Architecture,* Addison--Wesley 1995.

[10] Tom Shanley, *Pentium Pro and Pentium II System Architecture,* Addison-Wesley 1998.

[11] Tom Shanley, *Protected Modę Software Architecture,* Addison-Wesley 1996.

[12] Tom Shanley, *Pług and Play System Architecture,* Addison-Wesley 1995.

[13] Hans-Peter Messmer, *The Indispensahle PC Hardware Book,* Addison-Wesley 1997.

[14] M. Morris Mano, *Architektura komputerów.* Wydawnictwa Naukowo-Techniczne 1988.

115] Ryszard Goczyński, Michał Tuszyński, *Mikroprocesory 80286, 80386 i i486,* HELP 1991.

[16] Gary Syck, *Turboasembler, Biblia użytkownika,* LT&P 1994.

[17] Zdzisław Poręba, *Mikroprocesory RISC rodziny Power PC,* Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego 1995.

[18] Edward Stolarski, Jerzy M. Zając, *Pamięci półprzewodnikowe* Semper 1993.

119] Stanisław Kruk, *Procesor Pentium,* PLJ 1998.

[20] Michael L. Schmit, *Procesory Pentium, narzędzia optymalizacji.* Mikom 1995.

[21] *iAPX 286/10, Advanced Information,* Intel.

122] Henryk Małysiak. Bolesław Pochopień, Eugeniusz Wróbel, *Mikrokomputery klasy IBM PC,* Wydawnictwa Naukowo-Techniczne 1992.

[23] Leonid Bułhak, Ryszard Goczyński, Michał Tuszyński, *DOS 5.0 od środka,* HELP 1992.

|24] Andrzej Dudek, *Jak pisać wirusy,* SCS Press 1993.

[25] Jan Pieńkos, Janusz Turczyński, *Układy scalone TTL w systemach cyfrowych.* Wydawnictwa Komunikacji i Łączнos'ci 1980.

[26] Charles A. Holt, *Electronic circuits, digital and analog,* Wiley 1978.

[27] F. van Gilluwe, *The Undocumented PC Second Edition,* Addison-Wesley 1997.

[28] William Stallings, *Organizacja i architektura systemu komputerowego,* Wydawnictwo Naukowo-Techniczne 2000.

[29] Charles Petzold, *Programowanie Windows,* Read Me 1999.

[30] Wiesław Marciniak, *Przyrządy półprzewodnikowe i układy scalone.* Wydawnictwo Naukowo-Techniczne 1987.

Adresy internetowe:

[31] http://www.micron.com/products/modules/

[32] http://www.crucial.com/

[33] www.intel.com/

Skorowidz

URZĄDZENIA TECHNIKI KOMPUTEROWEJ

Krzysztof Wojtuszkiewicz

CZĘŚĆ 1

Jak działa komputer?

MIKOM

URZĄDZENIA TECHNIKI KOMPUTEROWEJ

CZĘŚĆ 1
Jak działa komputer?

Książka przeznaczona jest dla słuchaczy szkół policealnych o specjalności technik informatyk, dla uczniów technikum o specjalności systemy mikroprocesorowe oraz dla każdego, kto jest zainteresowany tym jak działa komputer. Całość pracy składa się z dwóch części - pierwsza omawia architekturę i działanie komputera, druga działanie urządzeń peryferyjnych i ich współpracę z jednostką centralną.

W części pierwszej omówiono budowę i działanie układów wchodzących w skład jednostki centralnej oraz architekturę współczesnego komputera klasy PC. Poszczególne tematy obejmują: podstawy działania układów cyfrowych i mikroprocesorowych, działanie mikroprocesorów rodziny Intel x86, budowę płyt głównych komputerów klasy IBM PC, standard Plug and Play, standardy magistrali rozszerzającej i budowę zasilaczy komputerów typu IBM/PC. Omawiane są też zagadnienia dotyczące pamięci: działanie pamięci półprzewodnikowych (w tym SDR i DDR DRAM, RDRAM), działanie pamięci cache oraz pamięci wirtualnej.

Obecne wydanie książki zostało gruntownie zmienione w celu uwzględnienia zarówno zmian w technice komputerowej jak i dostosowania podręcznika do wymagań nowej formy egzaminu potwierdzającego kwalifikacje zawodowe w zawodzie technik informatyk.

Krzysztof Wojtuszkiewicz jest absolwentem Wydziału Elektroniki Politechniki Wrocławskiej. Tam też ukończył studia podyplomowe na Wydziale Informatyki i Zarządzania. Od ponad 15-tu lat zajmuje się układami mikroprocesorowymi, architekturą komputerów i urządzeniami peryferyjnymi. Prowadzi między innymi zajęcia z Architektury Komputerów w Wyższej Szkole Informatyki i Zarządzania Copernicus we Wrocławiu. Przedmiot Urządzenia Techniki Komputerowej wykłada od początku istnienia specjalności technik informatyk. Uczestniczył w tworzeniu programu tego przedmiotu, współpracuje także z Okręgową Komisją Egzaminacyjną we Wrocławiu.

Jest autorem książek z zakresu urządzeń techniki komputerowej oraz wykorzystania symulatora układów elektronicznych PSpice.

Wolny czas lubi spędzać z bliskimi znajomymi, na włóczędze po górach, wycieczkach rowerowych oraz, w chwilach natchnienia, na malowaniu obrazów.

MIKOM
z każdym bitem serca
www.mikom.pl

ISBN-13: 978-83-01-15035-8
ISBN-10: 83-01-15035-1