helion.pl

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

# W sercu PC – według Petera Nortona




Autor: Scott H. A. Clark
Tłumaczenie: Wereszczyński Zygmunt
ISBN: 83-7197-796-4
Tytuł oryginału: Peter Norton's New Inside The PC
Format: B5, stron: 640

Książka „W sercu PC – według Petera Nortona" to pozycja zarówno dla początkujących, jak i dla zaawansowanych użytkowników komputerów. Jeśli chcesz dobrze zrozumieć działanie komputera, umieć go optymalnie skonfigurować, a w razie pojawienia się problemów poradzić sobie z ich usunięciem, sięgnij po tę pozycję. Dzięki książce rozpoznasz ogromną liczbę podzespołów, urządzeń peryferyjnych i innych opcjonalnych rozszerzeń przeznaczonych dla nowoczesnych „pecetów". Książka stanowi również niezbędną pomoc przy konfiguracji i obsłudze komputera.

Nazwisko Petera Nortona – jednego z wiodących światowych ekspertów w dziedzinie komputerów PC – to gwarancja najwyższego poziomu przekazanych w książce informacji. Jej najnowsze wydanie zawiera gruntownie przerobiony i zaktualizowany materiał, opisujący najnowsze tendencje w świecie komputerów osobistych.

Co można zyskać, przeczytawszy tę książkę?

- Znajomość podstawowych podzespołów płyty głównej komputera PC
- Wiedzę na temat różnych procesorów (ich wydajności, wspólnych cech i różnic)
- Solidne podstawy do poznawania zagadnień sieciowych
- Zrozumienie zaawansowanych zagadnień dotyczących pamięci komputerów PC
- Szeroką wiedzę na temat kart grafiki i wyświetlaczy
- Poznanie różnych metod druku komputerowego
- Poszerzenie wiedzy na temat zagadnień komunikacyjnych, w tym modemów i łącz szerokopasmowych
- Znajomość dysków IDE i SCSI
- Zrozumienie najważniejszych etapów rozruchu komputera PC
- Zapoznanie się z językami komputerowymi
- Możliwość spojrzenia w przyszłość technologii komputerowych

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

# Spis treści


**O Autorze** ..... **15**

**Wprowadzenie** ..... **17**

**Część I Szklana skrzynka** ..... **23**

**Rozdział 1. Komputer stacjonarny** ..... **25**

Trzy główne części ..... 26
Jednostka centralna ..... 26
Co tu mamy? — Pięć systemów w komputerze ..... 26
Pięć systemów jako całość ..... 38
Wyświetlacz ..... 38
Urządzenia wejściowe ..... 39
Wewnątrz komputera ..... 39

**Rozdział 2. Komputery przenośne** ..... **41**

Od komputerów walizkowych do laptopów i dalej ..... 41
Komputery przewoźne ..... 42
Prawdziwie komputery przenośne ..... 42
Laptopy i nasz przykładowy Dell Inspiron 8100 ..... 42
Nowe, cieńsze laptopy ..... 49
Komputery PC mniejsze niż laptopy ..... 50
Wyposażenie spotykane prawie wyłącznie w komputerach przenośnych ..... 51
Co to za nazwy: PCMCIA, karta PC i CardBus? ..... 51
Lekcje pobrane od przenośnych komputerów ..... 53
Ograniczenie miejsca prowadzi do zamkniętych rozwiązań ..... 54
Ograniczone miejsce ogranicza możliwości rozbudowy ..... 54
Ograniczone miejsce oznacza większą kontrolę jakości ..... 56
Jeśli wystarczająco dużo ludzi czegoś chce... ..... 56
Przedłużanie żywotności baterii ..... 57
Jeśli tylko jest to możliwe, zawsze korzystaj z zasilania sieciowego ..... 57
Dbaj o odpowiednie ładowanie baterii ..... 58
Korzystanie z funkcji oszczędzania energii ..... 58
Podsumowanie ..... 62

**Rozdział 3. Komputery kieszonkowe i ręczne** ..... **63**

Włóżmy komputer do kieszeni ..... 64
„Starożytność" ..... 64
Firma Psion ..... 64
Jabłko Newtona ..... 64

PalmPilot ... 66
Piszmy poprawnie ... 67
Graffiti ... 68
PIM-y, a nie Palmy ... 69
Palmy, a nie PIM-y ... 71
Linux pod ręką — Agenda ... 77
Urządzenia z Windows ... 78
Windows CE 1.0 i 2 ... 78
Windows CE 3.0 ... 79
Komputery ręczne ... 80
Porównania ... 80

**Rozdział 4. Zrozumieć bity i bajty ... 83**
Czym są informacja i dane? ... 83
Jaki jest rozmiar pytania? ... 84
Jaki jest rozmiar odpowiedzi? ... 84
Jak bity i bajty obsadzają kody ASCII? ... 88
Symbole i kody ... 89
Kody sterujące ... 92
Unicode ... 93
Zmniejszanie liczby bitów i bajtów a rzeczywistość ... 96
Podsumowanie ... 97

**Część II Wewnątrz komputera ... 99**

**Rozdział 5. Pierwsze spojrzenie na płyty główne i procesory ... 101**
Bezpieczna „chirurgia" jednostki centralnej ... 102
Zasady unikania uszkodzeń spowodowanych elektrycznością statyczną ... 103
Ostrożnie z siecią zasilającą — grozi śmiercią! ... 104
Rejestrowanie zmian i przywracanie poprzedniego stanu ... 104
Fundament, czyli płyta główna ... 106
Miejsce dla procesora ... 110
„Karmienie" komputera, czyli zasilacz ... 111
Przestrzeń do zapełnienia, czyli złącza i wnęki ... 112
Co się dzieje na magistrali? ... 113
Mechanizm „plug-and-play" ... 116
Procesory w komputerach PC ... 118
Czym jest procesor? ... 119
Rodzina Pentium ... 121
Procesor Itanium firmy Intel ... 125
Jeśli nasz komputer nie ma procesora firmy Intel... ... 126
Różne ulepszenia ... 128
Podsumowanie ... 129

**Rozdział 6. Karty graficzne i monitory ... 131**
Źle się dzieje, gdy nie można pokazać informacji ... 131
Czym jest podsystem wyświetlania? ... 131
Podstawowa charakterystyka systemu graficznego ... 132
Piksele ... 133
Rozdzielczość ... 134
Odstęp między plamkami i rozmiar plamki ... 135
Jak powstają obrazy na ekranie (przegląd) ... 139
Znaki i obrazy rastrowe ... 143
Gdzie i w jaki sposób jest tworzony i przechowywany obraz? ... 144
Jak powstaje obraz na ekranie? ... 146

Właściwy dobór kolorów ... 148
Regulacja monitora ... 149
Modyfikacja informacji w obrazie ... 149
Sposób porozumiewania się z wyświetlaczem ... 150
Zasady działania wyświetlaczy ... 151
Lampy kineskopowe ... 151
Wyświetlacze ciekłokrystaliczne ... 153
Podsumowanie ... 158

**Rozdział 7. Dyski i inne nośniki danych w komputerze PC ... 159**
Tradycyjne dyski w komputerze PC ... 159
Stacje dyskietek dla komputerów PC ... 160
Dyski twarde dla komputerów PC ... 162
Pochodzenie IDE, EIDE, ATA, ATAPI i innych ... 165
Początki historii ... 165
EIDE, ATA i ATAPI ... 166
Podłączanie urządzeń IDE ... 169
Magistrala SCSI ... 170
Architektura SCSI ... 170
Kilka „smaczków" SCSI ... 171
Urządzenia SCSI ... 173
Kontroler SCSI ... 174
Mieszanie dysków IDE i dysków SCSI ... 175
Podsumowanie ... 176

**Rozdział 8. Wymienne nośniki danych ... 177**
Wariacje na temat nośników danych w PC ... 177
Różnorodność nośników danych w komputerze PC ... 177
Główne i dodatkowe nośniki danych ... 178
Wymienne magnetyczne nośniki danych ... 178
Taśma magnetyczna ... 178
Wymienne dyski twarde ... 180
Dyski twarde z wymiennym nośnikiem ... 181
Dyskietki specjalne ... 181
Napędy Iomega ZIP ... 182
Dyski twarde na kartach PC, karty z paskiem magnetycznym i inne podobne urządzenia ... 183
Wymienne optyczne nośniki danych ... 184
Płyty kompaktowe ustanawiają standard ... 185
Dyski CD-ROM nie różnią się zbytnio — różnią się zaś czytniki ... 186
Odmienna technika: dyski do jednokrotnego zapisu (CD-R) ... 187
CD-RW umożliwia wielokrotny zapis ... 189
Płyta DVD (Digital Versatile Disc) ... 190
Urządzenia do magnetooptycznego zapisu danych ... 193
Podstawy magnetooptycznej metody zapisu danych ... 193
Elektroniczne nośniki danych dla komputerów PC ... 195
Karty pamięci typu flash ... 195
Miniaturowe karty pamięci ... 196
Podsumowanie ... 198

**Rozdział 9. Porty szeregowe ... 199**
Rozmowa przez cienką rurkę, czyli porty szeregowe ... 199
Podstawy komunikacji szeregowej ... 200
Szereg portów szeregowych ... 202
Łącze szeregowe bez użycia przewodów, czyli IrDA ... 202
Uniwersalna magistrala szeregowa (USB) ... 203

IEEE 1394 FireWire ... 208
Jeśli potrzeba dużo portów szeregowych... ... 211
Rozwiązanie z kartą rozszerzającą ... 211
Rozwiązanie z siecią lokalną ... 211
Rozwiązanie ostateczne ... 212
Podsumowanie ... 212

**Rozdział 10. Porty równoległe ... 213**
Szybsza ścieżka ... 213
Oryginalny jednokierunkowy port drukarkowy firmy IBM ... 213
Dwukierunkowe porty drukarkowe ... 214
Ulepszone porty drukarkowe ... 214
Konstrukcja mechaniczna ... 217
Porty „drukarkowe" nie tylko dla drukarek ... 218
Porty drukarkowe wcale nie są przeznaczone dla drukarek ... 218
Podsumowanie ... 219

**Rozdział 11. Sieci przewodowe i bezprzewodowe ... 221**
Czym jest sieć? ... 221
Dlaczego sieć jest potrzebna? ... 222
Cóż złego jest w sieci? ... 223
Rodzaje sieci ... 224
Sieci lokalne (LAN) ... 224
Sieci rozległe (WAN) ... 227
Intranety i Internet ... 228
Budowa sieci ... 230
Różne topologie ... 230
Karty i urządzenia sieciowe ... 233
Instalacja kablowa ... 233
Ethernet, czyli sieć w każdym kącie ... 233
Rozwiązania dla małych sieci ... 236
Sieci bezprzewodowe ... 237
Sieci o specjalnej konstrukcji ... 237
Sieci bezprzewodowe IEEE 802.11b i 802.11a ... 237
Sieci bezprzewodowe Bluetooth ... 238
Podsumowanie ... 239

**Rozdział 12. Modemy i dostęp szerokopasmowy ... 241**
Być w zasięgu ... 241
„Śpiewające" komputery ... 242
Dlaczego modemy są potrzebne? ... 242
Co to jest bod? ... 244
Standardy — im więcej, tym lepiej! ... 247
Różnorodność modemów ... 248
Modemy kablowe ... 250
Wszystko cyfrowe ... 253
Różne odmiany DSL ... 254
Sieci ISDN ... 257
T1 i inne szybkie łącza cyfrowe ... 258
Co wybrać? ... 259
Podsumowanie ... 260

**Rozdział 13. Urządzenia wejściowe ... 261**
Klawiatura jest kluczem do wszystkiego ... 261
Podstawy działania klawiatury ... 262
Różne konstrukcje klawiatur ... 262

Zadania klawiatury komputera ... 265
Kody klawiszy i sterownik klawiatury w jednostce centralnej ... 265
Powiadamianie programów o wciśniętych klawiszach ... 266
Różne klawiatury dla różnych potrzeb ... 266
Alternatywa dla pisania, czyli rozpoznawanie głosu ... 270
Słuchanie i rozumienie to trudne zadania ... 272
Mowa naturalna i przerywana ... 272
W skrócie: jak to działa? ... 273
Rozumienie ... 274
Jak daleko już zaszliśmy? ... 274
Pokazywanie punktu, czyli kręcenie myszą ... 275
Różne rodzaje myszy ... 275
Wiele przycisków i kółko ... 279
Inne urządzenia wejściowe dla komputerów PC ... 281
Skanery ... 281
Cyfrowe kamery i aparaty fotograficzne ... 283
Joysticki i inne urządzenia podłączane do portu gier ... 285
Podsumowanie ... 285

**Rozdział 14. Drukarki ... 287**
Zadania i znaczenie drukarek ... 287
Języki opisu strony ... 288
PCL ... 288
PostScript ... 289
HP-GL/2 ... 289
Technologie druku ... 289
Druk uderzeniowy ... 289
Drukarki atramentowe ... 291
Monochromatyczne drukarki laserowe ... 291
Kolorowe drukarki laserowe ... 293
Drukarki ze stałymi barwnikami ... 296
Drukarki sublimacyjne ... 298
Inteligentne drukarki stronicowe ... 299
„Bezmyślne" drukarki stronicowe ... 301
Uzyskiwanie prawie poprawnego koloru ... 301
Modele kolorów ... 302
Programy do korekty koloru i profile drukarek ... 305
Podsumowanie ... 307

**Część III Czarne skrzynki w szklanej skrzynce ... 309**

**Rozdział 15. Tajemnice płyty głównej ... 311**
Architektura procesora ... 311
RISC ... 312
CISC i szczegóły architektury procesorów x86 ... 312
Układy „zgodne z Intelem" ... 313
Nowa architektura IA64 ... 315
EPIC ... 315
X86-64, czyli alternatywa z firmy AMD ... 316
Jednostka interfejsu magistrali ... 316
Rozdzielanie instrukcji i danych ... 317
Przewidywanie działań i ich wykonywanie ... 317
Rejestry są miejscem tymczasowego przechowywania danych ... 318
Obliczanie adresów ... 323

Jednostka arytmetyczno-logiczna....328
Pamięć podręczna pierwszego poziomu....332
Architektura układów towarzyszących procesorowi....334
Pamięć....335
Porty....340
Przerwania, czyli siła napędowa....342
Nasłuch kontra przerwania....342
Tablica wektora przerwań....343
Jak powstają przerwania?....344
Programy obsługi przerwań....346
Usługi BIOS w pamięci ROM....347
Usługi DOS i BIOS w pamięci RAM....347
Czym jest kanał DMA?....348
Dlaczego DMA popadł w niełaskę?....349
Powrót DMA....350
Dostosowywanie się do zegara....350
Porównanie asynchronicznego i synchronicznego działania komputerów....351
Różne zegary do różnych celów....351
Co to jest superskalowanie?....353
Magistrale systemowe: ISA, PCI i AGP....354
Pierwotna magistrala ISA....354
Magistrala PCI....355
PCI jako oś „północ-południe" w komputerze PC....359
AGP, czyli boczna ścieżka do ulepszonej grafiki....360
Wiele magistral PCI w jednym komputerze....362
Mostek południowy....363
Co zyskano dzięki oddzieleniu mostka północnego od południowego?....364
PCI-X, czyli kolejna wersja PCI....364
PXI-X i 3GIO....366
Ocena wydajności za pomocą testów....367
FLOPS, SPEC, MIPS i BogoMips....368
Podsumowanie....368

**Rozdział 16. Proces rozruchu....369**
Czym jest BIOS?....369
Historia BIOS-u....370
Odwrotna inżynieria BIOS-u....371
Pamięć CMOS i inne programy....372
Modyfikacja parametrów zapisanych w pamięci CMOS....372
Pamięć NVRAM....373
Test wykonywany po włączeniu zasilania (POST)....374
Przebieg testu POST....374
Programy obsługi urządzeń i oprogramowanie sprzętowe....376
Jakie postępowanie jest właściwe?....377
Odporność BIOS-u na błędy....378
Przejście komputera do normalnej pracy....378
Szczegółowa kolejność działań podczas rozruchu systemu....380
Główny rekord rozruchowy....385
Proces rozruchu systemu operacyjnego....386
Przyspieszanie procesu rozruchu komputera....388
Komputery z funkcją oszczędzania energii pozornie uruchamiają się szybciej....389
Powszechnie stosowane opcje dostrajania BIOS-u....390
Zabezpieczenie pamięci CMOS....393
Producenci BIOS-ów....393
Podsumowanie....394

**Rozdział 17. Pamięć komputera PC ... 395**
Podstawowy obszar działania procesora ... 395
Dlaczego pamięć jest miejscem, w którym dzieje się wszystko? ... 396
Co trzeba wiedzieć o scalonych układach pamięci i o modułach? ... 396
Pamięć ROM i RAM ... 397
Co to jest parzystość? ... 399
Co to jest ECC? ... 399
Wewnętrzna organizacja układów pamięci ... 400
Różne odmiany pamięci RAM ... 402
Pamięci statyczne RAM (SRAM) ... 403
Pamięci dynamiczne RAM (DRAM) ... 403
Odmiany układów DRAM: FPM, EDO, VRAM, SDRAM oraz DDR RAM ... 405
Szybsze pamięci SDRAM ... 406
Rambus ... 407
Pamięci DDR SDRAM ... 411
PC1600 i PC2100 ... 411
Rynek pamięci DDR ... 411
Spodziewana wydajność pamięci DDR ... 412
Odmiany pamięci ROM i NVRAM ... 412
Pamięci ROM z programowaną maską ... 413
Adresowanie pamięci: segmenty firmy Intel ... 415
Ograniczenia wprowadzone przez firmy Intel i IBM ... 416
Nowszy model pamięci ... 417
Pamięć pamięci nierówna ... 418
Adresy logiczne, segmentowe, wirtualne, liniowe i fizyczne ... 419
Pamięć nie dostrzegana przez procesor ... 421
Zarządzanie pamięcią w komputerze PC ... 424
Przydział pamięci w systemie Windows ... 425
Polecenie MEM ... 430
Osławiona bariera 640 kB i sposoby jej przełamywania ... 431
Gospodarka pamięcią w systemie Windows ... 432
System Windows ma specjalne wymagania co do pamięci ... 433
Wirtualne maszyny DOS w systemie Windows ... 434
Wirtualna maszyna Windows ... 435
Sposoby wspomagania systemu Windows w zarządzaniu pamięcią ... 435
Programy do zarządzania pamięcią ... 436
Ile naprawdę potrzeba pamięci RAM? ... 437
Podsumowanie ... 438

**Rozdział 18. Jak są przechowywane dane na dyskach? ... 439**
Dyski twarde ... 439
Struktury fizyczne ... 440
Struktura logiczna dysku działającego w systemie Windows ... 448
Różnice między dyskami twardymi a dyskietkami ... 469
Główny rekord rozruchowy i partycje ... 469
Tablice rozszerzonych partycji DOS ... 473
Porządkowanie chaosu ... 474
Oznaczenia literowe dysków w systemie Windows ... 474
Pojemność twardego dysku C: ... 475
Systemy RAID ... 475
Dyski optyczne ... 476
Romeo i Joliet odwiedzają High Sierra ... 477
Universal Data Format (UDF) ... 478
Czym są kolorowe księgi? ... 478

Formaty plików ... 482
Pliki tekstowe ASCII ... 482
Pliki binarne (nie korzystające z kodów ASCII) ... 483
Kompresja danych ... 486
Autonomiczne programy do kompresji danych ... 487
Podsumowanie ... 488

**Rozdział 19. Zaawansowane zagadnienia sieciowe ... 489**
Model warstwowy sieci ... 489
Interfejs sieciowy wymaga sieciowego systemu operacyjnego ... 490
Czym jest sieciowy system operacyjny? ... 490
Powszechnie znane sieciowe systemy operacyjne ... 491
TCP/IP, czyli internetowa „ryba Babel" ... 494
Czym jest Internet? ... 494
Internet jest i zarazem nie jest podobny do innych sieci ... 495
Protokoły „do wszystkiego" ... 496
Adresy IP ... 499
Nazwy węzłów i domeny ... 499
Porty i gniazda sieciowe ... 501
Zapewnienie jakości usług w przyszłym Internecie ... 503
Podsumowanie ... 504

**Rozdział 20. Akceleratory graficzne ... 505**
Obrazy wektorowe i rastrowe ... 506
Trójwymiarowe obrazy wektorowe ... 508
Jak powstaje obraz w komputerze? ... 509
Obrazy pełne życia ... 511
Płaszczyzny bitowe i głębia kolorów ... 512
Układy RAMDAC ... 516
Program obsługi karty grafiki ... 516
Więcej na temat szybkości działania AGP ... 516
Obsługa tekstur w AGP ... 518
AGP Pro ... 518
Poziomy wydajności kart AGP ... 519
Adresowanie boczne i zwiększanie częstotliwości zegarowej w AGP ... 520
Świat grafiki trójwymiarowej ... 521
Akceleracja grafiki trójwymiarowej ... 522
Magia 256 bitów ... 526
Testy wydajności przetwarzania grafiki trójwymiarowej ... 527
Podsumowanie ... 528

**Rozdział 21. Wnętrze przyszłego komputera PC ... 529**
Świat po zmianie ... 529
To, co jest pewne ... 532
Twoja własna szklana skrzynka ... 533

**Dodatki ... 535**

**Dodatek A System operacyjny: pośrednik między człowiekiem a komputerem ... 537**
Czym jest system operacyjny i dlaczego jest on potrzebny? ... 538
Jak zrezygnować z systemu operacyjnego i dlaczego jest to zły pomysł? ... 538
Epoka systemu DOS ... 539
Działanie systemu operacyjnego ... 540

DOS nie żyje! Niech żyje Windows! ... 545
Obecnie używane odmiany Windows ... 545
Windows jest środowiskiem sterowanym zdarzeniami ... 549
Wielozadaniowość bez wywłaszczania i z wywłaszczaniem ... 550
Wybór systemu operacyjnego dla komputera PC ... 551
Windows 98 i Windows Me ... 552
Windows 2000 i Windows XP ... 552
Linux ... 553
Unix ... 554
W jaki sposób można uniknąć dokonywania wyboru ... 554
Podsumowanie ... 555

**Dodatek B Jak ludzie wydają polecenia komputerom? ... 557**
Jak zaprząc komputer do pomocy zwykłym ludziom? ... 557
Języki asemblera zmniejszają wysiłek umysłowy ... 558
Praca na wyższym poziomie: niech komputer wykonuje więcej zadań ... 562
Podział pracy ... 568
Programowanie na poziomie BIOS-u ... 569
Programu użytkowe (aplikacje) ... 569
System operacyjny jako produkt pośredniczący ... 569
Jak pracować, by nie wymyślać ponownie koła? ... 570
Biblioteki programowania ... 570
Konsolidacja programów i programy modułowe ... 571
Programowanie obiektowe ... 572
Ułatwienia w „programowaniu" komputerów PC przez zwykłych użytkowników ... 574
Podsumowanie ... 576

**Słowniczek ... 577**

**Skorowidz ... 617**

Rozdział 15.

# Tajemnice płyty głównej

W tym rozdziale zajmiemy się — nazywaną przez wiele osób „wnętrznościami" komputera — płytą główną. Zamontowane są na niej wszystkie najważniejsze części komputera PC, jak również bezpośrednie połączenia do tych części — z wyjątkiem części podłączanych za pomocą zewnętrznych magistral, takich jak: SCSI, USB lub IEEE 1394. Największe znaczenie wśród wszystkich elementów komputera ma procesor. To właśnie on decyduje o zaklasyfikowaniu komputera do określonej grupy.

## Architektura procesora

Konstruktorzy komputerów mówią często o *architekturze* danego modelu. Cóż określenie to ma oznaczać? Przecież zwykle dotyczy ono budynków, a nie komputerów!

W słowniku amerykańskiej odmiany języka angielskiego słowo *architekt* zdefiniowano jako „projektant" (czegokolwiek), zaś architektura to „struktura" (również czegokolwiek). A zatem architektura komputerów oznacza opis ułożenia poszczególnych części komputera oraz zasady ich budowy, które umożliwiają między nimi współpracę i określone działanie.

Podstawą każdej konstrukcji komputera jest procesor (CPU). Projektanci tego układu scalonego właśnie podczas jego tworzenia ostatecznie decydują o tym, jakie możliwości będzie posiadał komputer. Na przykład procesor ma określoną liczbę wyprowadzeń. Ich liczba i przeznaczenie wskazuje na rozmiar pamięci, która może być z nim połączona, na rodzaj informacji przetwarzanych w procesorze, reakcję na zdarzenia zewnętrzne itp.

Procesory z rodziny x86 mają wiele cech wspólnych. Ten zestaw jednorodnych właściwości określa architekturę x86, zaś konstrukcja samego komputera PC jest częścią bardziej ogólnej budowy x86. Zanim przejdziemy do opisu szczegółów związanych z x86, warto omówić różnice występujące między dwiema podstawowymi konstrukcjami procesorów, oznaczanymi skrótami RISC i CISC.

# RISC

*RISC* jest skrótem od nazwy *Reduced Instruction Set Computer* (czyli komputer o zredukowanym zestawie instrukcji). Procesor tego typu obsługuje stosunkowo małą liczbę instrukcji. Czyż nie brzmi to dziwnie? Oczywiście, chociaż trzeba pamiętać, że niekiedy jakość jest znacznie ważniejsza niż ilość.

Tradycyjnie producenci komputerów konstruują coraz bardziej skomplikowane procesory, by sprostać nowym wymaganiom ekstrawaganckich funkcji. Takie myślenie prowadzi do tworzenia sprzętu, dzięki któremu obsługiwanych jest coraz więcej rodzajów instrukcji, lecz odbywa się to kosztem czasu ich wykonywania. Im większy jest zestaw instrukcji, tym dłużej trwa ich wykonywanie.

Niektórzy producenci komputerów postępują odwrotnie; chcą zoptymalizować wydajność procesorów i zmniejszyć koszt ich produkcji, a zatem konstruują procesory obsługujące niewielką liczbę instrukcji. Typowym procesorem RISC jest PowerPC. Dzięki prostszym instrukcjom, ich wykonywanie może być przyspieszone bez konieczności wbudowywania nowych tranzystorów.

Czy jest to logiczne? Oczywiście, niekiedy tak. Jeśli zestaw instrukcji zostanie uproszczony, wszelkie wyrafinowane funkcje muszą być realizowane za pomocą oprogramowania, co zamiast wzrostu wydajności przetwarzania — może spowodować jej spadek.

Na obecnym etapie rozwoju technologii procesorów rozwiązania typu RISC stają się coraz bardziej skomplikowane. Obecnie układy RISC obsługują znacznie więcej instrukcji niż robiły to niegdyś. Z drugiej strony, tradycyjne procesory optymalizowane są z zastosowaniem sztuczek używanych dotychczas tylko w procesorach RISC.

Czy zatem, kiedy zamierzamy kupić nowy procesor, powinniśmy wybierać układ typu RISC? Zazwyczaj sprzedawca nie wie, czy dany procesor jest procesorem tego typu. Jedno można stwierdzić z całą pewnością: RISC nie jest już rozwiązaniem, bez którego nie można się obejść.

# CISC i szczegóły architektury procesorów x86

*CISC* oznacza *Complex Instruction Set Computer* (czyli komputer o złożonym zestawie instrukcji). Większość komputerów osobistych, także te z procesorami z rodziny x86, ma taką właśnie architekturę.

Bez przeszkód mógłbym zapełnić całą dalszą część książki dokładnym opisem zasad działania poszczególnych procesorów z rodziny x86. Znużyłoby to jednak tylko Czytelników i doprowadziło do łzawienia oczu z wysiłku włożonego w mozolną lekturę. Nie trzeba wcale poznawać wszystkich szczegółów, by uzyskać rzetelną wiedzę ogólną na temat działania procesorów. Ten podrozdział poświęcony jest przeglądowi najważniejszych właściwości architektury układów procesorowych. Celowo będzie się tu zacierać niektóre granice podziału między poszczególnymi elementami procesora lub grupować je na podstawie wzajemnych powiązań. Dzięki zastosowaniu ujednoliconego opisu, odnoszącego się do każdego procesora z rodziny, można uzyskać pełniejszy i jaśniejszy obraz niż przy szczegółowym opisie różnic między poszczególnymi układami.

Na rysunku 15.1 pokazano fotografię najnowszego procesora Pentium 4 firmy Intel, zaś na rysunku 15.2 widać fotografię procesora Celeron, produkowanego przez tę samą firmę.



**Rysunek 15.1.** *Układ Pentium 4 jest mikroprocesorem najnowszej generacji*




**Rysunek 15.2.** *Rodzina mikroprocesorów Celeron była projektowana z myślą o zrównoważeniu wydajności i ceny*

Trzeba pamiętać, że dalej w tym rozdziale wielokrotnie można będzie znaleźć odwołania do układów z rodziny x86. Pod tą wspólną nazwą rozumie się nie tylko serię mikroprocesorów produkowanych przez firmę Intel, łącznie z wieloma modelami Pentium, lecz również wszystkie klony tych procesorów, produkowane przez firmy konkurencyjne (na przykład AMD i VIA oraz Cyrix i IDT). Dla uproszczenia nie będziemy za każdym razem podkreślać tych rozróżnień. Jeśli dany komputer wyposażony jest w któryś z tych sklonowanych układów, należy przyjąć, że to, co napisano o procesorach x86, będzie prawdopodobnie prawdziwe także dla procesora w rozpatrywanym komputerze. Jeżeli różnice między poszczególnymi odmianami będą znaczące, zostanie to wyraźnie podkreślone.

Należy także pamiętać o innej ważnej sprawie. Pomimo deklarowanej zgodności układów produkowanych przez firmy AMD lub VIA z rodziną x86, niektóre z nich wymagają użycia specjalnych zestawów układów scalonych na płycie głównej. Jako przykład można podać procesory Duron i Athlon firmy AMD, które nie mogą działać z zestawami układów firmy Intel, a nawet wymagają podstawek o innych rozmiarach.

# Układy „zgodne z Intelem”

Na początku lat osiemdziesiątych ubiegłego wieku Intel był jedynym producentem procesorów z rodziny x86. Gwałtowny wzrost popytu na układy tego typu spowodował, że na rynku pojawiło się wielu producentów wytwarzających procesory „zgodne z Intelem”. Układy te obsługiwały ten sam zestaw instrukcji x86 i często były tańsze niż oryginalne produkty firmy Intel. W pewnych przypadkach wykazywały się nawet lepszą wydajnością.

Wiodącymi producentami intelowskich podróbek stały się firmy Cyrix i AMD. Firma Cyrix założona została w 1988 roku właśnie po to, aby rozpocząć produkcję mikroprocesorów zgodnych z układami Intela. Seria wytwarzanych przez nią mikroprocesorów 6x86 porównywalna jest z procesorami Pentium pod względem wydajności obliczeń stałoprzecinkowych, które odgrywają istotną rolę w typowych aplikacjach biurowych. Niestety, z powodu małej wydajności obliczeń używanych w grafice trójwymiarowej, seria ta nie zyskała wielkiego uznania wśród odbiorców. Firma Cyrix została przejęta przez National Semiconductor w roku 1997, a następnie przez firmę VIA w roku 1999. Na rysunku 15.3 pokazano fotografię procesora Cyrix 3, produkowanego już przez firmę VIA.

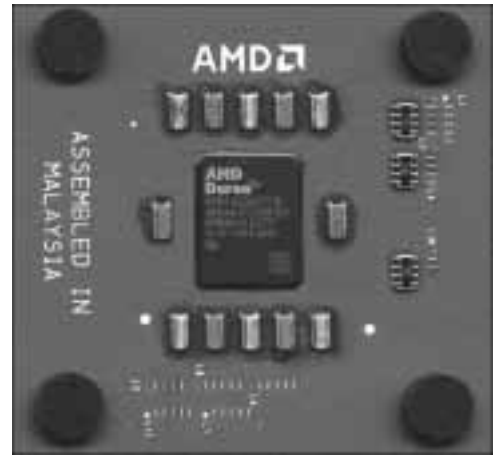



**Rysunek 15.3.**
*Procesor Cyrix 3 firmy VIA jest najnowszą konstrukcją na długiej liście produktów konkurujących na rynku z procesorami firmy Intel*

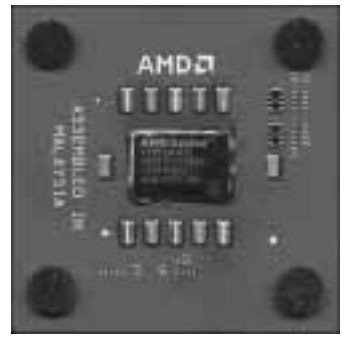
Firma AMD (skrót od angielskiej nazwy Advanced Micro Devices) swoim procesorem K7 Athlon zagroziła rynkowej pozycji Intela. Seria tych układów przewyższa procesory Pentium III pod wieloma względami i traktowana jest przez wielu najbardziej zagorzałych graczy jako jedyna platforma dla grafiki trójwymiarowej. Na rysunku 15.4 pokazano procesor K6-2, na rysunku 15.5 procesor Duron, a na rysunku 15.6. Athlon — wszystkie opracowane i wyprodukowane przez AMD.




**Rysunek 15.4.**
*Procesor K6-2 firmy AMD jest innym konkurentem dla układów Intela*




**Rysunek 15.5.**
*Kolejną pozycją w ofercie AMD jest rodzina procesorów Duron*




**Rysunek 15.6.** *Rodzina procesorów Athlon firmy AMD wybierana jest przez wielu użytkowników, szczególnie przez wymagających graczy*

# Nowa architektura IA64

IA64 to kodowa nazwa nowej technologii procesorów 64-bitowych, opracowanych wspólnymi siłami przez firmy Intel i Hewlett-Packard. Według zamierzeń projektantów w obecnym tysiącleciu mają się one stać powszechną platformą obliczeniową. Architektura IA64 zyskała poparcie wszystkich większych firm działających na rynku komputerowym z powodu swojej potencjalnie olbrzymiej wydajności.

Z technicznego punktu widzenia, na procesorze IA64 można uruchamiać programy pisane pierwotnie dla architektury x86. Jeżeli jednak chce się uzyskać maksymalną wydajność, należy stworzyć oprogramowanie zoptymalizowane dla obliczeń 64-bitowych.

W architekturze IA64 przewidziano szereg zaawansowanych metod przetwarzania danych, łącznie z instrukcjami wykorzystującymi długie słowa, predykcją, eliminacją rozgałęzień i spekulatywnym ładowaniem kodu. Jednak najbardziej interesujące w tym wszystkim jest wprowadzenie całkowicie nowego podejścia do obliczeń, nazywanego skrótowo EPIC.

# EPIC

Skrót *EPIC* pochodzi od angielskiego określenia *Explicitly Parallel Instruction Computing* (co oznacza bezpośrednie równoległe przetwarzanie instrukcji). Dzięki udostępnieniu możliwości przetwarzania równoległego już na poziomie instrukcji, przełamuje ono tradycyjnie obecne w architekturach RISC i CISC bariery dla programów wykonywanych sekwencyjnie. W tego typu rozwiązaniu system może wykonywać tyle instrukcji, ile wynika z możliwości obliczeń równoległych, jeśli tylko owe instrukcje są odpowiednio powiązane.

EPIC oznacza całkowicie nowe metody obliczeniowe. Samo przyjęcie tej technologii nie wystarczy, aby uzyskać efektywny wzrost wydajności. By osiągnąć taki efekt, należy od nowa napisać cały system operacyjny. W fazie opracowania znajduje się już specjalna, 64-bitowa wersja systemu Windows 2000. Istnieją również wersje Linuksa, które obsługują przetwarzanie 64-bitowe.

# X86-64, czyli alternatywa z firmy AMD

Jedną z wad intelowskiej technologii IA64 jest brak wstecznej zgodności. Procesor może wykonywać stare aplikacje 32-bitowe poprzez wykorzystanie emulacji, jednak uzyskiwana wówczas wydajność jest mała. W rzeczywistości każda operacja wykonywana poprzez emulowanie przebiega wolno.

Użytkownicy chcący zastosować architekturę 64-bitową i utrzymać tę samą wydajność, którą mają wykorzystywane przez nich aplikacje 32-bitowe, w zasadzie powinni obsługiwać oddzielne procesory 32-bitowe. Prowadzi to do powstawania dużych dylematów przy podejmowaniu decyzji o wdrożeniu nowej technologii na własnym sprzęcie, bowiem można nigdy nie osiągnąć tego, co jest najlepsze w obydwu technologiach!

W 64-bitowej architekturze zaprojektowanej przez firmę AMD zastosowano odmienne rozwiązanie. Zamiast całkowitej przebudowy procesora na 64 bity, dodano 64-bitowe rozszerzenia do istniejącej 32-bitowej architektury x86. Dzięki temu w jednym rdzeniu procesora można uzyskać najlepszą z możliwych wydajność w trybie 32-bitowym i jednocześnie zgodność z architekturą 64-bitową.

W nowym 64-bitowym rozszerzeniu firmy AMD dodano do rdzenia x86 osiem 64-bitowych rejestrów ogólnego przeznaczenia, 64-bitowe adresowanie pamięci oraz taki sam wskaźnik instrukcji. Rozszerzenia te będą zawarte w mającym się pojawić w niedługim czasie procesorze K8-Hammer.

# Jednostka interfejsu magistrali

Wyobraźmy sobie, że „odwiedzamy" procesor. Nasze pierwsze wrażenie powodowane jest jego wyglądem zewnętrznym, a dopiero potem tym, co zobaczymy w „przedpokoju". Rzeczywisty procesor połączony jest ze światem zewnętrznym za pomocą części swoich układów elektronicznych, którą nazywamy *jednostką interfejsu magistrali*. To ona „nadsłuchuje" (sprawdzając napięcie), co dzieje się na niektórych wyprowadzeniach i oczekuje na sygnały wejściowe oraz „informuje otoczenie" (ustalając napięcie) za pomocą innych wyprowadzeń. Istnieją także takie wyprowadzenia, które mogą „słuchać" lub „informować" w zależności od stanu napięcia na jeszcze innym wyprowadzeniu.

Ta część procesora odpowiedzialna jest również za *buforowanie* sygnałów wejściowych i wyjściowych. Oznacza to, że zawiera ona wzmacniacze sygnałów wyjściowych stosowane po to, by można było je odebrać w określonej liczbie układów odbiorczych. Są tu także odbiorniki, które mogą wykrywać sygnały wejściowe i pobierają w tym celu bardzo mało energii.

W niektórych układach z rodziny x86 jednostka interfejsu magistrali musi również dokonywać przekształceń poziomów napięcia. W procesorach tego typu zewnętrzne układy mogą pracować przy zasilaniu napięciem o wartości 5 V, zaś układy wewnętrzne zasilane są napięciem 3,3 V lub 2,5 V. Oznacza to, że poziom napięcia reprezentujący logiczną jedynkę na wyprowadzeniach zewnętrznych musi być nieco

wyższy niż 2 V, lecz wewnątrz procesora jedynce odpowiada poziom około 1 V. W obydwu przypadkach logicznemu zeru odpowiada poziom napięcia bliski 0 V. Odbiorniki w jednostce interfejsu magistrali i wzmacniacze wyjściowe muszą przyjmować i generować sygnały mieszczące się w tych zakresach dla wszystkich sygnałów wejściowych i wyjściowych, w zależności od tego, czy podłączone są do układów wewnętrznych procesora, czy do tego, co jest na zewnątrz względem procesora.

Oprócz tego, niektóre układy z rodziny x86 mają wewnętrzne generatory zegarowe (czasami mówi się o częstotliwości taktowania procesora), które zwielokrotniają częstotliwość zewnętrznego sygnału zegarowego (wówczas mówi się o *częstotliwości taktowania magistrali*). Jednostka interfejsu magistrali jest zatem odpowiedzialna za generowanie wewnętrznego sygnału zegarowego na podstawie sygnału zewnętrznego oraz za utrzymywanie przepływu informacji wewnątrz i na zewnątrz procesora zsynchronizowanego z tymi sygnałami.

# Rozdzielanie instrukcji i danych

Gdy sygnały docierają do wnętrza procesora, muszą być skierowane do odpowiednich bloków wewnętrznych. Porównując procesor z biurem, można sobie wyobrazić, że wygląda to tak, jak praca w recepcji.

Sygnały przychodzące stanowią mieszaninę dwóch rodzajów informacji, czyli instrukcji i danych. Odwołując się do innego porównania — z zakładem produkcyjnym — można sobie wyobrazić wyroby gotowe i materiały. W analogii biurowej byliby to pracownicy biura i klienci, którym pracownicy świadczą jakieś usługi.

Instrukcje umieszczane są w kolejce do dekodera instrukcji (w biurze byłaby to informacja: „Proszę podejść do okienka numer 3 i stanąć w kolejce"). Dane natomiast przetrzymywane są w poczekalniach, zwanych *rejestrami* (kolejna analogia biurowa: „Proszę tu zaczekać, wkrótce któryś z pracowników załatwi sprawę").

# Przewidywanie działań i ich wykonywanie

Instrukcje pobierane są z kolejki, interpretowane, a następnie uruchamiane. Zajmują się tym, odpowiednio, *jednostka pobierająca kod* (ang. *code prefetch unit*), *dekoder instrukcji* i *jednostka sterująca*. Te części procesora będziemy nazywać po prostu *blokiem obsługi instrukcji* (ang. *instruction handler*). Wykonuje on kilka zadań.

Procesory z rodziny x86 należą go kategorii CISC, w której występują instrukcje o różnej długości. Najkrótsze mają długość jednego bajta, zaś najdłuższe zawierają po kilkanaście bajtów.

Najpierw blok obsługi instrukcji sprawdza pierwszy bajt i na podstawie jego wartości określa, ile bajtów zawiera dana instrukcja. Następnie musi się upewnić, czy pozostałe bajty instrukcji są dostępne. Jeśli nie są, żąda pomocy od innych segmentów procesora w celu pobrania i dostarczenia bajtów z pamięci głównej.

Następnie blok obsługi instrukcji musi zdecydować, jakich danych wymaga przetwarzana instrukcja. Niektóre instrukcje same w sobie zawierają część lub wszystkie potrzebne im dane. Są to tzw. dane *bezpośrednie*. Inne instrukcje działają na danych obecnych w rejestrach procesora, jeszcze inne natomiast na zawartości komórek pamięci głównej; mogą też przesyłać tam wyniki swoich działań. Blok obsługi instrukcji musi zatem sprawdzić, czy wszystkie dane wymagane przez instrukcje są na swoim miejscu i czy są gotowe do użycia. W przeciwnym wypadku musi zaaranżować pobranie tych danych do procesora.

Na zakończenie tego ciągu czynności blok obsługi określa, co dana instrukcja nakazuje procesorowi, a następnie uaktywnia potrzebne części procesora, by zadania zostały wykonane. Najprostsze instrukcje odpowiadają elementarnym zadaniom realizowanym przez poszczególne bloki procesora. Współczesne układy z rodziny x86 mają specjalne obwody przeznaczone do wykonywania pewnych bardziej złożonych instrukcji w przypadku, gdy są one częściej używane. Dzięki temu takie instrukcje wykonywane są znacznie szybciej. W rzeczywistości, większość instrukcji rozpoznawanych przez procesor x86 wymaga podjęcia wielu działań przez różne elementy maszynerii liczącej w procesorze. Dekoder instrukcji wyszukuje te działania w wewnętrznej bibliotece zwanej *magazynem mikrokodu* i dostarcza pobrane stamtąd mikroinstrukcje do odpowiednich bloków procesora, gdzie będą one wykonywane.

# Rejestry są miejscem tymczasowego przechowywania danych

Rejestry odgrywają bardzo ważną rolę w każdym procesorze. Każdy komputer i każdy mikroprocesor, taki jak x86 (będący w istocie komputerem), musi dysponować miejscem na przechowywanie aktualnie przetwarzanych informacji.

Wiele układów z rodziny x86 różni się liczbą rejestrów i w wielu przypadkach mają one różne rozmiary. Pierwotne procesory 8086 i 8088 miały po 14 rejestrów przechowujących dane 16-bitowe. Pentium II ma znacznie więcej rejestrów, z których większość to rejestry 64-bitowe, chociaż występuje również kilka większych. W przypadku największych rejestrów (zwanych *buforami translacyjnymi przeglądania bocznego*, ang. *translation look-aside buffers*) jedynie część przechowywanych w nich bitów jest dostępna dla programu działającego w procesorze. Reszta ukryta jest przed tym programem, lecz dostępna dla samego procesora, co pomaga mu szybciej wykonać określone zadanie.

Jednym z powodów stosowania większej liczby rejestrów w ostatnich modelach procesorów z rodziny x86 jest to, że układy te zostały skomplikowane do tego stopnia, iż potrzebne są specjalne rejestry umożliwiające automatyczne testowanie procesora na zakończenie procesu produkcyjnego w fabryce. Producenci muszą być pewni, że układy opuszczające zakład działają poprawnie. Innym powodem zwiększenia liczby rejestrów jest obecność w tych procesorach dodatkowych bloków do *zarządzania systemem*, niezależnie od obecności bloków realizujących podstawowe zadania obliczeniowe. Bloki zarządzania systemowego realizują różnego rodzaju operacje specjalne, na przykład wyłączenie zasilania przy braku aktywności, co ma zaoszczędzić pobieraną energię.

Aby maksymalnie uprościć wykład, warto zająć się przede wszystkim zestawem rejestrów zdefiniowanych w procesorze 8086. Kolejne rejestry, dodane w kolejnych układach z rodziny x86, są bardzo podobne do pierwotnych, przynajmniej jeśli chodzi o zasadę działania. Omówimy je ogólnie, przy okazji wyjaśniania ich przeznaczenia. Czternaście rejestrów procesora 8086 (a także 8088) podzielono na pięć kategorii, które będą prezentowane w następnych sekcjach.

## Rejestry ogólnego przeznaczenia

Cztery rejestry, spośród czternastu istniejących w procesorze 8086, nazywa się rejestrami ogólnego przeznaczenia. Przede wszystkim stosowane są one do przechowywania wartości danych, które w określonym momencie są przetwarzane. Oznacza to, że można je dodawać, odejmować lub mnożyć. Można je także porównywać ze sobą. Liczba z rejestru ogólnego przeznaczenia może być łączona z liczbą przechowywaną w pamięci głównej. Oprócz tego, istnieje jeszcze wiele innych sposobów przetwarzania danych z tych rejestrów.

Jeśli wykonywana instrukcja wymaga tylko jednego bajta danych, bajt ten może być przechowywany w dowolnej części rejestru ogólnego przeznaczenia i może być dostępny bez względu na to, jaka jest zawartość drugiej części rejestru.

Rejestry tego typu mają proste nazwy, których używa się w instrukcjach języka programowania zwanego *asemblerem*. Nazwy te zostały zdefiniowane przez firmę Intel i wywodzą się z określeń jednobajtowych rejestrów stosowanych w mikroprocesorach 8080 (były one poprzednikami pierwszych układów z rodziny x86).

Program odwołujący się do rejestrów przechowujących liczby 16-bitowe posługuje się nazwami: AX, BX, CX i DX (w układach starszej generacji występowały rejestry o nazwach A, B, C i D, które mogły przechowywać tylko pojedyncze bajty). W tym kontekście dodaną literę X można traktować jako symbol *rozszerzenia* rejestru. Jeżeli chcemy odwołać się do dolnej połówki rejestru 16-bitowego, musimy używać nazw: AL, BL, CL i DL. Górne połówki rejestrów są nazywane: AH, BH, CH i DH.

W nowszych procesorach tej rodziny występują dłuższe rejestry, mające podobne nazwy — na przykład EAX lub EBX; oznaczają one długość dwukrotnie większą niż długość rejestrów AX lub BX. Pojedynczymi literami (A, B, C itd.) oznaczane są zatem rejestry 8-bitowe, zaś dodanie litery X w nazwie wskazuje na rejestr 16-bitowy. Jeśli za daną literą występuje dodatkowo litera H lub L, to oznacza ona górną lub dolną połówkę rejestru 16-bitowego. X występujące za literowym symbolem rejestru oraz E występujące przed takim znakiem oznacza rejestr 32-bitowy podobnego przeznaczenia jak krótszy rejestr o takim samym symbolu.

Można sądzić, że oznaczenia te zostały wybrane dlatego, że rozpoczynają się czterem początkowymi literami alfabetu. Jest w tym pewna słuszność, lecz dodatkowo mają one znaczenie mnemoniczne (czyli dłuższe niż sugerują oznaczenia literowe). W tych czterech rejestrach, mimo iż zwane są one rejestrami ogólnego przeznaczenia, występują pewne ograniczenia dotyczące ich zastosowania; ponadto rejestry te posiadają własną specyfikę.

- Rejestr AX (czyli AH w połączeniu z AL) jest najczęściej używany jako *akumulator* (ang. *accumulator*), czyli miejsce, do którego trafia końcowy wynik obliczeń. Można na przykład dodać wartość umieszczoną w innym rejestrze lub w jakiejś komórce pamięci do wartości umieszczonej w rejestrze A. Wynik pojawi się w tym rejestrze i zastąpi pierwotną wartość.
- Rejestr BX jest często używany do przechowywania części adresu zwanej segmentem. Użycie go w takim kontekście oznacza, że jest on traktowany jako rejestr *bazy* (ang. *base*), ponieważ adres segmentu oznacza początek (czyli bazę) danego obszaru pamięci. Rejestr BX (a także BL lub BH) może być również użyty do przechowywania danych innego rodzaju.
- Rejestr CX jest zazwyczaj przeznaczony do przechowywania liczby wskazującej na to, ile razy wykonana została dana operacja. Jeśli osiąga ona określoną wartość docelową, program musi wykonać skok do innego miejsca w kodzie. Konieczne porównanie i skok w jednej instrukcji jest możliwe jedynie wtedy, gdy *licznik* (ang. *counter*) wykonanych operacji przechowywany jest w tym właśnie rejestrze.
- Rejestr DX (a także DH i DL) nazywany jest zazwyczaj rejestrem *danych* (ang. *data*). Niekiedy jest on także wykorzystywany do przechowywania części adresu portu, a bywa także stosowany w połączeniu z rejestrem AX do przechowywania liczb 32-bitowych (na przykład wyników mnożenia dwóch liczb 16-bitowych).

## Rejestr znaczników

Jeden specjalny rejestr zwany jest *rejestrem znaczników* (jego mnemoniczny skrót to po prostu FLAGS). Jest on miejscem przechowywania szesnastu pojedynczych bitów, z których każdy oznacza jakiś stan. Najprostszym przykładem działania jednego ze znaczników jest sygnalizacja wyniku przeprowadzonego ostatnio porównania dwóch bajtów (tzn. określenie, czy były one równe, czy też nie). Inne znaczniki sygnalizują znak wyniku ostatniej operacji arytmetycznej (dodatni, ujemny lub zero, a także wskazują, czy nie nastąpiło przepełnienie rejestru). Jeszcze inne informują o stanie procesora, na przykład poprzez podanie odpowiedzi na następujące pytania: czy procesor reaguje na zewnętrzne sygnały przerwań, czy je ignoruje? Czy procesor działa w trybie krokowym? Czy ciągi bajtów przetwarzane są „w górę", czy „w dół"?

Nowsze procesory z rodziny x86 mają dłuższe rejestry zarówno dla danych, jak i dla znaczników, dlatego mogą one wyrażać więcej warunków właśnie za pomocą znaczników.

Wartości znaczników decydują o zachowaniu się procesora podczas wykonywania tzw. „instrukcji warunkowych", które będą omówione w tym rozdziale nieco dalej.

## Wskaźnik instrukcji

Kolejny rejestr specjalnego przeznaczenia przechowuje adres komórki w pamięci głównej, w której mieści się aktualnie wykonywana instrukcja. Nazywa się on, zgodnie ze swoim przeznaczeniem, *wskaźnikiem instrukcji* (ang. *instruction pointer*) i ma

mnemoniczny skrót IP. Wartość w tym rejestrze *niejawnie* adresuje komórkę pamięci, w której znajduje się wykonywana instrukcja. Aby uzyskać rzeczywisty adres, należy tę wartość połączyć w odpowiedni sposób z wartością przechowywaną w innym rejestrze, zwanym *rejestrem segmentu kodu* (ang. *code segment register*). Rejestr segmentu kodu przedstawiony będzie nieco później.

Wartość znajdująca się we wskaźniku instrukcji zmienia się w dwojaki sposób. Wyróżnia się *normalny przebieg* sterowania oraz *rozgałęzienia*.

## Normalny przebieg sterowania

Dopóki wykonywana instrukcja nie nakaże innego działania, wartość we wskaźniku instrukcji (zwanym również niekiedy *licznikiem programu*, ang. *program counter*) po zakończeniu operacji automatycznie przyrasta o długość tej instrukcji. Dzieje się tak dlatego, że w większości przypadków następna instrukcja, która ma być wykonana, znajduje się w pamięci tuż za bieżącą. Złożoność operacji powiększania zawartości wskaźnika instrukcji w procesorach z rodziny x86 wynika z tego, że są one procesorami typu CISC, czyli ich instrukcje mają różną długość. Ta zmienność wymusza powiększanie zawartości wskaźnika instrukcji o różne wartości.

## Instrukcje rozgałęzień

Powyższe stwierdzenia nie obowiązują w około 10% przypadków. Dotyczy to sytuacji, w których bieżąca instrukcja może nakazać procesorowi pobranie następnej instrukcji z jakiegoś innego miejsca niż zazwyczaj. Instrukcja taka nazywana jest instrukcją rozgałęzienia lub instrukcją *skoku*. Rozróżnia się dwa rodzaje takich instrukcji: *bezwarunkowe instrukcje rozgałęzienia*, w których położenie następnej komórki jest zawsze inne niż podczas zwykłego przebiegu sterowania i *warunkowe instrukcje rozgałęzienia*, w których podejmuje się decyzję o skoku do innej lokalizacji albo normalnym przejściu do kolejnej. Działanie instrukcji warunkowych określane jest na podstawie wartości pewnych bitów w rejestrze znaczników.

Innym sposobem zmiany przebiegu sterowania jest wykonanie instrukcji wywołującej podprogram. Umożliwia ono faktyczne przerwanie, na krótki czas, wykonywania programu i przywołanie, zamiast niego, innego programu (mówiąc dokładniej, chodzi o niewielkie programy wbudowane w innych miejscach programu głównego). Gdy taki miniprogram zakończy swoje zadanie, wykona instrukcję powrotu. Spowoduje ona, że procesor podejmie wykonywanie przerwanego, pierwotnego programu i zacznie swe działanie od instrukcji umieszczonej bezpośrednio za instrukcją wywołania podprogramu.

Taka strategia wywoływania podprogramów pozwala programistom zaoszczędzić ich wysiłki. Mogą oni napisać raz jakiś podprogram i wykorzystywać go z wielu różnych miejsc w większym programie, bez konieczności wstawiania w te miejsca wszystkich jego instrukcji. W rzeczywistości, oprócz zmniejszenia pracochłonności, możliwe jest również ograniczenie rozmiarów głównego programu, co w niektórych sytuacjach jest ogromną zaletą.

## Inne rejestry wskaźników

Jeszcze dwa spośród czternastu rejestrów oryginalnego procesora 8086 należą do grupy rejestrów wskaźników. Jeden z nich nazywany jest *rejestrem wskaźnika bazy* (ang. *base pointer register*, w skrócie *BP*), natomiast drugi *rejestrem wskaźnika stosu* (ang. *stack pointer register*, w skrócie *SP*). Każdy z nich przechowuje liczbę funkcjonującą jako część segmentowa adresu, jeśli procesor działa w trybie rzeczywistym. Podczas pracy w trybie chronionym liczba ta nazywana jest selektorem, lecz spełnia podobną rolę. Wykorzystywana jest jako wskaźnik obszaru pamięci używanego jako stos. Co to naprawdę oznacza, zostanie wyjaśnione w dalszych rozważaniach zawartych w tym rozdziale.

## Rejestry indeksowe

Dwa rejestry przeznaczone zostały do obsługi przemieszczania ciągów danych w pamięci (wielobajtowych sekwencji o dowolnej długości). Pierwszy z nich, zwany *źródłowym rejestrem indeksowym* (ang. *source index register*, w skrócie *SI*), może na przykład przechowywać adres początku ciągu, który ma być przemieszczony. Drugi, zwany *docelowym rejestrem indeksowym* (ang. *destination index register*, w skrócie *DI*), zawiera adres, pod którym ma zostać umieszczony przesuwany ciąg danych. Liczba bajtów, które mają być przesunięte, przechowywana jest zazwyczaj w rejestrze CX, pełniącym w takiej sytuacji rolę licznika. Oprócz zastosowań związanych z przemieszczaniem ciągów danych, rejestry indeksowe mogą być również wykorzystywane do wskazywania miejsc w tablicy zawierającej dane liczbowe, a także do wykonywania wielu innych operacji.

## Rejestry segmentowe

Ostatnia kategoria rejestrów to *rejestry segmentowe* (ang. *segment registers*). W procesorze 8086 są cztery takie rejestry. Mają one specjalne przeznaczenie, ponieważ stosowane są wyłącznie do obliczania adresów. W następnym punkcie wyjaśnione zostanie szczegółowo, na czym polegają niektóre sposoby zastosowania wartości przechowywanych w rejestrach segmentowych. Warto przedstawić teraz nazwy tych rejestrów i rodzaje przechowywanych przez nie adresów.

Pierwszy z nich nazywany jest *rejestrem segmentu kodu* (ang. *code segment register*, w skrócie CS). Przechowuje on wartość, która po połączeniu z wartością wskaźnika instrukcji określa adres następnej instrukcji przeznaczonej do wykonania.

Następny nazywany jest *rejestrem segmentu danych* (ang. *data segment register*, w skrócie DS). Zazwyczaj rejestr DS używany jest jako wskaźnik obszaru pamięci, w którym przechowywane są wartości danych. Może on być połączony na przykład z liczbami z rejestrów BX, SI lub DI, dzięki czemu istnieje możliwość określenia poszczególnego bajta lub słowa danych.

Trzeci rejestr to *dodatkowy rejestr segmentu* (ang. *extra segment register*, w skrócie ES). Zgodnie ze swoją nazwą, może on być dodatkowo wykorzystany przez programistę do dowolnego celu jako rejestr segmentowy, chociaż najbardziej naturalnym jego zastosowaniem są operacje na łańcuchach.

Ostatni określany jest jako *rejestr segmentu stosu* (ang. *stack segment register*, w skrócie SS). Wartość w nim przechowywana łączona jest z wartością w rejestrze wskaźnika stosu (SP) i wskazuje słowo danych aktualnie przetwarzanych na stosie. Więcej informacji na temat stosu można znaleźć nieco dalej. W niektórych instrukcjach rejestr SS może także funkcjonować w połączeniu z rejestrem BP.

# Obliczanie adresów

Podczas omawiania wskaźnika instrukcji (IP) podano, że jego zawartość sama z siebie nie określa miejsca w pamięci, w którym przechowywana jest instrukcja.

Procesory z rodziny 8086 przy każdym odwołaniu do pamięci (w celu pobrania z niej danych albo wpisania ich do niej) muszą wykonać mniej lub bardziej skomplikowaną operację łączenia dwóch rejestrów, by określić miejsce, którego to odwołanie dotyczy. Często procesor musi użyć wartości nie tylko z dwóch rejestrów lub większej ich liczby, lecz posłużyć się trzema różnymi tablicami danych, aby określić rzeczywisty adres komórki w pamięci (tak się dzieje w trybie chronionym).

## Od abstrakcji do rzeczywistości

Istnieje kilka powodów takiej komplikacji przy obliczaniu adresów w procesorze x86. Zapewne najważniejszym z nich jest to, że procesory z tej rodziny posługują się kilkoma rodzajami przestrzeni adresowej.

Na poziomie fizycznym (czyli oznaczającym to, co rzeczywiście dzieje się w komputerze) komórki pamięci są adresowane przez napięcia na przewodach doprowadzonych do każdego modułu lub układu scalonego. Sygnały dostarczane są z odpowiednich wyprowadzeń procesora. Poprzez obserwację napięcia na wyprowadzeniach procesora i oznaczanie jedynkami stanów wysokich, a zerami niskich, otrzymujemy liczbę binarną, którą nazywamy *adresem fizycznym*.

### Jakimi adresami posługują się programy?

Programy działające w komputerze PC nie używają adresów fizycznych, ponieważ nie pozwala na to konstrukcja rodziny procesorów x86. Programy muszą korzystać co najmniej z jednego poziomu pośredniego. Rzeczywisty adres fizyczny tworzony jest w wyniku połączenia dwóch lub więcej liczb zgodnie z pewnymi regułami adresowania. Ze względu na złożoność tego procesu, będziemy się tu posługiwać kilkoma określeniami: *adres logiczny* (zwany także *adresem wirtualnym*), *adres liniowy* i *adres fizyczny*. Każde z tych pojęć będzie wyjaśnione oddzielnie.

### Obliczanie adresów fizycznych w trybie rzeczywistym

Poznaliśmy już najprostszą metodę obliczania adresu komórki pamięci, stosowaną w procesorze x86. Adresy w trybie rzeczywistym wyrażone są w programie jako adresy logiczne, składające się z dwóch części, czyli z segmentu i offsetu.

**Szczegółowe informacje o adresowaniu**

Sygnały z niektórych wyprowadzeń procesora kierowane są bezpośrednio do modułów pamięci. Po drodze przechodzą one wprawdzie przez układy scalone wzmacniaczy, lecz nie są mieszane z innymi sygnałami.

Sygnały z pozostałych wyprowadzeń adresowych procesora są łączone w obwodach zwanych dekoderami adresów, decydujących o tym, który z układów lub modułów pamięci ma być w danym momencie uaktywniony. Wszystkie moduły (lub układy scalone) otrzymują wszystkie pozostałe sygnały, lecz moduły aktywne korzystają z tych sygnałów do określenia komórki pamięci, która ma być udostępniona.

Procesor z 64 liniami danych łączy się z ośmioma bajtami pamięci na raz. Oznacza to, że trzy mniej znaczące bity każdego adresu mogą być pomijane przy wskazywaniu położenia komórki w pamięci głównej. Są one wykorzystywane wewnątrz procesora do określenia, który z ośmiu odczytanych bajtów ma być użyty w dalszych operacjach. Wiedząc, że 2^3 równa się 8, a 2^25 to ponad 30 milionów, można stwierdzić, że do zaadresowania każdej komórki wewnątrz modułu pamięci o pojemności 32 MB wystarczają dokładnie 22 linie. Oprócz tego, możliwe jest (i stosuje się to) wykonanie wyprowadzeń adresowych w module w taki sposób, by pełniły one podwójną rolę. W określonej chwili służą one do odczytu adresu wiersza, a w innej do odczytu adresu kolumny. Producenci układów mogą dzięki temu zaoszczędzić w tym przypadku aż 11 wyprowadzeń oraz dodatkowo jedno wyprowadzenie sygnalizujące, czy linie adresowe zawierają jedenaście początkowych bitów adresu (od A3 do A13), czy sześć pozostałych (od A14 do A24).

Linie sygnalizujące w module pamięci (oznaczone jako CE, co jest skrótem od angielskiej nazwy *chip enable*) służą do włączania i wyłączania układów scalonych w module. Stan napięcia na takiej linii określa, czy moduł odpowiada na sygnały w pozostałych liniach wejściowych, czy też przechodzi w stan swoistego uśpienia, przez co ignoruje sygnały na wszystkich wejściach i nie wytwarza żadnych sygnałów wyjściowych.

Dekoder adresów pamięci pobiera sygnały z pozostałych linii adresowych (w tym przypadku z siedmiu linii o oznaczeniach od A25 do A31). Wystarcza to do wskazania dowolnego ze 128 różnych „banków" pamięci. Taki hipotetyczny komputer ma jednak tylko dwa banki pamięci. Każdy z nich ma pojemność 32 MB, co wystarcza do uruchomienia każdego współczesnego programu (jednak na pewno nie pozwala na uruchomienie kilku takich programów jednocześnie). Dekoder adresów pamięci musi sprawdzać sygnały na wszystkich siedmiu liniach, by nie uaktywniać modułów pamięci aż do momentu, w którym we wszystkich liniach, oprócz pierwszej, pojawi się zerowy stan napięcia. Poziom napięcia na pierwszej linii decyduje o tym, który z dwóch banków pamięci zostanie uaktywniony.

Wartość segmentu jest mnożona przez 16 (w zapisie szesnastkowym oznacza to po prostu przesunięcie o jedno miejsce w lewo), a następnie dodawana do wartości offsetu. Dopiero wtedy uzyskuje się adres fizyczny. Istnieje 65536 potencjalnych wartości numeru segmentu i taka sama liczba możliwości dla offsetu. Oznacza to, że poszczególne wartości segmentu wskazują oddzielne obszary pamięci o rozmiarze 64 kB. Wartość offsetu wskazuje konkretne miejsce w tym obszarze.

Ważną sprawą jest zrozumienie, że w trybie rzeczywistym istnieje wiele adresów logicznych (rozumianych jako pary liczb 16-bitowych, po jednej dla segmentu i po jednej dla offsetu), wskazujących na określony adres fizyczny. Przy powiększaniu wartości segmentu o 1 i jednoczesnym zmniejszeniu wartości offsetu o 16 adres fizyczny po prostu się nie zmieni. A zatem adres logiczny 01A0:4C67h jest dokładnie tym samym, co adres 01A1:4C57h (mała litera *h* występująca za adresem oznacza tu zapis szesnastkowy). Zwykle wystarczają dwie czterocyfrowe liczby oddzielone dwukropkiem, lecz

chcemy tu wyraźnie podkreślić rodzaj zapisu. Powinno być także jasne, że offset równy 4C57h jest o 16 mniejszy niż 4C67h, ponieważ te liczby różnią się tylko o jeden na pozycji „szesnastkowej". W opisie *trybu chronionego* zostanie pokazane, że może istnieć jeszcze więcej adresów logicznych wskazujących ten sam adres fizyczny.

## Obliczanie adresów fizycznych w trybie chronionym

Wszystkie procesory z rodziny x86 (z wyjątkiem najstarszych) mogą działać w więcej niż jednym trybie. Wszystkie rozpoczynają pracę w trybie rzeczywistym i posługują się w nim opisaną wyżej metodą obliczania adresów fizycznych.

Po wstawieniu do pamięci kilku specjalnych tablic z danymi, każdy procesor z rodziny x86, nowszy niż 80186, może rozpocząć pracę w trybie chronionym. Są trzy odmiany takiego trybu pracy. Liczby przechowywane w rejestrach segmentowych nie są wówczas po prostu mnożone i dodawane do wartości offsetu w celu uzyskania adresu komórki pamięci. Dzieje się to zupełnie inaczej niż w trybie rzeczywistym i dlatego liczby z rejestrów segmentowych nazywane są w trybie chronionym *selektorami*, a nie wartościami segmentów.

W przeciwieństwie do wartości segmentu, która wskazuje w pamięci określony obszar o rozmiarze 64 kB, selektor definiuje jeden wiersz w strukturze danych zwanej *tablicą deskryptorów*. W tym wierszu zawarte są trzy informacje o „wybranym" obszarze pamięci. Pierwsza precyzuje początek obszaru, druga jego rozmiar, zaś trzecia jest w istocie kombinacją liczb określającą pewne specyficzne właściwości (zwane *atrybutami dostępu*) tego obszaru.

Określenie *segment* nadal odnosi się do obszaru pamięci wskazywanego przez wartość selektora w rejestrze segmentu, lecz proces przejścia od wartości selektora do położenia segmentu w przestrzeni adresowej jest tu bardziej zagmatwany i dlatego określenia tego nie należy stosować zamiennie z selektorem.

Taka strategia obliczania adresu fizycznego ma trzy zalety. Po pierwsze, dowolna wartość selektora może wskazywać na dowolny obszar pamięci, bowiem nie ma związku między dwoma obszarami pamięci wskazywanymi przez selektory różniące się o jakąś ustaloną wartość. Druga korzyść wynika z faktu, że rozmiar segmentu definiowanego przez wartość selektora nie jest stały. Może on być równy jednemu bajtowi w niektórych przypadkach, a w innych osiągać aż 4 GB. Po trzecie, obecność atrybutów dostępu w tablicy deskryptorów umożliwia procesorowi kontrolę sposobu dostępu do obszaru pamięci wskazywanego przez dany selektor.

Określenie *adres logiczny* nadal oznacza kombinację dwóch liczb w zapisie szesnastkowym, oddzielonych dwukropkiem. Pierwsza liczba (zawsze dwubajtowa, czyli składająca się z czterech cyfr szesnastkowych) jest teraz nazywana *selektorem*, a nie *segmentem*. Druga liczba (cztero- lub ośmiobajtowa) nadal nazywa się *offsetem* i oznacza liczbę bajtów zawartych w segmencie od jego początku do wskazywanego miejsca.

Wspomniano już o trzech odmianach trybu chronionego. Pierwsza wprowadzona była w procesorze 286, zatem nazywana jest *trybem chronionym procesora 286*. Odmiany druga i trzecia wprowadzone zostały w procesorze 386 i nazywa się je odpowiednio: *trybem chronionym procesora 386* i *trybem wirtualnego procesora 86*.

Jedyna różnica między trybem chronionym 286 (który nadal obsługiwany jest nawet przez najnowsze procesory z rodziny x86 w celu zapewnienia wstecznej zgodności) a trybem chronionym 386 (który wykorzystywany jest prawie przez cały czas przez nowoczesne programy) to na ogół rozmiar liczb przechowywanych w tablicy deskryptorów. Wiążą się z tym różnice rozmiaru obszarów pamięci opisywanych przez te deskryptory oraz różna liczba atrybutów dostępu do tych obszarów.

## Obliczanie adresów fizycznych w trybie wirtualnego procesora 86

Tryb wirtualnego procesora 86 jest bardzo specyficznym trybem. Jest on uruchamiany zawsze w połączeniu z trybem chronionym 386, zaś uruchamiany program traktuje procesor jak 8086 działający w trybie rzeczywistym. Faktycznie procesor działa wówczas w trybie chronionym, a system operacyjny działający w trybie chronionym 386 służy jako tzw. *wirtualny monitor 86*.

Zaletą takiego trybu pracy procesora jest możliwość uruchamiania starszych programów napisanych dla systemu DOS przy jednoczesnym korzystaniu z zalet pracy w trybie chronionym. Zagadnienia te szerzej zaprezentowane zostały w następnym podrozdziale. Oprócz tego, możliwe jest uruchomienie jednocześnie kilku takich programów, każdego we własnym środowisku DOS, ale bez możliwości wymiany informacji między nimi. Taka sytuacja występuje wtedy, gdy uruchomi się starą aplikację DOS w oknie (lub w trybie pełnoekranowym) w systemie Windows 3.x, Windows 9x lub Windows NT. Rozruch komputera w trybie MS-DOS pozwala na jego pracę w trybie rzeczywistym, co jest możliwe we wszystkich wersjach Windows z wyjątkiem NT.

## Stronicowanie komplikuje obliczenia adresów

W przypadku procesora 386 i jego następców z rodziny x86 występuje jeszcze dodatkowa komplikacja przy obliczaniu adresów komórek pamięci. Nazywa się to *stronicowaniem* (ang. *paging*) i to właśnie stanowi podstawę trzeciego rodzaju adresowania. Wartość selektora w połączeniu z offsetem nadal nazywana tu będzie adresem logicznym, który jest faktycznie używany w programach. Po dodaniu offsetu do adresu bazowego otrzymuje się w wyniku selektor wskazujący tzw. *adres liniowy*. W procesorze 80286 oznaczał on to samo, co adres fizyczny, lecz w późniejszych konstrukcjach są to już różne wartości.

Należy zwrócić uwagę na to, że adres fizyczny jest określony przez napięcia na wyprowadzeniach adresowych procesora. Jest to tylko jedno z wielkiej liczby miejsc w *fizycznej przestrzeni adresów pamięci*.

Jeśli stosowane jest stronicowanie, adres liniowy oznacza po prostu abstrakcyjną lokalizację w pewnej hipotetycznej przestrzeni adresów pamięci. Przejście od adresu liniowego do adresu fizycznego odbywa się podobnie jak przejście od wartości selektora do adresu segmentu, chociaż jest to proces nieco bardziej skomplikowany.

32-bitowy adres liniowy dzielony jest na trzy części. Dziesięć najbardziej znaczących bitów wykorzystuje się jako indeks katalogu. Następne 10 bitów służy jako indeks tablicy, zaś 12 najmniej znaczących bitów wykorzystywanych jest jako offset.

Wiele osób zetknęło się prawdopodobnie ze *stronicowaniem* w kontekście operacji zapisu na dysk. Taki proces oznacza, że posiada się zbyt mało fizycznej pamięci RAM, by pomieścić cały kod i dane wymagane przez uruchomioną aplikację.

Procesor użyty w „inteligentnym" systemie operacyjnym, takim jak Windows 95/98 lub Windows NT, może zapisywać i pobierać z dysku zawartość pamięci poprzez skorzystanie z takiego samego procesu, jaki używany jest przy odwzorowywaniu adresów logicznych na adresy fizyczne.

Dzięki temu można uruchamiać jednocześnie więcej programów niż pozwalałaby pojemność fizycznej pamięci RAM. W systemach operacyjnych obsługujących stronicowanie w operacjach dyskowych mówi się o *pamięci wirtualnej*, ponieważ mogą one korzystać zarówno z pamięci wirtualnej, jak i z pamięci fizycznej.

## Wymuszanie trybu chronionego

Określenie „tryb chroniony" jest bardzo sugestywne. Wskazuje na to, że coś jest w jakiś sposób chronione, lecz co to naprawdę znaczy? Jedyną przyczyną stosowania trybu chronionego jest wykorzystanie go w prosty sposób do pozyskania możliwości wykonywania wielu zadań jednocześnie. W każdym systemie wielozadaniowym (myślimy o maszynie z systemem Windows, na której działa jednocześnie kilka programów) uruchomione programy działają tak, jakby były jedynymi działającymi w danym momencie. Chociaż pracują na tym samym komputerze, nie zakłócają wzajemnie swojej pracy — tak jest przynajmniej w założeniach systemu.

Aby wszystkie programy mogły pracować we właściwy sposób, trzeba powstrzymywać je przed wejściem w konflikt z pozostałymi programami. Do takiej ochrony musi być zaangażowany jakiś program nadrzędny — jest nim właśnie system operacyjny (na przykład Windows).

To jeszcze nie wszystko. Gdy system operacyjny w komputerze osobistym wywoła aplikację w trybie rzeczywistym, można za jej pomocą robić wszystko, co się zechce. Można zapisywać i odczytywać dane z dowolnego obszaru pamięci, wysyłać informacje do dowolnego portu lub pisać na całym ekranie. System operacyjny nie posiada odpowiednich środków, by powstrzymać takie działania. Dlatego właśnie Intel wydzielił różne tryby chronione w produkowanych przez siebie procesorach x86. Idea nie była zupełnie nowa, ponieważ zapożyczono ją z dużych komputerów (podobnie jak zapis stron pamięci na dysk), lecz po raz pierwszy zastosowano ją w mikrokomputerach.

Podstawową sprawą jest to, że każdemu programowi przydzielono pewien poziom ochrony i dlatego z pomocą uruchomionego programu można wykonywać tylko to, co jest dozwolone na danym poziomie ochrony. Istnieją cztery poziomy ochrony, oznaczane numerami od 0 do 3. Programom przydziela się jeden z tych sygnałów ostrzegawczych. Rdzeń systemu operacyjnego musi działać na poziomie zerowym i zazwyczaj w trybie chronionym jest to jedyny zestaw programów mający pozwolenie na takie działanie. Wszystkie aplikacje działają na poziomie o numerze 3. Dotychczas nie ma systemu operacyjnego, który korzystałby z poziomów 1 lub 2, mimo że występują one w każdym procesorze x86 w oczekiwaniu na moment, kiedy jakiś sprytny programista znajdzie dla nich zastosowanie.

Oprócz tego, każdy segment pamięci (tutaj jest to obszar określony przez zawartość różnych wierszy w jakiejś tablicy deskryptorów) ma pewne atrybuty dostępu. Tylko te programy, które dysponują odpowiednim poziomem ochrony, mają prawo do zmiany atrybutów, co powoduje, że mogą być wykonane tylko te operacje, na które pozwalają poszczególne atrybuty dostępu.

Przestrzeganie wszystkich opisanych tutaj reguł wymuszane jest przez procesor. Dlatego właśnie jest o tym mowa w rozdziale poświęconym wewnętrznej architekturze procesora, a w szczególności przy okazji omawiania sposobów obliczania adresów. Układy obliczające adresy wewnątrz procesora zapewniają również, że instrukcje wykonywane są bez naruszania żadnej z wymienionych reguł.

Naruszenie którejś z nich określane jest jako *wyjątek*. Wyjątki klasyfikowane są jako *błędy*, *pułapki* i *przerwania zadań*. Gdy wystąpi jakiś wyjątek, procesor przerywa zadanie wykonywane przez program, wyłącza się i zajmuje się innym działaniem. Odbywa się to bardzo podobnie do reakcji na przerwania zewnętrzne lub programowe, których mechanizmy omówione będą w tym samym rozdziale nieco dalej.

Najbardziej niesławnym wyjątkiem jest tzw. *Ogólny błąd ochrony typu 13* (ang. *General Protection Fault*). Powoduje on wyświetlenie na ekranie komunikatu o błędzie i prowadzi do zamknięcia programu, który spowodował taką sytuację. Można sądzić, że system Windows bardzo często się zawiesza. Tak jest naprawdę, lecz nie zdarza się to tak często, jak mogłoby się zdarzać przy braku wbudowanego w każdy procesor x86 systemu wymuszania ochrony.

# Jednostka arytmetyczno-logiczna

Na tym poziomie wygodnie jest przedstawić inny zespół logicznych części procesora, do których zaliczamy *jednostkę arytmetyczno-logiczną* (ang. *Arithmetic-Logic Unit*, w skrócie *ALU*) i kilka innych elementów mających podobne zadania. ALU dodaje, odejmuje, mnoży i dzieli liczby całkowite. Może ona także porównywać dwie liczby, aby określić, czy są one równe, a jeśli nie są — wskazać, która z nich jest większa.

## Niektóre proste operacje na liczbach całkowitych

Oprócz prostych operacji arytmetycznych (dodawania, odejmowania, mnożenia i dzielenia) i przeprowadzania porównań logicznych, jednostka arytmetyczno-logiczna może także dokonywać w różny sposób przesunięć bitów. Wyobraźmy sobie 16-bitowy rejestr jako 16 oddzielnych bitów, siedzących na krzesłach ustawionych w szeregu. Pierwszy rodzaj przesunięcia, zwany *przesunięciem arytmetycznym*, polega na tym, że wszystkie bity przesiadają się o jedno miejsce w lewo lub w prawo. Końcowy bit, dla którego zabraknie miejsca, jest po prostu tracony, zaś na opróżnione miejsce wstawiany jest bit o wartości zerowej.

Inny rodzaj przesunięcia nazywany jest *przesunięciem cyklicznym*. Polega ono na tym, że bit tracący miejsce na jednym końcu szeregu wstawiany jest na przeciwny jego koniec. Taki rodzaj przesunięcia wykorzystywany jest w operacjach mnożenia liczb oraz w niektórych operacjach logicznych, co sprawia, że jest niezwykle istotny dla programistów.

## Specjalne układy przeznaczone do bardziej skomplikowanych zadań

Dotychczas omawiane były ważne elementy spotykane w starszych konstrukcjach procesorów z rodziny x86. W najnowszych modelach spotyka się te same części, ale działają one dużo szybciej. Oprócz tego, występują tam także inne, nowe części, wyspecjalizowane w wykonywaniu dodatkowych zadań.

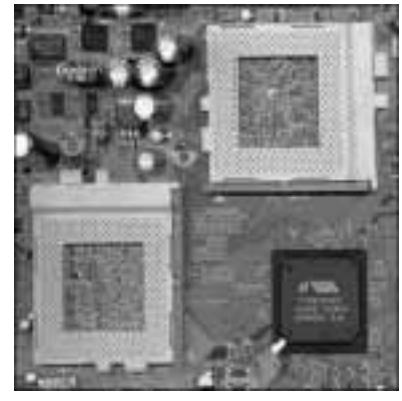
### SMP

Skrót ten oznacza *symetryczne przetwarzanie wieloprocesorowe* (ang. *Symmetric Multiprocessing*). Jest to specjalny rodzaj architektury komputera o dużej wydajności, w której poszczególne procesory są równocześnie udostępniane różnym zadaniom. „Symetryczność" oznacza, że dane zadanie może być przydzielone dowolnemu procesorowi, który nie jest akurat zajęty. Oprócz tego, w konstrukcji takiej mówi się również o skalowalności, czyli o możliwości instalowania dodatkowych procesorów, jeśli budowa płyty głównej na to pozwala.

Aby komputer zbudowany zgodnie z architekturą SMP mógł działać, muszą być spełnione różne wymagania. Po pierwsze, zestaw pomocniczych układów scalonych (tzw. *chipset*) na płycie głównej musi obsługiwać wiele procesorów. Nie wszystkie zestawy to potrafią. Po drugie, również procesor musi być przystosowany do pracy w systemie SMP. Układy takie jak: Celeron, Duron i kilka odmian procesora Athlon nie umożliwiają pracy w takiej konfiguracji. Po trzecie, system operacyjny musi wykrywać i wykorzystywać dodatkowe procesory. Nie można uruchomić oddzielnie kilku systemów operacyjnych na oddzielnych procesorach, ponieważ musi je obsługiwać jeden system. Ponadto, teoretyczny wzrost wydajności spowodowany zastosowaniem wielu procesorów okazuje się w praktyce nieco mniejszy, ponieważ procesory współużytkują pamięć i zasoby dyskowe.

Unix, Linux, Windows NT i Windows 2000 oraz Windows XP obsługują systemy wieloprocesorowe, natomiast Windows 9x i Windows ME nie mogą być wykorzystywane w tego typu konfiguracjach.

Na rysunku 15.7 pokazano fragment wieloprocesorowej płyty głównej z dwoma gniazdami procesorów.



**Rysunek 15.7.** *Ta płyta główna jest jedną z tych, które umożliwiają użycie wielu procesorów*

## Operacje zmiennoprzecinkowe

Począwszy od procesora 486DX, w każdym układzie z rodziny x86 znajdują się oddzielne układy do przetwarzania liczb zmiennoprzecinkowych. Jedną grupę stanowi zestaw rejestrów o długości 80 bitów przeznaczonych specjalnie do przechowywania liczb zmiennoprzecinkowych. Druga grupa obejmuje zestaw bramek logicznych (oraz specjalnych instrukcji mikrokodu, które uaktywniają te bramki), połączonych w taki sposób, by rzeczywiście wykonywały operacje na liczbach zmiennoprzecinkowych.

Instrukcje zmiennoprzecinkowe dodane zostały do zestawu instrukcji procesora wcześniej, jeszcze zanim wykonujące je układy pojawiły się wewnątrz procesora. Początkowo procesor przechwytywał te instrukcje i wywoływał specjalny program emulacyjny do ich wykonania. Później procesor mógł przechwytywać instrukcje zmiennoprzecinkowe i przekazywać je do oddzielnego układu, zwanego koprocesorem numerycznym. Obecnie procesor ma już wystarczającą wydajność, by takie instrukcje mógł przetwarzać sam.

## Rozszerzenia zestawu instrukcji x86: MMX, 3DNow! oraz KNI

Wprowadzenie stronicowania i dodatkowych trybów chronionych w procesorze 386 stanowiło prawdziwie rewolucyjny krok. Zestaw instrukcji x86 został wówczas wzmocniony przez duży pakiet instrukcji umożliwiających wykorzystanie nowych cech funkcjonalnych. Późniejsze zmiany konstrukcyjne miały już mniejszy wpływ na zawartość zestawu instrukcji.

Jeszcze całkiem niedawno wszystkie dodatki pojawiające się w zestawie instrukcji x86 (dotyczy to na przykład instrukcji specyficznych dla procesorów 486, początkowych modeli Pentium i Pentium Pro) wprowadzały drobne poprawki do poprzedniego zestawu. W ciągu ostatnich kilku lat w zestawie instrukcji x86 pojawiły się jednak dwie znaczące nowości oraz nastąpiło ich przyspieszenie. Interesujący może być fakt, że niektóre z tych dodatków nie były opracowane przez firmę Intel, lecz przez konkurujących z nią producentów klonów x86. Doszli oni do wniosku, że klienci będą kupować ich produkty nie tylko z powodu niskiej ceny, lecz także dlatego, że będą one — co chcieli osiągnąć poprzez wprowadzane zmiany — lepsze niż produkty Intela.

Pierwsze ulepszenie polegało na dodaniu tzw. „rozszerzeń multimedialnych". Firma Intel nazwała je w skrócie MMX. Jest to grupa instrukcji typu *jedna instrukcja, wiele danych* (ang. *single instruction, multiple data*, w skrócie SIMD). Zasada ich działania sprowadza się do tego, że Pentium MMX, Pentium II i nowsze modele procesorów x86 Intela do rejestrów zwykle używanych w operacjach zmiennoprzecinkowych mogą załadować *wiele danych* będących liczbami całkowitymi, a następnie wykonywać niektóre operacje równocześnie na wszystkich pozycjach danych.

Instrukcje MMX mogą wykorzystywać 64 bity w 80-bitowych rejestrach zmiennoprzecinkowych do przechowywania ośmiu liczb 8-bitowych, czterech liczb 16-bitowych albo dwóch liczb 32-bitowych. Następnie liczby te mogą być przetwarzane na raz (na przykład mogą być dodawane do liczby przechowywanej w innym rejestrze zmiennoprzecinkowym).

Takie działanie może się okazać całkiem przydatne i bardzo wydajne. W wielu programach multimedialnych oraz w niektórych rodzajach aplikacji biurowych wymagane jest właśnie powtarzanie tej samej operacji na wielu związanych pozycjach danych. Przy użyciu MMX wykonanie takiego zadania można znacząco przyspieszyć. Jako przykłady można podać cyfrowe przetwarzanie sygnału (stosowane w programach do obsługi tzw. „winmodemów") oraz niektóre operacje w grafice dwuwymiarowej (na przykład tzw. „odwzorowania tekstur").

Jedną z wad tej metody jest to, że nie można jednocześnie korzystać z rejestrów zmiennoprzecinkowych w obliczeniach i w operacjach MMX. Zazwyczaj, w większości współczesnych procesorów x86, jednostki obliczeniowe — stałoprzecinkowa i zmiennoprzecinkowa — działają równolegle, co przyspiesza wykonywanie zwykłych programów.

Firma AMD opracowała własną wersję rozszerzenia zestawu instrukcji i nazwała ją 3DNow!. Rozszerzono tu zestaw x86 dzięki dodaniu kilku instrukcji SIMD, wykonujących podobne operacje na rejestrach zmiennoprzecinkowych, co w zestawie MMX. Oprócz tego, dodano instrukcje dla operacji stałoprzecinkowych i zmiennoprzecinkowych oraz wprowadzono kilka zmian w rejestrach, dzięki czemu uzyskano większy stopień równoległości obliczeń. Poszczególne instrukcje z zestawu 3DNow! miały na celu — zgodnie z nazwą zestawu — przyspieszenie operacji przy wyświetlaniu grafiki trójwymiarowej, a także przyspieszenie obliczeń wymaganych w trakcie przetwarzania dźwięku i wyświetlania filmów.

Ostatnio wszystkie procesory produkowane przez AMD, Cyrix (oddział firmy National Semiconductor) i Centaur (oddział firmy IDT) obsługują rozszerzenia 3DNow! oraz MMX.

Intel musiał zareagować na wdrożenie koncepcji 3DNow!. W odpowiedzi zdecydował się na dalsze rozszerzanie zestawu MMX w przyszłych konstrukcjach procesorów, a nie na przyjęcie rozwiązań zastosowanych w 3DNow!. Zmodernizowane rozszerzenie nazwano początkowo MMX2, lecz obecnie mówi się o *nowych instrukcjach Pentium III*, a skrótowo nazywa się je KNI (skrót powstał od kodowej nazwy procesora, w którym zastosowane zostało po raz pierwszy, po dodaniu odpowiednich rozszerzeń układowych — *Katmai New Instruction*).

KNI oznacza zatem ulepszenia konstrukcji procesora w trzech kierunkach: nowe instrukcje, nowe rejestry i nowe sposoby sterowania dostępem do pamięci.

Nowe instrukcje SIMD są zarówno zmiennoprzecinkowe, jak i stałoprzecinkowe. Wykorzystują one nowe, specjalne rejestry o długości większej niż miały poprzednio rejestry zmiennoprzecinkowe. Instrukcje te umożliwiają pracę ze 128 bitami na raz (co oznacza osiem liczb dwubajtowych lub cztery liczby zmiennoprzecinkowe o zwykłej dokładności). Oprócz tego, uwzględniono w nowych instrukcjach sugestie przekazywane przez programistów, takie jak możliwość uśredniania z zaokrąglaniem w dół (wykorzystywana w kodowaniu sygnału wizyjnego przy pokazywaniu ruchu) oraz pewne elementy związane z rozpoznawaniem mowy.

Działanie wszystkich starszych procesorów x86 polegało na sekwencyjnym pobieraniu instrukcji, uruchamianiu ich i zapamiętywaniu wyników. Nie było możliwe rozpoczęcie pobierania nowej instrukcji przed zachowaniem wyniku działania instrukcji poprzedniej.

Nowa „architektura strumieniowej obsługi pamięci" określa metodę przetwarzania zastosowaną przez firmę Intel w procesorach Pentium III. Polega ona na umożliwieniu programiście pobierania kilku instrukcji z wyprzedzeniem przed ich wykonaniem, na ich wykonaniu i zachowaniu wyników tak szybko, jak będzie to możliwe. Proces ten przebiega bez zaburzania wykonywania dodatkowych instrukcji. Dzięki temu procesor nie musi oczekiwać na pobranie danych lub instrukcji z pamięci ani na zapis do pamięci, co powoduje, że jego wydajność wzrasta.

Zalety te można wykorzystać tylko w specjalnie napisanych programach, które uwzględniają dodatkowe instrukcje. To samo odnosiło się kiedyś do korzystania ze sprzętowych udogodnień w obliczeniach zmiennoprzecinkowych, gdy były one jeszcze nowością. Jedynie te programy, w których dokonano stosownych zmian, mogły korzystać z rozszerzeń procesora i dzięki temu mogły działać szybciej. Nie wszystkie programy można było jednak tak zmodyfikować.

Z biegiem czasu wiele programów zaczęło korzystać z jednostki zmiennoprzecinkowej. Jeśli uruchamia się je na komputerze z procesorem 486SX lub 386, które nie mają wbudowanej jednostki zmiennoprzecinkowej, okazuje się to bardzo męczące! Programy w takim przypadku muszą wywoływać program emulujący, by skompensować w ten sposób brak specjalnych układów w procesorze, zaś działanie takiego emulatora jest znacznie wolniejsze niż działanie specjalnego układu.

Chociaż rozszerzenie MMX pojawiło się już kilka lat temu, nadal nie jest ono powszechnie wykorzystywane w znajdujących się na rynku programach, nie licząc gier. Instrukcje 3DNow! są lepiej obsługiwane, przynajmniej jeśli chodzi o system Windows, ponieważ firma Microsoft włączyła je do najnowszych wersji programu sterującego DirectX. Dowolny program wykorzystujący do aktywacji sprzętu API DirectX będzie korzystał z ich zalet pod warunkiem, że uruchamiany jest na procesorze wyposażonym w rozszerzenie 3DNow!.

Prawdopodobnie Microsoft zapewni obsługę rozszerzenia KNI również w przyszłych wydaniach swoich programów sterujących i w innych programach systemowych. Jeśli tak się stanie i jeśli strumieniowa architektura potwierdzi zalety oferty Intela, Pentium III może jeszcze raz znacznie przewyższyć konkurencyjne produkty.

# Pamięć podręczna pierwszego poziomu

Pozostało jeszcze przedstawić ostatnią grupę funkcjonalną elementów procesora, czyli pamięć podręczną (ang. *cache memory*). Pewne odmiany tej pamięci stanowiły integralną część wszystkich procesorów firmy Intel, a także — począwszy od modelu 486 — klonów rodziny x86.

Pamięć podręczna (dla adresów) używana była jeszcze wcześniej, w postaci bufora transakcji (TLB) w procesorze 386 i w tej formie zachowała się również we wszystkich procesorach z rodziny 386. Jednak taki rodzaj pamięci podręcznej jest przeważnie ukryty i niedostępny dla buforowania danych i instrukcji, zatem nie będzie tutaj omawiany.

Do szerokiego zastosowania pamięci podręcznej przyczynił się fakt, że współczesne procesory są bardzo szybkie i z łatwością przewyższają szybkość działania pamięci głównej. Po wbudowaniu do procesora pamięci o niewielkiej pojemności, dorównującej szybkością samemu procesorowi, można ją wykorzystać jako bufor do chwilowego przechowywania danych i instrukcji pobieranych z pamięci głównej oraz wysyłanych do tej pamięci. Ze względu na szybkość, jaką oferuje pamięć podręczna, jest ona wyjątkowo droga i stosuje się ją tylko jako dodatek do pamięci głównej.

Jest to tzw. *pamięć podręczna pierwszego poziomu* (ang. *level 1 cache*), która przyspiesza działanie procesora w dwojaki sposób. Po pierwsze, buforuje ona dane przesyłane przez procesor do pamięci głównej i dane pobierane z tej pamięci.

Gdy informacja ma być zapisana do pamięci głównej, wtedy pamięć podręczna przyjmuje ją od procesora, który natychmiast może zająć się innymi operacjami. Układ sterujący pamięcią podręczną odpowiedzialny jest za przekazanie przechowywanej informacji na właściwe miejsce w pamięci głównej.

Drugą przyczynę przyspieszenia działania stanowi buforowanie przez procesor wielokrotnych odczytów informacji z określonego miejsca w pamięci głównej. Podczas pierwszego odczytu nie ma przyspieszenia. Podczas drugiego odczytu (i kolejnych) procesor żąda dostępu do tej samej komórki o tej samej zawartości. Jeśli dana wartość znajduje się już w pamięci podręcznej, jej układ sterujący dostarcza dane prawie natychmiast.

Podstawowe ograniczenie efektywności pamięci podręcznej wynika z faktu, że w porównaniu z rozmiarem pamięci głównej jej pojemność jest bardzo niewielka, zatem tylko nieliczna część zawartości ostatnio obsługiwanych komórek pamięci głównej może być przechowywana w pamięci podręcznej.

## Buforowanie pamięci nie zawsze jest korzystne

W czasach, gdy komputery PC były czymś nowym, wbudowywanie pamięci podręcznej do procesora nie miało sensu. Wynikało to z dwóch rzeczy. Po pierwsze, powodowało dodatkowe komplikacje i producenci układów scalonych w owych czasach często nie potrafili sobie radzić z takimi skomplikowanymi układami. Po drugie, układy procesorów działały bardzo wolno, zatem pamięć główna mogła z łatwością z nimi współpracować. Obecnie doszło jednak do sytuacji, w której procesory są kilkakrotnie szybsze niż układy pamięci, nawet na najlepszych płytach głównych, zaś dodatkowe problemy związane z wbudowaniem pamięci podręcznej do procesora można stosunkowo łatwo rozwiązać.

Dokładne badania współcześnie używanych programów komputerowych wykazały, że bardzo często wykorzystują one wielokrotnie te same instrukcje, a nawet te same

dane. Wynika to z faktu, że programiści bardzo często korzystają z pętli w programach ułatwiających wykonanie określonych zadań. Jeśli pętla jest mała i wszystkie jej instrukcje oraz powiązane z nimi dane mieszczą się w pamięci podręcznej, procesor podczas wykonywania pętli może działać z największą szybkością, ponieważ nie musi oczekiwać na dostęp do stosunkowo wolnej pamięci głównej.

W każdym przypadku, jeśli program chce zapisać jakieś informacje do pamięci, może to zrobić za pośrednictwem pamięci podręcznej. Pamięć ta zajmie się tym tak szybko, jak pozwolą jej na to znacznie wolniejsze układy zewnętrzne.

### Podwyższanie wydajności pamięci podręcznej

Mały pakiet pamięci podręcznej może mieć różną organizację i w różny sposób można z niej korzystać. Każda z użytych tu metod ma określone skutki, zarówno w postaci kosztów, jak i efektywności działania.

Oczywiście, w opisach spotykanych literaturze informatycznej można znaleźć bardzo wiele żargonowych określeń dotyczących pamięci podręcznych, na przykład takich jak: *buforowanie odczytu* (ang. *read caching*), *buforowanie odczytu z wyprzedzeniem* (ang. *read-ahead caching*), *buforowanie bez wstrzymywania zapisu* (ang. *write-through caching*), *odroczone buforowanie zapisu i odczytu* (ang. *deferred write and read caching*) oraz *pamięci całkowicie skojarzeniowe* (ang. *fully associative caches*), *pamięci z bezpośrednim odwzorowaniem* (ang. *direct mapped caches*) i *ustawiane pamięci skojarzeniowe* (ang. *set associative caches*). Te ostatnie mają także kilka odmian (dwukierunkowe, czterokierunkowe itp.). Mamy także pamięci podręczne rozdzielone na dwie części: jedna dla buforowania instrukcji, druga dla buforowania danych.

Można być pewnym, że producenci procesorów zbudowali je na podstawie wskazówek ośrodków badawczych w taki sposób, by zawierały najbardziej wydajną pamięć podręczną odpowiedniej pojemności, o ile tylko pozwala na to współczesna technologia produkcji układów scalonych.

Za chwilę powrócimy do kwestii buforowania, ponieważ pojawi się ona również w zagadnieniu dotyczącym architektury układów towarzyszących procesorowi, która jest tematem następnego podrozdziału. Omówimy także zastosowanie tzw. *zewnętrznej pamięci buforowej* w najnowszych konstrukcjach procesorów.

# Architektura układów towarzyszących procesorowi

Po lekturze dotychczasowego materiału Czytelnicy powinni dobrze rozumieć działanie elementów funkcjonalnych procesora. Zanim zakończymy opis architektury komputera PC, powinniśmy jeszcze poznać działanie oraz rozmieszenie zewnętrznych układów towarzyszących procesorowi. Najważniejsza jest tu pamięć główna, a następnie porty obsługujące wejścia i wyjścia. Wszystkie pozostałe układy komunikują się z procesorem właśnie za pomocą tych dwóch struktur.

# Pamięć

Pamięć główna i procesor są w komputerze obszarami, w których wykonywane są wszystkie obliczenia. Wynika to stąd, że dane i programy muszą być umieszczone w jakimś obszarze pamięci głównej, zanim procesor coś z nimi zrobi. Niektóre programy i bardzo małe porcje danych mogą tam przebywać stale. Przeważnie są one jednak wyrzucane z tego obszaru, kiedy nie są już potrzebne, i wówczas są albo niszczone (programy), albo zachowywane na trwałym nośniku (dane). Po takiej operacji zajmowane przez nie obszary pamięci są dostępne dla nowych programów i danych.

*Pamięć główna* w komputerze PC jest mieszanką pamięci RAM, ROM i miejsc, które można zapełnić. Znaczy to, że procesor może adresować fizyczną przestrzeń adresową o ustalonym rozmiarze. W niektórych obszarach tej przestrzeni umieszczone są układy pamięci o swobodnym dostępie do zapisu i odczytu (RAM). W innych miejscach znajdują się układy pamięci tylko do odczytu (ROM) lub przeważnie tylko do odczytu (pamięć nieulotna, tzw. NVRAM). Zazwyczaj w większej części przestrzeni adresowej nie znajduje się nic.

Nie zawsze tak było. W czasach, gdy komputery PC były jeszcze czymś nowym, pamięci były znacznie droższe w przeliczeniu za jeden bajt niż są obecnie. Ówczesne komputery nie potrafiły jednak adresować więcej niż 1 MB. Wielu właścicieli owych komputerów PC miało przestrzeń adresową wypełnioną prawie całkowicie układami pamięci RAM lub ROM.

## Maksymalny rozmiar pamięci fizycznej

Nowoczesne komputery mogą adresować bardzo wiele komórek pamięci. Począwszy od procesora 386, mogły one potencjalnie obsługiwać pamięci o pojemności do 4 GB, zaś przy układach Pentium i Pentium II teoretyczna granica wynosi aż 64 GB. Nawet przy obecnych, niskich cenach układów pamięci, nie ma zbyt wielu osób, które zbliżyłyby się do tej granicy.

Współczesne układy procesorów z rodziny x86 mają także dwie dodatkowe przestrzenie adresowe, nazywane pamięcią wirtualną (logiczną) i pamięcią liniową. Liniowa przestrzeń adresowa ma zwykle taki sam rozmiar, co przestrzeń fizyczna. Przestrzeń wirtualna jest o wiele większa, określa ją bowiem obszar dostępny na dysku.

Potencjalny rozmiar fizycznej przestrzeni adresowej nie jest taki sam jak maksymalny rozmiar pamięci, którą można zamontować w komputerze. Wiąże się to z tym, że producenci komputerów nie podłączają wszystkich linii adresowych (bezpośrednio lub za pośrednictwem dekoderów adresów) do gniazd pamięci. Nie jest to wymagane, ponieważ żaden klient nie żąda wstawienia do komputera pamięci o rozmiarze granicznym z możliwościami adresowymi procesora. Nawet gdyby tak było, żaden ze współczesnych systemów operacyjnych nie mógłby skorzystać z tak dużej pamięci. System Windows we wszelkich odmianach, oprócz NT, ograniczony jest do obsługi pamięci o pojemności nie większej niż 2 GB.

## Pamięci podręczne drugiego i trzeciego poziomu

Kończąc rozważania o architekturze procesora, opisano pamięć podręczną pierwszego poziomu (L1), wbudowaną we wszystkie ostatnio produkowane układy z rodziny x86. Idea użycia pamięci podręcznej narodziła się w rzeczywistości jeszcze przed skonstruowaniem procesora 486DX, który był pierwszym przedstawicielem rodziny x86 wyposażonym w ten typ pamięci. Gdy szybkość działania układów mikroprocesorowych stała się większa w najszybszych dostępnych układach pamięci DRAM, zainteresowano się pamięcią podręczną, lecz producenci nie byli jeszcze w stanie wbudować jej do układu procesora.

Pamięci podręcznej po raz pierwszy użyto w komputerach z procesorem 386. Płyta główna w tych komputerach zawierała dodatkowe, bardzo szybkie (i bardzo drogie) układy pamięci RAM oraz specjalny układ scalony, zwany kontrolerem pamięci podręcznej. Była to jedyna pamięć podręczna, która mogła być zastosowana w tego typu komputerach.

Pojemność takiej pamięci ograniczona jest głównie przez możliwości finansowe klientów, którzy chcieliby uzyskać większą wydajność, a nie przez technologię produkcji procesorów, tak jak w przypadku pamięci L1. Im więcej pamięci podręcznej, tym lepiej, lecz od pewnego momentu wzrost wydajności nie jest już tak duży, jakby wynikało ze wzrostu pojemności tej pamięci. Oznacza to, że nawet wtedy, gdy Intel i inni producenci rozpoczęli produkcję procesorów z niewielką pamięcią podręczną pierwszego poziomu (L1), producenci płyt głównych, którzy chcieli sprzedawać więcej, wyposażali je w pamięć podręczną drugiego poziomu (L2) o znacznie większej pojemności niż pamięć L1 w procesorze.

Pamięć L2 o danej pojemności jest mniej wydajna niż pamięć L1 o takiej samej pojemności z tego prostego powodu, że zewnętrzna częstotliwość zegarowa we współczesnych procesorach stanowi jedynie ułamek częstotliwości zegara wewnętrznego. Oczywiście, może to być użyteczny dodatek na płycie głównej, ponieważ taka pamięć dostarcza dane w jednym cyklu zegara, zaś pamięć DRAM, stosowana przeważnie jako pamięć główna, wymaga co najmniej dwóch cykli do przeprowadzenia tej samej operacji.

Ostatnio produkowane przez firmę Intel procesory, począwszy od Pentium Pro, zawierały dwa układy scalone, które tworzyły moduł procesora. Jeden z tych układów to właściwy procesor zawierający pamięć L1. Drugi układ scalony jest oddzielną pamięcią podręczną L2, która łączy się z procesorem przez oddzielną magistralę (nie tę, przez którą procesor komunikuje się ze światem zewnętrznym). Firma Intel nazywa taką konstrukcję architekturą dwóch niezależnych magistrali (ang. *Dual Inpependent Bus*, w skrócie *DIB*).

Taka konstrukcja posiada dwie zalety. Po pierwsze, magistrala główna może przenosić więcej danych, ponieważ nie jest zaangażowana w wymianę danych z pamięcią L2. Po drugie, szybkość wymiany danych między procesorem a pamięcią L2 jest taka, na jaką pozwala sama pamięć (w różnych układach była ona równa połowie lub pełnej szybkości działania rdzenia procesora, czyli znacznie szybsza niż magistrala główna). W systemie Xeon może na przykład występować procesor Pentium II oraz pamięć L2, działające z częstotliwością 450 MHz przy częstotliwości magistrali zewnętrznej zaledwie 100 MHz.

Firma Intel poinformowała o następnych planach produkcji modułów procesorów, w których zarówno pamięć podręczna L1, jak i L2 wbudowane będą do układu procesora, a dodatkowa pamięć podręczna trzeciego poziomu (L3) wbudowana będzie do modułu jako oddzielny układ scalony. Aby uniknąć nieporozumień, firma zaproponowała także zmianę numeracji pamięci podręcznej w taki sposób, by jej najmniejszą porcję położoną najbliżej procesora nazywać pamięcią L0 (zerowego poziomu), drugą porcję wewnątrz procesora, o większej pojemności, nazywać pamięcią pierwszego poziomu (L1), zaś trzecią, umieszczoną w module w postaci oddzielnego układu o pojemności prawdopodobnie kilku megabajtów, połączonego z procesorem za pomocą magistrali DIB — pamięcią drugiego poziomu (L2).

Klienci wymagający bezwzględnie pamięci podręcznej o dużej pojemności, mogą w komputerze z procesorem zawierającym pamięć L1 i L2 dodać zewnętrzną pamięć L3 na płycie głównej. Pamięci kolejnych poziomów mogą mieć coraz większe pojemności, co częściowo rekompensuje ich wolniejsze działanie w porównaniu z pamięcią poprzedniego poziomu. Kombinacja pamięci podręcznej wszystkich trzech poziomów jest największym osiągnięciem współczesnej technologii.

Obecnie taka kombinacja może być uzyskana tylko przy użyciu płyt z gniazdem procesora nazwanym Super Socket 7. Płyty takie mają na pewno własną pamięć podręczną i można do nich włączyć procesor K6-3 firmy AMD, wyposażony w moduły pamięci podręcznej L1 i L2.

## Spójność pamięci podręcznej

Cały sens stosowania pamięci podręcznej polega na tym, że taka lokalna, niewielka, lecz szybka pamięć zawiera wierną kopię tego, co jest przechowywane w jakimś obszarze pamięci głównej, która jest większa, bardziej oddalona i znacznie wolniejsza. Procesor może posługiwać się kopią podręczną tak samo, jakby była to zawartość pamięci głównej.

W większości sytuacji podobne działanie nie stwarza problemów, lecz w dwóch przypadkach mogą się one pojawić. Pierwszy pojawia się wtedy, gdy informacja z pamięci podręcznej nie dotarła jeszcze na swoje miejsce w pamięci głównej, a inne urządzenie w komputerze chce tę informację stamtąd odczytać. Drugi przypadek występuje wtedy, gdy procesor chce skorzystać z informacji zawartej w pamięci głównej, lecz nie jest poinformowany, że jakieś inne urządzenie zmieniło zawartość tej pamięci, ponieważ była ona wcześniej pobrana do pamięci podręcznej.

Każdy komputer PC — z wyjątkiem modelu PC Junior, produkowanego kiedyś przez IBM — miał możliwość korzystania z funkcji bezpośredniego dostępu do pamięci (ang. *direct memory access*, w skrócie *DMA*). Wymagało to zastosowania specjalnego układu sterującego, który może przyjmować od procesora polecenia przeniesienia zawartości jakiegoś obszaru pamięci w inne miejsce — albo do portu wyjściowego, albo pobrania jej z portu wejściowego. Układ sterujący bezpośrednim dostępem do pamięci realizuje to zadanie bez udziału procesora. Oprócz tego, duże komputery PC pełniące role serwerów (a także niektóre bardzo wydajne stacjonarne stacje robocze) mogą być wyposażone w wiele procesorów, które współużytkują pulę pamięci głównej.

W każdym przypadku, niezależnie od tego, czy używa się wielu procesorów, czy korzysta się z DMA (lub innych urządzeń do „zarządzania magistralą", takich jak na przykład szybkie sterowniki SCSI), możliwa jest niekiedy zmiana zawartości pamięci głównej nie tylko przez sam sterownik pamięci podręcznej procesora (lub sterownik pamięci L2 i L3, jeśli takie występują). Jeśli zdarzy się taka sytuacja, sterownik pamięci podręcznej podłączony do pamięci głównej musi być o tym fakcie poinformowany i musi „unieważnić" przynajmniej obraz tej pamięci ulokowany w pamięci podręcznej, zanim nie zastąpi jej nową, zaktualizowaną zawartością.

Jedynym sposobem uzyskania *spójności pamięci podręcznej* jest podłączenie jej sterownika do pamięci głównej i śledzenie wszystkich prób dostępu podejmowanych przez inne urządzenia. Sterownik pamięci podręcznej musi sprawdzać, czy każdy adres udostępnianej komórki pamięci pokrywa się z adresem komórki aktualnie przechowywanej w zarządzanej przez niego pamięci podręcznej. Każdy sterownik pamięci podręcznej ma zatem wbudowaną możliwość swoistego *podsłuchu magistrali* (ang. *bus snooping*), by sprawdzać, jakie urządzenie może dokonywać wpisów do pamięci.

Pojawia się również inny, nieco bardziej subtelny problem związany ze spójnością pamięci podręcznej. Jeśli procesor próbuje zapisać coś do pamięci pod adresem, pod którym umieszczony jest układ ROM lub nie ma w ogóle nic, taka próba się nie powiedzie. Jeśli sterownik pamięci podręcznej nie będzie poinformowany o tym fakcie, może przetrzymywać informację przekazaną przez procesor w taki sam sposób, jakby była to informacja obowiązująca, która powinna znaleźć się pod podanym adresem w pamięci głównej. Dopóki ta informacja pozostaje w pamięci podręcznej, procesor może ją pobrać w dowolnym momencie. Jednak jeśli będzie zwlekał tak długo, aż zostanie ona zmieniona, pobierze wartość rzeczywistą (jeśli taka istnieje) przechowywaną pod danym adresem.

Jedynym sposobem uniknięcia problemów tego typu jest przekazywanie do sterownika pamięci podręcznej informacji o tym, które obszary fizycznej przestrzeni adresowej *mogą być buforowane* (czyli w których znajduje się rzeczywiście pamięć RAM), a które nie (czyli te, które nie są obsadzone lub zawierają pamięć ROM). Prawie zawsze korzystne jest buforowanie odczytu z pamięci ROM, ale *nigdy* nie należy pozwalać na buforowanie zapisu do takich obszarów.

Nowoczesne programy do konfiguracji BIOS-u zawierają często procedury, za pomocą których można poinformować sterownik pamięci podręcznej o obszarach wymagających buforowania. Jeśli do programu zostaną wprowadzone poprawne dane konfiguracyjne, problemy nie wystąpią. Pomyłka w danych może jednak skutkować dziwnymi niespodziankami objawiającymi się podczas pracy komputera.

## Stosy

Wspomniano wcześniej, że procesor musi dysponować jakimś miejscem do chwilowego przechowywania przetwarzanej informacji. Do tego właśnie służą rejestry. Niekiedy ich liczba jest jednak zbyt mała, by przechować całą potrzebną informację i trzeba ją wówczas zgromadzić w innym miejscu.

Kłopot ten występuje szczególnie w systemach wielozadaniowych. Symulacja jednoczesnego wykonywania kilku zadań polega na wykonaniu niewielkiej części jednego zadania, przełączeniu się do drugiego, wykonaniu jego niewielkiej części, przełączeniu się do trzeciego itd.

Przy każdym przełączaniu zadań procesor musi zachować zawarte w rejestrach wartości, uzyskane w poprzednim zadaniu i dopiero wówczas może zająć się wykonywaniem kolejnego zadania. Do tego celu służą stosy. Nie jest to ich jedyne przeznaczenie, chociaż w większości są one tak właśnie wykorzystywane.

Koncepcja stosu jest prosta i można wyjaśnić jej działanie na przykładzie stosu talerzy. Kiedy chcemy odłożyć umyty przed chwilą talerz, układamy go na stosie w szafce. Kiedy chcemy wyjąć jakieś talerze, bierzemy je kolejno z wierzchołka stosu. Talerz położony na stosie jako ostatni będzie z niego zdjęty jako pierwszy.

Rolę stosu w komputerze pełni po prostu obszar pamięci skojarzony z rejestrem przechowującym wskaźnik adresowy. Jeśli jakaś porcja informacji ma być przesłana na stos w wyniku wykonania instrukcji PUSH, jest ona wpisywana pod adres wyznaczony przez wskaźnik adresowy, po czym następuje zmniejszenie o jeden wartości tego wskaźnika, który od tego momentu będzie pokazywał następną komórkę pamięci o niższym adresie. Przy pobieraniu informacji ze stosu za pomocą instrukcji POP odbywa się proces odwrotny: informacja jest odczytywana z komórki określonej przez zawartość rejestru, a po odczycie zawartość rejestru wskaźnikowego powiększana jest o jeden.

Maksymalny rozmiar stosu ustalany jest na podstawie rozmiaru liczb, które mogą być przechowywane w rozważanym rejestrze oraz na podstawie początkowej wartości wskaźnika stosu (w normalnych warunkach jest ona taka sama, jak rozmiar „segmentu" przydzielonego na potrzeby stosu). Na ogół to drugie ograniczenie jest wiążące, chociaż można także utworzyć bardzo duży stos korzystający z całej niemal pojemności rejestru.

Gdy program próbuje przesłać na stos więcej informacji niż można obsłużyć ze względu na ograniczenia rozmiaru stosu, wtedy wskaźnik stosu staje się ujemny. Procesor rejestruje ten fakt i zanim dojdzie do operacji na stosie, wygenerowany zostanie wyjątek.

W komputerze PC zawsze musi istnieć stos gotowy do użytku. Procesor przechowuje wskaźnik bieżącego położenia w stosie w rejestrze zwanym rejestrem wskaźnika stosu (SP). Jest to wartość offsetu w segmencie wskazywanym lub definiowanym przez zawartość rejestru segmentu stosu (SS).

W każdym poprawnie napisanym programie jedną z początkowych operacji jest utworzenie prywatnego stosu. Wynika to stąd, że programiści nie mogą przewidzieć, ile dostępnego miejsca znajduje się na stosach utworzonych wcześniej. Operacja tworzenia prywatnego stosu przez program polega na zarezerwowaniu pewnego obszaru pamięci, następnie przesłaniu bieżącej wartości wskaźnika stosu na stos utworzony wcześniej, a potem załadowaniu do rejestrów SS i SP nowych wartości wskazujących na zarezerwowany obszar pamięci. Kończąc tę operację, program pobiera stare wartości ze stosu i procesor odzyskuje swój poprzedni stan. Konieczne jest również przesłanie na stos zawartości wszystkich rejestrów, które będą modyfikowane przez program i pobranie ich ze stosu na zakończenie programu.

W komputerze PC można zdefiniować dowolną liczbę stosów w dowolnym momencie, ale tylko jeden z nich będzie stosem *bieżącym*. Jest nim stos określany za pomocą adresu logicznego `':[SS]` (jest to zapis specjalny, który należy odczytywać jako oddzielone dwukropkiem liczby szesnastkowe, przechowywane w dwóch rejestrach o nazwach podanych w nawiasie kwadratowym).

Przy okazji należy wspomnieć, że w ostatnio produkowanych procesorach x86 istnieje jeszcze inny stos. Jest on przeznaczony do obsługi ośmiu 80-bitowych rejestrów procesora, które wykorzystywane są do obliczeń zmiennoprzecinkowych, a także do wykonywania instrukcji MMX w procesorach Pentium MMX i Pentium II. Z tego specjalnego stosu w zwyczajnych okolicznościach korzystają tylko instrukcje MMX (i 3DNow! w klonach x86) oraz instrukcje zmiennoprzecinkowe. Nie należy go mylić z „normalnym" stosem, tworzonym i obsługiwanym przez instrukcje `PUSH` i `POP`, poza obszarem pamięci głównej w programach.

Stos odgrywa ważną rolę w programowaniu, ponieważ umożliwia tworzenie bardziej skomplikowanych struktur niż w przypadku korzystania z samych rejestrów procesora, służących do tymczasowego przechowywania informacji. Bez stosu nie byłaby możliwa wielozadaniowość.

# Porty

Gdy procesor tworzy adres fizyczny na swoich wyprowadzeniach adresowych, odwołuje się do określonego miejsca w przestrzeni adresów pamięci. Prosta zmiana stanu napięcia z wysokiego na niski na jednym z wyprowadzeń (nazywanym wyprowadzeniem uaktywniającym pamięć lub porty I/O) sygnalizuje, że adres na wyprowadzeniach procesora może być interpretowany jako wartość z innej przestrzeni adresowej.

Ponieważ te inne adresy przeważnie używane są podczas przekazywania informacji między procesorem i innymi urządzeniami, w tym także urządzeniami zewnętrznymi, określana przez nie logiczna przestrzeń adresowa nazywana jest przestrzenią portów wejścia-wyjścia komputera. Niezależnie od tego, że adresy w nowej przestrzeni określane są przez napięcia na tych samych wyprowadzeniach, które określają także „zwyczajne" adresy, przestrzeń portów I/O jest czymś zupełnie innym niż fizyczna przestrzeń adresowa.

Przede wszystkim jest ona znacznie mniejsza. Każdy procesor z rodziny x86, począwszy od 8086 i 8088 aż do najnowszych modeli Pentium II, ma taką samą przestrzeń I/O o rozmiarze 64 kB (65536). Wynika to stąd, że do jej obsługi procesory wykorzystują tylko 16 dolnych linii adresowych.

Oprócz tego, z adresami w przestrzeni portów I/O nie wiąże się żadna komplikacja w postaci selektorów lub stronicowania. Dysponujemy tylko 64 kB adresów portów (jednobajtowych), dlatego też do określenia żądanego portu potrzebna jest tylko jedna liczba 16–bitowa, którą można załadować do dowolnego rejestru ogólnego przeznaczenia (chociaż niektóre instrukcje wymagają, by adres portu załadowany był do rejestru DX).

Wszystkie układy z rodziny x86, z wyjątkiem procesora 8088, mogą odczytywać i zapisywać równocześnie co najmniej dwa (ale nie więcej niż osiem) bajty informacji z pamięci, zatem mogą one także obsługiwać taką samą liczbę kolejnych adresów portów podczas jednej operacji. Podobnie jak przy adresowaniu komórek pamięci, procesor może wskazywać adresy portów z rozdzielczością wynikającą z szerokości magistrali danych. Oznacza to, że Pentium może adresować blokowo po osiem portów, niezależnie od tego, że może wymieniać informacje z każdym portem jednobajtowym, jeśli jest to konieczne.

## Różnice między portami I/O a komórkami pamięci

Główna różnica między portami I/O a komórkami pamięci polega na odmiennym sposobie traktowania danych przesyłanych w te miejsca. Kolejne bajty wysyłane do portu trafiają zazwyczaj do jakiegoś urządzenia odbiorczego. Za każdym razem, przy kolejnych odczytach z portu, uzyskiwać można różne wartości (i żadna z nich nie musi być taka sama, jaką przesłano do portu) — wiąże się to z tym, że odbierana informacja pochodzi z zewnątrz. Takie zachowanie wyraźnie różni się od zachowania prawdziwych komórek pamięci, w których wartość odczytywana jest dokładnie taka sama jak wartość ostatnio zapisana.

Nie oznacza to, że pod adresami portów nie mogą być umieszczone układy pamięci, chociaż w praktyce rzadko spotyka się takie rozwiązania.

## Odwzorowanie wejść-wyjść w pamięci jako inne rozwiązanie dla portów

Co się stanie, jeśli adresy zostaną użyte „odwrotnie", czyli gdy jakieś urządzenie wejściowo-wyjściowe zostanie umieszczone w przestrzeni adresowej pamięci, zamiast w przestrzeni adresowej portów I/O? Otóż jest to możliwe i co więcej — może się to okazać niezwykle przydatne. Nie trzeba skomplikowanych sztuczek, by układy portu reagowały tak jak pamięć. Wystarczy po prostu odwrócić sygnał na linii MEM/IO#, a port będzie traktował próby dostępu do pamięci jako próby dostępu do portu i odwrotnie.

Przydatność takiego rozwiązania wynika z różnicy szybkości między magistralą ISA, obsługującą porty I/O, a magistralą obsługującą pamięć. Aby uzyskać maksymalną szybkość transmisji w urządzeniu wejściowo-wyjściowym, należy *odwzorować je w pamięci*, czyli spowodować, by pojawiło się ono w przestrzeni adresowej pamięci procesora. Najnowszym przykładem takiego rozwiązania jest tzw. zaawansowany port grafiki (ang. *Advanced Graphic Port*, w skrócie *AGP*), stosowany w komputerze PC do obsługi najszybszych kart graficznych.

Niestety, pociąga to za sobą dodatkową komplikację, której można by uniknąć, gdyby urządzenie było włączone w przestrzeń adresową rzeczywistego portu. Ze względu na różnorodność przekształceń liniowej przestrzeni adresowej na fizyczną przestrzeń adresową pamięci (stosowanie selektorów oraz stronicowania), bardzo łatwo można doprowadzić do sytuacji, w której urządzenie podłączone do portu pojawia się wielokrotnie w liniowym odwzorowaniu adresów lub znika całkowicie z tej przestrzeni. Jest to zjawisko niepożądane, ponieważ programy komunikujące się z urządzeniami, by móc działać poprawnie, muszą znać ich lokalizację.

Na szczęście, dysponujemy obecnie możliwością szybkiej obsługi urządzeń poprzez magistralę PCI, zatem zmniejsza się liczba powodów, dla których należałoby korzystać z odwzorowania portów w pamięci. Obecnie prawie wszystkie urządzenia peryferyjne, które wymagają dużej szybkości obsługi, podłączane są do magistrali PCI.

### Przepustowość portów

Pierwotnie porty podłączane były do linii danych i linii adresowych procesora tak samo jak układy pamięci. W miarę wzrostu szybkości działania procesorów i pamięci, zaczęto rozdzielać drogi przepływu danych do pamięci i do portów I/O. Łączą się one w zestawie układów pomocniczych płyty głównej, lecz poza tym miejscem działają niezależnie i z różnymi szybkościami.

Nowoczesne i bardzo dobrze wyposażone komputery PC korzystają na przykład z magistrali pamięci działającej z częstotliwością 166 MHz (oraz z zegarem wewnętrznym procesora o częstotliwości do 3,06 GHz), lecz magistrala PCI taktowana jest z częstotliwością do 66 MHz. Obsługiwana jest również magistrala ISA, taktowana — jak za dawnych czasów — z częstotliwością 8,33 MHz. Te ograniczenia przepustowości stosowane są po to, by nawet w najnowszych komputerach zapewnić możliwość korzystania ze starszych kart rozszerzających.

# Przerwania, czyli siła napędowa

Dotychczas opisano już wszystkie ważniejsze części komputera PC, zatem znana jest, przynajmniej w sensie statycznym, jego podstawowa architektura. Nie było jednak jeszcze mowy o pewnych kluczowych aspektach dynamicznych tej architektury i dlatego poświęcone im zostaną pozostałe rozważania. Pierwszą grupę zagadnień z tej dziedziny — i w pewnym sensie najważniejszą — stanowią przerwania.

## Nasłuch kontra przerwania

Wyobraźmy sobie małą kwiaciarnię w centrum handlowym. Jej właściciel musi obsługiwać odwiedzających go klientów, lecz w przerwach musi udawać się na zaplecze i wykonywać prace biurowe. Można zadać sobie pytanie: w jaki sposób właściciel dowiaduje się, że przybył klient i trzeba odłożyć pracę biurową, by go obsłużyć?

Są tu dwie podstawowe strategie działania. Jedna z nich polega na przerywaniu pracy biurowej w regularnych odstępach czasu, odchodzeniu od biurka i sprawdzaniu, czy pojawił się klient, którego należy obsłużyć. Jest to tzw. nasłuch (ang. *pooling*). Taka strategia sprawdza się, lecz jest bardzo mało wydajna, a to z dwóch powodów. Po pierwsze, klient, by zostać obsłużonym, musi czekać, aż nadejdzie czas przerwy w pracy biurowej właściciela. Po drugie, nawet kiedy nie ma klientów, właściciel nie może całkowicie poświęcić się pracy biurowej, ponieważ musi regularnie sprawdzać, czy ktoś się nie pojawił w sklepie i nie żąda obsługi.

Powszechnie stosowanym rozwiązaniem — przynajmniej w kwiaciarniach — jest instalacja czujnika wyzwalanego przy przejściu klienta przez drzwi. Zazwyczaj takie urządzenie składa się ze źródła światła i fotodetektora umieszczonego na przeciwnej framudze drzwi. Przerwanie wiązki światła przez wchodzącego (lub wychodzącego) klienta powoduje uruchomienie dzwonka na zapleczu, a tym samym właściciel zostaje powiadomiony, że trzeba kogoś obsłużyć. Początkowy koszt tego rozwiązania jest większy niż koszt nasłuchu, bowiem wiąże się z zakupem oraz instalacją czujnika i dzwonka. Na dłuższą metę taka inwestycja jest jednak opłacalna, ponieważ właściciel może dzięki niej pracować bardziej wydajnie.

Po raz pierwszy firma Intel zastosowała podobne rozwiązanie w procesorze 8086, a potem we wszystkich układach z rodziny x86. Szczegóły stosowania tej strategii są bardzo interesujące.

# Tablica wektora przerwań

Intel projektuje procesory z rodziny x86 w taki sposób, by podczas pracy w trybie rzeczywistym kończyły wykonywanie bieżącej instrukcji, przerywały wykonywane zadanie, zachowywały znacznik wskazujący miejsce zawieszenia zadania i rozpoczynały wykonywanie innego specyficznego zadania — jeśli pojawi się jedno z 256 określonych zdarzeń. Rodzaj wykonywanego specyficznego zadania zależy od rodzaju przerwania.

W procesorach firmy Intel odbywa się to w taki sposób, że po pojawieniu się sygnału przerwania określany jest najpierw jego rodzaj, a potem następuje skok pod specyficzny adres, umieszczony bardzo blisko początku przestrzeni adresowej pamięci. Z komórki o tym adresie pobierany jest wskaźnik do innej komórki, gdzie mieści się program realizujący właściwą obsługę danego rodzaju przerwania.

Należy zwrócić uwagę na brak wskazań bezpośrednich; konstruktorzy procesora mogliby na przykład wydać polecenie: „Jeśli wystąpi przerwanie o numerze 75, przejdź pod podany adres i wykonaj to, co nakazuje umieszczony tam program". Odbywa się to jednak inaczej — do procesora wysyłana jest instrukcja: „Pobierz wartość wskaźnika z pozycji o numerze 75 w tablicy wektora przerwań (ang. *interrupt vector table*, w skrócie *IVT*) i wykonaj program, na który on wskazuje".

Taki brak bezpośredniego adresowania ma kilka zalet, z których dość ważną jest możliwość zmiany „w locie" zachowań procesora w odpowiedzi na przerwanie. Wymaga to tylko zmiany wskaźnika dla danego rodzaju przerwania, by nie wskazywał on na przykład na program A, lecz na program B.

Tablica wektora przerwań w trybie rzeczywistym umieszczona jest wewnątrz początkowego 1 kB rzeczywistej fizycznej przestrzeni adresowej pamięci. W trybie wirtualnego procesora 86 jest to początkowy 1 kB liniowej przestrzeni adresowej, który może mieć różne położenie w przestrzeni fizycznej. W trybach chronionych 286 lub 386 wektor przerwań został zastąpiony przez podobną strukturę, nazywaną tablicą deskryptora przerwań (ang. *interrupt descriptor table*, w skrócie *IDT*), która może być umieszczona przez system operacyjny w dowolnym obszarze pamięci. Z powodów

praktycznych nie wolno umieszczać deskryptora przerwań w takim obszarze pamięci fizycznej, który mógłby być odwzorowany poza zakresem adresów dostępnych dla procesora. Inne ograniczenia tutaj nie występują.

Szczegółowe zasady obsługi przerwań w trybie chronionym mogą być bardzo skomplikowane i do ich opisu wprowadza się różne pojęcia, na przykład: „bramka przerwań", „bramka pułapki", „segment stanu zadania" itp. Na szczęście, można zrozumieć działanie systemu przerwań na podstawie ogólnego opisu ich działania w trybie rzeczywistym. Dalej pozostaje już tylko wiara, że konstruktorzy procesorów i systemów operacyjnych właściwie zaprojektowali trudniejsze zadania i w skomplikowanym środowisku trybu chronionego wszystko działa podobnie.

# Jak powstają przerwania?

W podanym poprzednio porównaniu z kwiaciarnią w centrum handlowym pokazano, że przerwanie może być użyte w celu sygnalizacji zewnętrznych zdarzeń (w tym przypadku było to wejście klienta). Niektóre przerwania w komputerze osobistym również pojawiają się w wyniku zdarzeń zewnętrznych. Są też przerwania będące wynikiem celowego działania programów, a także takie, które wywoływane są przez sam procesor.

## Przerwania sprzętowe

Pierwszy rodzaj przerwań, zwanych przerwaniami sprzętowymi, wykorzystywany jest jako główny środek zwracania uwagi procesora na to, co dzieje się poza nim. Na przykład przy każdym naciśnięciu klawisza klawiatura przesyła do jednostki centralnej pewien sygnał. Specjalny układ zwany *sterownikiem klawiatury* odbiera ten sygnał i następnie sygnalizuje procesorowi naciśnięcie klawisza.

Procesor ma dwa wyprowadzenia, za pomocą których mogą być mu sygnalizowane przerwania sprzętowe: normalne wejście przerwań (z symbolicznym oznaczeniem INTR) oraz wejście przerwań niemaskowalnych (wyprowadzenie z oznaczeniem NMI). W rzeczywistości w komputerze znacznie więcej zdarzeń wymaga powiadomienia procesora i wydaje się to w takiej sytuacji problemem. Na szczęście standardowa architektura komputera osobistego zawiera rozwiązanie tego kłopotu.

Standardową częścią pyty głównej, otaczającą każdy procesor, jest podsystem zwany sterownikiem przerwań. W pierwotnym komputerze PC/XT mógł on przyjmować sygnały z ośmiu linii i wysyłać do procesora sygnał przerwania. Następnie, po potwierdzeniu odbioru tego sygnału przez procesor, sterownik przerwań powinien poinformować procesor o tym, z której linii pochodziło przerwanie. W komputerze IBM PC/AT i wszystkich konstrukcjach późniejszych, liczba wejść sterownika przerwań została powiększona do 15 albo 16. Ta niejednoznaczność (15 *albo* 16) wynika z faktu, że jedno z wejść przerwań zbiera sygnały z ośmiu innych, lecz w niektórych sytuacjach może być wykorzystywane niezależnie.

Magistrala wejściowo-wyjściowa zawiera dziewięć ze wspomnianych szesnastu linii żądania przerwań (IRQ). Dowolne urządzenie podłączone do gniazda magistrali I/O może za pomocą tych linii informować procesor o konieczności obsługi. Pozostałe linie IRQ zarezerwowane są do użytku płyty głównej, na przykład przez kontroler klawiatury.

Karty rozszerzające, włączane do magistrali ISA, nie mogą w normalny sposób współużytkować linii przerwań. Każda z linii może być wykorzystana tylko przez jedną kartę rozszerzającą. Z drugiej strony, karty włączane do magistrali PCI lub CardBus współużytkują zazwyczaj przerwania z innymi kartami włączonymi do takiej nowoczesnej magistrali. Każde z przerwań spowodowanych obecnością sygnału na linii IRQ dociera ostatecznie do wyprowadzenia INTR procesora. Przerwanie niemaskowalne (NMI) jest przeważnie używane tylko przez układy zerujące cały komputer PC.

Różnica między przerwaniem niemaskowalnym a zwykłym da się wyjaśnić następująco: gdy zostanie przejęte przerwanie zwykłe (maskowalne), procesor może je odrzucić, jeśli dotarło do niego polecenie ignorowania przerwań. Fakt odłożenia przerwania zapamiętywany jest do momentu, w którym procesor może odpowiedzieć na żądanie. Wygląda to tak, jakby dorosła osoba mówiła dziecku ciągnącemu ją za rękaw: „Nie teraz, kochanie. Porozmawiam z tobą, jak skończę rozmawiać przez telefon."

Przerwanie niemaskowalne może zostać wykorzystane wtedy, gdy nie można dopuszczać do opóźnień. Funkcjonuje to podobnie do działania alarmu pożarowego. Układy zerujące używają tego mechanizmu do przerwania pracy procesora, ponieważ działa on zawsze, bez względu na to, że procesor może być całkowicie „otumaniony" i mógłby nieskończenie długo nie odpowiadać na zwykłe przerwania.

## Przerwania programowe

W firmie Intel oceniono, że bardzo dobrze będzie stworzyć takie rozwiązanie, które umożliwiałoby programom powodowanie przerwań w czasie swojej pracy. Polega to na przerwaniu pracy programu głównego i uruchomieniu innego specjalnego programu obsługującego zdarzenie zgłaszające sygnał przerwania.

Bez trudu można wykazać zaletę takiego mechanizmu. Twórca programu może na przykład wiedzieć, że w pewnym momencie program musi wysłać jakąś informację na ekran. Jednak program (i programista) nie muszą potrafić dokładnie określić, w jaki sposób ma się to odbywać. Wystarczy wówczas, że główny program przechwyci odpowiedni rodzaj przerwania przy określonych wartościach w niektórych rejestrach, wskazujących, co ma być wysłane na ekran, a program wywołany przez przerwanie wykona resztę pracy.

Programista tworzący program, którego zadaniem jest wysyłanie czegoś na ekran, nie musi znać wszystkich szczegółów takich operacji. Z kolei programista tworzący program, który faktycznie realizuje zapis na ekran, nie musi niczego wiedzieć o tym, dlaczego i kiedy dana informacja ma pojawiać się na ekranie.

### Wyjątki procesora

Po zrealizowaniu opisanego mechanizmu, specjaliści pracujący w firmie Intel doszli do wniosku, że byłoby dobrze, gdyby procesor mógł z niego korzystać do obsługi wykrywanych przez siebie błędów. Dotyczy to na przykład takich sytuacji, w których program żądający dostępu do obszaru pamięci, w stosunku do którego nie ma odpowiednich uprawnień (często nazywa się to ogólnym błędem ochrony lub błędem strony), będzie wyzwalał odpowiednie przerwanie. Wywołany program obsługi takiego przerwania może następnie wyświetlić na ekranie okno z informacją o błędzie i zakończyć działanie programu głównego, który ów błąd spowodował.

## Programy obsługi przerwań

Programy reagujące na sygnał przerwania nazywane są, zgodnie ze swoim przeznaczeniem, *programami obsługi przerwań* (ang. *interrupt service routines*, w skrócie *ISR*). Każde z 256 możliwych przerwań musi mieć własny program obsługi lub programów tych musi być co najmniej tyle, ile faktycznie występuje przerwań. Niekoniecznie muszą to być różne programy, ponieważ niektóre z nich wykorzystuje się do obsługi wielu przerwań.

Po wywołaniu programu obsługi przerwania jest on wykonywany i kończy się specjalną instrukcją RET, oznaczającą powrót do programu głównego. Procesor traktuje tę instrukcję jako polecenie powrotu do czynności wykonywanych przed wystąpieniem przerwania.

Prosty program obsługi przerwania może zawierać tylko instrukcje RET i w takim przypadku będzie programem, który nie podejmuje żadnego działania. Jeżeli posiada się takie instrukcje umieszczone gdzieś w pamięci, należy po prostu tak konstruować pozycje tablicy wektora przerwań, które mają być obsłużone, by ignorowały wskazanie na tę instrukcję.

### Przerwanie zakłócające program obsługi innego przerwania

W dotychczasowych rozważaniach nic nie wskazywało na to, że programy obsługi przerwań same zabezpieczają się przed innymi przerwaniami. Niekiedy nie można pozwolić, by były one przerywane w ciągu krótkiego czasu w trakcie swojego działania; w takim przypadku można włączyć maskę zabraniającą przerwań. Nie jest to jednak rozwiązanie godne dobrego programisty, jeżeli komputer pozostaje w takim stanie przez czas dłuży niż jest to niezbędnie konieczne.

Mamy zatem dość często do czynienia z sytuacjami, w których programy obsługi przerwań są przerywane. Jeśli tak się zdarzy, wykonywanie takiego programu jest przerywane dokładnie tak samo, jak każdego innego programu. W miejsce działającego programu obsługi przerwania pojawia się nowy, który jest wykonywany i kończy się instrukcją RET. W tym momencie procesor powraca do wykonywania poprzedniego programu obsługi przerwania i dopiero po jego zakończeniu następuje powrót do właściwej aplikacji.

### Warstwowa struktura programów obsługi przerwań

Nie tylko inne przerwania mogą przerywać działanie programów obsługi przerwań. Często spotyka się sytuacje, w których jeden program obsługi przerwania wywołuje inny, ten natomiast wywołuje jeszcze inny itd. Po załadowaniu programu obsługi przerwania do pamięci, zastępuje on adres zapisany w określonym miejscu tablicy wektora przerwań swoim własnym adresem. Jeśli program napisany był poprawnie, na początku kopiuje pierwotny adres do jakiegoś miejsca przechowywania w swojej własnej strukturze. Następnie, jeśli do pamięci zostanie załadowany inny program obsługujący to samo przerwanie, również wykona on taką samą czynność. Jeśli taka sytuacja się powtarza, to cały zespół poprawnie skonstruowanych programów obsługi przerwań działa bardzo dobrze i użytkownik nie musi się martwić tym, co dzieje się bardzo głęboko w pamięci jego komputera.

## Usługi BIOS w pamięci ROM

Można zadać sobie pytanie, skąd komputer pobiera programy obsługi przerwań? Otóż każda maszyna opuszcza fabrykę z pamięcią ROM, która zawiera BIOS. Ten układ scalony (a właściwie najczęściej są to dwa układy) może zawierać różne dane i programy. Większość z nich to programy obsługi przerwań realizujących najbardziej elementarne czynności, które musi wykonywać każdy komputer PC — na przykład odpowiadanie na sygnały z klawiatury i wpisywanie znaków na monochromatycznym ekranie. Pod koniec tego rozdziału zostaną szczegółowo omówione wpisy w pierwotnej tablicy wektorów przerwań, wskazujące na programy ich obsługi, zawarte w układzie BIOS.

W wielu komputerach PC występują również pamięci ROM na kartach rozszerzeń, zawierające dodatkowe programy BIOS. Jednym ze znanych powszechnie przykładów jest karta grafiki, która obecnie prawie zawsze ma bardzo rozbudowany BIOS w pamięci ROM. Zawiera on inne programy obsługi przerwań związanych z zapisem na ekran za pomocą układów karty. Na początku procesu rozruchu komputera taka karta musi zatem umieścić w tablicy wektorów przerwań adresy swoich własnych programów, aby zastąpić nimi adresy domyślne, pochodzące z BIOS-u płyty głównej.

## Usługi DOS i BIOS w pamięci RAM

Podczas ładowania systemu operacyjnego do pamięci ładuje się wiele dodatkowych programów obsługi przerwań. Prawie każdy system operacyjny można zatem w pewnym sensie traktować jako duży zbiór programów obsługi przerwań. Większość z usług świadczonych przez system operacyjny na rzecz programów odbywa się za pomocą programów obsługi przerwań obecnych albo w tym systemie, albo w pamięci ROM BIOS.

Po załadowaniu programu rezydentnego (na przykład programu obsługi myszy), w komputerze pojawia się następny program obsługi przerwania. Prawie zawsze w takim przypadku program nie tylko ładowany jest do pamięci w sposób umożliwiający korzystanie z niego w dowolnym momencie, lecz także musi wpisać swój własny adres początkowy w odpowiednie miejsce tablicy wektora przerwań, by jego użytkowanie było możliwe.

Powszechnie spotykanym odstępstwem od tej zasady jest pewna klasa programów obsługi urządzeń zwanych *urządzeniami blokowymi* (ang. *block devices*). Są to programy, które na przykład obsługują symulowane dyski (np. dyski wirtualne w pamięci RAM) lub uaktywniają występujące w komputerze dyski specjalnego typu. Programy takie włączają się same w system operacyjny, zarówno za pomocą wpisu w odpowiednie miejsce tablicy wektora przerwań, jak i za pomocą wpisu na *listę powiązaną* (ang. *linked list*), nazywaną *kaskadą programów obsługi urządzeń*.

Obok bezpośrednich operacji procesora, przerwania są jedną z najważniejszych przyczyn działania komputera. W rzeczywistości ich rola jest tak ważna, że stały się one bardzo cennym i poszukiwanym zasobem, ponieważ niekiedy jest ich za mało w porównaniu z wymaganiami sprzętu. Taka sytuacja prowadzi często do konfliktów, występujących wtedy, gdy nie można znaleźć wolnego przerwania dla urządzenia zainstalowanego przed chwilą w komputerze.

Na szczęście magistrala ISA powoli wychodzi z użycia i zastępowana jest przez USB oraz FireWire (IEEE 1394), w których nie występują tego typu problemy. Nowe magistrale wymagają jednego, a niekiedy więcej przerwań, by same mogły działać, lecz obsługują wielką liczbę urządzeń peryferyjnych bez potrzeby korzystania z dodatkowych przerwań.

Kolejną specjalną funkcją standardowej architektury PC, działającą podobnie do przerwań, jest kanał bezpośredniego dostępu do pamięci (DMA). Wspomniano o nim już nieco wcześniej w tym rozdziale, podczas omawiania pamięci podręcznej; teraz nadeszła pora na podanie szczegółów.

# Czym jest kanał DMA?

Najczęściej dzieje się tak, że przemieszczanie każdego bajta informacji wewnątrz pamięci głównej komputera lub między pamięcią a portem I/O odbywa się dwuetapowo. Najpierw procesor pobiera dany bajt do któregoś ze swoich rejestrów, a potem wysyła ten bajt z rejestru na miejsce jego przeznaczenia.

Taka procedura się sprawdza, chociaż nie jest pozbawiona wad. Po pierwsze, procesor nie może zająć się czymś innym podczas operacji przenoszenia bajta. Po drugie, cały proces odbywa się w dwóch różnych etapach. Przeniesienie pojedynczego bajta w taki sposób nie niesie ze sobą jeszcze żadnych zagrożeń, lecz przy bardzo dużej ilości bajtów mogą pojawić się problemy.

W komputerze PC można znaleźć kilka urządzeń, które wymagają przenoszenia wielobajtowej informacji. Najpierw stosowana była do tego dyskietka. Karty dźwiękowe przesyłają w taki sposób tyle informacji, że wymagają więcej niż jednego kanału DMA. Podobnie jest w przypadku skanerów, które korzystają z kanału DMA; któż jednak może przewidzieć, jakie jeszcze urządzenia wymagające takiej obsługi pojawią się w przyszłości?

Konstruktorzy pierwszego komputera PC zdawali sobie z tego sprawę, dlatego zdecydowali o dołączeniu dodatkowego mikrosterownika, przeznaczonego wyłącznie do obsługi takich problemów. Jest to specjalny układ mikroprocesorowy, wykonujący tylko jedno zadanie, noszący nazwę sterownika bezpośredniego dostępu do pamięci (lub po prostu układu DMA). Współpraca tego układu z procesorem głównym polega na tym, że procesor może przekazać mu część swoich zadań związanych z przenoszeniem pewnej liczby bajtów z kolejnych komórek pamięci, począwszy od określonego adresu do kolejnych komórek zlokalizowanych pod innym adresem początkowym. Istnieje również możliwość polecenia sterownikowi DMA, by wysłał określoną liczbę bajtów z kolejnych komórek pamięci na określony adres portu.

Procesor może także nakazać sterownikowi DMA pobranie określonej liczby bajtów ze wskazanego portu i zachowanie ich w kolejnych komórkach pamięci, począwszy od określonego adresu. Przy każdej takiej operacji układ DMA tworzy tzw. kanał DMA. Oznacza to, że określona część tego układu zajmuje się określonym zadaniem.

Układ DMA nie jest jednak w stanie wskazać, która jego część ma się zajmować poszczególnym zadaniem. O wyborze kanału decyduje urządzenie żądające wykonania takiej operacji (czyli procesor lub jakieś urządzenie wejściowe albo wyjściowe). Oznacza to, że kanały DMA również znajdują się w grupie cennych zasobów i konflikty w dostępie do nich mogą uniemożliwiać uzyskanie pełnej wydajności.

Do tej pory w komputerze PC nie spotyka się zbyt wielu urządzeń, które wymagają użycia kanału DMA, zatem problem braku wolnych kanałów jest mniej groźny niż problem braku wolnych przerwań. Taka sytuacje nie musi jednak trwać bez końca, ponieważ w miarę wzrostu liczby nowych urządzeń peryferyjnych przesyłających coraz większe porcje informacji może się okazać, że sterownik DMA nie będzie mógł znaleźć wolnego kanału.

# Dlaczego DMA popadł w niełaskę?

Początkowo traktowano DMA jako cudowne rozwiązanie wszystkich problemów. Sterownik stacji dyskietek w pierwszym komputerze PC korzystał z DMA, co pomogło przyspieszyć stabilizację działania tego z natury powolnego urządzenia. Kilka lat później niektóre sterowniki dysków twardych oraz sterowniki magistrali SCSI również korzystały z mechanizmów bezpośredniego dostępu do pamięci.

Następnie wszystko zaczęło się zmieniać. Zaczęto stosować szybsze procesory, lecz magistrala ISA, obsługująca wejścia i wyjścia nie mogła być przyspieszona. Dopiero niewielkie przyspieszenie jej zegara z początkowej częstotliwości 4,77 MHz do 8,33 MHz w PC/AT przyniosło efekty. Każdy producent klonów komputera PC, któremu udało się przyspieszyć działanie magistrali (niektórzy uzyskiwali nawet 12 MHz), igrał z losem, ponieważ z całą pewnością niektóre karty rozszerzeń używane przez klientów mogły współpracować tylko ze standardową magistralą, a nie z jej przyspieszoną wersją.

Gdy wzrosła szybkość działania procesorów, wzrosła również szybkość działania pamięci głównej, chociaż nie w sposób bardzo znaczący. Częstotliwość taktowania magistrali pamięci podwyższano od początkowej wartości 4,77 MHz do 16 MHz,

potem do 33 MHz i ostatecznie do 133 MHz, a przy okazji dokonywano oczywiście innych ulepszeń. Magistrala ISA nadal jednak musiała być taktowana częstotliwością 8,33 MHz. Oznaczało to, że sterownik DMA jest nieustannie powstrzymywany przez tę powolną magistralę (w szczególności dotyczy to operacji wymiany danych między pamięcią a portem przez magistralę ISA). Pomimo teoretycznej możliwości realizacji zadania bez udziału procesora, sterownik DMA nie mógł wykonać go szybciej niż sam procesor, obsługujący urządzenia peryferyjne za pomocą mechanizmów odwzorowania pamięci. Kilka latem temu pojawiły się zatem i zaczęły święcić tryumfy *programowane wejścia-wyjścia* dla urządzeń peryferyjnych z odwzorowaniem portów w pamięci. Mechanizmy DMA nadal wykorzystywane są jako efektywny sposób przekazywania danych między różnymi obszarami pamięci, natomiast przy wymianie danych z portami nie są już po prostu potrzebne.

## Powrót DMA

DMA powrócił jednak na scenę i okazało się, że stanowi lepsze rozwiązanie niż środki zastosowane w pierwszym komputerze PC. Powodem tego zwrotu są po prostu nowe technologie stosowane w magistralach obsługujących wejścia i wyjścia (PCI, USB i CardBus). Dzięki nim przekaz danych do urządzeń peryferyjnych i odbiór danych z tych urządzeń odbywa się prawie tak samo szybko jak przekaz danych na magistrali pamięci głównej. Obecnie po raz drugi DMA jawi się jako możliwość zaoszczędzenia czasu pracy procesora i jego otoczenia.

Także sam system DMA został ulepszony i obecnie korzystać może z dostępnej przepustowości magistrali w sposób znacznie bardziej wydajny niż kiedyś. Najnowsza wersja tego mechanizmu, zwana Ultra DMA, stosowana jest przy transmisji danych do urządzeń IDE z przepływnością 33 MB/s (czyli dwukrotnie większą niż poprzednio). Wykorzystano tu możliwość transmisji danych, wyzwalanej obydwoma zboczami impulsów zegarowych, a nie tylko samym zboczem opadającym, co było dotąd powszechnie przyjęte i nadal jest standardem w przypadku większości magistrali.

Dla użytkownika PC oznacza to powrót kanałów DMA. Ponownie stały się one zalecanym sposobem przenoszenia dużych porcji danych między urządzeniami peryferyjnymi i pamięcią, a coraz więcej urządzeń zaczyna korzystać z tego rozwiązania.

# Dostosowywanie się do zegara

Częstotliwości zegara w komputerze PC były wzmiankowane już kilkakrotnie. Rzeczywiście, od wielu lat jedną z najczęściej wymienianych liczb w reklamach komputerów jest wartość częstotliwości zegarowej (czytamy na przykład: „Kup nasz wspaniały, nowy komputer z dwoma procesorami Pentium II Xeon 450 MHz"). Cóż tak naprawdę oznacza ta częstotliwość? Które części komputera faktycznie taktowane są taką częstotliwością? Czy w komputerze są jeszcze jakieś inne generatory zegarowe? Jeśli tak, czemu służą? Odpowiedź na te pytania Czytelnik znajdzie w tym podrozdziale.

# Porównanie asynchronicznego i synchronicznego działania komputerów

Układy elektroniczne wymagają pewnego czasu na wykonanie jakichś operacji, mimo iż wydają się nadzwyczaj szybkie. Czas ten zmienia się w zależności od egzemplarza. Możliwe jest zbudowanie komputera, który nie byłby wyposażony w generator zegarowy, a który działałby z taką szybkością, na jaką pozwalałyby jego poszczególne elementy. Dane żądane od pamięci mogłyby stać się dostępne dopiero po pojawieniu się takiego żądania. Wyniki obliczeń byłyby odsyłane do pamięci dopiero wtedy, gdy staną się dostępne. Żaden element nie traciłby czasu na niepotrzebne oczekiwanie, ponieważ po prostu nie musiałby nadążać za jakimś centralnym zegarem.

Taki komputer *można* zbudować, chociaż na pewno nie jest to łatwe zadanie. Znacznie prościej projektuje i buduje się komputer, w którym każdy element jest synchronizowany z centralnym zegarem odmierzającym rytm pracy. Jeśli „tykanie" tego zegara jest wystarczająco wolne, można mieć pewność, że wszystkie elementy ukończą wykonywanie swoich zadań przed pojawieniem się następnego „tyknięcia", nakazującego wykonanie następnej czynności. Konstrukcja komputerów synchronicznych jest tak prosta, że obecnie prawie każdy komputer jest zbudowany właśnie w ten sposób. Dotyczy to szczególnie komputerów PC.

# Różne zegary do różnych celów

Praca synchroniczna nie oznacza jednak, że wszystkie elementy komputera muszą działać w rytm tego samego zegara. Można korzystać (i obecnie stało się to standardem) z wielu różnych zegarów w komputerze, które służą do różnych celów. Kilka kolejnych sekcji tego rozdziału zawiera krótki przegląd najważniejszych generatorów zegarowych, występujących w każdym komputerze PC. Może się okazać, że w konkretnym egzemplarzu komputera tych generatorów jest jeszcze więcej.

## Zegar procesora

Najbardziej znany jest zegar „tykający" wewnątrz układu scalonego procesora. Decyduje on o szybkości działania najszybszej części komputera. Szczególnie obecnie tylko rdzeń procesora działa z taką częstotliwością i żaden inny element komputera nie jest w stanie zbliżyć się do tej wartości. Wyjątkiem jest tu pamięć podręczna L2 w Pentium Pro lub Pentium II Xeon, umieszczona wewnątrz modułu procesora i taktowana taką samą częstotliwością. W procesorze Pentium II częstotliwość taktowania pamięci L2 jest dwukrotnie mniejsza niż częstotliwość taktowania rdzenia. We wszystkich pozostałych układach z rodziny x86 zewnętrzna pamięć podręczna jest taktowana z taką samą częstotliwością, jak magistrala obsługująca pamięć główną.

Magistrala prowadząca z procesora do pamięci głównej zazwyczaj taktowana jest częstotliwością kilkakrotnie mniejszą niż rdzeń procesora. Procesor korzysta ze zwielokrotnionej częstotliwości taktowania zewnętrznej magistrali. Znaczy to, że układy, które rzeczywiście tworzą generator zegarowy, mieszczą się poza procesorem, który synchronizuje swój własny, niezbyt stabilny generator z ustaloną wielokrotnością częstotliwości zewnętrznego sygnału zegarowego.

Na przykład w komputerze z procesorem Pentium II 400 MHz częstotliwość taktowania pamięci głównej wynosiła 100 MHz. Oznaczało to, że wewnętrzny zegar procesora działał z częstotliwością czterokrotnie większą niż zegar zewnętrzny. Nieco szybsze maszyny z procesorami 450 MHz korzystały z takiej samej częstotliwości zewnętrznej, lecz wewnętrzna częstotliwość taktowania procesora była w nich uzyskiwana po pomnożeniu częstotliwości zewnętrznej przez współczynnik równy 4,5.

## Zegar pamięci głównej

Generator taktujący procesor używany jest również do taktowania modułów pamięci głównej i wszystkich związanych z nią układów (należy pamiętać, że jest on umieszczony poza procesorem i jego częstotliwość jest ułamkową częścią częstotliwości wewnętrznej procesora). Taka wartość częstotliwości taktowania występuje na magistrali zewnętrznej.

Zdarza się obecnie, a kiedyś było to powszechne, że tylko zewnętrzna pamięć podręczna (L2 i L3) umieszczona na płycie głównej mogła nadążyć za tym szybkim zegarem. W takim przypadku należało umożliwić pracę wolniejszym układom pamięci DRAM, tworzącym jej główny blok, poprzez dodanie co najmniej jednego tzw. „cyklu oczekiwania" (ang. *wait state*). Jest to po prostu opóźnienie wprowadzane między cyklem zegara procesora lub sterownika zewnętrznej pamięci podręcznej, oznaczającym żądanie dostępu do układu pamięci, a cyklem oznaczającym odczyt wyniku tej operacji. Niekiedy sprytni konstruktorzy omijali problem cykli oczekiwania dzięki podzieleniu pamięci na bloki i zorganizowaniu do niej dostępu „z przeplotem" lub w jakiś inny sposób. Obecnie cykle oczekiwania są również potrzebne, przynajmniej od czasu do czasu, jeśli układy DRAM pamięci głównej nie są w stanie odpowiadać tak szybko, jak wynika z częstotliwości magistrali zewnętrznej. Jest to jedna z przyczyn stosowania zewnętrznej pamięci podręcznej (L2 lub L3), ponieważ pamięć główna w takich przypadkach nie nadąża za procesorem.

Na szczęście w wielu ostatnio produkowanych komputerach PC, w szczególności w oznaczanych jako „wydajne", układy scalone DRAM wykorzystywane jako pamięć główna mogą działać bez wprowadzania cykli oczekiwania przy takiej samej częstotliwości taktowania, jaka występuje na magistrali zewnętrznej. Dlatego właśnie w takich komputerach nie są wymagane (i w związku z tym nie są montowane) pamięci podręczne na płytach głównych. Pamięci podręczne L1 i L2 wewnątrz procesora lub w jego module nadal są oczywiście potrzebne, lecz działają one ze znacznie większymi częstotliwościami niż pamięć główna.

## Zegary magistrali I/O

Już wielokrotnie wspominano, że magistrala ISA nie może pracować z częstotliwością większą niż 8,33 MHz. Taki sygnał jest wyodrębniany za pomocą podziału częstotliwości tego samego zegara, z którego pochodzi sygnał taktujący pamięć główną. Jeśli sygnał taktowania pamięci głównej ma częstotliwość 100 MHz, to współczynnik podziału częstotliwości wynosi 12. Mała częstotliwość zegarowa w przypadku magistrali ISA jest stosowana w celu zachowania możliwość współpracy nawet z bardzo starymi kartami ISA. W niektórych najnowszych konstrukcjach po prostu nie stosuje się już magistrali ISA, ponieważ jest ona niezwykle powolna oraz ze względu na chęć uniknięcia problemów z konfliktami przerwań.

Współczesne komputery mają jednak dodatkowe magistrale I/O, które działają znacznie szybciej niż ISA, chociaż jeszcze nie tak szybko jak pamięć główna. Częstotliwość taktowania magistrali PCI wynosi zazwyczaj 33 MHz, zaś w złączu AGP, w najnowszych komputerach jest równa 266 MHz. Częstotliwości te uzyskiwane są metodą podziału lub zwielokrotniania częstotliwości sygnału zegarowego pamięci głównej.

### Pozostałe zegary w komputerze PC

Wiele podsystemów w komputerze PC musi działać synchronicznie z zegarami o różnych częstotliwościach, które nie są synchronizowane z zegarem taktującym pamięć główną, procesor i magistralę I/O. Przykładem takiego urządzenia jest monitor z lampą kineskopową, w którym wiązka elektronowa przemiata ekran z częstotliwością określoną przez rozdzielczość obrazu i wymagane odświeżanie.

Także napędy dysków wymagają zegarów działających z odpowiednimi częstotliwościami, które są ustalonymi wielokrotnościami częstości obrotowej talerzy dysku. Zegar używany w modemie decyduje o poprawnej szybkości transmisji bitów.

Zegarów może być tyle, ile jest różnych urządzeń charakteryzujących się różnymi szybkościami i wymaganiami. Nie można bowiem przyspieszyć danego urządzenia poza jego ograniczenia konstrukcyjne, a niekiedy nie można również zmienić szybkości działania na inną niż przewiduje standard dla całej grupy takich urządzeń.

## Co to jest superskalowanie?

Chełpiąc się tym, że nowy komputer jest szybszy niż starsze modele, które należy wymienić, producenci często wskazują na superskalowalną (ang. *super-scalar*) wydajność. Cóż jednak naprawdę to oznacza? Po prostu, znaczy to, że jeśli częstotliwość zegarowa w nowym komputerze jest taka sama jak w starym, ten nowy będzie wykonywał programy szybciej. Jeśli w nowym komputerze częstotliwość zegarowa jest dwukrotnie większa niż w starym, to wzrost szybkości wykonywania programów w porównaniu ze starym będzie nie większy niż dwa razy. Oznacza to, że wydajność wzrasta szybciej niż wynika to ze wzrostu częstotliwość zegarowej. Gdyby obowiązywała tu proporcjonalna relacja wydajności i częstotliwości zegarowej, wydajność można by regulować za pomocą tej częstotliwości. Jeśli jest większa, mówi się o superskalowaniu.

Może istnieć wiele przyczyn superskalowalnej wydajności. Zwykle wynika ona z połączenia kilku ulepszeń, takich jak: dodanie większej liczby jednostek wykonujących instrukcje (stałoprzecinkowe i zmiennoprzecinkowe), obsługa nowszych i bardziej wydajnych instrukcji (na przykład MMX i 3DNow!), korzystanie z dłuższych potoków instrukcji czy wreszcie większe pojemności pamięci podręcznej L1 lub L2. Wzrost wydajności może być również spowodowany „spekulatywnym wykonywaniem" przez pewne części procesora instrukcji, na które jeszcze nie nadeszła kolej, lecz przewiduje się ich wykonanie. Takie działanie może być przydatne, jeśli okaże się, że wyniki takich instrukcji są potrzebne — będą one wówczas dostępne od razu. Wzrost wydajności może być również spowodowany użyciem większej liczby procesorów.

Bez względu na przyczyny, superskalowanie w najprostszym znaczeniu powoduje, że nowy komputer działa szybciej niż można się było spodziewać. Jest to na pewno cenna właściwość, lecz nie tak tajemnicza jak sugeruje jej nazwa.

# Magistrale systemowe: ISA, PCI i AGP

Magistrala określa sposób komunikacji pomiędzy poszczególnymi częściami komputera. Wyznacza ona tory sygnałów przesyłanych między elementami funkcjonalnymi maszyny. Pełna definicja magistrali jest wystarczająco szczegółowa, by poszczególni producenci mogli produkować różne części bez obaw, że nie będą one ze sobą współpracować.

Omówimy teraz wiele ważnych standardów określających budowę elementów komputera i jego architekturę. Niezależnie od faktu, że konstrukcja procesora głównego jest głównym czynnikiem rozwoju komputerów, także magistrale odgrywają w tym bardzo znaczącą rolę.

## Pierwotna magistrala ISA

W pierwszym komputerze PC, wyprodukowanym przez firmę IBM, zastosowano procesor mający zaledwie osiem linii danych i dwadzieścia linii adresowych. Był on połączony z innymi podzespołami komputera za pomocą wspólnej magistrali, która zawierała te linie i kilka dodatkowych linii sterujących. Dane przepływały przez magistralę z szybkością określoną przez wewnętrzny zegar procesora. Było to działanie synchroniczne i zupełnie proste.

Ta prostota była zarówno zamierzona, jak i konieczna. W owym czasie nie było możliwe umieszczenie całej potrzebnej pamięci na płycie głównej. Często stosowano karty rozszerzające z układami pamięci. Większość innych podzespołów wymagających wspomagania przez dodatkowe układy scalone również była montowana w postaci kart rozszerzeń. Należały do nich np. sterowniki stacji dyskietek (a później także sterowniki twardych dysków, wprowadzonych w PC/XT). Te zasady obowiązywały również w podsystemie grafiki oraz w portach szeregowych i równoległych. Każdy z takich podzespołów był początkowo taktowany sygnałem o częstotliwości 4,77 MHz. Po zbudowaniu PC/AT, częstotliwość podwyższono najpierw do wartości 6 MHz, a następnie do 8,33 MHz — i taka wartość pozostała do tej pory.

Zanim procesory zaczęły działać znacznie szybciej niż w początkowych komputerach, spośród podzespołów komputera PC wydzielono magistralę obsługującą wejścia i wyjścia (I/O).

### Rozwój magistrali ISA

Magistrala ISA sprawowała się dobrze w komputerze PC/AT. Wkrótce pojawiły się jednak klony komputerów, które mogły działać dużo szybciej. Szybsze procesory zastosowane w tych maszynach wymagały szybszej obsługi wejść i wyjść, a także szybszych

połączeń z układami pamięci systemowej. Konieczny efekt próbowano osiągnąć różnymi metodami. Firma IBM wprowadziła na przykład tzw. *MicroChannel*. Wielu producentów klonów stosowało odmienną konstrukcję, o nazwie *Extended Industry Standard Architecture* (w skrócie *EISA*). Utworzono także grupę producentów, nazwaną *Video Equipment Standards Association* (w skrócie *VESA*), której początkowym zadaniem była pomoc w opracowywaniu nowych standardów podsystemów grafiki, lecz przekształciło się to we wspieranie standardów wielu innych podzespołów używanych w komputerze PC. Oczywiście, podane tu przykłady nie wyczerpują listy wszystkich metod, za pomocą których próbowano ulepszyć stary, dobry standard magistrali ISA.

# Magistrala PCI

Skrót *PCI* pochodzi od angielskiej nazwy magistrali *Peripheral Component Interconnect* (czyli połączenie urządzeń peryferyjnych). Była ona promowana głównie przez firmę Intel jako nowy sposób łączenia podzespołów, który początkowo miał podwyższyć efektywność działania płyty głównej. Później rozszerzono jej zastosowanie również do obsługi połączeń z urządzeniami zewnętrznymi.

Ważną cechą konstrukcyjną magistrali PCI była możliwość przystosowania jej do różnych komputerów lub do innego sprzętu. Nie ograniczało się to tylko do zastępowania przez nią magistrali ISA ani do działania tylko w komputerach zgodnych z PC.

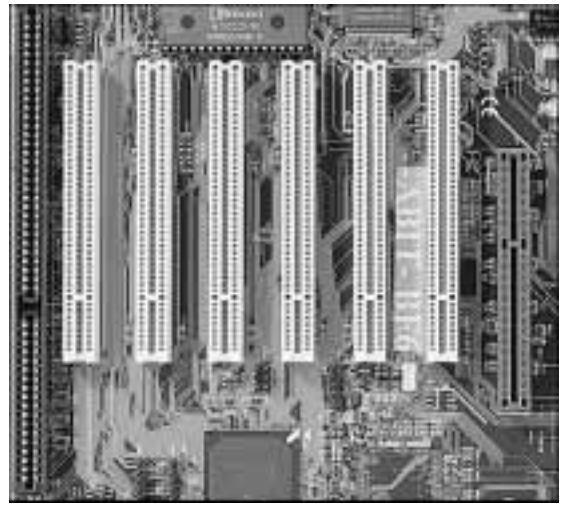
Jak się okaże za chwilę, PCI nie oznacza tylko pojedynczej konstrukcji. Istnieje kilka ważnych odmian tej magistrali, na przykład CompactPCI, której używa się w sterownikach przemysłowych i przyrządach pomiarowych. Podstawowa wersja magistrali PCI jest także stosowana w komputerach wieloprocesorowych (z procesorami x86, Alpha, PowerPC i innymi). Istnieją również cztery warianty magistrali dla komputerów PC, różniące się szerokością magistrali danych i częstotliwością taktowania. Omówimy te warianty, jednak aby uprościć te zagadnienia, nie będziemy szczegółowo prezentować ani standardu CompactPCI, ani zastosowań magistrali PCI w komputerach z procesorami innymi niż x86.

## Podstawy magistrali PCI

Zasadniczo pojawienie się PCI zmusiło do zmiany sposobu traktowania połączeń między poszczególnymi podzespołami komputera PC. Początkowo dotyczyło to tylko tych podzespołów, które były umieszczone wewnątrz jednostki centralnej, lecz w późniejszych wersjach konstrukcyjnych PCI objęło również połączenia z elementami zewnętrznymi.

Magistrala PCI powstała w wyniku poszukiwań bardziej wydajnego połączenia układów scalonych otaczających procesor i doprowadzenia tych połączeń do takiego stanu, by stały się mniej zależne od rodzaju zastosowanego procesora. Później, gdy firma Intel skonstruowała złącze magistrali PCI, wykazało się ono wielkimi zaletami w porównaniu z magistralą ISA.

Ponieważ była to pierwsza próba rozwiązania problemów występujących w magistrali ISA, nową magistralę zaprojektowano zgodnie ze stanem ówczesnej technologii, nie zwracano przy tym uwagi na wszystkie problemy charakterystyczne dla magistrali ISA, a trwające bez zmian aż do dziś. Jednak po zdefiniowaniu magistrali PCI i wdrożeniu jej na szeroką skalę okazało się, że również ona nie jest pozbawiona wad. Nadal jednak jest to magistrala odporna i elastyczna, co udowadniają jej liczne zastosowania i odmiany. Na rysunku 15.8 pokazano fragment płyty głównej z sześcioma złączami PCI.



**Rysunek 15.8.** *Przykład typowej płyty głównej z sześcioma złączami PCI, jednym złączem ISA (pierwsze z lewej) i jednym złączem AGP (pierwsze z prawej)*

## Wydajność magistrali PCI

W porównaniu z magistralą ISA magistrala PCI posiada około połowę aktywnych linii, lecz pośród nich jest dwukrotnie więcej linii danych, dzięki czemu całość może działać czterokrotnie szybciej. Aby zapewnić stabilną pracę magistrali przy tej szybkości, złącza wyposażone są aż w 124 styki, a większość z nich podłączona jest do „masy" lub do jednego z napięć zasilających. Większa szybkość działania przy mniejszej liczbie linii spowodowana jest głównie użyciem tego samego zestawu linii do przekazu danych i adresów (proces ten zachodzi w różnym czasie).

Aby zwiększyć niezawodność, zastosowano także bit parzystości — zarówno dla adresów, jak i dla danych. Magistrala PCI wykrywa zatem błędy adresów i danych. Jeśli taki błąd wystąpi, sterownik magistrali powiadomi o tym urządzenia zaangażowane w transmisję i nakaże im powzięcie decyzji o uruchomieniu procedury korekcji błędu.

Magistrala PCI została zaprojektowana w taki sposób, by nie trzeba było stosować rezystorów dopasowujących (w odróżnieniu od magistrali SCSI). Dzięki temu użytkownik może łatwiej z niej korzystać. Wykorzystano także sygnały odbite od niedopasowanych końców linii. Taka konstrukcja ogranicza jednakże długość pojedynczego segmentu magistrali do zaledwie kilkunastu centymetrów, a także liczbę kart lub innych urządzeń, które można na tym odcinku podłączyć (nie więcej niż 10). Można znacznie wydłużyć zasięg magistrali PCI za pomocą układów mostkowych, montowanych między poszczególnymi segmentami.

## Automatyczna konfiguracja urządzeń na magistrali PCI

W definicji magistrali PCI uwzględniono automatyczną konfigurację podłączonych do niej urządzeń (*plug-and-play*). Każda karta umieszczona w złączu PCI (a także każde urządzenie podłączone do tej magistrali na płycie głównej) musi być wyposażone w lokalną pamięć, przeznaczoną na dane konfiguracyjne. System odczytuje te dane oraz zapisuje je podczas konfiguracji.

## Wiele urządzeń nadrzędnych i współużytkowanie przerwań na magistrali PCI

Można wskazać dwa problemy, z którymi nie mogli sobie poradzić konstruktorzy starszej od PCI magistrali ISA. Jeden z nich to obsługa wielu urządzeń podłączonych do magistrali, które mogą w różnym czasie przejmować kontrolę nad komunikacją w całej magistrali. Drugi związany jest z obsługą wielu urządzeń, które współużytkują te same linie przerwań, kiedy chcą zasygnalizować procesorowi swoje potrzeby. Rozwiązanie problemu wielu urządzeń nadrzędnych w magistrali PCI polega na zastosowaniu jednostki arbitrażu jako niezbędnej części systemu. „Konflikt" przerwań rozwiązano w prosty sposób, pozwalając każdemu urządzeniu na przejmowanie linii przerwań, określając następnie, podczas procesu potwierdzania, które to są urządzenia, a na koniec obsługując żądania.

## Czym jest magistrala podpodłogowa?

Magistrala PCI (zwana także *magistrala podpodłogową*, ang. *mezzanine bus*) była konstruowana jako zastępstwo magistrali ISA, lecz w rzeczywistości stała się czymś innym. Podobnie ja ISA, rozprowadza ona sygnały między procesorem, różnymi urządzeniami na płycie głównej a kartami umieszczonymi w jej złączach. Aby proces taki miał miejsce, magistrala PCI musi buforować sygnały i w niektórych przypadkach zmieniać także ich zależności czasowe.

W przeciwieństwie do magistrali ISA, definicja magistrali PCI nie odwołuje się do żadnego konkretnego procesora ani nawet do rodziny procesorów. Zaletą takiej niezależności jest obecnie fakt, że wielu producentów może wytwarzać karty PCI, których można używać w szerokiej gamie systemów mikrokomputerowych. Szerszy rynek prawie zawsze oznacza zaś niższe ceny produktów, jak również lepsze zaspokojenie popytu — jeśli spojrzeć na to ze strony klientów.

Oprócz tego, standard PCI w jawny sposób zawiera odwołanie do połączeń za pomocą mostków. Mogą to być mostki między magistralami PCI lub między magistralą PCI a innymi standardowymi magistralami, na przykład magistralą pamięci i magistralą ISA w komputerze PC lub magistralą VME w zastosowaniach przemysłowych.

Magistrala PCI została skonstruowana tak, by mogła działać z obowiązującą wówczas pełną szybkością magistrali płyty głównej i w pełni obsługiwać równoległą transmisję danych. Jeśli uwzględni się rok jej powstania, łatwo stwierdzić, że miała ona 32 linie danych, zaś jej częstotliwość taktowania wynosiła 33 MHz; a zatem była ona czterokrotnie szybsza i miała dwukrotnie więcej linii danych niż zastępowana przez nią magistrala ISA.

PCI zyskała wiele zastosowań zaraz po wprowadzeniu na rynek. Wyraźnie potwierdziła swoją wyższość nad poprzednimi standardami magistrali w komputerze PC. Było to tak widoczne, że producenci PC chcieli stosować ją w swoich produktach, a i klienci byli z niej bardzo zadowoleni. Wiele firm zaczęło wstawiać karty PCI w nowe złącza. Przede wszystkim były to karty grafiki, sterowniki dysków twardych i karty sieciowe, ponieważ to w ich przypadku najbardziej potrzebne było zwiększenie szybkości.

Gdy zwiększono częstotliwość taktowania płyty głównej do 66 MHz, magistrala PCI nadal pozostała przy wartości 33 MHz. Gdy procesory rozpoczęły obsługę danych 64-bitowych, PCI nadal obsługiwała tylko 32 bity. Dopiero niedawno standard PCI został rozszerzony i obecnie uwzględnia zarówno wersję 64-bitową, jak i taktowanie częstotliwością 66 MHz. Rozszerzenie było jeszcze cenniejsze, ponieważ zapewniało zgodność z wcześniejszymi i przyszłymi urządzeniami. Można zatem (przy pewnym ograniczeniu wydajności) posługiwać się starszymi kartami 32-bitowymi, taktowanymi częstotliwością 33 MHz w nowych złączach 64-bitowych z zegarem 66 MHz i odwrotnie. Taka obustronna zgodność jest rzeczywiście bardzo rzadko spotykana w historii rozwoju standardów.

To niezwykłe działanie uzyskano w bardzo prosty sposób. Do 64-bitowej magistrali PCI dołączono po prostu dodatkowy segment do złącza, podobnie jak w 16-bitowym rozszerzeniu złącza starej, 8-bitowej magistrali ISA. Obecnie można zatem wstawić starszą kartę 32-bitową do 64-bitowego złącza PCI, a także nową kartę 64-bitową do starszego złącza 32-bitowego. Sztuczka polega na tym, że nowsze karty muszą się same skonfigurować w taki sposób, by działać jako karty 32-bitowe w 32-bitowych złączach PCI.

Podobnie działa zegar 66 MHz ze starszymi kartami PCI, ponieważ sterownik danego segmentu magistrali PCI rozpoznaje obecność wolniejszych kart i w przypadku ich wykrycia zmniejsza częstotliwość taktowania w tym segmencie. Z drugiej strony, każda nowsza karta PCI może działać wolniej. Definicja magistrali PCI przewiduje, że karty lub magistrale mogą być taktowane z dowolną częstotliwością, aż do 0 MHz. Oczywiście, większość korzysta z maksymalnej dozwolonej dla danego standardu częstotliwości taktowania, bowiem w przeciwnym wypadku nie byłoby możliwe osiągnięcie odpowiedniej wydajności.

Odstępstwem od tej zasady jest system skonfigurowany w taki sposób, by oszczędzać energię, gdy nie pracuje on z maksymalną wydajnością. W takich sytuacjach można by spowalniać magistralę PCI i uzyskać tym samym dodatkową oszczędność, lecz niewiele współcześnie produkowanych komputerów potrafi to zrobić. Maszyny pozwalające na taką dodatkową oszczędność musiałyby posiadać specjalną konstrukcję.

Dlatego właśnie, mimo iż użycie mieszaniny kart PCI o różnych standardach w różnych złączach jest dozwolone, prowadzi do znacznego spadku wydajności magistrali. Można zatem oczekiwać pojawienia się komputerów PC wyposażonych w segmenty nowej magistrali PCI o większej szybkości i w segmenty magistrali taktowanej wolniej. Jeśli komputer ma nową, szerszą magistralę (prawdopodobnie na odcinku taktowanym szybciej), użytkownicy powinni umieścić wszystkie starsze, 32-bitowe karty w wolniejszych segmentach i zarezerwować szybsze segmenty, o większej szerokości tylko dla tych kart, które rzeczywiście tego wymagają.

Na zakończenie warto zauważyć, że karty PCI produkowane są z dwiema różnymi wartościami napięć znamionowych. Wszystkie starsze karty pracują z napięciami 5 V, zaś nowsze używają napięć 3,3 V. Oznacza to, że dzięki mniejszej wartości napięcia zasilającego układy scalone w nowszych kartach, wydziela się mniej ciepła i mogą one działać szybciej. Różnice konstrukcyjne polegają na różnym położeniu wycięć ustalających w złączach, co zapewnia prawidłowy montaż kart i uniemożliwia popełnienie pomyłki. Specyfikacja zapewnia także możliwość produkcji kart „uniwersalnych", które mogą być wstawiane do dowolnego złącza, niezależnie od występującego w nim napięcia zasilania.

Najnowszym standardem w rodzinie PCI jest PCI-X; standard ten zostanie przedstawiony jednak nieco dalej.

# PCI jako oś „północ-południe" w komputerze PC

Ze względu na to, że PCI jest magistralą „podpodłogową" i dzięki temu może służyć jako połączenie między różnymi magistralami, można ją traktować jako uniwersalne połączenie różnych części komputera, zarówno wewnątrz jednostki centralnej, jak i poza nią. Firma Intel wykorzystała tę zaletę w celu rozdzielenia różnych funkcji i w nowszych konstrukcjach dokonała podziału zestawów pomocniczych układów scalonych płyty głównej na dwie grupy. Innowacja polega na zgrupowaniu jednych funkcji w układzie zwanym *mostkiem północnym*, a pozostałych w *mostku południowym*. Połączenie między tymi układami zapewnione jest dzięki magistrali PCI.

## Układ scalony mostka północnego

Procesor połączony jest z pamięcią główną przez szybszą i szerszą ścieżkę danych niż ta, która użyta została do połączenia z magistralą PCI. Ta szybsza magistrala nazywana jest magistralą *zewnętrzną*, *systemową* lub *węzłową*. Procesor nie może być połączony bezpośrednio z modułami pamięci, zatem wymagane są przynajmniej jakieś bufory i dekodery adresów. Nie może on również być bezpośrednio połączony z magistralą PCI. Zadaniem mostka północnego jest zatem buforowanie (wzmacnianie) sygnałów procesora, a następnie przekazywanie ich we właściwe miejsca.

Procesor „rozmawia" z urządzeniami zewnętrznymi i z pamięcią poprzez te same linie. Zmienia tylko poziom napięcia na jednym z wyprowadzeń sterujących, by wskazać, że w danym momencie mają być dostępne urządzenia I/O albo pamięć. Układ scalony mostka północnego korzysta z sygnału sterującego przy rozdzielaniu komunikatów dotyczących pamięci i tych, które dotyczą urządzeń zewnętrznych.

Wymagane do tego elektroniczne układy interfejsu (zwane czasami *układami towarzyszącymi* płyty głównej, ang. *glue logic*) zostały scalone w jeden bardzo duży układ (prawie tak skomplikowany jak procesor), który nazywany jest mostkiem północnym. Obecność tych układów w mostku północnym umożliwia także obsługę bardzo szybkiej magistrali specjalnego przeznaczenia, zwanej *zaawansowanym portem graficznym* (ang. *Advanced Graphic Port*, w skrócie *AGP*).

Należy pamiętać, że podany tu opis dotyczy najnowszych rozwiązań konstrukcyjnych w zestawach układów pomocniczych dla płyt głównych firmy Intel. Istnieją inni producenci podobnych zestawów, którzy oferują podobne rozwiązania, a także grupa producentów wytwarzających całkowicie odmienne zestawy. Oprócz tego, Intel oferuje różne wersje zestawów układów pomocniczych optymalizowanych do różnych celów. Podczas opisu tych zagadnień pominiemy te różnice i omówimy hipotetyczny zestaw układów pomocniczych, obsługujący wszystkie najnowsze rozwiązania (być może taki zestaw nigdy nie zostanie wyprodukowany).

# AGP, czyli boczna ścieżka do ulepszonej grafiki

Magistrala AGP jest w istocie wyspecjalizowaną magistralą PCI, zawierającą szereg ulepszeń wymaganych przy obsłudze podsystemu graficznego o bardzo dużej wydajności. Oczywiście, taki podsystem graficzny będzie sterowany za pomocą dobrego akceleratora graficznego. Bez wątpienia, musi on również zawierać duży bufor kadru, umożliwiający uzyskanie na ekranie obrazu o bardzo dużej rozdzielczości i głębi kolorów. Taki podsystem graficzny wymaga od magistrali szybkiego połączenia z procesorem i bezpośredniego połączenia z niektórymi obszarami pamięci głównej, by wymiana informacji z pamięcią nie angażowała procesora. Magistrala AGP spełnia te wymagania.

Akcelerator graficzny AGP może być zamontowany na płycie głównej albo może mieć postać karty wstawionej do specjalnego złącza AGP. Obecnie najczęściej spotyka się rozwiązanie w postaci karty, ponieważ dzięki temu producenci mogą wybierać spośród różnych kart grafiki, które przystosowane są do różnych celów (między innymi chodzi tu o na przykład o zrównoważenie kosztów i wydajności przy generowaniu grafiki trójwymiarowej).

Złącze AGP w komputerze zastępuje zawsze jedno tradycyjne złącze PCI i umieszczane jest najbliżej procesora.

## AGP 1X, 2X, 4X, 8X i Pro

Magistrala AGP, podobnie jak PCI, z której się wywodzi, rozwinęła się z biegiem czasu i w związku z tym powstało kilka związanych z nią standardów. Pierwotnie obsługiwała ona tzw. pojedynczą (AGP 1X) i podwójną (AGP 2X) szybkość przetwarzania. Obecnie najszybszą wersją jest AGP 8X. Większość kart grafiki AGP obsługuje obecnie standard 2X, podobnie jak większość aktualnie produkowanych płyt głównych. Można się jednak spodziewać, że wkrótce pojawią się płyty główne i karty grafiki obsługujące standard 8X.

Na rysunku 15.9 pokazano fragment płyty głównej ze złączami PCI i AGP.

Różnice między wspomnianymi standardami częściowo polegają na stosowaniu innych napięć roboczych, a częściowo na sposobach wykorzystania sygnału zegarowego w synchronizacji danych. W najszybszych magistralach AGP zastosowano wiele pomysłów pochodzących z nowego standardu Rambus, dotyczącego bardzo szybkiej obsługi pamięci głównej. W złączu AGP do obsługi kart z różnymi napięciami znamionowymi zastosowano sprawdzone wcześniej rozwiązania z magistrali PCI.



**Rysunek 15.9.**
*Pokazany tu fragment rysunku 15.8 obejmuje złącze PCI (z lewej) i AGP (z prawej) w celu ukazania różnic w budowie mechanicznej tych złączy*

Szybkość komunikacji z podsystemem AGP jest ważna, lecz nie jest to jedyny znaczący czynnik. Równie ważna jest wydajność przetwarzania systemu. Przy uzyskiwanym obecnie stopniu integracji układów scalonych nie jest jeszcze możliwe zamontowanie na niewielkiej karcie PCI lub AGP wszystkich wymaganych podzespołów w celu uzyskania wymaganych parametrów. Dużo układów umieszczonych na jednej karcie może powodować jej przegrzewanie lub przeciążenie złącza AGP ponad dopuszczalną wartość.

Sposobem rozwiązania tego typu problemów stał się następny standard, zwany AGP Pro. Zaproponowano w nim, by karty AGP były „grube" (czyli miały wiele podzespołów oraz dodatkowo dołączane moduły). Karty tego typu zajmują więcej miejsca niż pozwala na to standardowy odstęp między złączami PCI, wynoszący około 2 cm.

Karty AGP Pro mogą zajmować więcej miejsca niż przewiduje się zwykle na dwa, a nawet trzy złącza (złącze AGP i sąsiednie złącza PCI). W każdym przypadku producent może skorzystać ze złącza AGP i dodatkowo ze złącz PCI. Jeżeli posiada się kartę umieszczoną w więcej niż jednym złączu, można uzyskać dodatkowy dostęp do zasilania oraz do niektórych linii magistrali PCI, które nie występują w magistrali AGP.

Dotychczas powstała wstępna definicja standardu AGP Pro i pojawiło się kilka deklaracji producentów, którzy wyrazili chęć wytwarzania takich kart. W rzeczywistości na rynku nie ma jeszcze kart tego typu i dopiero gdy się pojawią, można będzie ocenić wydajność wyposażonych w nie komputerów.

## Czym jest GART?

Konstrukcja AGP miała umożliwić kartom grafiki bezpośredni dostęp do głównej pamięci systemu. Nie miałoby jednak sensu zezwalanie na dostęp do tej całej pamięci, która użytkowana jest częściowo przez uruchomione programy (część na sam kod programu, a część na jego dane). Tylko mała część dostępnej w systemie pamięci RAM może być używana przez kartę grafiki. W przypadku AGP jest ona przeznaczona na przechowywanie tekstur obrazów, dzięki czemu może z nich korzystać każdy lokalny bufor wyświetlanego w danym momencie obrazu. Jeśli trzeba skorzystać z tekstury, można ją pobrać z pamięci i użyć w odpowiednim fragmencie obrazu.

W definicji magistrali AGP występuje zatem uwaga, że karta grafiki może mieć dostęp tylko do fragmentu pamięci głównej. Ten fragment oglądać można w specjalnym oknie AGP. *Tablica odwzorowań adresów grafiki* (ang. *Graphic Address Remapping Table*, w skrócie *GART*) decyduje o tym, do którego fragmentu pamięci dostęp będzie możliwy.

Tablica GART tworzona jest albo w lokalnej pamięci na karcie AGP, albo w pamięci głównej. Określa ona obszary pamięci głównej, do których karta AGP może mieć bezpośredni dostęp. Zmiana zawartości tablicy powoduje zmianę położenia tych obszarów. Na karcie AGP znajdują się specjalne układy, wskazujące położenie tych obszarów i korzystające z ich zawartości poprzez mostek północny.

# Wiele magistral PCI w jednym komputerze

Wspomnieliśmy już, że magistrala AGP jest w rzeczywistości zamaskowaną magistralą PCI. Jest ona jednak całkowicie oddzielona od tej magistrali, co oznacza, że każda karta AGP może porozumiewać się z procesorem lub korzystać z pamięci głównej niezależnie od stanu magistrali PCI.

Mostek północny można zbudować w taki sposób, by obsługiwał nie tylko samą magistralę AGP i jedną PCI. Nie ma przeciwwskazań do posiadania kilku magistral PCI. Przede wszystkim funkcje mostka północnego obejmują połączenia między magistralami PCI i AGP a magistralą zewnętrzną, a także połączenia między kilkoma magistralami PCI, jedną AGP a magistralą zewnętrzną.

Główną przyczyną tego, że pojawienie się takich konstrukcji komputerów PC stanowi jedynie kwestię czasu, są dwa standardy magistral PCI: 32-bitowej, taktowanej zegarem 33 MHz i 64-bitowej, taktowanej zegarem 66 MHz. Dzięki temu można stosować mieszaninę starszych i nowszych kart bez utraty wydajności. Można także oczekiwać, że pewnego dnia, oprócz dodatkowych magistral PCI, w komputerze pojawi się także magistrala PCI-X, wychodząca z mostka północnego.

Jako alternatywne rozwiązanie proponuje się, by szybka i szeroka magistrala PCI (lub PCI-X) wychodząca z mostka północnego, po przejściu przez kilka szybkich i szerokich złącz, kończyła się na mostku PCI-PCI. Mostek ten ma zapewniać połączenie z innym segmentem magistrali PCI, charakteryzującej się tradycyjnymi właściwościami.

Powodem, dla którego stosuje się wiele magistral PCI wychodzących bezpośrednio z mostka północnego jest możliwość obsługi większej liczby złączy, niż może obsługiwać pojedyncza magistrala. Zwykle, gdy uwzględnia się obciążenie wnoszone przez różne elementy płyty głównej, pojedynczy segment magistrali PCI nie powinien mieć więcej niż cztery złącza. Większa ich liczba jest potrzebna przede wszystkim w serwerach lub komputerach PC, które wykorzystywane są w zastępstwie dużych maszyn.

Istnieje jeszcze jeden sposób uzyskania większej liczby magistral PCI w jednym komputerze. Jest to połączenie mostkowe takich magistral. Opisywany na początku książki przykładowy komputer PC ma poczwórną kartę adaptera grafiki. Karta ta ma wbudowany układ mostkowy PCI-PCI, a cztery karty graficzne podłącza się do drugiej magistrali PCI, poza mostkiem łączącym ją z magistralą PCI na płycie głównej.

W zaprezentowanej tu tabeli podano różne standardy interfejsów używanych dawniej i obecnie.

| Standard | Zastosowanie | Przepustowość |
|---|---|---|
| ISA | Karty dźwiękowe | Od 2 MB/s do 8,33 MB/s |
| | Adaptery SCSI | |
| | Karty graficzne | |
| | Karty sieciowe | |
| | Karty z portem gier | |
| | Modemy wewnętrzne | |
| EISA | Karty sieciowe | 33 MB/s |
| | Adaptery SCSI | |
| PCI | Karty graficzne | 266 MB/s |
| | Adaptery SCSI | |
| | Karty dźwiękowe | |
| | Karty sieciowe | |
| | Modemy wewnętrzne | |
| AGP 4X, 8X | Karty graficzne | od 528 MB/s do 2 GB/s |

# Mostek południowy

Na „południowym" krańcu magistrali PCI mieści się układ scalony zwany *mostkiem południowym*. Zawiera on pozostałe podzespoły wchodzące w skład tzw. *logiki towarzyszącej płyty głównej* (czyli zestawu układów pomocniczych), których nie umieszczono w mostku północnym. Mostek południowy zawiera układy interfejsu odpowiedzialne za przekazywanie sygnałów między magistralą PCI a znacznie od niej wolniejszą magistralą ISA i innymi interfejsami występującymi w komputerze PC. Współczesne płyty główne mają niewiele złącz ISA, głównie z tego powodu, że karty produkowane wcześniej w tym standardzie zostały zmodyfikowane i współpracują obecnie z magistralą PCI.

Do układu mostka południowego podłączone są stacje dyskietek. Układ ten obsługuje zazwyczaj także dwa kanały EIDE (główny i wtórny), do których można podłączać po dwa urządzenia EIDE, chociaż mógłby obsługiwać cztery takie kanały. W mostku południowym są również oddzielne magistrale do obsługi klawiatury, myszy, standardowego portu równoległego (ECP lub EEP) oraz co najmniej jednego portu szeregowego. Oddzielne magistrale przewidziane są do obsługi złącz USB, a także IEEE 1394 (FireWire). Układ mostka południowego ma zawierać również interfejs obsługujący złącza kart PC — będzie to zapewne 32-bitowy interfejs CardBus z magistralą Zoomed Video, podłączoną bezpośrednio do podsystemu grafiki.

Jeśli w komputerze jest kilka magistral PCI, do każdej z nich można podłączyć oddzielny układ mostka południowego. Nie jest to jednak obowiązkowe.

Jeśli w komputerze PC mają być złącza ISA, będą one podłączone do mostka południowego. Magistrala ISA jest zatem w tej architekturze dodatkiem do magistrali PCI i nie jest połączona w jakiś bezpośredni sposób z procesorem, co było charakterystyczne w jej pierwotnej wersji.

Interfejsy ISA, IDE oraz CardBus taktowane są częstotliwością 8,33 MHz (czyli jest to dokładnie cztery razy mniej niż w standardowej magistrali PCI). Jedynym wyjątkiem jest tu kanał EIDE ATA66, w którym zastosowano zegar o częstotliwości 16,7 MHz, lecz działa on tylko wtedy, gdy urządzenia podłączone są za pomocą specjalnego kabla zgodnego ze specyfikacją ATA66. Pozostałe porty podłączone do mostka południowego korzystają z mniejszych i różnych częstotliwości taktowania (łącze Zoomed Video między kartą PC a sterownikiem wizyjnym działa z częstotliwością 33 MHz, niezależnie od częstotliwości 8,33 MHz stosowanej na odcinku między kartą PC a mostkiem południowym).

## Co zyskano dzięki oddzieleniu mostka północnego od południowego?

Dzięki wprowadzeniu standardowego podziału układów elektronicznych potrzebnych w komputerze PC na mostek północny i południowy, firma Intel umożliwiła wykonanie wielu interesujących dodatków. Jednym z nich jest stacja dokująca, do której można podłączyć komputer PC (najczęściej będzie to komputer przenośny). Stacja dokująca ma własny zestaw urządzeń peryferyjnych, które wspomagają funkcje podłączonego komputera.

Inna możliwość rozszerzenia wynika z tego, że mostek południowy może być produkowany przez inną firmę niż mostek północny. Dopóki obydwa układy poprawnie współpracują z magistralą PCI, obydwa będą także bez problemów współpracować ze sobą.

Po wbudowaniu do komputera wielu mostków południowych (i tylko jednego mostka północnego), można podłączać wiele urządzeń wskazujących obsługiwanych poprzez oddzielne porty, a także mieć wiele takich portów.

## PCI-X, czyli kolejna wersja PCI

Nowa, szybka magistrala PCI o szerokości 64 bitów, zwana PCI-X jest czterokrotnie szybsza niż magistrala tradycyjna. Okazuje się, że w pewnych zastosowaniach jest to niewystarczające. Z myślą o dokonaniu kolejnego skoku technologicznego, w grupie producentów o nazwie PCI Special Interest Group (w skrócie PCI-SIG) rozważa się wprowadzenie rozszerzonej postaci protokółu PCI, również zwanego PCI-X.

PCI-X jest rozszerzeniem zgodnym z istniejącą magistralą PCI. Ta nowa architektura działać ma z częstotliwością do 133 MHz, dzięki czemu szybkość transmisji strumieniowej sięgać będzie powyżej 1 GB/s, niezależnie od możliwości wykrywania błędów. Taka przepustowość jest bardzo ważna w przemysłowych serwerach, w których stosuje się następujące rozwiązania:

- gigabitowy Ethernet,
- interfejsy Fibre Channel,
- interfejsy Ultra3 SCSI,
- połączenia klastrów.

Standard PCI-X nie będzie konkurował ze standardem AGP w dziedzinie zastosowań graficznych.

Ten nowy, bardzo elegancki protokół jest wspólnym dziełem firm Compaq, Hewlett-Packard oraz IBM. Prace miały doprowadzić do kontynuacji stosowania rozwiązań w stylu PCI w komputerach PC, gdy objętość przekazywanych danych przewyższa przepustowość najszybszych magistral PCI wykonanych zgodnie z obowiązującymi standardami. Największe wrażenie robi fakt, że nowa, większa przepustowość magistrali pozostaje w zgodzie z istniejącymi urządzeniami, obsługującymi 64 bity i zegar 66 MHz. Kluczowym zagadnieniem rozwiązanym przez konstruktorów były ograniczenia klasycznego protokołu magistrali PCI, które przy wyższych szybkościach stawały się „wąskim gardłem"; szczęśliwie ograniczenia te można było stosunkowo łatwo usunąć.

Ograniczenie szybkości w konwencjonalnych systemach PCI wynika stąd, że w każdym cyklu zegara każde urządzenie musi obserwować ruch na magistrali i decydować, czy przekazywana tam informacja jest przeznaczona dla niego, a jeśli tak jest — przejmować dane. Po zwiększeniu częstotliwości zegarowej z 33 MHz na 66 MHz okazało się to kłopotliwe, chociaż możliwe. Dalsze podwojenie częstotliwości przy obecnym stanie technologii produkcji układów elektronicznych po prostu nie jest już możliwe.

Proponowane rozwiązanie tego ograniczenia jest bardzo proste. W każdym urządzeniu należy dodać rejestr (na chwilowe przechowywanie danych), który przechwytuje dane poza cyklami zegarowymi, niezależnie od tego, czy są one przeznaczone dla danego urządzenia, czy nie. Przed pojawieniem się następnego cyklu zegara urządzenie sprawdza dane i podejmuje wymagane działania. Dzięki temu czas na podjęcie decyzji i działanie podwaja się. Oznacza to także, że odpowiedź pojawia się nie po jednym, lecz po dwóch cyklach zegara za pobudzeniem — czyli do całej transakcji dodawany jest jeden cykl zegarowy.

Pozwoliło to na zwiększenie częstotliwości taktowania do 166 MHz. Dodanie jednego cyklu obniża wydajność zaledwie o 10%, ponieważ większość transakcji na magistrali PCI wymagało dziewięciu cykli, a teraz wymaga dziesięciu. Ogólny wzrost wydajności nie jest zatem dwukrotny, lecz nieco mniejszy.

Aby można było zastosować protokół PCI-X, każde urządzenie podłączone do magistrali musi być zgodne z jego wymaganiami. Ważną część definicji PCI-X stanowi zdolność do wykrywania tej właściwości. Podobnie jak przy poprawce ATA66 do magistrali EIDE, w definicji PCI-X zezwala się na stosowanie tego protokołu tylko wtedy, gdy wszystkie urządzenia mogą bez kłopotów pracować przy wyższej częstotliwości taktowania. Ponieważ nie zmieniono ani mechanicznej konstrukcji złączy, ani elektrycznej i logicznej struktury interfejsu, do magistrali PCI-X można podłączać

starsze karty — podobnie jak nowe karty PCI-X podłączać można do starszych złączy PCI. W takich przypadkach wszystkie urządzenia będą działać zgodnie ze starszą definicją magistrali PCI.

Jedną z zalet architektury PCI jest zgodność wstecz. Dotyczy to również PCI-X, ponieważ jej wsteczna zgodność ze starszymi urządzeniami PCI wynika z następujących cech:

- Karty PCI-X mogą być używane w konwencjonalnych systemach PCI, jednak wówczas ich domyślna szybkość wynika z częstotliwości taktowania danej magistrali (33 MHz lub 66 MHz).
- Konwencjonalne karty PCI mogą być używane w systemach PCI-X. Działają one wówczas ze swoją znamionową szybkością (33 MHz lub 66 MHz).
- W systemach PCI-X można włączać karty PCI i PCI-X do tej samej magistrali, lecz szybkość działania takiej magistrali wynika z szybkości działania najwolniejszej karty.

Grupa PCI-SIG dość szczegółowo informuje o pracach nad PCI-X. Od czasu powstania, w czerwcu 1992 roku, zachowuje ona charakter organizacji stworzonej przez producentów, utrzymującej specyfikacje magistrali PCI. Kieruje nią rada nadzorcza, w skład której wchodzą obecnie przedstawiciele największych firm: AMD, Compaq, Hewlett-Packard, IBM, Intel, Microsoft, Phoenix Technologies, ServerWorks oraz Texas Instruments.

# PXI-X i 3GIO

Skrót 3GIO powstał od angielskiej nazwy *Third-Generation I/O* (czyli wejścia-wyjścia trzeciej generacji). Grupa PCI-SIG podpisała z grupą roboczą Arachoe porozumienie postulujące utrzymywanie i rozpowszechnianie specyfikacji architektury 3GIO. Grupa robocza Arachoe ma status niezależnego zrzeszenia producentów, w skład którego wchodzą firmy: Compaq, Dell, IBM, Intel i Microsoft. Zajmuje się ona tworzeniem propozycji standardów dla grupy PCI-SIG.

3GIO oznacza szeregowe połączenie I/O, w którym zmniejszono liczbę styków w interfejsie. Jego cele są następujące:

- wzrost efektywności finansowej,
- maksymalizacja przepustowości pojedynczej linii interfejsu,
- zapewnienie wysokiej skalowalności,
- zabezpieczenie inwestycji poczynionych przez klienta,
- ułatwienie przenikania się technologii.

Można zapytać, dlaczego powiązano 3GIO z PCI-X? Kluczowy problem wiąże się z tym, że 3GIO wzmacnia model programowy PCI. Jest ono rozważane jako rozszerzenie rodziny PCI. W rzeczywistości grupa PCI-SIG rozważa nadanie 3GIO nazwy bardziej odpowiadającej „stylowi" PCI. Powstaje poważne pytanie, czy 3GIO pokrywa się z PCI-X?

Odpowiedź brzmi: **nie**. Standard PCI-X jest zoptymalizowany pod kątem aplikacji wymagających bardzo dużej przepustowości, które działają na serwerach lub na stacjach roboczych. Z drugiej strony, 3GIO ma charakter ogólny i ma prowadzić do optymalizacji wydajności **szeregowej** magistrali I/O w aplikacjach o mniejszym znaczeniu. Obydwa standardy są po prostu przeznaczone dla innych segmentów rynku.

W standardzie 3GIO zastosowano model warstwowy w celu zapewnienia różnych połączeń w operacjach wejściowo-wyjściowych przy dużej szybkości transmisji. Składa się on z pięciu warstw. Są to:

- Warstwa konfiguracyjna, obejmująca model *plug-and-play* przeniesiony z PCI.
- Warstwa oprogramowania, obejmująca model programu sterującego PCI.
- Warstwa transakcyjna, korzystająca z protokołu pakietowego.
- Warstwa łącza danych, zapewniająca integralność danych.
- Warstwa fizyczna, która stanowi podstawę łączy 3GIO, zawierających dwie pary linii sterowanych różnicowo i pełniących rolę pary nadawczej i odbiorczej. Wbudowany zegar, który taktuje przepływ danych, umożliwia początkową szybkość transmisji wynoszącą 2,5 GB/s, z możliwością jej powiększenia do 10 GB/s.

Jedną z oczywistych zalet zastosowania warstw jest fakt, że wpływ przyszłych żądań zmian szybkości transmisji oraz metod kodowania ogranicza się tylko do warstwy fizycznej. Konstrukcja produktu końcowego może być zatem uproszczona, zaś jego wydajność — podwyższona.

W czasie trwania prac nad tą książką nie istniały jeszcze liczące się prototypy, które demonstrowałyby faktyczne działanie 3GIO w rzeczywistym środowisku. Ocenia się, że produkty 3GIO pojawią się na rynku pod koniec roku 2003.

# Ocena wydajności za pomocą testów

Kiedy chcemy porównać wydajność posiadanego systemu z innymi, korzystamy głównie z różnych zestawów dostępnych na rynku testów (ang. *benchmarks*). Chodzi przede wszystkim o testy zestawiające wydajność różnych urządzeń i oprogramowania. Różne zestawy testów korzystają z różnych standardów i mechanizmów, co może prowadzić do zamierzonych lub niezamierzonych odchyleń oceny na korzyść bądź na niekorzyść konkretnego produktu. Z tego właśnie powodu, przed porównywaniem wyników należy dokładnie dowiedzieć się, do czego służy dany zestaw.

Najbardziej popularne programy testujące można pobrać ze stron o następujących adresach sieciowych:

- Ziff Davis CPU Mark 99: *ftp://ftp.zdnet.com/zdbop/winbench/cpumk99.exe*
- Cli Benchmark: *http://www.ncpro.com/clibench/download/clibenchsmp/zip*

- Wintune 98: *ftp://wintune.winmag.com/wintune_43.exe*
- Winbench 99: *ftp://ftp.zdnet.com/pub/zdbop/winbench/wb9911up.exe*
- SiSandra 2000: *http://www.simtel.net/pub/simtelnet/win95/util/san649.zip*

## FLOPS, SPEC, MIPS i BogoMips

FLOPS jest skrótem od angielskiej nazwy *floating-point operations per second* (czyli operacje zmiennoprzecinkowe na sekundę). Jest to jednostka wydajności powszechnie stosowana w testach oceniających wydajność operacji zmiennoprzecinkowych w procesorze. Dokładniej rzecz ujmując, jest ona miarą wydajności jednostki zmiennoprzecinkowej procesora.

Operacje zmiennoprzecinkowe obejmują dowolne obliczenia na liczbach ułamkowych. Występują one szczególnie licznie w obliczeniach naukowych i w programach obsługujących grafikę trójwymiarową.

Ponieważ FLOPS nie uwzględnia czynników wynikających z rzeczywistych okoliczności, takich jak na przykład obciążenie, nie jest traktowany jako dobre narzędzie pomiarowe. Z tego właśnie powodu konsorcjum producentów utworzyło organizację nazwaną *Standard Performance Evaluation Corporation* (w skrócie *SPEC*), która ma zapewniać bardziej wiarygodne mierniki wydajności.

MIPS jest skrótem od angielskiego określenia *million instructions per second* (czyli milion instrukcji na sekundę). Za pomocą tej właśnie liczby (tj. jednego miliona) wyrażana jest liczba instrukcji kodu maszynowego, które są wykonywane przez system w ciągu jednej sekundy. Innymi słowy, jest to miara wydajności obliczeniowej procesora i niczego więcej. Oznacza to, że nie zawiera ona oceny wydajności rzeczywistej, ponieważ rzeczywiste aplikacje często ograniczone są przez wiele czynników nie związanych z procesorem. Można jednak wykorzystywać MIPS do ogólnej oceny szybkości działania systemu.

BogoMips stosuje się w systemie Linux. Każde jądro Linuksa wymaga użycia pętli czasowej skalibrowanej z szybkością działania procesora w systemie. Aby uzyskać kalibrację, jądro musi zmierzyć podczas procesu rozruchu szybkość działania tej pętli. Wartość BogoMips podaje informację o ogólnej szybkości działania procesora. Wartość ta nie stanowi jednak rzetelnej oceny ogólnej wydajności całego systemu.

Więcej informacji na temat wartości BogoMips można znaleźć w dokumentach HOWTO, pod adresem *http://www.linuxdoc.org/HOWTO/mini/BogoMips*.

# Podsumowanie

W tym rozdziale bardziej szczegółowo zaprezentowano płytę główną, która stanowi rdzeń każdego komputera PC. W następnych rozdziałach omówione zostaną m.in. proces rozruchu, podczas którego komputer się „ożywia" oraz pamięć, dzięki której działanie takie jest możliwe.