9-10 1986

informatyka

Prof. Vasilii Zakharov o lokalnej sieci ETHERNET

Cm* — komputer o strukturze hierarchicznej

Litery polskie dla IBM PC

WYDAWNICTWO MOT SIGMA

Cena 120 zł

informatyka

9-10 1986

Nr 9-10
Miesięcznik Rok XXI
Wrzesień-październik 1986

Organ Komitetu Informatyki MNSZWiT oraz Komitetu Naukowo-Technicznego NOT ds. Informatyki

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

Zakł. Graf. „Tamka". Zam. 0677-1300/86. Obj. 4,0 ark. druk. Nakład 8350 egz. P-75.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 120 zł
Prenumerata roczna 1440 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

W NAJBLIŻSZYCH NUMERACH:

- Wojciech Cellary o systemach wielomikroprocesorowych
- Krzysztof Zieliński, Jadwiga Indulska i Roman Krasowski o lokalnej sieci komputerowej UMMLAN-2
- Paweł Maćków o komunikacji szeregowej w mikrokomputerach IBM PC i kompatybilnych
- Jacek Ambroziak o systemach ekspertowych
- Stanisław Wrycza o przesłankach i uwarunkowaniach modelowania konceptualnego
- Piotr Strutyński o wielodostępnych systemach operacyjnych dla IBM PC/XT
- Daniel Tabak o mikroprocesorach Intel 80386 i nowych mikroprocesorach 32-bitowych

VASILII ZAKHAROV
Uniwersytet Genewski
Genewa
Szwajcaria

Lokalna sieć komputerowa ETHERNET (I)

Celem tego artykułu jest omówienie właściwości, elementów składowych i obecnego stanu rozwoju lokalnej sieci komputerowej Ethernet, zgodnej zarówno z modelem architektury systemu otwartego ISO (ang. International Organization for Standardization), jak i z normą IEEE 802.3. Podstawą tego przeglądu będą rzeczywiste problemy praktyczne występujące w dużych środowiskach użytkowników. Przed przejściem do zagadnień technicznych, celowe jest jednak krótkie przedstawienie szerszego kontekstu i uwarunkowań rozwoju technologii lokalnych sieci komputerowych, na tle całej dziedziny telekomunikacji.

Zdefiniowanie lokalnej sieci komputerowej jest znacznie trudniejsze niż zdefiniowanie sieci rozległej. Rozległa sieć komputerowa (ang. wide area network, WAN) obejmuje szerszy obszar geograficzny, środowisko jej użytkowników jest duże i zróżnicowane, a transmisja danych następuje na wspólnej częstotliwości nośnej, zarówno na poziomie usług (ang. services), jak i rzeczywistych ośrodków transmisji, którymi są zazwyczaj linie telefoniczne lub kanały akustyczne jakiegoś innego multipleksowanego ośrodka. Typowa sieć lokalna obejmuje znacznie mniejszy obszar o maksymalnych odległościach nie przekraczających 10 km, lecz nie jest to zasadnicze ograniczenie. Wtedy szybkość transmisji jest znacznie większa, rzędu 10—100 Mb/s, choć istnieją sieci lokalne o znacznie mniejszych szybkościach.

Wydaje się, że najważniejszym czynnikiem odróżniającym sieci lokalne od sieci rozległych i innych sieci komputerowych jest cel ich stosowania oraz problem częstotliwości nośnej. Sieci rozległe powstały, aby zaspokoić potrzebę terminalowego dostępu do oddalonych, dużych komputerów macierzystych (ang. mainframe hosts), a wymagania na usługi pocztowe powstały później. W rzeczywistości, odkrycie, że usługi pocztowe zdominowały cały ruch sieciowy, było zaskoczeniem nawet dla twórców takich sieci, jak np. ARPA. Z drugiej strony, cała technologia i architektura sieci rozległych jest oparta na znanych właściwościach kanałów telekomunikacyjnych z lat siedemdziesiątych, z uwzględnieniem ich stopy błędów. W czasie rozwoju tej technologii wyłoniła się znana norma X.25 i architektura sieci z komutacją pakietów.

Celem tworzenia lokalnych sieci komputerowych jest oczywiście także zdalny dostęp terminalowy i usługi pocztowe, ale równie ważna jest potrzeba współdzielenia zasobów i łączenia urządzeń inteligentnych — nie tylko terminali. Tak więc, wymagania związane z transferem plików nie są nieodzowne. Jeśli ponadto przyjmuje się, że dla sieci lokalnych nie jest konieczne stosowanie kanałów o wspólnej częstotliwości nośnej, to nie ma powodu, aby w tych sieciach stosować bardzo skomplikowane protokoły używane w sieciach rozległych.

Do podstawowych wymagań, które powinny być spełnione przez lokalną sieć komputerową należy zaliczyć:

- dużą szybkość transmisji i przepustowość,
- mały koszt dołączania urządzeń i prosty sposób realizacji połączeń,
- możliwość dołączenia dużej liczby urządzeń o równoprawnym dostępie do kanału.

Do tej listy można dodać jeszcze wymaganie, aby rzeczywisty ośrodek transmisji był tani i prosty, lecz to nie musi być zawsze spełnione, jak choćby w wypadku niektórych systemów szerokopasmowych (ang. broadband). Wymienione cechy są całkowicie odmienne od właściwości rozległych sieci komputerowych o wspólnej częstotliwości nośnej. Odpowiedź na pytanie: czy protokół X.25 z przełączeniem pakietów byłby odpowiedni dla sieci lokalnej — jest zdecydowanie negatywna.

Pod jednym względem jednak, lokalne i rozległe sieci komputerowe są podobne. Mianowicie, w obu wypadkach wykorzystuje się architekturę warstwową modelu referencyjnego OSI (ang. open system interconnection). Jest to sprawa niezwykle ważna i wynika nie tylko z potrzeby zapewnienia modularności i kompatybilności wewnątrz określonej sieci, lecz także z konieczności łączenia sieci, tzn. sprzęgania jednej sieci z inną w celu umożliwienia wzajemnej wymiany usług.

Do wyliczonych cech sieci lokalnych należy dodać uzasadnienie, dlaczego te cechy są tak istotne we współczesnych systemach informacyjnych. Po pierwsze: wszystkie te właściwości są pożądane ze względu na potrzebę porozumiewania się ludzi i urządzeń w określonym środowisku. Dobrym przykładem takiej potrzeby są usługi pocztowe. Po drugie: powstało całkowicie nowe wymaganie współdzielenia zasobów, np. dużej maszyny obliczeniowej, bazy danych, centralnego archiwum plików lub wysokiej jakości drukarki, wynikające głównie z przyczyn ekonomicznych. Z podobnych przyczyn wynika trzecie, choć całkowicie odmienne wymaganie, polegające na potrzebie współdzielenia jednoużytkowego oprogramowania w środowisku wieloużytkowym. Gwałtowny wzrost liczby komputerów osobistych i indywidualnych stanowisk roboczych (ang. personal workstations) spowodował znaczne zwiększenie kosztów ponoszonych na oprogramowanie, ponieważ obecnie wymaga się zakupu oddzielnej licencji dla każdej instalowanej kopii tego oprogramowania. Jedno z rozwiązań tego problemu polega na użyciu sieci lokalnej w celu połączenia użytkowników i współdzielenia tego samego oprogramowania, do czego wystarcza zakup pojedynczej licencji lub niewielu więcej. W środowiskach uniwersyteckich jest to jeden z głównych czynników przemawiających za wprowadzeniem lokalnych sieci komputerowych.

WYBÓR SIECI LOKALNEJ

Choć istnieje wiele różnych realizacji lokalnych sieci komputerowych, wybór ich architektury, ośrodka transmisji lub metody dostępu jest dość ograniczony. Jeśli chodzi o topografię sieci lokalnych, to stosuje się struktury punktowe (ang. point-to-point), gwiazdowe, pierścieniowe i magistralowe. Nieograniczone struktury hybrydowe i siatkowe należy wyeliminować, ponieważ wymagają użycia węzłów przełączających w celu dokonania trasowania (ang. routing), co wprowadza zbyt duże ograniczenia szybkościowe i wydajnościowe. Struktura punktowa jest nie tylko ogromnie kosztowna dla dużej liczby urządzeń, lecz wymaga dodania **n** połączeń dla każdego nowego urządzenia dołączanego do sieci złożonej z **n** jednostek. Z tego względu nie nadaje się do stosowania w sieciach lokalnych. Konfiguracja gwiazdowa, choć wymaga tylko jednego połączenia dla każdego nowego urządzenia, narzuca zależność od centralnego sterowania, co wyklucza jej stosowanie w sieciach lokalnych, ze względu na ograniczenia przepustowości i niezawodności. W praktyce, w nowoczesnych sieciach lokalnych przyjęto tylko strukturę magistralową i pierścieniową, choć stosuje się wiele różnych metod dostępu, ośrodków i rodzajów transmisji.

Wybór między magistralą a pierścieniem jest dość trudny, ponieważ oba rodzaje struktury mają swoje zalety. Pierścienie z pewnością umożliwiają — w sprzyjających warunkach — lepsze wykorzystanie dostępnego pasma niż wielodostępna magistrala, taka jak Ethernet, lecz mają inne wady. Pierścienie z przekazywaniem znamienia lub szczelin (ang. token-passing ring, slotted ring) są wrażliwe na przerwanie linii i rozsynchronizowanie (ang. cumulative

timing jitter). Ten drugi czynnik ogranicza w praktyce liczbę węzłów w każdym pierścieniu. Systemy magistralowe są prostsze i narzucają mniej ograniczeń, co w zupełności kompensuje ich wadę polegającą na gorszym wykorzystaniu dostępnego pasma. W chwili obecnej wydaje się, że systemy magistralowe są bardziej przydatnym rozwiązaniem i pozostaną takim jeszcze przez kilka lat. Dzieje się tak z dwóch powodów. Po pierwsze, przy szybkości transmisji 10 Mb/s system magistralowy jest zdolny do obsługi prawie wszystkich żądań, oprócz zupełnie wyjątkowych. Po drugie: są już handlowo dostępne elementy opartej na magistrali sieci lokalnej Ethernet. Z uwagi na dłuższe doświadczenie w eksploatacji sieci Ethernet, systemy pierścieniowe nie są obecnie dla niej konkurencyjne pod względem technicznym, lecz mogą stać się takimi w ciągu roku lub dwóch lat. Aby uzasadnić to stwierdzenie, wystarczy podać, że w chwili obecnej istnieje ponad 10 000 instalacji sieci Ethernet. Choć wpływ firmy IBM jest przemożny i można oczekiwać coraz szerszego rozpowszechnienia jej rozwiązania pierścienia z przekazywaniem znamienia, tzw. IBM Token Ring, wydaje się, że z powodu obecnego ograniczenia szybkości do 4 Mb/s i braku pełnego zakresu elementów na rynku, dominująca pozycja sieci Ethernet jeszcze przez jakiś czas będzie niezagrożona.

Mimo że preferowanym rodzajem sieci jest Ethernet, wciąż otwarta pozostaje kwestia podziału pasma (rodzaju transmisji) i wyboru ośrodka transmisji. Jeśli chodzi o rodzaj transmisji, to w grę wchodzą dwie możliwości: system szerokopasmowy lub transmisja w pasmie podstawowym. Transmisja w pasmie podstawowym (ang. baseband system) polega na bezpośrednim przesyłaniu informacji przez ośrodek (kabel współosiowy lub włókno światłowodowe) w postaci cyfrowej z wykorzystaniem całego dostępnego pasma kanału. W systemach szerokopasmowych (ang. broadband systems) stosuje się podział częstotliwości na pasma i używa modemów cyfrowych. W normie IEEE 802, dla systemów szerokopasmowych zaleca się użycie 75-omowego kabla współosiowego jako ośrodka transmisji oraz podział pasma na kanały według zasad obowiązujących w telewizji kablowej (ang. Community Access TV, CATV), tzn. szerokość kanału 6 MHz i odstęp między kanałami 192 MHz. Sygnał jest modulowany zwykle metodą AM/PSK (ang. amplitude modulation — phase shift keying, modulacja amplitudy z kluczowaniem fazy).

Podstawowa zaleta systemów szerokopasmowych polega na szerokim rozpowszechnieniu techniki CATV i na możliwości mieszania różnych usług w tym samym ośrodku transmisji. Przykładowo — jedna grupa kanałów może służyć do przekazów telewizyjnych, inna do przesyłania sygnałów akustycznych, a jeszcze inna do transmisji cyfrowej w sieci lokalnej. W praktyce jednak zbyt duży jest koszt modemów, a ich maksymalna szybkość transmisji na ogół nie przekracza 2 Mb/s. Niekiedy twierdzi się, że systemy szerokopasmowe umożliwiają większy zasięg niż transmisja w pasmie podstawowym, lecz różnica jest minimalna i wynika z nieco szybszej propagacji sygnałów przez 75-omowy kabel CATV niż przez 50-omowy kabel sieci Ethernet. Z drugiej strony, jeśli stosuje się standardowy mechanizm dostępu do ośrodka według normy IEEE 802.3, to ograniczenie długości sieci jest takie samo dla obu systemów i wynosi ok. 2,8 km dla pojedynczej sieci Ethernet.

Systemy transmisji w pasmie podstawowym są łatwiejsze do zainstalowania, ich ośrodek transmisji jest tani, a koszty sprzęgania są znacznie mniejsze niż dla systemów szerokopasmowych, szczególnie dlatego, że większość elektroniki jest cyfrowa i może być zintegrowana w układy o dużym stopniu scalenia. Obecnie znacznie łatwiej dostępne są elementy sieci Ethernet do transmisji w pasmie podstawowym, dlatego nie należy się dziwić, że właśnie ta technika w tej chwili dominuje w realizacji lokalnych sieci komputerowych.

SYSTEM TRANSMISJI W PASMIE PODSTAWOWYM

W chwili obecnej istnieją trzy wersje specyfikacji sieci Ethernet:

— **wersja 1,** opublikowana przez firmy DEC, Intel i Xerox, w sierpniu 1980 roku [1a],

— **wersja 2,** opublikowana przez te same firmy w listopadzie 1982 roku [1b],

— **norma IEEE 802.3** dotycząca wrażliwego na nośną dostępu wspólnego z wykrywaniem kolizji (ang. carrier-sense multiple access with collision detection CSMA/CD), opublikowana w 1985 roku [2] i stanowiąca przedmiot prac ISO jako DIS 8802/3 (Draft International Standard).

Wersja pierwsza jest oryginalną specyfikacją opracowaną przez trzy wymienione firmy, lecz opartą na wcześniejszych badaniach prowadzonych w Xerox Palo Alto Research Center.

Druga wersja, niewiele różniąca się od pierwszej, tym niemniej niezgodna z nią, powstała aby uprzedzić wyniki prac nad normą IEEE, lecz nie zastępuje tej normy.

Norma IEE 802.3 różni się od obu poprzednich wersji pod względem zalecanych układów kodowania i nadajników-odbiorników (ang. transceivers). Choć zasadnicza idea pozostaje niezmieniona, inna jest realizacja.

W dalszej części artykułu będzie mowa tylko o normie IEEE 802.3.

W sieci Ethernet wszystkie węzły (urządzenia użytkowe, komputery macierzyste — ang. hosts, itd.) są dołączone do wspólnego kanału, którym jest 50-omowy kabel współosiowy. Każde dołączone urządzenie ma możliwość sprawdzenia, czy kanał jest wolny, przez wykrycie obecności lub nieobecności ruchu w kanale. Jest to wrażliwość na nośną (litery CS w skrócie CSMA/CD), polegająca na ciągłym nasłuchu sygnału częstotliwości nośnej przed rozpoczęciem transmisji. Jeżeli kanał jest wolny (brak sygnału nośnej), to dowolne urządzenie żądające transmisji może ją wykonać. Oznacza to wspólny dostęp do kanału (litery MA w skrócie CSMA/CD). Oczywiście, wysyłana informacja zawiera adres docelowy i adres źródłowy.

Ponieważ wszystkie stacje współzawodniczą w dostępie do tego samego ośrodka i więcej niż jedna stacja może rozpocząć transmisję w tym samym odcinku czasowym, musi istnieć mechanizm rozstrzygania konfliktów. Ten mechanizm polega na wykrywaniu obecności więcej niż jednego komunikatu (tzn. kolizji), wycofaniu się w takim wypadku wszystkich nadających stacji i ponownym rozpoczęciu transmisji po upływie losowego opóźnienia.

Aby umożliwić wszystkim węzłom wykrywanie kolizji, ograniczono od dołu długość pakietu danych, wprowadzając tzw. minimalną długość ramki równą 512 bitów. Przy szybkości transmisji 10 Mb/s na przebiegnięcie ramki przez całą sieć Ethernet potrzeba 51 μs. Biorąc pod uwagę czas propagacji sygnałów w ośrodku, otrzymuje się ograniczenie na maksymalną długość sieci Ethernet, wynoszące ok. 2,8 km.

W rzeczywistości ustalenia zawarte w normie IEEE 802.3 uwzględniają wiele więcej szczegółów. Przykładowo, każda stacja wykrywająca kolizję powoduje zagłuszenie kanału przez pewien okres. Dla stacji, która skończyła transmisję, istotna jest informacja, że w innym miejscu ośrodka nastąpiła kolizja. Choć w tym artykule nie ma miejsca na głębsze wnikanie w szczegóły, należy wiedzieć, że informacja jest przesyłana w ramkach (pakietach), które oprócz danych zawierają adres źródłowy i docelowy, tryb adresowania (przesyłanie jednoadresowe, wieloadresowe lub rozgłaszanie), pole typu oraz ciąg kontrolny CRC. Między kolejnymi ramkami musi istnieć minimalny odstęp 9,6 μs. Sygnał cyfrowy jest kodowany bifazowo (ang. Manchester encoding) ze względu na samotaktowanie (ang. self-clocking) i przechodzenie sygnału przez poziom zerowy w każdym bicie (w połowie okresu).

Specyfikacja sieci Ethernet zawiera szereg istotnych ograniczeń, jak np. ograniczenie minimalnej odległości między dwoma sąsiednimi nadajnikami-odbiornikami do 2,5 m. Ponadto każdy segment może łączyć co najwyżej 1024 węzły, choć nie jest to krytyczne ograniczenie, gdyż pojedynczy węzeł może obsłużyć wiele urządzeń dołączonych przez koncentrator. Jeśli chodzi o strukturę sieci, to oprócz ograniczenia długości najistotniejszym ograniczeniem jest zakaz tworzenia zamkniętych pętli. W praktyce sieci lokalne Ethernet mają strukturę drzewiastą, co omówimy dokładniej w jednym z następnych punktów.

ZGODNOŚĆ Z ARCHITEKTURĄ OSI

Każda nowoczesna sieć Ethernet działa według protokołów zgodnych z warstwową strukturą modelu OSI. Dotąd omówiono jedynie najniższy poziom, tj. warstwę fizyczną (ang. physical layer), obejmującą zasady dostępu do ośrodka, zdefiniowane w normie IEEE 802.3. Powyżej tej warstwy, na poziomie łącza, obowiązują zasady sterowania określone w normie IEEE 802.2 (ang. logical link control, LLC). Protokoły wyższych warstw nie są jeszcze znormalizowane w tym sensie, że nie ma odpowiednich uzgodnień międzynarodowych, w normach IEEE lub ISO, lecz część z nich

jest już zdefiniowana, a opublikowania dokumentów normalizacyjnych oczekuje się w ciągu najbliższych lat. W chwili obecnej stosuje się kilka różnych protokołów o znaczeniu wewnętrznym.

W niektórych sieciach, na poziomie transportowym stosuje się protokoły X.25, lecz nie są one dobrze dostosowane do sieci Ethernet. Firma DEC ma własny zbiór protokołów wysokiego poziomu, a firma Xerox używa świetnego protokołu XNS (ang. Xerox Network System). Niestety oba te rodzaje protokołów w rzeczywistości nie pozwalają na tworzenie systemów otwartych, tj. łączenie sieci w heterogeniczne środowiska z mieszanymi komputerami macierzystymi (ang. hosts). W praktyce najszerzej stosuje się protokoły TCP/IP (ang. Transmission Control Procedures — Internetworing Protocols), opracowane przez amerykańską agencję DARPA (ang. Defense Advanced Research Project Agency).

Protokół IP odpowiada warstwie sieciowej w architekturze ISO. Jest wywoływany przez protokoły wyższych warstw komunikacji między komputerami macierzystymi i w praktyce zapewnia tzw. usługi datagramowe (ang. datagram service) między źródłem i miejscem przeznaczenia danych. Protokół IP obejmuje także mechanizm fragmentacji i montowania dłuższych datagramów, lecz nie zapewnia sterowania przepływem ani integralności danych.

Na wyższym poziomie niż obsługiwany przez IP, protokół TCP umożliwia niezawodną komunikację między procesami w różnych komputerach macierzystych i odpowiada w przybliżeniu warstwie transportowej, a częściowo także warstwie sesji modelu ISO. Protokół TCP umożliwia również kontrolę przepływu sterowania, zarządzanie transmisją bloków danych (takich jak rekordy), multipleksowanie kilku procesów w celu współbieżnego korzystania z kanału komunikacyjnego i — utrzymanie informacji o stanie.

Oprogramowanie zgodne z protokołami TCP/IP istnieje już dla szerokiego zakresu systemów operacyjnych, m.in. firm DEC i IBM, stanowi standard komunikacyjny dla Unixa w wersji BSD4.2 i jest rozwijane dla szeregu indywidualnych stanowisk roboczych. Dla wyższych poziomów niż obsługiwane przez TCP/IP istnieją protokoły odpowiadające poszczególnym usługom, jak np. Telnet — zapewniający dostęp odległych terminali (ang. remote terminal access), FTP — umożliwiający przesyłanie plików (ang. file transfer) i protokoły poczty elektronicznej.

Przewiduje się, że w odpowiednim czasie specyfikacje wszystkich protokołów powyżej warstwy sieciowej zostaną opracowane przez ISO. Narazie jednak nie ma w powszechnym użyciu protokołów alternatywnych wobec TCP/IP. Jedyna ewentualność może dotyczyć protokołów SNA (ang. system network architecture), lecz nie ze względu na powszechność ich stosowania w sieciach lokalnych, a z uwagi na silne wpływy firmy IBM, w której je opracowano.

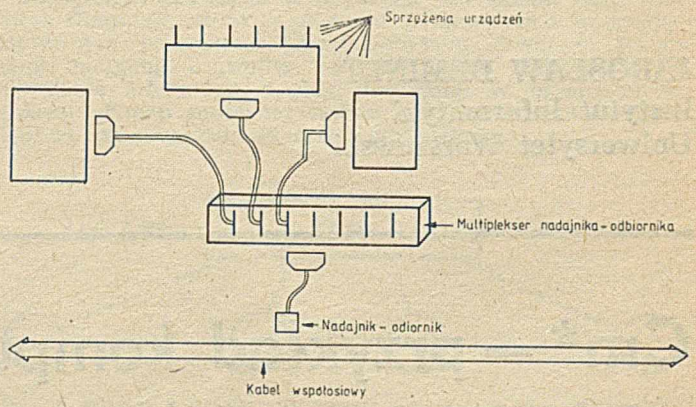
ŁĄCZENIE SIECI ETHERNET

Podstawowym celem budowy sieci lokalnej ETHERNET jest wzajemne sprzęganie urządzeń przy użyciu fizycznego ośrodka transmisji. Łączonymi urządzeniami mogą być terminale, komputery osobiste, stanowiska robocze, współdzielone komputery obliczeniowe, różnego rodzaju stacje obsługi lub bramy do innych sieci. Według normy IEEE 802.3 ośrodkiem transmisji może być tylko 50-omowy kabel współosiowy, choć w praktyce istnieje pewna dowolność wyboru kabla. Norma IEEE zaleca użycie kabli sztywnych o minimalnym promieniu zgięcia 25 cm. Stosuje się jednak również cieńsze kable typu RG58, które są bardziej elastyczne i dziesięć razy tańsze od zalecanych, tworząc tzw. sieci Cheapernet. Kable RG58 mają jednak większą tłumienność, wskutek czego maksymalna długość linii bez repeterów (ang. repeaters) jest ograniczona do 200 m, a nie do 500 m jak w wypadku kabli sztywnych.

Nadajnik-odbiornik

Każde urządzenie łączy się z kablem sieci Ethernet przez odczep (ang. tap) i nadajnik-odbiornik (ang. transceiver). W sieci Cheapernet nadajnik-odbiornik jest zwykle częścią płyty sprzęgającej (ang. interface board), natomiast prawie we wszystkich połączeniach ze sztywnym kablem stanowi oddzielną jednostkę. Jako odczepu używa się zwykle złącza typu N lub innych rodzajów złącz, które przebijają kabel i mogą być usunięte w razie potrzeby. Prawie we wszystkich wypadkach odczep stanowi integralną jednostkę z nadajnikiem-odbiornikiem, choć istnieje jedno bardzo dobre rozwiązanie, w którym oddzielny odczep łączy się z nadajnikiem-odbiornikiem opcjonalnie dobieranym złączem. Z drugiej strony nadajnik-odbiornik ma 15-stykowe złącze typu D do kabla dołączeniowego (ang. drop cable), tzw. AUI (ang. attachment unit interface). Ten kabel dołączeniowy o maksymalnej długości 50 m jest częścią segmentu sieci i musi być uwzględniony w obliczeniach długości całej sieci.

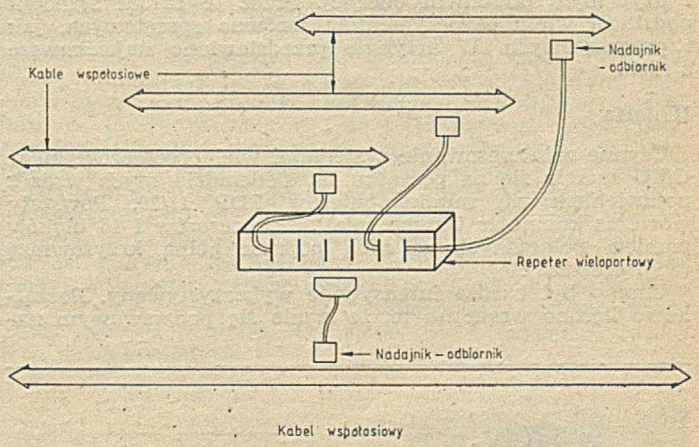
Standardowy nadajnik-odbiornik spełnia funkcje nadawania i odbierania danych, wykrywania kolizji przez badanie sygnału SQE (ang. signal quality error) i zabezpieczenia sieci przed zbyt długimi pakietami danych (ang. jabber function). Ostatnio opracowano rozszerzenie typowego rozwiązania umożliwiające dołączenie (przez kable dołączeniowe) ośmiu jednostek do pojedynczego nadajnika-odbiornika z jednym odczepem (rys. 1). Zaletą tego rozwiązania jest mniejszy koszt jednostkowy dołączania urządzeń i uniknięcie wielokrotnego uszkodzenia głównego kabla. Ponadto unika się w ten sposób 2,5-metrowego ograniczenia na minimalną odległość sąsiednich nadajników-odbiorników.



Rys. 1. Wielokrotny nadajnik-odbiornik sieci Ethernet

Repeter

Maksymalna długość kabla dla segmentu sieci Ethernet wynosi 500 m. Aby umożliwić zwiększenie całej długości sieci, norma IEEE zezwala na połączenie pięciu takich segmentów sprzężonych przez tzw. repetery (ang. repeaters). Repeter przesyła w obu kierunkach zsynchronizowane pakiety danych, regeneruje preambułę ramki i umożliwia automatyczny podział oraz rekonfigurację segmentów sieci.



Rys. 2. Wieloportowy repeter sieci Ethernet

Dzięki użyciu repeterów, sprzęgających kable sieci Ethernet w zwykły sposób za pomocą nadajników-odbiorników, można zwiększyć całkowitą długość sieci do 5×500 m. Wliczając dodatkowo 8 kabli dołączeniowych po 50 m, otrzymuje się całkowitą długość sieci równą 2900 m (w rzeczywistości jest to nieco mniej, ze względu na opóźnienia powstające w repeterach).

Nie wszyscy zdają sobie sprawę, że faktycznie istnieją dwa rodzaje nadajników-odbiorników. Ponieważ repeter jest w istocie przedłużeniem kabla, musi przesyłać wszystkie sygnały, włącznie z występującymi w czasie przerwy

międzyramkowej o długości 9,6 μs. Tak więc każdy repeter wymaga dwóch specjalnych nadajników-odbiorników, które nie badają sygnału SQE.

Podobnie jak w wypadku nadajników-odbiorników niektórzy producenci wytwarzają wieloportowe repetery umożliwiające łączenie kilku segmentów w konfigurację gwiazdową (rys. 2). Jednakże globalne ograniczenie długości sieci pozostaje nadal obowiązujące. W drugiej części artykułu omówimy m.in. repetery odległe, które umożliwiają tworzenie tzw. bezwęzłowych segmentów sieci.

LITERATURA

[1] DEC, Intel, Xerox: The Ethernet — A Local Area Network Data Link Layer and Physical Layer Specifications. Maynard (MA). Santa Clara (CA), Stamford (CT), September 1980 (Version 1), November 1982 (Version 2)

[2] IEEE: Standards for Local Area Networks — Carrier Serse Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. ANSI/IEEE Std 802.3, New York (NY), 1985.

Opracował: **Janusz ZALEWSKI**

POLSKIE TOWARZYSTWO INFORMATYCZNE

Pierwsza wersja artykułu została przedstawiona podczas Szkoły Jesiennej PTI „Współczesne kierunki rozwoju informatyki" — Mrągowo, 4—8 listopada 1985 r.

JAROSŁAW DEMINET
Instytut Informatyki
Uniwersytet Warszawski

Cm* — przykład komputera wieloprocesorowego o strukturze hierarchicznej

Cm* jest wielomikroprocesorowym komputerem, zbudowanym w latach 1975—1980 na Uniwersytecie Carnegie-Mellon (CMU) w Pittsburghu (stan Pensylwania, USA) i używanym do chwili obecnej. Przez te 10 lat stanowił podstawę wielu eksperymentów zarówno sprzętowych, jak i programowych. W artykule przedstawiono najciekawsze z nich.

Historia

Czynne zainteresowanie systemami wieloprocesorowymi w CMU zaczęło się na początku lat siedemdziesiątych wraz z pojawieniem się minikomputerów PDP 11/20. Powstały wówczas komputer C.mmp miał 16 procesorów i 16 bloków pamięci połączonych ze sobą poprzez pełną krzyżownicę (ang. crossbar).

C.mmp był bardzo intensywnie wykorzystywany. Początkowo liczono wręcz na to, że stanie się podstawowym narzędziem pracy. Bardzo dużo trudu włożono w oprogramowanie narzędziowe (co najmniej kilkanaście osób przez przeszło 5 lat); stworzono kompletny system operacyjny Hydra, skrośny kompilator języka Bliss-11, skrośny asembler, specjalny program konsolidujący, program obsługi łącza sieci Arpanet itp. Niestety, jednak w momencie, gdy cały sprzęt i oprogramowanie były już gotowe, stały się przestarzałe. Podstawową wadą komputera była mała przestrzeń adresowa (64 KB). Wprawdzie całkowita pamięć programów mogła być większa, ale wymagało to skomplikowanego programowania. W tym czasie pojawiły się nowe, duże komputery i C.mmp pozostał jedynie narzędziem eksperymentalnym. W 1980 roku został rozebrany.

W wyniku badań nad C.mmp sformułowano kilka wniosków, które wykorzystano przy następnych projektach badawczych. Oto najbardziej interesujące:

- Można zbudować komputer wieloprocesorowy przy wykorzystaniu gotowych układów minikomputerowych, lecz należy liczyć się z koniecznością znacznych modyfikacji (150 dodatkowych układów scalonych do 400-układowego procesora). Takie modyfikacje są tym trudniejsze, im większy jest stopień scalenia użytych układów (jest trudniej dostać się „do środka").

- W praktyce nie da się wykorzystać zachwalanej często cechy systemów wieloprocesorowych, jaką jest rzekomo możliwość dzielenia kodu programu między kilka procesów. W rzeczywistości jeden blok pamięci był zdolny obsłużyć co najwyżej 4 programy równocześnie. Ta nauka jest chyba ważna do dziś. Wprawdzie dzisiejsze pamięci są szybsze, ale i procesory korzystają z pamięci w krótszych odstępach czasu.

- Po odrzuceniu dzielenia kodu okazuje się, że zdecydowana większość odwołań do pamięci dotyczy kodu, zmiennych lokalnych i stosu, a więc obszarów niejako prywat-

Mgr JAROSŁAW DEMINET ukończył w 1978 r. Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Pracuje w Instytucie Informatyki UW, w Zakładzie Systemów i Oprogramowania. W latach 1979—1980 był na 18-miesięcznym stażu w Carnegie-Mellon University w Pittsburghu, gdzie zajmował się oprogramowaniem systemowym i eksperymentalnym dla komputera Cm*.

nych dla zadania. Jedynie 3—10% odwołań dotyczy pamięci dzielonej między kilka programów.

● W systemie wieloprocesorowym niezbędne okazuje się „zaszycie" niektórych operacji w mikroprogramach procesorów. W początkowej wersji C.mmp procesory nie były mikroprogramowane. Gdy tylko pojawiły się mikroprogramowane komputery PDP 11/40, zmodyfikowano je tak, aby umożliwić dopisywanie fragmentów mikroprogramu, i zdefiniowano — na przykład — nowe instrukcje przesłań grupowych oraz instrukcje manipulowania pewnymi specjalnymi, chronionymi obiektami. Przyspieszyło to pracę całego systemu.

Gdy prace nad C.mmp dobiegały końca, zaczęto prowadzić przygotowania do nowego projektu, który szedłby dalej w podobnym kierunku. Założono, że architektura nowego komputera powinna umożliwić połączenie nawet kilku tysięcy procesorów i gigabajtów pamięci operacyjnej, przy czym powinno być możliwe stosunkowo łatwe dodawanie nowych procesorów i pamięci.

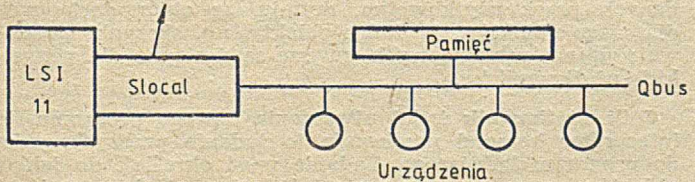
W tym samym czasie (1975 r.) firma DEC opracowała pierwsze mikrokomputery LSI-11 (na których jest wzorowana Mera 60). Są to mikrokomputery 16-bitowe, na pojedynczej płytce drukowanej, z możliwością zaadresowania do 56 KB pamięci, bez żadnych mechanizmów sprzętowej ochrony zasobów. Komunikacja z pamięcią i urządzeniami odbywa się przez asynchroniczną magistralę Q-bus. Na płytce procesora znajdują się 4 układy scalone, tworzące wspólnie kompletny procesor oraz układy o małym i średnim stopniu scalenia odpowiedzialne przede wszystkim za współpracę z magistralą. Sam procesor ma 8-bitową budowę wewnętrzną, co powoduje dość wolną pracę (przy cyklu zegara ok. 400 ns, wykonanie typowej instrukcji trwa od 3 do 10 μs). Pamięć użyta w Cm* była zbudowana z układów dynamicznych o pojemności 4 Kb, później wymienionych na układy 16 Kb.

Przy opracowywaniu szczegółów architektury Cm*, kierowano się łatwością zaimplementowania systemu operacyjnego, zbliżonego w swej koncepcji do Hydry. Nad takim systemem, o nazwie StarOS, pracowano pod kierunkiem Anity Jones. Po pewnym czasie grupa podzieliła się: pięć osób pozostało w oryginalnym zespole, trzy — przystąpiły do prac nad mniejszym i prostszym systemem, nazwanym Medusa. Założeniem twórców Medusy było maksymalne wykorzystanie możliwości architektury Cm*. Odmienność obu systemów rzuca światło na problematykę specyfikowania produktów sprzętowych i programowych.

Struktura komputera

Przy opracowywaniu architektury nowego komputera, trzeba było przede wszystkim rozwiązać problem struktury sieci połączeń między procesorami a pamięcią. Pełna krzyżownica zupełnie się nie nadawała do tego celu. Liczba połączeń jest w niej bowiem rzędu n^2, gdzie n jest liczbą składników. Już w C.mmp dla 16 procesorów złożoność krzyżownicy była taka, jak całej reszty konfiguracji (5 tys. układów scalonych).

Każda wersja struktury stanowi pewien kompromis ze względu na różne kryteria: liczbę węzłów, czas połączenia, stopień zrównoleglenia, prostotę ustalenia trasy połączenia itp. W wyniku dokładnej analizy zdecydowano się wybrać strukturę hierarchiczną. W ten sposób powstała koncepcja Cm*. Cm jest skrótem od ang. computer module, natomiast gwiazdka oznacza iterację (teoretycznie — nieograniczoną).



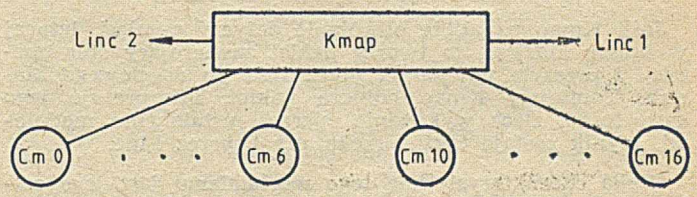
Rys. 1. Element struktury komputera Cm*

Podstawowym elementem struktury jest moduł (Cm), składający się z procesora, pamięci i ewentualnych urządzeń dołączonych do magistrali, tak jak w zwykłym mikrokomputerze (rys. 1). Między procesor a magistralę włączono dodatkowy element — Slocal (ang. local switch). Slocal pełni dwie różne funkcje. Po pierwsze, wspomaga procesor, nadrabiając niektóre jego braki (np. pozwala na rozróżnienie trybu pracy programów systemowych i użytkowych, przy czym te drugie nie mają prawa wykonywać niektórych potencjalnie niebezpiecznych instrukcji). Należy podkreślić, że przy dużym stopniu scalenia modyfikacja procesora była bardzo trudna (potrzebne było do tego 80 układów scalonych).

Slocal rozszerza także możliwości adresowe LSI-11. Przestrzeń adresowa jest podzielona na 16 stron wirtualnych po 4 KB każda. Każda strona może być odwzorowana na dowolny blok pamięci o takim samym rozmiarze, leżący w tym samym module. Slocal zawiera rejestry sterujące tym odwzorowaniem. Podobnie jak w innych komputerach, istnieją osobne zestawy rejestrów dla programów, pracujących w trybie systemowym i w trybie użytkowym.

Po drugie, Slocal umożliwia komunikację modułu z resztą komputera. W szczególności odwołania do niektórych stron (niektórych adresów) są kierowane na wyższe piętro hierarchii.

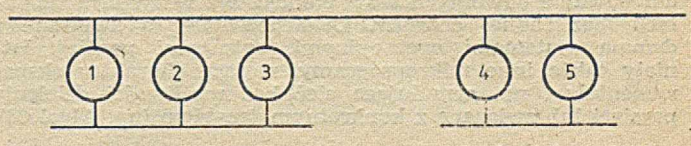
Następnym poziomem w hierarchii jest klaster (ang. cluster). Obejmuje on do 14 modułów połączonych z wyspecjalizowanym procesorem komunikacyjnym, zwanym Kmap (ang. mapping controller, rys. 2). Kmap pośredniczy w wymianie informacji między modułami. W ograniczonym zakresie Kmap może też nadzorować pracę procesorów LSI-11, na przykład, zgłaszając przerwania.



Rys. 2. Klaster z wyspecjalizowanym procesorem komunikacyjnym

Czas cyklu procesora Kmap wynosi 160 ns. Kmap jest mikroprogramowany poziomo, a mikrorozkaz ma 80 bitów szerokości i jest zapisany w pamięci RAM, dostępnej z zewnątrz.

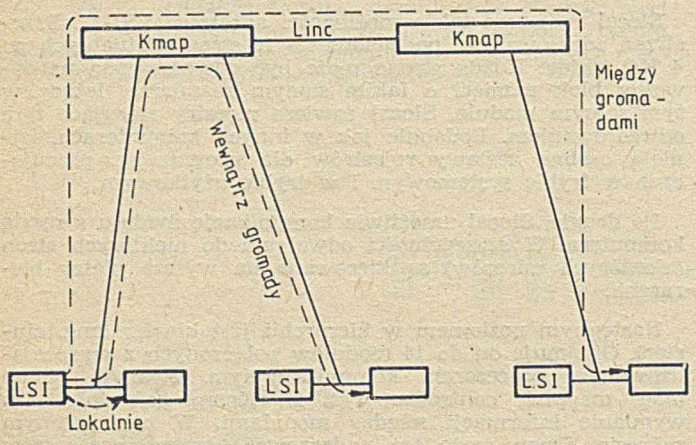
Kmap ma dwa porty, do których można dołączyć szybkie magistrale prowadzące do innych klastrów (zwane Linc). Magistralami tymi są przesyłane kilkusłowowe pakiety danych, z szybkością ok. 2 Mb/s. Koncepcja Cm* nie nakłada żadnych ograniczeń na liczbę różnych magistrali Linc ani na ich konfigurację. Ogranicza się natomiast do 64 liczbę klastrów dołączonych do jednej magistrali. Przykładowym rozwiązaniem jest macierz 32×32 klastry, połączone 32 magistralami poziomymi i 32 pionowymi. W takiej konfiguracji każda transmisja między klastrami wymaga co najwyżej jednego „pośrednika". Konfiguracja miałaby ok. 13 tys. procesorów i do 4 GB pamięci operacyjnej. Rzeczywistość jest bardziej prozaiczna. Uruchomiona konfiguracja Cm* składa się z 5 klastrów, 50 procesorów i 3 MB pamięci (rys. 3). Na rysunku 4 przedstawiono sposób odwoływania się procesora do pamięci. Adres jest porównywany przez Slocal z rejestrami sterującymi. Jeśli ich zawartość wskazuje, że nastąpiło odwołanie do strony umieszczonej w lokalnej pamięci modułu, to Slocal wyznacza adres fizyczny i przekazuje go bezpośrednio na magistralę. Dalej operacja przebiega tak samo, jak w wypadku zwykłego mikrokomputera.



Rys. 3. Uruchomiona konfiguracja Cm*

Jeśli nastąpiło odwołanie do strony oznakowanej uprzednio jako nielokalna, to Slocal przekazuje wszystkie informacje (adres i ewentualne dane) do procesora Kmap. Ten z kolei decyduje o dalszym losie operacji, na podstawie swojego mikroprogramu i posiadanych danych. Na ogół Kmap określa, w którym module leży adresowana pamięć i prze-

kazuje do niego odpowiednie żądanie. Slocal docelowego modułu przekazuje adres i ewentualne dane na magistralę. Wynik operacji (potwierdzenie zapisu lub przeczytane dane) wraca tą samą drogą, przez Kmap, do procesora-zleceniodawcy.



Rys. 4. Sposób odwoływania się do pamięci

Jeśli Kmap uznał, że odwołanie dotyczy pamięci, leżącej w innym klastrze, to tworzy pakiet zawierający opis operacji i przesyła ten pakiet przez Linc do docelowego klastra. Tamtejszy Kmap odbiera pakiet, interpretuje jego zawartość i zleca wykonanie operacji ustalonemu modułowi. Wynik zostaje także przekazany poprzez Linc.

Cechą charakterystyczną tego mechanizmu jest jego całkowita przezroczystość dla programu, z którego pochodzi odwołanie do pamięci. Nie ma żadnej różnicy jakościowej między odwołaniem do pamięci lokalnej, wewnątrz klastru i między klastrami. Różni się natomiast czas, w którym żądana operacja zostanie zakończona. Odwołanie do pamięci lokalnej trwa ok. 3 μs; do pamięci wewnątrz klastra — minimum 9 μs; do pamięci w innym klastrze — powyżej 27 μs. Dwa ostatnie parametry dotyczą warunków optymalnych; gdy liczba nielokalnych odwołań rośnie, wtedy następuje „zatkanie" procesora Kmap i operacje mogą trwać nawet 200 μs. W praktyce z architektury Cm* wynika, że możliwie duża część odwołań powinna odnosić się do pamięci lokalnej. W szczególności, tam powinien znajdować się kod programu i jego stos zawierający dane lokalne.

Kmap z założenia miał być bardzo elastyczny i uniwersalny; możliwość mikroprogramowania zapewniła spełnienie tego założenia. Zmiana mikroprogramu pozwala zrealizować różne strategie adresowania. W szczególności Kmap może rozpoznawać odwołania do niektórych adresów, jako żądania wykonania specjalnych funkcji. W najprostszych wypadkach mogą to być żądania zmiany zawartości wewnętrznych rejestrów opisujących przekształcanie adresów. W bardziej skomplikowanych — żądania przesłania bloku kilkuset słów pod wskazany adres, być może w odległym module. Wreszcie, Kmap może sam obsługiwać struktury danych, np. kolejki i stosy: zapisanie słowa pod wskazany adres powoduje dodanie go do stosu, a przeczytanie spod tego adresu — pobranie z wierzchołka stosu.

Istnieją co najmniej trzy w pełni sprawdzone mikroprogramy. Jeden, bardzo prosty, stosowany do testowania sprzętu i do prostych eksperymentów programowych, umożliwia adresowanie dowolnych słów pamięci we wszystkich modułach, bez żadnej ochrony. Dwa pozostałe są związane z dwoma systemami operacyjnymi, opisanymi poniżej. Istniały także inne mikroprogramy, mające charakter eksperymentalny, np. symulujące sieć bez możliwości dzielenia pamięci lub związane z konkretnym językiem (np. Algol 68).

Cm* został obudowany aparaturą wspomagającą uruchamianie sprzętu i oprogramowania. Przede wszystkim do procesorów Kmap są dostawiane dodatkowe mikrokomputery LSI-11, które umożliwiają zapisanie ich mikroprogramu, oglądanie stanu rejestrów wewnętrznych, zapisywanie do nich nowych wartości, ustawianie punktów przerwań i pracę krokową.

Użytkownik komunikuje się z modułami przez 10 linii szeregowych (po 2 na klaster). Wszystkie te linie są sterowane przez specjalizowany komputer PDP 11/20, zwany Cm* Host. Host pozwala dodatkowo zatrzymać i uruchomić procesory w poszczególnych modułach, a także załadować do nich programy.

Oprogramowanie i mikroprogramy dla Cm* przygotowywano skrośnie, na komputerze DEC-10, a następnie transmitowano — początkowo przez linie szeregowe, a potem przez dwa szybkie łącza pamięciowe. W miarę upływu czasu Cm* był rozbudowywany o nowe urządzenia, mające często charakter eksperymentalny (np. inteligentny, mikroprogramowany sterownik dyskowy, bardzo szybki monitor ekranowy, łącze sieci lokalnej Ethernet).

System StarOS

StarOS jest systemem operacyjnym, opartym na dojściach (ang. capability). Dojście można interpretować jako wskaźnik obiektu, określający dodatkowo typ obiektu i zbiór operacji, którymi dysponuje posiadacz dojścia. W szczególności posiadacz może (ale nie musi) mieć prawo powielania dojścia oraz usuwania go. Dojścia są przechowywane w pamięci, ale dostęp do nich jest możliwy tylko za pośrednictwem operacji zaimplementowanych w procesorze komunikacyjnym Kmap. Każdy obiekt może składać się z części danych dostępnych bezpośrednio i z części dojść.

Struktura zbioru obiektów systemu StarOS odzwierciedla strukturę komputera. Każdy klaster jest traktowany jako osobna jednostka organizacyjna, mająca własny program, tworzący obiekty i zarządzający posiadaną pamięcią. Każdy tworzony obiekt jest opisywany przez pozycję w słowniku. Położenie obiektu w słowniku jest stałe przez cały czas istnienia obiektu, mimo że obiekt może zostać przesunięty, choćby do innego modułu (pozostając jednak wewnątrz klastra). Dojścia wskazują obiekt, podając numer klastra oraz numer pozycji w słowniku.

Podstawowymi obiektami są strony pamięci. Tylko do nich program może mieć szybki dostęp za pośrednictwem zwykłych instrukcji procesora. Jeśli strona pamięci znajduje się w module wykonującym program, to Kmap przy pierwszym odwołaniu zapisuje odpowiednią wartość do rejestru wewnątrz przełącznika Slocal, tak aby następne odwołania były wykonywane całkowicie lokalnie. W odwołaniach do stron położonych w innych modułach oraz do wszelkich innych obiektów, niezależnie od ich położenia, musi pośredniczyć Kmap. Jednak i tu stosuje się różne formy optymalizacji, na przykład Kmap zapisuje w swojej szybkiej pamięci roboczej pomocniczą informację o fizycznym położeniu obiektu.

Komunikacja między programami odbywa się za pośrednictwem tzw. skrzynek pocztowych (ang. mailbox). Poszczególne programy mogą mieć dojścia do różnych skrzynek, z prawem wysyłania lub odbierania komunikatów. Każdy komunikat może zawierać zarówno zwykłe zmienne, dostępne bezpośrednio dla procesora, jak i dojścia.

Początkowo w systemie istnieje kilka programów usługowych, np. program tworzący obiekty, program ładujący i system plików. Każdy program sam jest obiektem; część pamięciowa zawiera parametry, natomiast część dojściowa — dojścia do stron pamięci, zawierających kod i dane programu oraz dojścia do skrzynek pocztowych.

Aby użytkownik mógł skorzystać z systemu, StarOS tworzy dla niego program — interpreter komend. Po zidentyfikowaniu użytkownika, na podstawie uprzednio zapisanych informacji, program ten tworzy inicjalny zbiór dojść posiadanych przez użytkownika, definiujący zbiór dostępnych dla niego obiektów. Niektóre obiekty, np. skrzynki pocztowe programów usługowych, zawsze są dostępne dla wszystkich użytkowników.

Jeśli użytkownik żądał utworzenia nowego programu, to interpreter komend przekazuje to żądanie przez skrzynkę do programu ładującego. Żądanie musi określać, do jakich obiektów nowy program ma mieć dostęp, a także musi zawierać dojście do skrzynki, przez którą program ładujący ma potwierdzić wykonanie zlecenia.

Po przyjęciu żądania program ładujący musi zająć odpowiedni obszar pamięci na sam program oraz na jego skrzynki i strony pamięci. W tym celu wysyła żądanie do programu obsługi obiektów i powrotną pocztą otrzymuje dojścia do nowych obiektów. Po utworzeniu i wypełnieniu inicjalną zawartością stron nowego programu, program ła-

dujący tworzy obiekt — program. Odbywa się to przez tzw. wzmocnienie dojścia. System zostaje poinformowany o tym, że obiekt, który dotychczas był zwykłą stroną, od tej pory ma typ „program". Posiadanie dojścia do takiego obiektu nie uprawnia do swobodnego modyfikowania go — można tylko polecać dokonanie na nim operacji (np. zmiany parametrów) programowi, który utworzył obiekt (w wypadku programów — programowi ładującemu). StarOS umożliwia dynamiczne definiowanie nowych typów i programów ich obsługi.

Po wpisaniu parametrów i dojść do dostępnych obiektów do obiektu-programu, dojście do niego zostaje odesłane użytkownikowi powrotną pocztą. Może on polecić wykonanie programu, przesyłając dojście programowi szeregującemu, który z kolei — według ustalonej strategii — przydziela poszczególnym programom czas poszczególnych procesorów i wpisuje odpowiednie parametry w obszarze dzielonym z jądrem systemu. Jądro dokonuje faktycznego uruchamiania i wywłaszczania programów.

Mechanizm dojść szczególnie dobrze pasuje do architektury wieloprocesorowej. Bardzo łatwo można zorganizować duży zbiór programów, dzielących w dowolny sposób swe zasoby położone w różnych modułach, a nawet klastrach. Dzięki temu, że program odwołuje się do systemu za pośrednictwem skrzynek, istnieje możliwość dzielenia pracy programów systemowych (może istnieć na przykład kilka programów obsługi plików, każdy dostępny dla innych programów użytkowych).

Od początku zakładano, że użytkownik będzie tworzył zespoły programów (ang. task force), współpracujących przy wykonaniu określonych zadań. StarOS sam powinien umieć rozmieścić różne obiekty (programy, skrzynki, strony pamięci), aby zoptymalizować wydajność systemu. W praktyce jednak nie udało się tego osiągnąć.

Jednym z problemów związanych z architekturą systemu hierarchicznego jest problem odśmiecania, tzn. usuwania zbędnych obiektów. Obiekt, do którego nie istnieją żadne dojścia, może zostać usunięty. Znajdowanie martwych obiektów jest zadaniem odśmiecacza. Trzeba się jednak liczyć z tym, że równolegle będą pracowały inne programy, które mogą kopiować posiadane dojścia lub tworzyć nowe obiekty. Aby uniknąć zmylenia odśmiecacza, przyjęto zasadę, że w czasie jego pracy Kmap specjalnie oznacza nowe obiekty oraz nowe dojścia — mówi się, że maluje je na żółto. Wskazywane obiekty nie zostaną na pewno usunięte w tym cyklu odśmiecania. To jednak nie wystarcza — kopiowaniem dojść zajmuje się Kmap, a jego jurysdykcja rozciąga się tylko na jeden klaster. Są więc kłopoty z kolorowaniem obiektów położonych w innych klastrach. Przyjęto zatem dodatkowo malowanie na czerwono tych obiektów, do których istnieją dojścia spoza klastra. W czasie normalnego odśmiecania, czerwone obiekty są ignorowane. Dopiero gdy ich liczba będzie zbyt duża, włączy się superodśmiecacz, działający równocześnie we wszystkich klastrach.

System Medusa

Medusa jest systemem prostszym i mniej elastycznym niż StarOS. Przede wszystkim, z góry jest określony zbiór możliwych typów obiektów i każdy z nich jest obsługiwany przez program stanowiący część systemu operacyjnego. Każdy obiekt jest opisywany przez deskryptor, podobny w swej koncepcji do dojścia. Deskryptory mogą jednak być zapisane tylko w specjalnych obiektach, zwanych listami deskryptorów.

Podstawowym obiektem dynamicznym jest zespół obejmujący od kilku do kilkudziesięciu procesów. Struktura zespołu jest sztywna — proces nie może przenieść się do innego zespołu, nie może też istnieć poza zespołem. Z każdym zespołem jest związana dzielona lista deskryptorów, opisująca obiekty dostępne dla wszystkich procesów zespołu. Każdy proces ma swoją własną listę deskryptorów dla obiektów, do których tylko on ma dostęp.

Podstawowym obiektem statycznym jest strona pamięci. Może dla niej istnieć tylko jeden deskryptor, a więc może ona być dzielona przez cały zespół, albo przydzielona dla jednego procesu. Dla innych obiektów liczba istniejących deskryptorów może być większa, ale nigdy nie może przekroczyć liczby ustalonej podczas tworzenia obiektu. Medusa notuje przy każdym obiekcie adresy wszystkich jego deskryptorów. Z jednej strony ułatwia to odśmiecanie (można łatwo określić, kiedy zostaje usunięty ostatni deskryptor), z drugiej — znacznie komplikuje koordynację pracy procesorów Kmap w systemie obejmującym kilka klastrów.

Medusa traktuje całą konfigurację jednolicie. Każdy program systemowy obsługuje wszystkie procesy, być może położone w kilku klastrach. Do komunikacji procesów z systemem oraz procesów między sobą służą potoki, podobne do potoków w Unixie. Każdy proces systemowy ma kilka potoków wejściowych, odpowiadających różnym funkcjom. Deskryptory tych potoków są zapisane w systemowych listach przydzielonych każdemu modułowi. Polecenie wykonania funkcji jest interpretowane przez Kmap, jako wysłanie komunikatu poprzez odpowiedni potok.

Autorzy zwrócili szczególną uwagę na podział systemu na zespoły i na zdefiniowanie funkcji wewnątrz zespołów. Chodziło o to, aby pogodzić koncepcję podziału na warstwy (funkcje należące do warstwy wyższej korzystają z funkcji należących do warstwy niższej) z podziałem tematycznym (np. funkcje związane z obsługą plików powinny móc korzystać ze wspólnych danych). Należało także zapewnić, aby system nie zablokował się niezależnie od obciążenia. Ostatecznie każdy zespół systemowy podzielono na warstwy, z których każda jest związana z jednym potokiem danych. Cały zbiór warstw wszystkich zespołów tworzy acykliczny graf wywołań. Poza uniknięciem zakleszczenia, ułatwia to także obsługę sytuacji awaryjnych.

Formalne usankcjonowanie istnienia zespołów spowodowało konieczność innego spojrzenia na szeregowanie procesów. Zazwyczaj struktura zespołu jest taka, że wstrzymanie jednego procesu może spowodować spowolnienie lub wręcz zablokowanie pozostałych procesów. Zagadnienie przydziału procesorów procesom zespołu staje się podobne do zagadnienia przydziału i wymiany stron w pamięci wirtualnej, z groźbą migotania włącznie. Aby tego uniknąć, wprowadzono dwie zasady. Po pierwsze — system stara się szeregować całe zespoły, równocześnie przydzielając procesory wszystkim procesom zespołu (można to porównać do koncepcji pola roboczego). Po drugie — nawet gdy proces zawiesza się w oczekiwaniu na jakieś zdarzenie zewnętrzne, system przez pewien czas nie przydziela procesora innemu procesowi, aby umożliwić szybkie wznowienie procesu, np. po otrzymaniu komunikatu od innego członka zespołu. Dostrajanie parametrów takiego systemu wydaje się jednym z najbardziej fascynujących zagadnień związanych z systemami wieloprocesorowymi.

Twórcy Medusy poświęcili wiele uwagi problemowi reagowania na błędy oprogramowania i awarie sprzętu. Wprowadzono m.in. pojęcie procesu siostrzanego (ang. buddy). Każdy proces może zażądać, aby w razie powstania sytuacji wyjątkowej zawiadamiać dowolny inny proces wewnątrz zespołu. Zakłada się bowiem, że proces awaryjny może nie być w stanie sam poprawić swego stanu (np. awaria mogła spowodować uszkodzenie kodu). Proces siostrzany w czasie pomagania procesowi awaryjnemu ma pełny dostęp do wszystkich obiektów tego ostatniego.

Medusa była tworzona bardzo efektywnie, z zastosowaniem wszelkich reguł sztuki dobrego programowania. Właściwy system operacyjny (nie licząc mikroprogramu i oprogramowania skrośnego) ma ok. 20 tys. linii w Blissie. Wszystkie te programy zostały najpierw w całości zaprojektowane i napisane, a dopiero potem rozpoczęto ich uruchamianie. Z jednej strony spowodowało to konieczność sporych modyfikacji dalszych programów w miarę uruchamiania poprzednich, ale z drugiej — pozwoliło skrócić okres uruchamiania do ok. 2 miesięcy. Uruchamianie rozpoczęto od programów testujących i śledzących. Przez cały czas bezwzględnie przestrzegano zasady zgodności programów z dokumentacją i niewprowadzania doraźnych ulepszeń i uzupełnień zmieniających specyfikację programów. Przestrzeganie tych zasad pozwoliło, na przykład, autorowi artykułu w ciągu dwóch tygodni przejąć odpowiedzialność za kontynuowanie uruchamiania, modyfikowania i pielęgnowania systemu.

Problematyka językowa i eksperymenty programowe

Zdecydowana większość oprogramowania dla Cm* (w tym — oba systemy operacyjne) została napisana w języku Bliss-11. Nie zapewniał on wprawdzie żadnych specjalnych mechanizmów programowania równoległości, ale dzięki temu był bardzo elastyczny, a poza tym istniał jego dosko-

nały kompilator skrośny. Dla użytkowników Medusy przeniesiono język C.

W początkowej fazie prac na Cm* został przeniesiony z C.mmp kompilator Algolu 68. Kompilator ten wykrywał możliwe zrównoleglenia w programie (np. niezależne obliczanie podwyrażeń) i generował odpowiedni kod. Okazało się jednak, że w ten sposób można zająć najwyżej kilka procesorów; wewnętrzne zrównoleglenie w programie sekwencyjnym jest niewielkie.

Najbardziej udanym językiem stworzonym specjalnie dla Cm* był AMPL, łączący w sobie synchronizację przepływu danych ze strukturalnością Moduli i pewnymi koncepcjami CSP. Moduły mogą się ze sobą komunikować tylko przez komunikaty, wysyłane do różnych skrzynek odbiorczych. Komunikaty mogą oznaczać żądanie wykonania pewnych operacji i udostępnienia wyniku. Eksperymenty dowiodły, że dla niektórych problemów, programy w AMPL, korzystające z systemu Medusa, działają niewiele wolniej niż standardowe zespoły, korzystające z dzielonej pamięci.

Na Cm* przeprowadzono sporo eksperymentów programowych. Na próbnej 10-modułowej wersji prowadzono doświadczenia z algorytmami, badanymi uprzednio na C.mmp: sortowaniem, iteracyjnym rozwiązywaniem układów równań różniczkowych, programowaniem całkowitoliczbowym i rozpoznawaniem mowy. Powstały pierwsze, kolejkowe modele zachowania się skomplikowanej struktury Cm*. Badano czas obsługi typowego żądania przy znanej częstotliwości odwołań do dzielonej pamięci. Później, w miarę stabilizowania ostatecznych wersji systemów operacyjnych, liczba użytkowników zwiększała się. Niestety, nie udało się jednak zwabić na Cm* żadnych użytkowników zainteresowanych dużymi projektami, dla których byłby on narzędziem do uruchomienia programów, na przykład z dziedziny sztucznej inteligencji, albo grafiki. Wszyscy użytkownicy Cm* byli zainteresowani wieloprocesorowością samą w sobie i z natury rzeczy starali się uruchamiać proste, przykładowe programy.

Zachowanie się algorytmów uruchamianych na Cm* było określone przez dwie główne cechy architektury: asynchroniczność pracy poszczególnych procesorów i niejednorodność pamięci. Jest to najlepiej widoczne na przykładzie algorytmu rozwiązywania układu równań różniczkowych, polegających na iteracyjnym przetwarzaniu macierzy kwadratowej. Macierz ta była zapisana w pamięci, a każdy proces odpowiadał za iterację jednego prostokątnego fragmentu. Zdecydowanie najlepiej działał algorytm całkowicie asynchroniczny, w którym iteracje przebiegały niezależnie jedna od drugiej. Dla wielu procesów pracujących w różnych klastrach, liczby iteracji dla poszczególnych fragmentów mogły się różnić nawet o rząd wielkości (w czasie, gdy proces umieszczony w tym samym module co macierz wykonał 20 tys. iteracji, proces w innym klastrze wykonał ich tylko ok. 2 tys.). Okazało się, że wprawdzie algorytm gwarantuje zawsze zbieżność obliczeń, ale szybkość zbieżności zależy istotnie od tego, które fragmenty macierzy są obsługiwane przez najbliższe (a więc i najszybsze) procesy. Bardzo istotne okazało się także rozdzielenie macierzy między kilka modułów, co pozwoliło uniknąć wąskiego gardła powstającego przy dostępie kilkudziesięciu procesów do jednej pamięci. Po starannym dostrojeniu parametrów udało się doprowadzić do prawie liniowego przyspieszenia. Później opracowano jeszcze lepsze wersje algorytmu, korzystające z podziału macierzy na sześciokąty i minimalizujące obszar dzielonej pamięci.

Algorytm szybkiego sortowania okazał się zupełnie nieodpowiedni do rozwiązywania problemów związanych z dostępem do dzielonej pamięci. Już przy kilkunastu procesach rozdzielonych między dwa klastry, wydajność systemu przestawała się zwiększać. W pierwszej chwili może zaskakiwać fakt, że punkt równowagi między sortowaniem szybkim i bąbelkowym (tzn. punkt, poniżej którego należy przejść na to drugie) okazał się być w tym samym miejscu, co dla sortowania sekwencyjnego ($n = 10$).

Podjęto próbę skonstruowania równoległego oprogramowania do symulowania zdarzeń, wzorowanego na Simuli 67. Okazało się to trudne i niezbyt eleganckie, ale możliwe. Przy tej okazji przeprowadzono eksperymenty dotyczące optymalnego rozmieszczania procesów w modułach, przy założeniu, że kod procesu musi znajdować się w tym samym module, co wykonujący go procesor. Ręczna optymalizacja pozwoliła zmniejszyć czas pracy o ok. 30%, co wskazuje na wagę problemu.

Poważniejszą pracą była symulacja systemu energetycznego, składającego się z węzłów o znanej charakterystyce (napęcie, obciążalność) i linii przesyłowych. Symulacja polega na iteracyjnym rozwiązywaniu układu równań liniowych z uwzględnieniem zachowania się węzłów. Okazało się, że standardowo używany algorytm zawiera w sobie sporo równoległości i łatwo dał się przenieść na Cm*. W rezultacie przyspieszenie algorytmu było prawie liniowe, a szybkość wersji 6-procesorowej bliska szybkości programu pracującego na komputerze DEC-20.

Uruchomiono także kilka algorytmów związanych z modelowaniem procesów chemicznych, m.in. z analizą zachowania się cząsteczek wody, przy zastosowaniu metody Monte-Carlo. Dla 45 procesorów osiągnięto 30-krotne przyspieszenie. Należy jednak zaznaczyć, że LSI-11 ma bardzo powolną arytmetykę zmiennopozycyjną, zajmującą większość czasu procesora, a więc odwołania do danych następują w programach numerycznych z niewielką częstotliwością.

Podobne ograniczenie odnosi się do interpretacji eksperymentu z szybką transformacją Fouriera (FFT). Podzielono w nim 1024-elementowy wektor na 32 równe odcinki, a każdy proces dwukrotnie wykonywał obliczenia dla swojego odcinka, wymieniając jego zawartość z innymi. Udało się osiągnąć ponad 25-krotne przyspieszenie w stosunku do wersji jednoprocesorowej, lecz użycie szybszego procesora (np. 11/23 z procesorem zmiennopozycyjnym) zapewne pogorszyłoby ten wynik.

* * *

Realizacja Cm* nie dała pełnej odpowiedzi na pytanie — czy opłaca się korzystać z gotowego sprzętu przy projektowaniu systemów wielomikroprocesorowych. W trakcie przygotowywania oprogramowania systemowego programiści zmagali się z ograniczeniami, wynikającymi z niedostępności wnętrza LSI-11 dla procesorów Kmap, które ze względu na swą wydajność i położenie pełniły rolę nadzorczą.

Cm* jest z pewnością systemem niezrównoważonym. W przeciwieństwie do procesorów Kmap, procesory LSI-11 wydają się zdecydowanie zbyt słabe, aby z nich budować tak złożony komputer. Dostępne niewiele później procesory LSI 11/23 (a zwłaszcza jednokostkowe 11/73) na pewno uprościłyby konstrukcję modułu, zapewniając mechanizmy ochrony zasobów, zarządzanie 22-bitową przestrzenią adresową itp.

Kmap był zaprojektowany wyłącznie z układów o małym i średnim stopniu scalenia. W dwa lub trzy lata później projekanci użyliby zapewne procesorów segmentowych i matryc programowalnych, co pozwoliłoby zmniejszyć rozmiary, pobór mocy i koszt (oryginalny Kmap kosztował 7 tys. dolarów; koszt jednego modułu wynosił 6 tys. dolarów — 3 tys. za LSI-11 i Slocal, 3 tys. za 128 KB pamięci).

Brak dużego zainteresowania społeczności uniwersyteckiej korzystaniem z Cm* był spowodowany przede wszystkim małą przestrzenią adresową i powolną arytmetyką. Właściwie każdy poważny problem wymagałby wyjścia poza 64 KB i programowego manipulowania przestrzenią adresową, a to stanowiłoby poważne utrudnienie, czego dowiodły liczne doświadczenia z C.mmp, m.in. z programem rozpoznawania mowy. Gdy więc w pobliżu stał VAX 11/780, choć jednoprocesorowy, lecz z 32-bitowym słowem adresowym i z prawdziwą arytmetyką zmiennopozycyjną, to użytkownicy wybierali właśnie ten komputer.

Eksperymenty programowe wykazały, że problemy związane z niejednorodnością struktury komputera mogą być fascynujące. Dzięki modyfikowalności adresowania można dodatkowo eksperymentować, na przykład z różnymi zasymulowanymi strukturami sieciowymi.

Wydaje się, że byłby możliwy powrót do koncepcji zbliżonej do Cm*, z wykorzystaniem nowych procesorów 32-bitowych (68020, 32032), pamięci 256 Kb, macierzy programowalnych i ewentualnie specjalnie wypiekanych układów VLSI. Można by się wówczas przekonać, jak pracuje system złożony, na przykład, z tysiąca modułów, z których każdy ma w przeciwieństwie do LSI-11 — pokaźną moc obliczeniową.

Konkurencyjnym rozwiązaniem jest konstrukcja sieci komputerowej (opartej np. na Ethernet), wyposażonej w jeden system operacyjny, pozwalający na automatyczny podział zasobów, włącznie z podziałem czasu poszczególnych procesorów. W tym kierunku idą obecne prace w CMU.

dokończenie na str. 30

ANDRZEJ GECOW
Instytut Problemów Jądrowych
Świerk

Litery polskie dla IBM PC

Przejście[1] od użycia maszyny do pisania, ręcznej czy nawet elektrycznej, **do przygotowywania tekstów przy użyciu komputera i drukarki** o tak wielkich możliwościach jak współczesna drukarka mozaikowa, **porównałbym do przejścia od pisania ręcznego do stosowania maszyn** do pisania. **Chodzi tu** nie tyle o możliwość powielania tekstu, co **o wygodę doprowadzenia tekstu do postaci niemal idealnej, z niedostępnymi praktycznie dotąd środkami wyrazu**, jak: różne kroje liter, pogrubianie, podkreślanie, indeksowanie, odsyłacze itp. Przygotowanie tekstu na ekranie przy użyciu programów redakcyjnych (tzw. edytorów) jest tak wielkim skokiem jakościowym, że **nie można już zaniechać stosowania tej techniki**, mimo wysokiej ceny komputera.

Jednakże, aby stosować tę technikę **do polskich tekstów**, z efektem nie gorszym od dotychczasowych, **niezbędne**[2] **jest użycie polskich liter. Niestety**, ani **polska drukarka** D-100, ani tym bardziej zagraniczne, **nie mają w swoim repertuarze gotowych znaków polskich.** Trzeba przyznać, że sytuacja z polskimi znakami jest dość dziwna. W Polsce, gdzie od szkoły podstawowej tyle czasu i energii poświęca się nauczaniu ortografii z użyciem polskich liter, m.in. ż i ó, okazują się one zbędne, gdyż produkuje się drukarki bez tych liter, **a nawet nie ma normy przypisującej im kody!**

Definiowanie znaków drukarki

Współczesne drukarki mozaikowe, oprócz drukowania określonych znaków o kształtach zdefiniowanych we własnej pamięci, **potrafią** jeszcze **drukować rysunki. Otrzymują wtedy dane z komputera w inny sposób, tzw. graficzny** (mówi się, że pracują w trybie graficznym, w przeciwieństwie do trybu znakowego). Posiadanie tej możliwości przez drukarkę określa jej atrybut "graficzna". **W trybie graficznym drukarka może wypisać** każdą kombinację kropek, a więc **także litery polskie, jednak z wielu powodów korzystanie z tej drogi jest niewygodne** i niezbyt zgrabne. **W większości** współczesnych **drukarek** (np. Seikosha GP500A), choć na szczęście nie we wszystkich, **jest to jednak jedyna przewidziana droga definiowania własnych znaków. Są też takie drukarki, jak np. Star SG-15 lub SG-10, w których można** zdefiniować **własne znaki** oprócz standardowych i **używać ich w normalnym trybie znakowym, do którego przygotowane są wszelkie edytory**, systemy operacyjne **i całe oprogramowanie**, w tym m.in. systemy wejścia-wyjścia znakowego różnych języków programowania.

Współczesna **drukarka mozaikowa** jest złożonym urządzeniem o rozbudowanej elektronice, zaopatrzonym w pamięć i rozpoznającym w potoku danych skierowane do niego rozkazy. Zawiera pamięć typu ROM, w której przechowuje pełną informację o kształcie liter i innych znaków drukowanych **w trybie znakowym. Z komputera otrzymuje tylko jednobajtowy** numer **(kod) znaku** traktowany jako **indeks do tablicy** zapisanej w tej pamięci, **z której pobiera kilkunastobajtową informację o kształcie znaku. Zysk z posługiwania się poza drukarką**, czyli w komputerze i pomiędzy komputerem a drukarką, **tylko kodem znaków** (kilkunastokrotnie bardziej zwięzłym od kodowanego opisu znaku) **jest oczywisty i niemały. Przypisanie kodom znaków ich** kształtu, czyli **znaczenia, określa w zakresie** siedmiu bitów, tj. **do** dziesiętnej **wartości** 127, ogólnie przyjęta **norma ASCII. Bajtowa** (czyli 8-bitowa) **organizacja pamięci** współczesnych mikrokomputerów w naturalny sposób **umożliwia rozszerzenie zakresu** wartości **kodów** o 1 bit, czyli **do wartości** dziesiętnej **255** (kod zwany Extended ASCII). **Przypisanie znaczenia kodom z zakresu 128-255 bywa różne** w różnych komputerach. **IBM PC** stał się niezaprzeczalnym standardem, należy więc spodziewać się, że **przyjęty w nim wybór znaczenia kodów stanie się** także **powszechnym standardem.**

W normie ASCII, tj. **w zakresie** wartości dziesiętnych **kodu 0-127, przewidziane są**, co prawda, pewne **wartości dla znaków narodowych (tab. 1), jednak** powszechnie używane **znaki** amerykańskie **(USA) są** na tyle **cenne w praktyce**, że skutki rezygnacji z nich na korzyść znaków polskich byłyby uciążliwe (także w Polskiej Normie PN/T-42112.01 obowiązującej od 1 stycznia 1980 roku przyjęto znaki amerykańskie). Inaczej jest ze znakami **z zakresu** wartości dziesiętnych **128-255**, gdzie **wiele znaków proponowanych w IBM PC jest w naszych warunkach praktycznie zbędnych.**

1) Druk pogrubiony stanowi konspekt. Jest to przykład nowych możliwości, jakie dostajemy do dyspozycji i propozycja nowego zwyczaju, jednego z takich, jakie prawdopodobnie zrodzą się w wyniku wykorzystywania tych możliwości.

2) Polskie litery istnieją. Uważam (ja, nie Redakcja!), że można mieć różne poglądy na temat celowości dalszego stosowania polskich liter w życiu, a szczególnie w technice komputerowej. Nie są one wygodne, ani nie widać żadnego zysku z ich użycia, stwarzają natomiast wiele problemów. Poglądy te nie mają jednak znaczenia w dniu dzisiejszym, kiedy przechodzimy do następnej epoki w dziejach pisma i stoimy przed koniecznością wprowadzenia polskich liter do nowych technik.

Tabela 1. Znaki narodowe w zakresie podstawowego ASCII w drukarkach SG-10 i SG-15

Kod dziesiętny	35	64	91	92	93	94	96	123	124	125	126
USA	#	@	[	\	]	^	`	{	!	}	~
Francja	£	à	°	ç	§	^	`	é	ù	è	¨
Niemcy	#	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
Anglia	£	@	[	\	]	^	`	{	!	}	~
Dania	#	@	Æ	Ø	Å	^	`	æ	ø	å	~
Szwecja	#	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Włochy	#	§	°	ç	é	^	ù	à	ò	è	ì
Hiszpania	#	@	¡	Ñ	¿	^	`	¨	ñ	}	~

Wspomniane już **drukarki SG-15 i SG-10 mają** oprócz pamięci typu ROM, także sporą **pamięć** typu **RAM** wykorzystywaną do różnych celów. **Można do niej** m.in. **wpisać z komputera informację o kształcie zdefiniowanych przez siebie znaków, zastępujących chwilowo znaki ze stałego repertuaru drukarki. Właściwość ta** wydaje się naturalną konsekwencją rozwoju technologii i **najprawdopodobniej rozpowszechni się jako standard.** Można oczekiwać w przyszłości podobnej właściwości elektroniki sterującej monitorami ekranowymi. **W przypadku** potrzeby wprowadzenia **polskich znaków jest to właściwość wymarzona,** pozwalająca rozwiązać ten problem wyjątkowo prosto, ogólnie i w najbardziej właściwy sposób. **Entuzjazm ten** nie **jest podyktowany** jedynie lenistwem (które wyjątkowo u programisty jest w dużym stopniu cechą pozytywną), ale **rzeczywistą** poprawnością i **optymalnością tego rozwiązania, zachowującego wszelkie ogólne właściwości** trybu znakowego, **a przez to w naturalny sposób zgodnego z całym oprogramowaniem,** którego ojczyzną w małej części jest Polska.

Wybór kodów dla polskich znaków

Wykorzystując tę właściwość drukarki zbudowałem **program o nazwie POLDRUK, ładujący do jej pamięci informację o polskich literach.** Projektując go napotkałem na dwa podstawowe problemy:

- **jakie znaki zastąpić polskimi?**
- **jakimi metodami można korzystać z przyjętego rozwiązania?**

Oba te problemy są ze sobą ściśle powiązane.

Najbardziej pożądanym rozwiązaniem byłoby umieszczenie znaków polskich w podstawowym zakresie kodów, tzn. do wartości dziesiętnej 127, **w którym ogólnie każdy kod jest dostępny przez naciśnięcie jednego lub najwyżej dwóch klawiszy** (dodatkowo klawisza SHIFT tzn. "duże litery"). **W tym zakresie mieszczą się kody znaków narodowych** ASCII, z których należałoby skorzystać. Jak wspomniano, **jednak rezygnacja ze znaków amerykańskich** (#, @, [, \,], ^, `, {, !, }, ~) **byłaby znaczną stratą,** szczególnie w opisach oprogramowania, gdzie znaki te są często używane. Opisy oprogramowania są przecież największą z przewidywanych grup tekstów polskich. Użycie pojedynczych klawiszy do uzyskania znaków polskich, choć niewątpliwie wygodne, **wymagałoby** dorysowywania, naklejania lub innych **przeróbek wskazujących przypisanie liter klawiszom,** gdyż nie istnieje prosta reguła pozwalająca to zapamiętać. O ile na klawiszach można dorysować dodatkowe znaki, to zmiana kształtu liter **na ekranie monitora wiąże się z przeróbką elektroniki** i odejściem od standardu, czego (w zakresie kodów do wartości 127) należy koniecznie uniknąć. Za wyborem tego zakresu kodów do reprezentacji polskich liter przemawia natomiast organizacja najszerzej znanego edytora - **WordStar,** który **dopuszcza w tekście użytkownika tylko znaki z podstawowego zbioru ASCII,** a kodów powyżej wartości 127 używa jako sterujących. Edytor ten w obecnej postaci **nie zapewnia jednak wygodnego dostępu do wszystkich, coraz ciekawszych, możliwości nowoczesnych drukarek.**

Alternatywnym rozwiązaniem jest umieszczenie polskich liter w zakresie kodów powyżej wartości dziesiętnej **127. W tym zakresie nie ma ogólnie przyjętego standardu wprowadzania kodów za pomocą nie więcej niż dwóch klawiszy.** W wypadku IBM PC za standard, ale niedostatecznie prosty i niedostatecznie ogólny, można uznać sposób polegający na wpisywaniu wartości dziesiętnej kodu na klawiaturze numerycznej, przy jednoczesnym wciśnięciu klawisza ALT. Wymaga to jednak naciśnięcia zwykle czterech klawiszy na jeden znak. Ten brak standardu jest kłopotliwy tylko przy wprowadzaniu danych z klawiatury do programów, ale niewątpliwie jest istotny.

Tekst polski, w którym obecność polskich znaków jest konieczna, **powstawać będzie przy użyciu edytora.** Dość powszechnie znany jest **Professional Editor,** który **zapewnia bardzo wygodny** i naturalny **mechanizm, doskonale nadający się do wprowadzania polskich liter** z dowolnymi wskazanymi wartościami kodów. Są to definiowane i zapamiętywane na pomocniczym pliku makrowywołania (tzw. makrosy). Ciąg znaków reprezentowany przez dany makros jest wprowadzany podczas pisania tekstu przez naciśnięcie klawisza litery lub cyfry (stanowiącej nazwę tego makrosu) wraz z klawiszem ALT. **Małe litery polskie mogą** więc **być wprowadzane przez naciśnięcie klawisza litery łacińskiej, której są modyfikacją, wraz z klawiszem ALT.** Sądzę, że bardziej wygodnej reguły być nie może. Pozostają więc do określenia reguły wprowadzania dużych liter i jednej z liter ż lub ź. **Duże litery** kreskowane (tj. C, S, Z, N, O) lub z ogonkiem (tj. Ą i Ę) **można uzyskać przez drukowanie dwóch znaków na jednym miejscu. Z uwagi na rzadkość występowania** tych liter, cenę makrosów, które są wygodnym narzędziem do różnych innych celów, oraz trudności ze znalezieniem prostej reguły wskazania makrosu (klawisze do naciśnięcia wraz z ALT), przyjęto ten sposób ich drukowania. **Dostawianie ogonka (czyli haka) po napisaniu dużego A lub E wykonuje makros h, a dostawianie kreski nad C, S, Z, N i O - makros i, kojarzący się ze zmiękczeniem.** Duże litery z ogonkiem lub kreską widać więc na ekranie w postaci trzech znaków.

Omówiona grupa liter polskich małych i dużych **stanowi** pewną jednolitą **całość, dlatego należałoby konsekwentnie makros o nazwie z przeznaczyć dla litery ż. Kształt** graficzny **ż i ł należy zdefiniować** w drukarce **dla liter dużych i małych.**

Litera **ł ma** na ekranie **piękny odpowiednik** w postaci znaku funta **£, a na odpowiednik ż** na ekranie **można wybrać** literę **æ i Æ**, występującą w postaci małej i dużej, a zbędną w polskich tekstach. **Kształt małego æ jest** bardzo **zbliżony do x, więc x można przyjąć jako nazwę makrosu dla ż** (ż występuje znacznie częściej od ź, przez co szybciej można przyzwyczaić się do takiego wyboru). Przy wyznaczaniu makrosu **dla ł i ż przyjęto następną literę w alfabecie po użytej dla makrosu małej litery ł i ż, czyli odpowiednio m i y.** Dowolność wyboru kodu pozwala na zgrabne **przypisanie polskim literom znaków** wyświetlanych **na ekranie,** tak że ich znaczenie w większości przypadków jest oczywiste, a w pozostałych **łatwo się kojarzy** z właściwym kształtem litery. Wybrane i **proponowane jako standard** w przyszłych rozwiązaniach **przypisanie znaków i kodów przedstawiono w tab. 2.**

Tabela 2. Polskie litery w zakresie wartości kodów 128-255

Kod dziesiętny	160	130	155	135	158	164	162	159	156	145	146	166	39
Litera polska	ą	ę	ć	ś	ż	ń	ó	ł	Ł	ż	Ż	'	'
Znak zastąpiony	á	é	¢	ç	¿	ñ	ó	£	£	æ	Æ	ª	'
Litera użyta z klawiszem ALT	a	e	c	s	z	n	o	l	m	x	y	h	i

Professional Editor ma jeszcze drugą przewagę nad WordStarem – **pozwala bardzo łatwo wprowadzić dowolne kody** między wartościami 128 a 255, którym odpowiada wiele cennych **znaków, jak np. Σ, α, β, π, σ, μ, δ, ∈, ∩, ≡, ±, ≥, ≈,** znaki do rysowania tabel itp., **oraz rozkazy do drukarki kontrolujące zmiany kroju liter, podkreślenia, pogrubienia,** indeksowanie itp. W efekcie **umożliwia on pełne wykorzystanie możliwości drukarki.**

Rozwiązanie polegające na przyjęciu kodów znaków polskich **w zakresie** wartości **128-255** (Extended ASCII) nie jest jednak całkiem ogólne, gdyż **nie zapewnia ogólnej metody wprowadzania znaków polskich jako danych z klawiatury do programów.** W tym wypadku pozostaje zaproponowanie standardu w zakresie kodów 0-127, chyba że powstanie ogólna opcja dla wszelkich systemów operacyjnych lub modyfikacja klawiatury nie naruszająca pozostałych standardów.

W zakresie podstawowego kodu ASCII zastąpionio polskimi literami znaki narodowe, przy czym zamiast rezygnować z nawiasów kwadratowych **"[" i "]",** które stosuje się np. do odsyłaczy literaturowych, **użyto znaków "&", "$" i dodatkowo podkreślenia,** które drukarka wykonuje innymi środkami – znaczenie tych znaków na ekranie jest znacznie łatwiej zapamiętać. **Duże litery,** tak jak w rozwiązaniu poprzednim, **tworzy się przez podwójne pisanie na jednym miejscu** – taką możliwością dysponuje WordStar. **W wypadku liter Ż i Ł,** dla których w drukarce trzeba zdefiniować także **literę dużą,** pożądane jest **umieszczenie** jej **na jednym klawiszu z małą, rozróżnianą** jak w wypadku pozostałych liter **klawiszem SHIFT. Obok klawisza Z, jest** odpowiedni do tego celu **klawisz ze znakami "\" i "|",** które przywodzą **na** myśl przekreślenie **litery Ż, a w pobliżu klawisza L – klawisz ze znakami "'" i "~",** który mógłby oznaczać literę **ł. Przyjęte** w tym rozwiązaniu **przypisanie znaków i kodów przedstawiono w tab. 3.**

Tabela 3. Polskie litery w zakresie znaków narodowych podstawowego kodu ASCII

Kod dziesiętny	35	64	91	92	93	94	96	123	124	125	126	38	36	95
USA	#	@	[	\	]	^	`	{	¦	}	~	&	$	_
Polska	ż	ą	[	ś	]	ń	ł	ć	Ż	ó	Ł	ę	ś	.

Jak wynika z tych rozważań, przynajmniej na razie nie wystarczy jeden wybór kodów dla znaków polskich. **Przewiduję, że wybór w zakresie rozszerzonego kodu ASCII będzie coraz bardziej** ogólny i **wystarczający. Należy dążyć do jak najszybszego znormalizowania tego wyboru** (lub innego, bardziej uzasadnionego).

Układ klawiszy w polskich maszynach do pisania różni się od klawiatury IBM PC. **Dla osób piszących bez patrzenia na klawisze,** jest to problem wymagający **zmiany znaczenia klawiszy na polski standard. Dokonuje** tego **program POLKLAW przeznaczony do wprowadzania dużej ilości tekstu na plik. Zamiana znaczenia klawiszy jest w pełni określona.** Program POLKLAW ma możliwość poprawiania tekstu jedynie w zakresie wpisywanej linii. Powstający plik oraz tekst wyświetlany na ekranie są w standardzie przedstawionym tabelą 2 tak, że **plik nadaje się do poprawiania edytorem** Professional Editor i drukowania drukarką SG po zdefiniowaniu znaków polskich programem POLDRUK.

* * *

Podstawowym **zadaniem programu POLDRUK jest wprowadzanie do** pamięci **drukarki** informacji o kształtach **znaków polskich** i przypisanie im kodów **według opisanych reguł. Doświadczenie kilku miesięcy jego użytkowania wykazuje, że potrzeba zastosowania wyboru kodów, w zakresie podstawowego kodu ASCII, występuje wyjątkowo rzadko, a proponowany standard w zakresie Extended ASCII, przy użyciu edytora Professional Editor** (i innych), **okazuje się** wystarczająco wygodny i w zasadzie **wystarczający.**

Poza tym, POLDRUK ma możliwość zmiany przyporządkowania kodów znakom polskim, uzupełnienia o duże litery kreskowane i z ogonkiem oraz **wprowadzenia własnych znaków o dowolnym kształcie i dowolnym kodzie według danych w postaci obrazka** przygotowanego na pliku za pomocą dowolnego edytora.

Posiadacze drukarek prostszych typów, nie mających możliwości wprowadzenia nowych znaków, **oraz użytkownicy, dla których znaki na ekranie powinny mieć porządne, polskie kształty, mogą** (w większości urządzeń) **przeprogramować pamięć stałą,** w której przechowuje się informacje o kształcie liter – wprowadzając do niej, zamiast istniejących tam znaków, znaki polskie. Nie polecałbym jednak zmieniania w ten sposób znaków z podstawowego kodu ASCII, natomiast zmiana znaczenia kodów powyżej wartości dziesiętnej 127, wydaje się rozwiązaniem poprawnym i pożądanym. **Przed taką zmianą trzeba jednak** określić, przedyskutować i **przyjąć jeden standard.** Przedstawiony wybór można traktować jako propozycję takiego standardu. Przeprogramowania pamięci typu ROM zawierającej znaki można dokonać w IPJ (tel. Warszawa 79-88-06).

ROMAN GRABOWICZ
JANUSZ ZALEWSKI
Warszawa

System operacyjny PC-DOS (2)

Po wstępie do systemu PC-DOS i przedstawieniu organizacji systemu plików, przechodzimy do omówienia poleceń systemowych.

POLECENIA SYSTEMU PC-DOS

Polecenia systemu PC-DOS, ze względu na miejsce ich rezydowania, można podzielić na dwa rodzaje: polecenia wewnętrzne i zewnętrzne.

Polecenia wewnętrzne (inaczej — wbudowane), po załadowaniu systemu stale przebywają w pamięci operacyjnej. W tym wypadku wykonanie polecenia odbywa się natychmiast po jego wydaniu.

Polecenia zewnętrzne rezydują na dysku w postaci plików z rozszerzeniem nazwy COM lub EXE. Aby wykonać polecenie zewnętrzne, system operacyjny najpierw ściąga odpowiedni plik do pamięci operacyjnej. Użytkownik może dołączyć swoje własne polecenia zewnętrzne przez utworzenie plików z rozszerzeniem nazwy COM lub EXE, używając translatorów i linkera. Przy wydawaniu polecenia pisze się tylko jego nazwę oraz ewentualnie listę parametrów.

Wśród poleceń systemu PC-DOS można wyróżnić następujące grupy:

- polecenia parametryzowania i konfigurowania systemu; ich działanie polega na ustawieniu odpowiednich parametrów, które pozostają w mocy przez cały czas działania mikrokomputera (aż do ponownego załadowania systemu operacyjnego),
- polecenia operowania plikami realizujące takie funkcje jak: usuwanie plików, drukowanie, wyświetlanie, formatowanie, zmiana nazwy, kopiowanie itd.; ich parametrami mogą być w niektórych wypadkach także nazwy standardowych urządzeń zewnętrznych,
- polecenia realizujące funkcje specyficzne dla dysku stałego,
- polecenia operowania skorowidzami, realizujące takie funkcje jak: zakładanie, usuwanie lub wyświetlanie skorowidza, zmiana skorowidza bieżącego itp.,
- polecenia filtrowania, służące do pobierania danych ze standardowego urządzenia zewnętrznego, przetwarzania ich i wyprowadzania na standardowe urządzenie zewnętrzne,
- makropolecenia, tj. polecenia złożone, składające się z kilku poleceń wewnętrznych lub zewnętrznych i dodatkowych poleceń pozwalających na zmianę przepływu sterowania,
- inne polecenia.

W opisie poleceń systemu PC-DOS przyjęto następujące konwencje notacyjne:

- słowa kluczowe, które występują w poleceniu obligatoryjnie, zapisywane są dużymi literami,
- pozycje zapisane w formie polecenia małymi literami powinny być zastąpione parametrami podanymi przez użytkownika,
- pozycje podane w nawiasach kwadratowych występują opcjonalnie,
- jeżeli pozycje oddzielone są kreską pionową |, to tylko jedna z nich może występować w poleceniu,
- wielokropek ... oznacza, że pozycja go poprzedzająca może występować dowolną liczbę razy,
- znaków "[", "]", "|", "..." nie umieszcza się w poleceniu,
- wszystkie elementy poleceń można pisać zarówno dużymi jak i małymi literami lub dowolną ich kombinacją (wprowadzone małe litery zamieniane są przez PC-DOS na duże).

Przyjęto następujące konwencje, dotyczące zapisu parametrów poleceń:

d: — oznacza stację dyskietek lub dysk stały; jeżeli zostanie opuszczony, to przyjmuje się stację domyślną,

path — odpowiada sekwencji [/][dirname][/dirname[...]] [1]) i oznacza ścieżkę dostępu przez nazwy skorowidzów; jeżeli po ścieżce dostępu podaje się nazwę pliku, to musi ona być także poprzedzona znakiem /; jeżeli występuje początkowy znak /, to ścieżka rozpoczyna się od skorowidza systemowego, jeżeli nie — to od skorowidza bieżącego,

filename — nazwa pliku (por. cz. 1 artykułu),

ext — rozszerzenie nazwy pliku, występujące po kropce,

filespec — odpowiada sekwencji [d:]filename[.ext],

file — odpowiada sekwencji [d:][path] filename [.ext].

Polecenia parametryzowania systemu

DATE [mm-dd-yy]

Pozwala wprowadzić do systemu nową lub zmienić już istniejącą datę. Datę wprowadza się jako parametr polecenia, gdzie: **mm** oznacza miesiąc, **dd** — dzień, a **yy** — ostatnie dwie cyfry roku. Jest to polecenie wewnętrzne.

TIME [hh:mm:ss.xx]

Pozwala wprowadzić lub zmienić już istniejący czas systemowy. Aktualny czas podaje się w przedstawionej postaci, gdzie **hh** oznacza godzinę, **mm** — minuty, **ss** — sekundy, a **xx** — setne części sekundy. Jeżeli końcowa część wprowadzonego czasu zostanie pominięta, to system przyjmuje wartość 0. Jest to polecenie wewnętrzne.

VER

Jest to polecenie wewnętrzne, wyświetlające numer aktualnie używanej wersji systemu PC-DOS.

PROMPT [text]

Jest to polecenie zewnętrzne, które pozwala na zmianę znaku gotowości systemu. Standardowo — system wyświetla symbol domyślnej stacji dyskowej oraz znak większości „>". Użytkownik może go zmienić na dowolny tekst, zawierający także znaki o specjalnym znaczeniu. Przy każdorazowym sygnalizowaniu przez system stanu gotowości wyświetla się nowo wprowadzony tekst, np. bieżącą datę, aktualny czas, bieżący skorowidz itp.

BREAK [ON|OFF]

Jest to polecenie wewnętrzne o standardowo przyjętej wartości parametru OFF. Oznacza, że w trakcie realizacji programu sprawdza się, czy naciśnięto klawisze CTRL-BREAK (równoznaczne z przerwaniem programu). Dotyczy tylko realizacji wejścia-wyjścia na urządzenia standardowe, na drukarkę oraz dodatkowe urządzenia zewnętrzne, a nie dotyczy dysków. Przyjęcie parametru ON umożliwia przerwanie programu w każdej chwili, gdy odwołuje się on do funkcji systemowych. Polecenie BREAK bez parametrów powoduje wyświetlenie aktualnie przyjętej wartości parametru.

---

[1]) Ze względu na brak w drukarni znaku **ukośnika** (ang. backslash), w tekście artykułu zastąpiono go znakiem dzielenia „/". Znak ukośnika jest symetryczny do znaku dzielenia „/" względem osi pionowej i w kodzie ASCII ma przypisaną wartość 92.

SYS d:

Polecenie to pozwala przenieść pliki IBMBIO.COM i IBMDOS.COM systemu operacyjnego na dysk d. Normalnie pliki te są zaznaczone jako ukryte, niewidoczne przy zwykłym korzystaniu ze skorowidza, umieszczone w ściśle określonym miejscu na dysku — przed innymi plikami. Z tego powodu polecenie SYS może działać tylko na dyskach sformułowanych poleceniem

FORMAT d:/S

lub

FORMAT d:/B

albo odpowiednio przygotowanych przez dystrybutora. Jest to polecenie zewnętrzne.

ASSIGN [x=y [...]]

Polecenie to powoduje, że wszystkie odwołania do dysku **x** będą kierowane faktycznie do dysku **y**. Pozwala to na realizowanie operacji wejścia-wyjścia na innym dysku i odpowiada przypisaniu urządzeniu fizycznemu różnych symboli logicznych. Polecenie ASSIGN bez parametrów przywraca stan pierwotny, tzn. numerom urządzeń fizycznych nadaje te same numery logiczne. Jest to polecenie zewnętrzne.

GRAPHICS

Polecenie to umożliwia wyprowadzenie na drukarkę kopii ekranu z zawartością graficzną. Do tego celu konieczna jest drukarka z możliwościami graficznymi i karta kolorowego monitora graficznego. Wykonanie kopii ekranu jest możliwe przez naciśnięcie klawiszy SHIFT-PRSC lub programowo przez realizację sekwencji rozkazów:

PUSH BP
INT 5
POP BP.

Polecenie zwiększa rezydentną część systemu PC-DOS o 736 bajtów. Jest to polecenie zewnętrzne.

CTTY device

Polecenie to zmienia standardowe wejście-wyjście z klawiatury i monitora na inne urządzenie podane jako parametr. Nowe urządzenie powinno mieć możliwość realizowania zarówno operacji wejścia, jak i wyjścia. Parametr może przyjmować wartości AUX, COM1, COM2 lub określać inne urządzenia znakowe. W tym ostatnim wypadku użytkownik odpowiada za zainstalowanie odpowiedniego programu obsługi urządzenia (ang. device driver). Skutki działania polecenia CTTY odnoszą się tylko do programów korzystających z wywołań funkcji systemowych. Inne programy, które nie używają funkcji systemu, np. BASIC, nie mogą wykorzystać polecenia CTTY. Polecenie CTTY CON przywraca przypisanie standardowych urządzeń wejścia-wyjścia do klawiatury i ekranu monitora. Jest to polecenie wewnętrzne.

MODE

Rozróżnia się cztery formy tego polecenia:

1. MODE LPT #: [n][,[m][,P]]

2. MODE n lub MODE [n],m[,T]

3. MODE COMn:baud[,parity[,databits[,stopbits[,P]]]]

4. MODE LPT #:=COMn

Polecenie ustala tryb pracy drukarki, kolorowego monitora graficznego lub asynchronicznego układu transmisji i przeadresowuje wyprowadzanie do układu transmisji asynchronicznej zamiast na drukarkę. Użycie polecenia w formacie 1, 3 lub 4 powoduje powiększenie rezydentnej części systemu operacyjnego o około 256 bajtów.

Postać pierwsza służy do ustalania trybu pracy drukarki. Poszczególne parametry oznaczają:

— numer drukarki 1, 2 lub 3
n — liczbę znaków w linii, 80 lub 132
m — liczbę wierszy na cal, 6 lub 8
P — ponawianie prób przy błędach przeterminowania (ang. time-out).

Postać druga służy do ustalania trybu pracy karty kolorowego monitora graficznego. Poszczególne parametry oznaczają:

n — tryb pracy monitora (kolorowy, czarno-biały, graficzny)
i — liczbę znaków w wierszu
m — przesuwanie obrazu w prawo lub w lewo, R lub L
T — wyświetlanie obrazu kontrolnego w celu regulacji prawidłowego wyświetlania.

Postać trzecia służy do ustalania parametrów protokołu transmisji. Poszczególne parametry oznaczają:

n — numer układu transmisji asynchronicznej
baud — szybkość transmisji
parity — N bez kontroli parzystości, 0 kontrola nieparzystości, E kontrola parzystości (wartość domyślna)
databits — 8 lub 7 (wartość domyślna)
stopbits — 1 lub 2
P — znaczenie jak w postaci pierwszej.

Postać czwarta umożliwia wykorzystanie układu transmisji asynchronicznej jako wyjścia dla strumienia danych skierowanych do drukarki. Poszczególne parametry oznaczają:

— numer drukarki 1, 2 lub 3
n — numer układu transmisji asynchronicznej, 1 lub 2

Jest to polecenie zewnętrzne.

SET [name=[parameter]]

Procesor poleceń posiada bufor, do którego wpisywane są ciągi znaków charakteryzujące środowisko. Dostępne są one dla każdego polecenia lub programu aplikacyjnego, który w zależności od ich znaczenia podejmuje odpowiednie działania. Polecenie to użyte bez parametru powoduje wyświetlenie zawartości bufora, a użyte jedynie z parametrem name= usuwa ten parametr z bufora.

POLECENIA OPEROWANIA PLIKAMI

FORMAT[d:][/S][/1][/8][/V][/B]

Jest to polecenie zewnętrzne, które formatuje dysk, kontroluje jakość nośnika magnetycznego, blokując zapis na ścieżkach, na których powierzchnia magnetyczna została uszkodzona, inicjalizuje tablicę alokacji plików FAT (ang. File Allocation Table) oraz tworzy kopię plików systemowych. Polecenie FORMAT musi być wykonane przed pierwszym użyciem nośnika. Jeżeli zastosuje się je do nośnika już używanego, to niszczy wszystkie informacje na nim zapisane. Poszczególne parametry mają następujące znaczenie:

S — pliki systemowe IBMBIO.COM, IBMDOS.COM, COMMAND.COM są także kopiowane na formatowany dysk
1 — dysk jest formatowany jednostronnie
8 — dysk jest formatowany z ośmioma sektorami na jednej powierzchni ścieżki (wartość domyślna wynosi 9 sektorów)
V — żądanie umieszczenia etykiety nośnika
B — na dyskach z ośmioma sektorami na ścieżce rezerwuje miejsce na pliki IBMBIO.COM i IBMDOS.COM.

VOL[d:]

Polecenie to wyświetla etykietę dysku. Jest poleceniem wewnętrznym.

ERASE file

Jest to polecenie wewnętrzne, które usuwa wyszczególniony plik (lub pliki) z określonego skorowidza na określonej stacji dyskowej. Nazwy plików i ich rozszerzenia mogą być zmienne, zapisane przy użyciu znaków "*", "?", np. poleceniem

ERASE *.*

usuwa się wszystkie pliki z bieżącego skorowidza na domyślnej stacji dyskowej.

DELETE file

Polecenie to działa dokładnie tak samo jak ERASE.

RENAME file filename[.ext]

Polecenie to zmienia nazwę i rozszerzenie nazwy pliku podanego jako pierwszy parametr na nazwę i rozszerzenie nazwy pliku podanego jako drugi parametr. W poleceniu RENAME można używać nazw zmiennych, tj. znaków "*", "?". Jest to polecenie wewnętrzne.

COPY

Jest to polecenie wewnętrzne. Rozróżnia się dwie jego postaci:

1. COPY [/A][/B]file[/A][/B]
[d:][path][filename[.ext]][/A][/B][/V]
2. COPY [/A][/B]file[/A][/B][+file[/A][/B]...]
[d:][path][filename[.ext]][/A][/B][/V]

Polecenie kopiuje plik (lub pliki) wymieniony jako pierwszy parametr na plik (lub pliki) wymieniony jako drugi parametr. Dla postaci drugiej dokonuje się także łączenia plików. Parametry V, A, B mają następujące znaczenie:

V — system operacyjny zapewnia kontrolę poprawności zapisu pliku wynikowego; parametr ten odpowiada działaniu polecenia VERIFY ON, ale z ograniczeniem do polecenia COPY

A — plik traktowany jest jako plik testowy (w kodzie ASCII); pliki źródłowe są kopiowane aż do napotkania pierwszego znaku End-Of-File (Ctrl-Z), ale bez tego znaku, a w pliku wynikowym znak End-Of-File jest dodawany przez to polecenie

B — cały plik, łącznie ze znakiem End-Of-File, jest kopiowany, a w pliku wynikowym znak ten nie jest dodawany.

Postać pierwsza powoduje kopiowanie pliku. Plik wynikowy może mieć taką samą nazwę jak plik źródłowy, ale musi znajdować się na innej stacji dyskowej lub w innym skorowidzu. Jeżeli plik wynikowy ma być w tym samym skorowidzu co źródłowy, to musi mieć inną nazwę.

Postać druga umożliwia łączenie plików źródłowych oddzielonych znakiem "+" i zapisanie ich na plik wynikowy. Stosowanie parametrów A i B umożliwia odpowiednie zapisanie znaku End-Of-File.

Polecenie to dopuszcza stosowanie nazw zmiennych, a także używanie nazw standardowych urządzeń zewnętrznych. Przykładowo, polecenie

COPY CON fileA

powoduje zapisanie (kopiowanie) danych wprowadzonych z klawiatury na plik fileA, na stacji domyślnej, w bieżącym skorowidzu.

DISKCOPY [d:] [d:][/1]

Jest to polecenie zewnętrzne, które kopiuje zawartość całej dyskietki źródłowej (pierwszy parametr) na dyskietkę wynikową (drugi parametr). Jeśli dyskietka wynikowa jest niesformatowana lub sformatowana niezgodnie z dyskietką źródłową, to polecenie formatuje ją przed kopiowaniem. Polecenie to nie ma zastosowania do dysku stałego. Parametr /1 powoduje, że kopiowana jest tylko pierwsza strona dyskietki. Z logicznego punktu widzenia polecenie

DISKCOPY A: B:

(o ile dyskietka B jest sformatowana) odpowiada poleceniu

COPY A: B:

ale fizyczne rozmieszczenie informacji na dyskietkach wynikowych może być różne.

COMP [d:][path][filename[.ext]] [d:][path][filename[.ext]]

Jest to polecenie zewnętrzne, które powoduje porównanie zawartości pliku podanego jako pierwszy parametr z zawartością pliku podanego jako drugi parametr. Dopuszcza się stosowanie nazw zmiennych, co umożliwia porównanie dwóch grup plików ze sobą. Jeżeli polecenie wykryje w plikach niezgodne bajty, to sygnalizuje błąd podając numer bajtu, licząc od początku pliku, w którym wykryto niezgodność oraz zawartość bajtów w obu plikach.

DISKCOMP [d:] [d:][/1][/8]

Polecenie to porównuje zawartość fizyczną dwóch całych dyskietek — nie ma zastosowania do dysku stałego. Parametry posiadają następujące znaczenie:

1 — porównanie tylko pierwszych stron dyskietek,
8 — porównanie tylko ośmiu sektorów na ścieżce.

Jeżeli wystąpi niezgodność między dyskietkami, to polecenie sygnalizuje numer strony dyskietki (0 lub 1) oraz numer odpowiedniej ścieżki. Jest to polecenie zewnętrzne. Polecenia COMP i DISKCOMP najczęściej wykorzystuje się dla weryfikacji poprawności wykonania poleceń COPY i DYSKCOPY.

VERIFY [ON|OFF]

Wykonanie polecenia wewnętrznego

VERIFY ON

powoduje, że po każdym zapisie na dysk system operacyjny dokona odczytu zapisanej informacji, aby sprawdzić, czy została ona zapamiętana poprawnie. Powoduje to zwiększenie czasu zapisu danych. Parametr OFF oznacza wyłączenie tej kontroli.

CHKDSK [d:][filename[.ext]][/F][/V]

Polecenie zewnętrzne CHKDSK analizuje skorowidze oraz tzw. tablicę FAT (ang. File Allocation Table) na wyszczególnionej lub domyślnej stacji dysków oraz generuje raport o stanie dysku. Jeżeli podana jest nazwa pliku, to polecenie wyświetli liczbę niespójnych obszarów, z których składa się plik. Parametry polecenia mają następujące znaczenie:
F — polecenie dokonuje korekty błędów znalezionych w skorowidzu i tablicy FAT; bez tego parametru wyświetla się jedynie raport błędów
V — wyświetla się dokładniejszy opis znalezionych błędów oraz informacje o przebiegu kontroli.

Polecenie umożliwia odnalezienie bloków informacji, które nie należą do żadnego pliku i umieszczenie ich w plikach o nazwach FILEnnnn.CHK.

RECOVER

Wyróżnia się dwa formaty tego polecenia zewnętrznego:
1. RECOVER file
2. RECOVER d:

W wypadku uszkodzenia dysku i straty części danych, polecenie to pozwala odzyskać dane zapisane w sektorach nieuszkodzonych.

Polecenie w postaci pierwszej powoduje odczytanie określonego pliku i usunięcie z niego uszkodzonych sektorów, których adresy umieszcza się w specjalnej tablicy. Stają się one niedostępne. Dalsze korzystanie z pliku wymaga najczęściej dodatkowych akcji użytkownika. Polecenie dopuszcza stosowanie nazw zmiennych, jednak tylko jeden plik (pierwszy odpowiadający wyspecjalizowanej nazwie) zostaje poprawiony.

Polecenie w postaci drugiej stosuje się w wypadku uszkodzeń nośnika w obrębie sektorów opisujących skorowidz. Pozwala ono odzyskać wszystkie pliki na określonym dysku.

TYPE file

Polecenie wewnętrzne TYPE powoduje wyświetlenie zawartości pliku na ekranie. Należy zwrócić jednak uwagę, że jedynie pliki tekstowe ukazują się w postaci łatwo czytelnej dla człowieka.

PRINT [[d:][filename[.ext]][/T][/C][/P]..]

Polecenie zewnętrzne PRINT służy do wyprowadzania plików na drukarkę. Pełni ono funkcję spoolera i jest jedynym poleceniem, które umożliwia działanie wieloprogramowe. Wyspecyfikowane pliki przed drukowaniem są umieszczane w kolejce, która może obejmować do dziesięciu plików. W trakcie drukowania użytkownik może kontynuować pracę wykonując inne polecenia systemu. Poszczególne parametry mają następujące znaczenie:
P — wartość domyślna, plik umieszczany jest w kolejce do wydruku
C — plik jest usuwany z kolejki do wydruku
T — wszystkie pliki są usuwane z kolejki do wydruku.

Wydanie polecenia PRINT bez wyspecjalizowania pliku i parametrów powoduje wyświetlenie listy nazw plików znajdujących się w kolejce do wydrukowania. Przy pierwszym wykonaniu polecenia PRINT rezydentna część systemu operacyjnego powiększa się o około 3200 bajtów. Przed rozpoczęciem pierwszego wydruku wyświetlane jest pytanie, na które urządzenie zewnętrzne ma być skierowany wydruk. Użytkownik może wybrać jedno z urządzeń: LPT1, LPT2, LPT3, PRN (wartość domyślna), COM1, COM2 lub AUX. Urządzenie to staje się niedostępne dla innych poleceń, dopóki nie zostaną wydrukowane wszystkie pliki z kolejki. Pliki umieszczane w kolejce do wydruku muszą znajdować się w skorowidzu bieżącym. Dopóki nie zostanie zakończone drukowanie pliku, nie może on być skasowany (za pomocą polecenia DELETE lub ERASE), ani w żaden sposób zmieniony. Dyskietki z plikami do wydruku nie można usuwać z kieszeni stacji. Dopuszcza się stosowanie nazw zmiennych.

WALDEMAR DWORAKOWSKI
Instytut Maszyn Matematycznych
Warszawa

Komputery osobiste IBM PC (2)

Druga część artykułu poświęcona jest dodatkowym układom sterowników urządzeń we-wy, które są stosowane w mikrokomputerach IBM PC i kompatybilnych, w celu rozbudowy ich możliwości funkcjonalnych. Przedstawiony opis stanowi jedynie pobieżny przegląd niektórych układów sterowników, pojawiających się na rynku polskim. Z oczywistych przyczyn artykuł ten nie może być kompendium wiedzy o wyposażeniu komputerów IBM PC. Dokładne informacje o konstrukcji i parametrach sterowników można znaleźć w publikacjach fachowych w języku angielskim lub w dokumentacji technicznej tych urządzeń, choć niestety, nie zawsze dokumentacja dostarczana przez producenta zawiera poszukiwane informacje.

STEROWNIK GRAFIKI KOLOROWEJ

Płytka grafiki kolorowej została pierwotnie opracowana rzez firmę IBM, jednak obecnie jest produkowana przez wielu innych wytwórców. Umożliwia ona uzyskiwanie kolorowych i monochromatycznych obrazów graficznych lub znakowych, odpowiednio na monitorach barwnych i monochromatycznych, a po dodaniu modulatora — także na ekranie telewizora. Układ sterownika posiada wbudowaną pamięć RAM o pojemności 16 KB, umożliwiającą przechowywanie wielu stron tekstu (np. do 4 stron o wielkości 25 linii po 80 znaków) lub jednego obrazu graficznego. Płyta może pracować w jednym z ośmiu trybów:

- tryb znakowy 25 linii po 40 znaków czarno-białych,
- tryb znakowy 25 linii po 40 znaków w kolorze (16 kolorów),
- tryb znakowy 25 linii po 80 znaków czarno-białych,
- tryb znakowy 25 linii po 80 znaków w kolorze (16 kolorów),
- tryb graficzny 200 linii po 320 punktów w kolorze (4 kolory),
- tryb graficzny 200 linii po 320 punktów czarno-białych,
- tryb graficzny 200 linii po 640 punktów czarno-białych,
- tryb graficzny 100 linii po 160 punktów w kolorze (16 kolorów).

Tryb przedostatni nie jest wykorzystywany w systemie i nie ma dla niego oprogramowania. Dla pozostałych trybów pracy istnieje oprogramowanie podstawowe w systemie BIOS, umożliwiające przełączanie trybów i wykonywanie podstawowych operacji na ekranie.

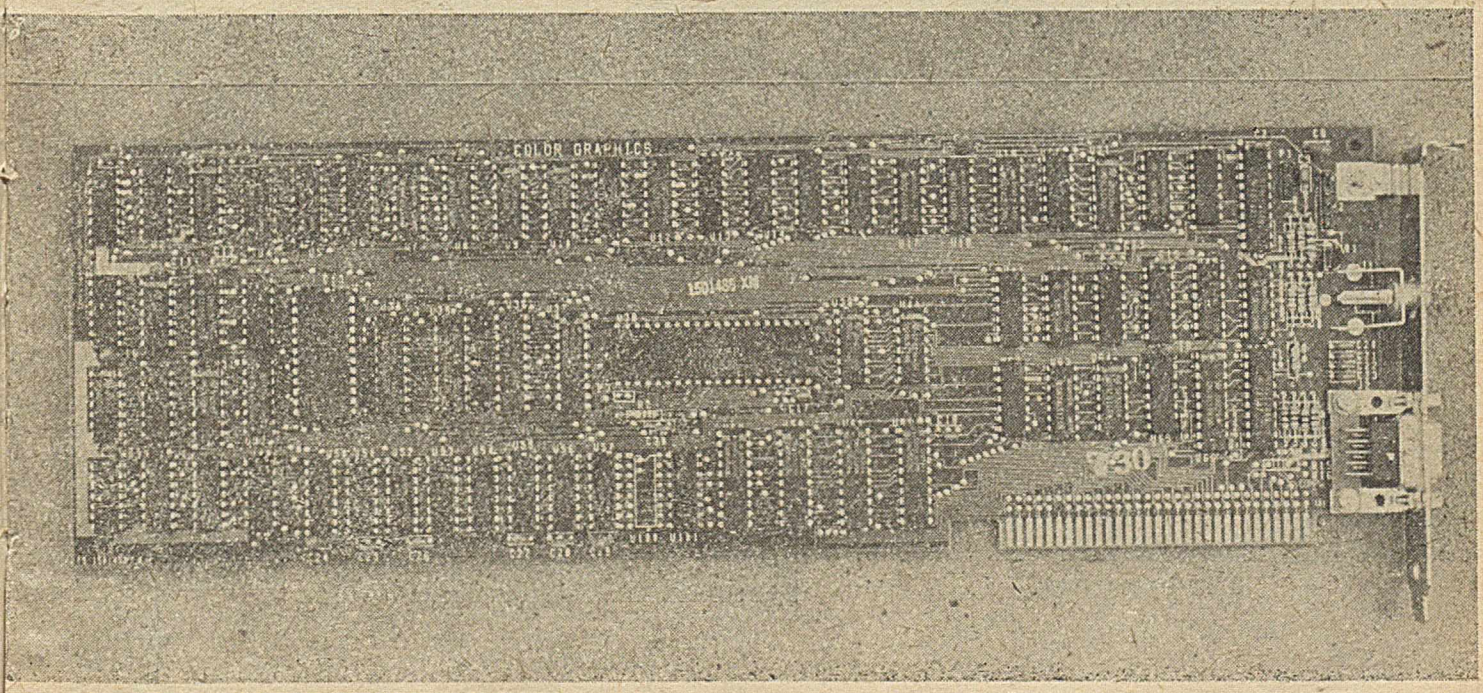
Do operacji tych należą:

- ustawienie trybu pracy,
- ustawianie wielkości kursora,
- pozycjonowanie kursora,
- odczyt pozycji kursora,
- odczyt pozycji pióra świetlnego,
- wybór strony aktywnej (wyświetlanej),
- przesunięcie obszaru ekranu do góry,
- przesunięcie obszaru ekranu do dołu,
- odczyt znaku wraz z atrybutem,
- zapis znaku wraz z atrybutem,
- zapis znaku bez atrybutu,
- ustawienie palety kolorów,
- zapisanie punktu graficznego,
- odczytanie punktu graficznego,
- zapisanie znaku z interpretacją,
- odczytanie trybu pracy sterownika.

Stosowanie tego sterownika pozwala na uzyskiwanie obrazów alfanumerycznych za pomocą matrycy znakowej 8×8 punktów oraz pola znaku 5×7 punktów. Umożliwia to znośną czytelność tekstu przy użyciu odpowiedniej jakości monitora, jednak do pracy z tekstem jest zalecany sterownik monochromatycznego monitora alfanumerycznego.

STEROWNIK DYSKU STAŁEGO WINCHESTER

Sterowniki dysku stałego stanowią rodzinę wzajemnie kompatybilnych urządzeń inteligentnych do obsługi pamięci masowych, opartych na dyskach typu Winchester. Najbardziej popularny w Polsce sterownik, oparty na ele-



Fot. 1. Płyta grafiki kolorowej (fot. PZ KAREN)

mentach firmy Western Digital, zawiera układy WD10C20, WD11C00, WD101OA oraz mikrokomputer jednoukładowy 8049. Dodatkowo ma on pamięć typu 2764 (EPROM 8K×8 bitów) oraz pamięć RAM 2K×8 bitów. Pamięć EPROM zawiera oprogramowanie sterujące pracą sterownika i stanowi rezydentną nakładkę (rozszerzenie) systemu BIOS. Mikrokomputer jednoukładowy zapewnia testowanie oraz ułatwia sterowanie dyskiem (pozycjonowanie głowic). Należy dodać, że sam napęd dysku stałego zazwyczaj ma własny mikrokomputer jednoukładowy.

Podczas inicjowania systemu, oprogramowanie zawarte w pamięci EPROM (po testowaniu płyty głównej) przejmuje sterowanie i przeadresowuje przerwanie (a właściwie wektor tego przerwania) związane z obsługą dysku elastycznego, przechodząc do obsługi dysku stałego. Podczas inicjowania systemu dokonywane jest też testowanie pamięci RAM zawartej na płytce i innych podzespołach, np. napędu dyskowego.

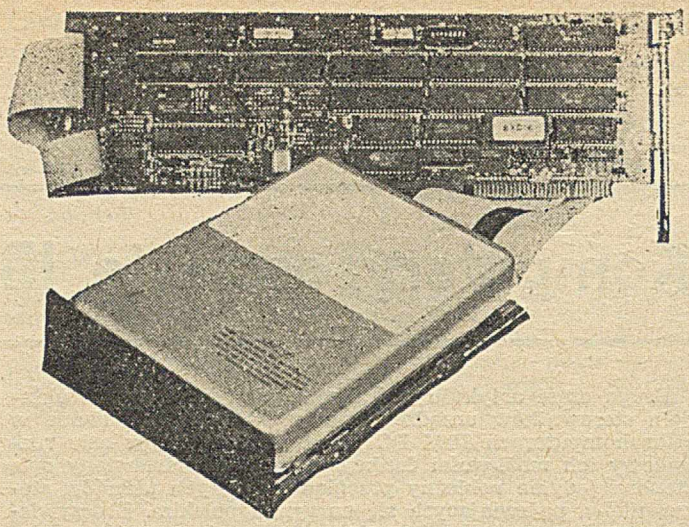
Sterownik pozwala na równoczesną współpracę z czterema napędami dyskowymi o pojemności od 5 do ok. 100 MB. Niestety, struktura oprogramowania systemowego uniemożliwia obsługę ciągłych obszarów o pojemności większej niż ok. 34 MB.

Oprogramowanie systemowe umożliwia tworzenie, niezależnie od fizycznej konfiguracji dysku stałego, wielu tzw. stref logicznych (ang. partitions), dzięki czemu możliwy jest podział jednego dużego dysku o pojemności do 130 MB na cztery logiczne napędy o pojemności ok. 32 MB. Podział dysku, aczkolwiek zapisany na dysku, jest niezależny od sterownika i zależy jedynie od oprogramowania systemowego.

Fizyczny zapis na dyskach stałych jest zbliżony do zapisu na dyskach elastycznych. Dysk składa się z 2 do 16 powierzchni, na każdej powierzchni może znajdować się ok. 300—600 ścieżek, które z kolei składają się z sektorów. Dokładne liczby sektorów, ścieżek i powierzchni są zależne od rodzaju napędu dyskowego. Zapis na dysku jest dokonywany zazwyczaj metodą MFM, przy czym szybkość transmisji jest znacznie większa niż w wypadku dysków elastycznych (wynosi zwykle ok. 5 Mb/s). Sposób zapisu na dyskach Winchester nie jest objęty żadnym standardem, co oznacza, iż wymiana sterownika może doprowadzić do niemożności odczytania zapisanej poprzednio informacji.

Prędkość obrotowa dysków Winchester wynosi zazwyczaj ok. 3600 obrotów na minutę. Przy tej prędkości głowica nie może ślizgać się po powierzchni, lecz unosi się nad nośnikiem, na wysokości kilku mikrometrów [1]). Nawet chwilowe

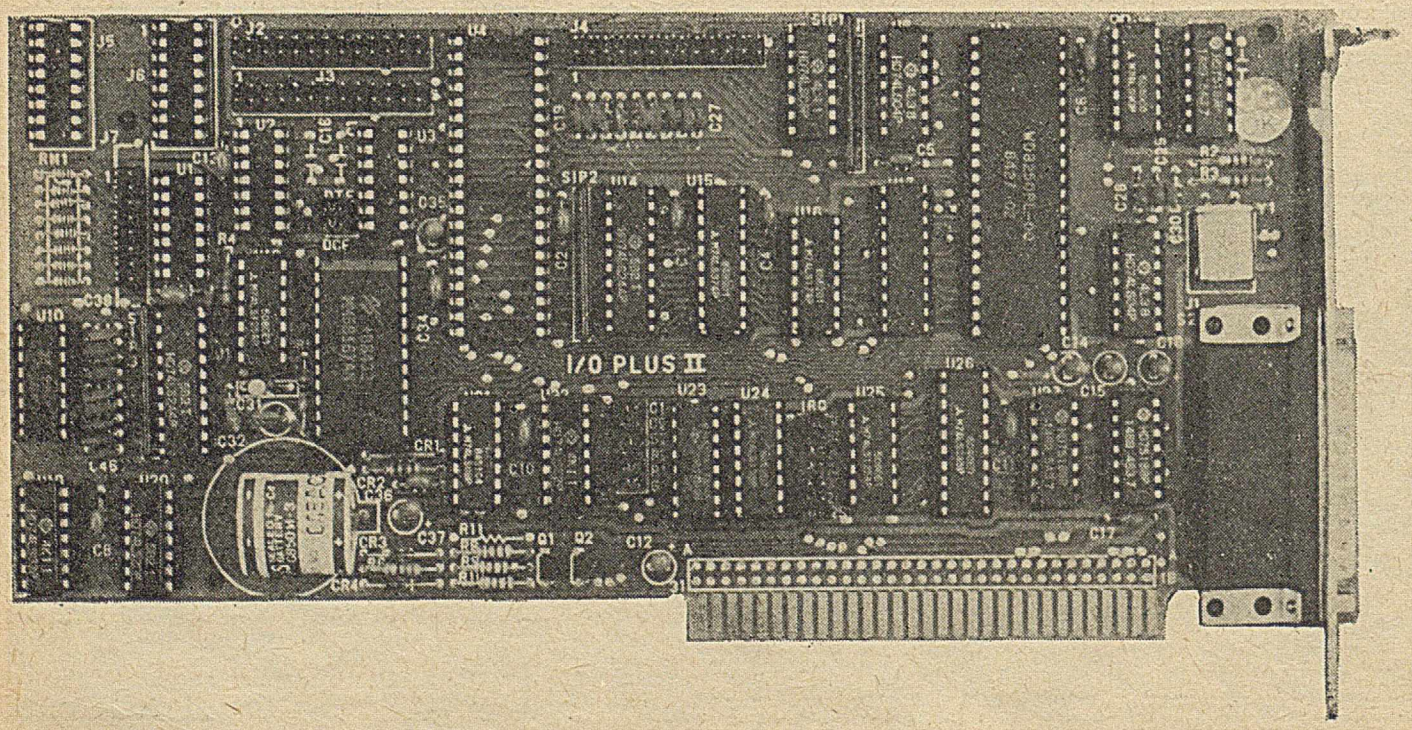
[1]) Por.: Pamięci dyskowe typu Winchester. Informatyka, nr 6, str. 26—28, 1983.



Fot. 2. Dysk Winchester ze sterownikiem (fot. PZ KAREN)

zetknięcie się głowicy z powierzchnią dysku, przy prędkości liniowej ok. 100 km/h, może spowodować zniszczenie zarówno nośnika jak i głowicy. Narzuca to oczywiste wymagania czystości powierzchni i powietrza wewnątrz hermetycznie zamkniętej obudowy dysku. Otwieranie obudowy praktycznie musi prowadzić do kompletnego zniszczenia napędu dyskowego.

W związku z dużą wrażliwością głowicy oraz nośnika, uderzenie głowicy o nośnik — nawet przy zatrzymanym dysku — może spowodować uszkodzenie nośnika. Dlatego napędy dyskowe Winchester powinny być w najwyższym stopniu chronione przed wstrząsami i uderzeniami. Przed jakimkolwiek przemieszczaniem urządzeń zawierających dyski Winchester należy, przy użyciu odpowiednich programów, pozycjonować głowicę na określone, nieużywane ścieżki, które tworzą tzw. strefę spoczynkową (ang. landing zone). Różnią się one od normalnych ścieżek pokryciem i umożliwiają stosunkowo bezpieczne przenoszenie napędów. Niestety, ścieżki te są ścieżkami wewnętrznymi (położonymi blisko środka) i mała prędkość liniowa utrudnia rozruch głowicy. Oprócz pozycjonowania należy uniemożliwić ruch głowic podczas przemieszczania, w sposób opisany w instrukcji eksploatacji napędu. Nowsze konstrukcje napędów dyskowych mają automatyczną blokadę pozycjonera po wyłączeniu napędu. Należy jednak unikać przenoszenia urządzeń zawierających dyski Winchester i przenosić je jedynie w wypadku bezwzględnej konieczności.



Fot. 3. Płyta sterownika we-wy (fot. PZ KAREN)

Ostatnio pojawiły się na rynku pamięci masowe Winchester z nośnikami wymiennymi. Ich konstrukcja różni się od niewymiennych jedynie tym, że jeden lub dwa talerze dyskowe są zamknięte w częściowo hermetycznej obudowie wyjmowanej z napędu. Niestety, wysoki koszt jeszcze nie pozwala na stosowanie tych pamięci w komputerach osobistych.

Typowe dyski Winchester są jednak jednostkami z nośnikiem niewymiennym i dlatego istnieje problem zabezpieczenia informacji zawartej na nośniku. W związku ze stopniową obniżką cen dysków Winchester rozpowszechnia się używanie drugiego dysku jako obszaru do dokonywania okresowego zapisu informacji z dysku głównego (roboczego). Oprócz tego stosowane jest przechowywanie informacji na dyskach elastycznych i na specjalnych minikasetach. Niekiedy sterowniki dysków Winchester mogą sterować równocześnie szybkimi napędami taśmowej pamięci kasetowej (tzw. tape streamer).

Zazwyczaj dostarczane jest także oprogramowanie do przesyłania informacji z dysków stałych na kasety. System operacyjny DOS 2.0 i następne jego wersje umożliwiają oznaczanie plików, które zostały zmodyfikowane od ostatniego zapisu, co pozwala na selektywny zapis informacji na nośniku zewnętrznym.

STEROWNIK TRANSMISJI ASYNCHRONICZNEJ

Sterownik transmisji asynchronicznej jest stosunko prostym urzadzeniem umożliwiającym transmisję przez układ zgodny z zaleceniem V24 i z normą RS-232. Dodatkowo niektóre wersje tego sterownika mogą pracować przez interface pętli prądowej 20 mA. Sterownik może być ustawiany programowo na szybkość 110, 150, 300, 600, 1200, 2400, 4800 lub 9600 bodów. Można także ustawiać parzystość przy nadawaniu i odbiorze (tzn. badanie parzystości, badanie nieparzystości lub brak badania). Długość nadawanego znaku może być ustawiona na 7 lub 8 bitów. Znak jest uzupełniany, oprócz opcjonalnie dodawanego bitu parzystości, także jednym bitem startowym i jednym lub dwoma bitami końcowymi (bitami stopu). Dodatkowe 2—4 bity, które towarzyszą każdemu przesyłanemu znakowi, ograniczają rzeczywistą szybkość transmisji do 36% względem szybkości nominalnej. Jest to jednak immanentna cecha transmisji asynchronicznej.

Sprzętowo sterownik transmisji jest oparty na układzie INS8250, który zapewnia zarówno konwersję równoległo-szeregową, jak i wytwarzanie wzorcowych częstotliwości nadawania i odbioru oraz wykrywanie sytuacji specjalnych. Układ sterownika może wytwarzać przerwanie w wypadkach gotowości do nadawania, gotowości do przekazania odebranego znaku, zmiany stanu linii sterowania modemem, wykrycia przez odbiornik błędu lub stanu BREAK. Oprogramowanie zawarte w systemie BIOS lub w DOS nie wykorzystuje przerwań.

W każdym komputerze IBM PC mogą być zainstalowane równocześnie dwa sterowniki asynchronicznej transmisji szeregowej. Każdy z nich może pracować w systemie jako pierwszy lub drugi. System BIOS umożliwia współistnienie trzech sterowników, jednak ten trzeci musi być innego typu lub — co najmniej — zmodyfikowany.

Zestaw linii interface'u, zastosowany w tym sterowniku, wystarcza do dołączenia typowych modemów, aczkolwiek oprogramowanie systemowe zawarte w BIOS nie zapewnia pełnej obsługi tych linii. Istnieje jednak oprogramowanie umożliwiające współpracę przez modemy z wieloma typami systemów komputerowych, przy zapewnieniu emulacji wielu typów terminali. Typowym i chyba najlepszym programem tego typu jest CROSSTALK XVI.

Obecnie istnieją i są bardzo popularne płytki wielofunkcyjne, które zazwyczaj zawierają jeden lub dwa układy transmisji szeregowej. Oczywiście zakup takiej płytki wielofunkcyjnej jest bardziej opłacalny niż zakup wielu płytek, które spełniają te same funkcje.

PŁYTA WEJŚCIA-WYJŚCIA ZE STEROWNIKIEM DYSKOWYM

Płyta ta zawiera następujące elementy składowe: sterownik dysków elastycznych 5,25 cala, interface drukarki równoległej, zegar czasu astronomicznego z podtrzymaniem bateryjnym oraz 2 układy interface'u szeregowego V24. Umożliwia współpracę z drukarkami posiadającymi interface Centronics, Logabax lub IRPR. Dobór typu interface'u jest dokonywany przez zmianę oprogramowania wbudowanego w BIOS. Standardowo oprogramowanie jest dostosowane do pracy z interfacem Centronics.

Częstotliwość pracy zegara czasu astronomicznego jest stabilizowana przez rezonator kwarcowy. Dodatkowe oprogramowanie umożliwia wstępne ustawianie zegara, przepisywanie czasu z zegara czasu astronomicznego do zegara systemowego oraz odczyt czasu. Praca zegara podczas wyłączenia komputera jest podtrzymywana przez wbudowany akumulator. Układy interface'u szeregowego umożliwiają jednoczesną współpracę z dwoma urządzeniami o układach stykowych — zgodnych z zaleceniem V24 — pełnodupleksową transmisję asynchroniczną przy szybkościach do 9600 bodów, zarówno bezpośrednio z urządzeniami, jak i przez łącze telefoniczne przy wykorzystaniu modemów.

Niektórzy producenci, np. Computex, opracowali płytki kompatybilne o rozszerzonych możliwościach sterownika dyskowego. Płytka MF-2 produkowana przez tę firmę, umożliwia współpracę z napędami dla dyskietek o gęstości zapisu 96 i 48 ścieżek na cal (ang. tracks per inch, TPI), przy czym dla napędów 96 TPI możliwe jest przełączanie gęstości zapisu przez dołączone oprogramowanie.

Inni producenci, na płytce tego typu (oprócz wymienionych poprzednio funkcji) umieszczają układ współpracy z manipulatorem drążkowym (ang. game paddle), który umożliwia cyfrowy odczyt ustawienia czterech kanałów analogowych (potencjometry 100-omowe) i czterech kanałów cyfrowych (zestyki). Pomiar w kanałach analogowych jest realizowany przez pośredni pomiar rezystancji, za pomocą układów czasowych (na płytce następuje konwersja oporności na czas). Procesor musi sam określać czas przez odczyt portu i na tej podstawie wnioskować o ustawieniu drążka każdego z dwóch manipulatorów.

Płytka ta, dzięki połączeniu wielu funkcji, jest bardzo ekonomiczna i zwiększa niezawodność systemu. Ponadto, ułatwia jego rozszerzanie, ponieważ zajmuje tylko jedno miejsce (gniazdo) płyty systemowej zamiast pięciu.

EDMUND PIERZCHAŁA
Środowiskowe Centrum Obliczeniowe
CYFRONET
Instytut Energii Atomowej
Otwock-Świerk

Interakcyjne środowisko języka Lisp (2)

Pierwsza część artykułu zawierała przegląd najważniejszych cech Lispu, decydujących o przydatności tego języka do budowy środowisk interakcyjnych. W tej części omówiono najlepiej rozwinięte i najpowszechniej używane środowisko tego języka — Interlisp.

Drugą część artykułu opracowano na podstawie [1] i [2].

Powszechnie wiadomo, że postęp w dziedzinie projektowania i produkcji oprogramowania jest znacznie wolniejszy niż postęp technologii wytwarzania sprzętu komputerowego. Mówi się od lat o kryzysie w dziedzinie produkcji oprogramowania. Jest on wywołany przede wszystkim osiągnięciem przez produkty programowe takiego poziomu złożoności, przy którym programista posługujący się dostępnymi narzędziami traci praktycznie możliwość panowania nad tymi produktami. Owa bariera złożoności towarzyszy programiście od zarania dziejów informatyki — nie bez powodu przecież wiele wysiłku włożono w latach pięćdziesiątych w opracowanie języków symbolicznych, a następnie — także w następnych dziesięcioleciach — języków wysokiego poziomu. Nie do pomyślenia było, na przykład, pisanie skomplikowanego programu obsługi bazy danych bezpośrednio w języku maszynowym. Wykonanie takiego zadania staje się realne dzięki użyciu języka wysokiego poziomu.

DROGA DO ŚRODOWISK INTERAKCYJNYCH

Wydaje się, że rozwój języków wysokiego poziomu napotkał pewien próg, którego pokonanie będzie wymagało zmiany sposobu myślenia o programowaniu jako takim.

Przykładem podejścia niekonwencjonalnego jest tzw. programowanie logiczne, zwane też programowaniem w logice, (ang. logic programming) i język Prolog, będący przedstawicielem nowej kategorii języków programowania — języków bardzo wysokiego poziomu.

Inną jeszcze próbą pokonania kryzysu software'owego są badania w dziedzinie automatycznego programowania. Ich ostatecznym celem — najogólniej rzecz biorąc — jest zbudowanie programów, które byłyby zdolne tłumaczyć nieprecyzyjne i bardzo ogólne wymagania określające systemy oprogramowania, jakie dziś zleceniodawcy przekazują do zrealizowania zawodowym programistom, na programy w istniejących językach wysokiego poziomu. Wykonanie tak postawionego zadania jest jednak bardzo trudne.

Oprócz ciągłego doskonalenia warsztatu obserwuje się znaczący postęp w dziedzinie metodologii programowania. Programowanie strukturalne umożliwia, dzięki umiejętnemu rozłożeniu problemu na możliwie niezależne problemy cząstkowe, rozwiązywanie coraz bardziej skomplikowanych zadań. Słusznie stawia się pytanie, gdzie złożoność systemów oprogramowania jest rzeczywiście usprawiedliwiona, które z nich mogłyby być zastąpione mniej skomplikowanymi ideowo odpowiednikami? Czy dobry język programowania wysokiego poziomu i jego translator muszą być skomplikowane? Czy możemy obejść się bez monstrualnych, nieprzyjaznych użytkownikowi systemów operacyjnych? Język Pascal i system operacyjny UNIX mogą być odpowiedziami na te pytania.

Tam jednak, gdzie ani prostota, ani strukturalizacja nie mają wstępu (na przykład ze względu na zbyt słabe rozpoznanie danej dziedziny, jak to jest chociażby w wypadku sztucznej inteligencji), można próbować w jeszcze inny sposób pokonywać barierę złożoności, zakładając ścisłą współpracę człowieka i komputera w interakcyjnych środowiskach programowania.

Interakcyjne środowisko programowania powinno zapewniać [1]:

1) gospodarowanie różnymi generacjami modułów programu (uzyskiwanymi w wyniku poprawiania błędów lub rozszerzania programu);

2) zarządzanie zestawami testów i ich poprawnych wyników (włączając w to efekty uboczne), tak aby odpowiednie zestawy danych testowych były wykonywane automatycznie lub półautomatycznie, gdy fragmenty programu ulegają zmianie, ewentualnie zaś niezgodności z wynikami wzorcowymi były sygnalizowane użytkownikowi;

3) gospodarowanie formalną i nieformalną dokumentacją segmentów programu i automatyczne generowanie dokumentacji na podstawie programów;

4) tłumaczenie na inne dialekty języka;

5) sprawdzanie zgodności między modułami programu:

6) tłumaczenie z języków wysokiego poziomu (specjalistycznych lub uniwersalnych) na wybrany język bazowy z uwzględnieniem sygnalizacji błędów translacji i wykonania w języku bazowym, komentarzy itp.;

7) wsparcie dla przyjętej metodologii programowania (jeśli system ma pomagać, na przykład, w programowaniu zstępującym, to naturalne jest, aby pamiętał on kolejne kroki dekompozycji i wzajemne powiązania między abstrakcyjnym krokiem a jego dekompozycją);

8) wsparcie dla sesji interakcyjnej. Na przykład pamiętanie historii sesji umożliwia użytkownikowi cofnięcie się do poprzednio wykonanych komend systemu, ich przeredagowanie i ponowne wykonanie. Anulowanie (ang. undo facility) pozwala cofnąć efekty niektórych spośród poprzednio wykonanych komend;

9) specjalizowane redagowanie programów za pomocą edytora rozumiejącego przynajmniej składnię wybranego języka, pozwalającego wobec tego odwoływać się do jego naturalnych obiektów;

10) możliwość optymalizacji programów.

Zintegrowanie środowiska interakcyjnego (a więc istnienie ścisłych powiązań między jego elementami składowymi) pozwala uniknąć dublowania niektórych czynności przez system. Jeśli, na przykład, edytor wrażliwy na składnię zapisuje redagowany program w formie odzwierciedlającej jego strukturę, nie zaś w postaci ciągu znaków, to kompilator wchodzący w skład środowiska nie musi w ogóle wykonywać analizy leksykalnej ani składniowej.

INTERLISP

Historia tego systemu sięga 1966 roku, kiedy to w firmie Bolt, Beranek and Newman Inc. rozpoczęto implementację Lispu na komputerze PDP. Obecna jego nazwa pochodzi z 1972 roku, w którym część zespołu implementatorów przeniosła się do Xerox Palo Alto Research Center i dalszy rozwój systemu stał się wspólną troską obu grup. Początkowy brak środowiska użytkowników systemu budowanego „od zera" nie zniechęcił jego twórców do eksperymentowania ze śmiałymi i nie spotykanymi wcześniej rozwiązaniami. Przyjęto dla tych eksperymentów tę samą zasadę, co dla programowania wspomaganego potem przez środowisko — stopniowego i swobodnego rozwijania progra-

mu na podstawie nabywanych doświadczeń. Ten styl programowania zadecydował o wymaganiach nałożonych na tworzące się środowisko i o jego ostatecznym kształcie. System nie powstał przez realizację raz na zawsze przyjętych założeń projektowych, lecz rozwijał się drogą ewolucji.

Interlisp zapewnia użytkownikowi dużą swobodę eksperymentowania i przystosowywania go do indywidualnych wymagań. Autorzy [2] przyznają, że to właśnie, jak też chętne i szybkie uwzględnianie sugestii użytkowników dotyczących rozszerzeń systemu, przyczyniło się do popularności Interlispu.

Pakiet obsługi plików

Interakcyjne uruchamianie programu składa się w dużej mierze z testowania jego fragmentów na przemian z poprawianiem wykrytych błędów. Struktury danych reprezentujące program są umieszczone w pamięci operacyjnej, gdzie mogą być modyfikowane za pomocą wewnętrznego edytora (tzn. pracującego na reprezentacji programu zapisanej w pamięci wewnętrznej, w odróżnieniu od edytora zewnętrznego, pracującego na reprezentacji programu zapisanej w pamięci zewnętrznej, np. dyskowej). Środowisko programowania z edytorem wewnętrznym byłoby niekompletne bez narzędzi pozwalających aktualizować kopię programu zapisaną w pamięci masowej, która służy do przechowywania jej między sesjami interakcyjnymi czy przenoszenia na inny sprzęt. Konieczna jest również możliwość wyświetlania treści programu w postaci czytelnej dla człowieka (funkcja PRINT nie troszczy się o czytelność s-wyrażeń, wyprowadzając je na całą szerokość ekranu). Funkcja PrettyDef, wyposażona w zestaw reguł umożliwiających czytelny zapis programu na pliku zewnętrznym, stała się zalążkiem pakietu gospodarowania plikami. Z biegiem czasu PrettyDef rozszerzono tak, aby mogła zapisywać zawartość list własności, tablic i innych obiektów, umożliwiając równocześnie użytkownikowi definiowanie własnych komend dla pakietu.

W kolejnym kroku ewolucji pakiet uzyskał zdolność śledzenia zawartości poszczególnych plików zawierających fragmenty uruchamianego programu, jak też zmian ich treści spowodowanych modyfikacjami istniejących i wprowadzaniem nowych definicji. Wywołania funkcji MarkAsChanged, śledzącej zmiany obiektów składających się na uruchamiany program, zostały umieszczone w takich miejscach systemu jak edytor, funkcje przeznaczone do definiowania funkcji i struktur danych (rekordów) czy DWIM (który może modyfikować definicje użytkownika, korygując błędy leksykalne — patrz niżej). Dzięki temu pakiet mógł uzyskać pewien stopień samodzielności, podczas gdy początkowo był on wyłącznie narzędziem wywoływanym jawnie przez programistę.

Bardzo ciekawym produktem jest wewnętrzny edytor Interlispu. Konwencjonalny edytor umożliwia odwołania do linii programu przez ich numery lub wzorce fragmentów tekstu. Edytor Interlispu porusza się po strukturach drzewiastych (s-wyrażeniach w pamięci operacyjnej), reprezentujących definicje funkcji, własności atomów lub wartości zmiennych. Pozwala schodzić w głąb tych struktur po różnych ścieżkach, odwoływać się do poszczególnych elementów listy i wykonywać wiele dogodnych dla programisty operacji (jak na przykład usuwanie lub wstawianie par nawiasów, grupowanie elementów, zmiana pozycji elementów), nie mających swoich bezpośrednich odpowiedników w repertuarze możliwości zwykłego edytora tekstowego.

Masterscope

Programowanie w Lispie ma charakter eksperymentalny. Częste zmiany definicji lub listy parametrów formalnych funkcji nie są niczym niezwykłym. Pojawia się naturalna potrzeba posiadania narzędzia, które w sposób interakcyjny ułatwiałoby użytkownikowi orientowanie się w strukturze jego programu. Służy do tego Masterscope. Utrzymuje on bazę danych przechowującą rezultaty przeprowadzanych przez niego analiz, dotyczących struktury programu. Posługując się tą bazą, może wskazywać, na przykład, wszystkie miejsca wywołań danej funkcji, miejsca gdzie określony atom zmienia wartość lub jest używany jako zmienna wolna. Może także współpracować z edytorem, wskazując mu miejsca dokonania pewnych poprawek, spełniające określone warunki.

Pakiet DWIM

Nie każda pomyłka w wyrażeniu napisanym przez użytkownika środowiska interakcyjnego musi powodować odrzucenie tego wyrażenia i sygnalizację błędu. Proste pomyłki literowe w nazwach funkcji czy komend, „oczywiste" pomyłki w liście argumentów (na przykład podanie atomu, tam gdzie interpreter spodziewa się listy złożonej z atomów) mogą być z powodzeniem poprawiane przez sam system, gdyż w większości tego rodzaju wypadków kontekst jednoznacznie określa to, o co chodziło programiście. Do tego celu służy DWIM (Do What I Mean). Współpracuje on z pozostałymi elementami środowiska, włączając w to pakiet obsługi plików, edytor i interpreter. Jeżeli — przykładowo — w definicji podanej przez użytkownika zostanie wykryty błąd, polegający na przykład na wywołaniu funkcji niezgodnym z jej definicją, to DWIM poprawia go i informuje o tym pakiet obsługi plików.

Asystent programisty

Jest on jakby aktywnym pośrednikiem między użytkownikiem a środowiskiem interakcyjnym. Dzięki pamiętaniu poprzednio wykonywanych komend, obliczanych s-wyrażeń, ich wyników oraz efektów ubocznych, umożliwia on programiście cofnięcie się do dowolnego etapu poprzedzającego bieżące obliczenia, ewentualnie przeredagowanie i powtórzenie poszczególnych kroków lub anulowanie niektórych spośród nich. Asystent jest dostępny nie tylko w pętli READ-EVAL-PRINT interpretera, lecz także z innych miejsc systemu (na przykład z edytora), dzięki czemu w istotny sposób zwiększa zakres możliwości środowiska. Asystent może być także wywołany z wnętrza programu użytkownika. Ta właśnie dostępność jednego narzędzia wewnątrz innych jest wyrazem zintegrowania środowiska.

UNIKALNOŚĆ INTERLISPU

Ważną i docenianą przez użytkowników cechą tego środowiska jest jego twórczy i przyjazny stosunek do użytkownika. Nie to jednak decyduje o jego unikalności. Interlisp jest przede wszystkim bardzo silnie zintegrowanym zestawem narzędzi wspomagających i uzupełniających się wzajemnie. Inną istotną cechą Interlispu jest jego modyfikowalność, czyli swoboda, z jaką użytkownik może dopasowywać go do własnych potrzeb. Dzięki temu Interlisp może wspomagać programowanie nie narzucając określonego stylu programowania, który w dużym stopniu zależy od programisty. Jest to szczególnie cenne w wypadku środowiska języka, który w znacznej mierze służy do przeprowadzania różnego rodzaju eksperymentów, a w mniejszym stopniu do programowania „produkcyjnego".

Powstanie i rozwój Interlispu przypadły na lata rozkwitu wielodostępnych systemów komputerowych. Każdy, kto pracował interakcyjnie w takim systemie orientuje się, że komfort pracy zależy od obciążenia systemu, tzn. od liczby aktywnych w danej chwili użytkowników. Konkretna cecha środowiska interakcyjnego uważana w pewnych warunkach za jego zaletę, w innych warunkach staje się wadą. Przykładem może być automatyczna korekcja leksyki dokonywana przez DWIM. W słabo obciążonym systemie o krótkim czasie odpowiedzi może ona być bardzo pomocna, natomiast w bardzo zajętym systemie staje się balastem wpływającym ujemnie nie tylko na czas odpowiedzi, lecz także na zajętość procesora centralnego, o który ubiegają się inni użytkownicy.

Problemy takie przestają istnieć w jednodostępnym środowisku komputera osobistego. Procesor nie wykonujący w danej chwili żadnych obliczeń na rzecz jedynego użytkownika, stoi bezczynnie. Nie ma więc potrzeby usprawiedliwiać stosowania narzędzi konsumujących czas procesora, o ile tylko przyczynia się to do oszczędzania czasu programisty. Można posunąć się dalej, zatrudniając bezczynny procesor przy pracy pożytecznej dla środowiska, normalnie wykonywanej na wyraźne życzenie użytkownika lub samego systemu, jak na przykład aktualizowanie bazy danych Masterscope czy kompilacja ostatnio modyfikowanych funkcji.

LITERATURA

[1] Sandewall E.: Programming in an interactive environment — The LISP experience. ACM Computing Surveys, Vol. 10, No. 1, pp. 35—37, March 1978

[2] Teitelman W., Masinter L.: The Interlisp Programming Environment. Computer, Vol. 14, No. 4, pp. 25—34, April 1981.

ZBYSZKO KRÓLIKOWSKI
Instytut Automatyki
Politechnika Poznańska

Generowanie planów wykonywania transakcji w systemach rozproszonych baz danych (3)

Nowy kierunek badań

W pierwszej części artykułu wykazano, że szczególne miejsce w problematyce związanej z przetwarzaniem transakcji w systemach rozproszonych baz danych (SRBD) zajmują metody wykonywania relacyjnej operacji połączenia. Wynika to z faktu, że koszt wykonywania tej operacji w zdecydowanym stopniu determinuje koszt wykonywania całej transakcji. W systemie rozproszonej bazy danych operacja połączenia wymaga zwykle zarówno transmisji dużych woluminów danych, jak i czasochłonnego przetwarzania.

W drugiej części artykułu dokonano przeglądu dotychczas opracowanych algorytmów generowania planów wykonywania transakcji w SRBD. Wspólną ich cechą jest bardzo uproszczony model kosztów wykonywania transakcji. W modelu tym uwzględnia się jedynie koszt transmisji danych, pomijając koszty ich przetwarzania na stanowiskach komputerowych. Przyjęcie takiego modelu kosztów jest równoznaczne z założeniem, że wąskim gardłem SRBD jest transmisja danych. W konsekwencji zdecydowana większość opracowanych dotychczas algorytmów generowania planów wykonywania transakcji koncentruje się wyłącznie na redukcji woluminów danych, przesyłanych między stanowiskami SRBD.

W tej części powyższe założenie zostanie rozpatrzone w kontekście najnowszych tendencji rozwojowych w dziedzinie sieci komputerowych oraz zostanie przedstawiony nowy kierunek badań w ramach omawianej problematyki.

Wspomniane podejście, w którym w modelu kosztów wykonywania transakcji w SRBD uwzględnia się jedynie koszt transmisji danych między stanowiskami systemu, było uzasadnione w SRBD starszego typu, realizowanych na dużych, szybkich komputerach i stosunkowo wolnych sieciach komunikacyjnych, jak ARPANET (10 kilobitów/s) [4, 15]. Wówczas bowiem mamy rzeczywiście do czynienia z silną dominacją czasów transmisji nad czasami przetwarzania. Jednakże postęp technologiczny odwrócił proporcje tych czasów. Z jednej strony obserwujemy ostatnio gwałtowny wzrost szybkości transmisji w sieciach komputerowych (wersje eksperymentalne sieci na światłowodach charakteryzują się szybkością transmisji rzędu gigabitów/s). Przyrost ten jest znacznie szybszy niż przyrost mocy obliczeniowej komputerów. Z drugiej natomiast — obserwuje się silne tendencje do stosowania małych (mini i mikro) komputerów w roli stanowisk SRBD [9]. Oznacza to relatywne zmniejszenie mocy obliczeniowej stanowisk w stosunku do możliwości transmisji sieci, a więc wzrost kosztów przetwarzania w stosunku do kosztów transmisji. Obecnie koszty przetwarzania stają się zatem co najmniej tak samo ważne jak koszty transmisji danych. Wniosek ten uzasadnia również szczegółowa analiza pomiarów przeprowadzonych w eksperymentalnych i komercyjnych SRBD [1, 2, 5, 12, 13, 14] oraz maszynach baz danych [1, 16]. Na jej podstawie, przy uwzględnieniu wspomnianych tendencji rozwojowych, można stwierdzić, że wąskim gardłem SRBD są stanowiska komputerowe, których obciążenie sięga 95%, przy obciążeniu sieci komunikacyjnych sięgającym 30%.

Przedstawione okoliczności sprawiają, że konieczne jest nowe podejście do problemu optymalizacji planów wykonywania transakcji w SRBD. Próbę takiego podejścia podjęto w badaniach nad systemami rozproszonych baz danych, prowadzonych w Instytucie Automatyki Politechniki Poznańskiej. W pierwszej kolejności należało opracować nowy model kosztów wykonywania transakcji w SRBD, który stałby się podstawą opracowania nowych algorytmów generowania planów wykonywania transakcji.

MODEL KOSZTÓW

W przyjętym modelu kosztów wykonywania transakcji [3, 8, 11] uwzględniono następujące trzy grupy kosztów:

- **transmisji** danych między stanowiskami komputerowymi SRBD,
- **przetwarzania** — koszty wykonywania operacji relacyjnych, w tym koszty dostępu do pamięci zewnętrznej,
- **systemowe** — koszty generowania i wykonywania akcji lokalnych, (akcja lokalna jest tutaj rozumiana jako sprawowanie nadzoru nad rozproszonym wykonaniem transakcji), koszty organizacji synchronizacji transakcji, koszty zabezpieczeń przed upadkiem systemu oraz koszty tworzenia tymczasowych plików danych.

Poniżej przedstawiono szczegółowe koszty wykonywania transakcji uwzględnione w modelu, w podziale na wymienione grupy.

1. Oszacowanie kosztów transmisji danych (TRANSMISSION-COST)

$$\text{TRANSMISSION-COST} = 2*\left(\left(C(R_i)*\text{width}(R_i)/PS_i\right)*A+B\right) \quad (1)$$

gdzie PS_i jest rozmiarem strony transmitowanej relacji R_i (w bajtach), A — kosztem obsługi przez stanowisko komputerowe jednej strony przesyłanej relacji, B — kosztem przygotowania sesji transmisyjnej, $C(R_i)$ — licznością relacji R_i, $\text{width}(R_i)$ — szerokością krotki relacji R_i (w bajtach).

Współczynik 2 we wzorze (1) wynika z uwzględniania faktu, że koszty obsługi transmisji są ponoszone zarówno na stanowisku wysyłającym dane, jak i na stanowisku przyjmującym dane.

2. Oszacowanie kosztów przetwarzania (PROCESSING-COST)

W ramach kosztów przetwarzania uwzględnia się:

— **koszty ładowania danych z pamięci zewnętrznej do pamięci operacyjnej komputera**, czyli koszty dostępu (ACCESS-COST), określone następująco:

$$\text{ACCESS-COST} = C(R_i)*\text{width}(R_i)*\text{COST-A} \quad (2)$$

gdzie COST-A jest kosztem dostępu do jednego bajtu relacji R_i; a $C(R_i)$ oraz $\text{width}(R_i)$ są określone jak w punkcie 1.

— **koszty wykonywania operacji relacyjnych** (RELATIONAL-OPERATION-COST). W tej grupie kosztów wyróżniamy koszty operacji połączenia oraz projekcji. Nie ma potrzeby uwzględniania w tej grupie kosztów innych operacji relacyjnych, na przykład operacji selekcji, gdyż są one stałe, niezależne od planu wykonywania transakcji;

— **koszty operacji połączenia relacji** R_i i R_k (JOIN-COST)

$$\text{JOIN-COST} = (\text{NMT}_i*\text{COST-MT}+(C(R_i)-\text{NMT}_i)*\text{COST-NMT})*(\text{width}(R_i)+\text{width}(R_k)) \quad (3)$$

gdzie COST-MT jest kosztem poniesionym przez stanowisko komputerowe dla obsługi jednego bajtu krotki, która znajdzie się w relacji wynikowej połączenia relacji R_i i R_k na atrybucie połączeniowym $R_i.A$; COST-NMT — kosztem poniesionym przez stanowisko komputerowe dla obsługi jednego bajtu krotki, która nie znajdzie się w relacji wynikowej połączenia relacji R_i i R_k na atrybucie połączeniowym $R_i.A$; NMT_i — liczbą trafionych krotek relacji R_i.

Dla operacji połączenia $R_i \overset{\bowtie}{A} R_k$[1])

$$NMT_i = C(R_i) * SF(R_k \cdot A) \quad (4)$$

W wypadku, gdy operacja połączenia jest wykonywana na kilku atrybutach połączeniowych, współczynnik selektywności SF[2]) we wzorze (4) jest iloczynem współczynników selektywności tych atrybutów.

— koszty operacji projekcji (PROJECTION-COST)

$$PROJECTION\text{-}COST = C(R_i) * width(R_i) * COST\text{-}P \quad (5)$$

gdzie COST-P jest kosztem poniesionym przez stanowisko komputerowe dla obsługi jednego bajtu relacji, na której jest wykonywana operacja projekcji. Koszty przetwarzania (PROCESSING-COST) są szacowane według następującego wzoru:

$$PROCESSING\text{-}COST = ACCESS\text{-}COST + RELATIONAL\text{-}OPERATION\text{-}COST \quad (6)$$

Uczestniczące w szacowaniu kosztów przetwarzania koszty COST-A, COST-MT, COST-NMT i COST-P są zależne od sprzętu i oprogramowania systemu zarządzania rozproszoną bazą danych (SZRBD).

3. Oszacowanie kosztów systemowych (SYSTEM-COST)

W tej grupie kosztów uwzględniamy następujące koszty składowe:

— LOCAL-ACTION-COST — suma kosztów generowania i wykonywania na stanowisku komputerowym jednej akcji lokalnej,

— SYNCHRO-COST — suma kosztów organizacji współbieżnego wykonywania transakcji, np. zakładania i zdejmowania blokad dla zapisu (odczytu) oraz obsługi zmiennych synchronizacji,

— TDF-COST — koszt utworzenia jednego tymczasowego pliku danych.

W obliczeniach kosztów wykonywania transakcji według prezentowanego modelu, w kosztach TDF-COST i SYNCHRO-COST przyjmuje się średnie wartości kosztów dla stanowisk wysyłających i przyjmujących tymczasowe pliki danych.

— INSTALL-COST — suma kosztów inicjowania wykonywania programów obsługujących funkcje systemowe,

— INTERPRETATION-COST — suma kosztów dekompozycji, optymalizacji i generowania planu wykonania transakcji oraz inicjowania wykonywania programów obsługujących operacje relacyjne.

Koszty systemowe są szacowane według następującego wzoru:

$$SYSTEM\text{-}COST = NS * (TDF\text{-}COST + LOCAL\text{-}ACTION\text{-}COST + SYNCHRO\text{-}COST + INSTALL\text{-}COST + INTERPRETATION\text{-}COST) \quad (7)$$

gdzie: NS — jest liczbą stanowisk komputerowych zaangażowanych w wykonywaniu transakcji, przy czym jeśli stanowisko bierze udział w dwóch kolejnych operacjach wykonywanych w ramach transakcji, to koszty systemowe są liczone podwójnie.

Ostatecznie koszt wykonywania transakcji w SRBD (TRANSACTION-COST) jest określany według następującego wzoru:

$$TRANSACTION\text{-}COST = TRANSMISSION\text{-}COST + PROCESSING\text{-}COST + SYSTEM\text{-}COST = (NS * (TDF\text{-}COST + LOCAL\text{-}ACTION\text{-}COST + SYNCHRO\text{-}COST + INSTALL\text{-}COST + INTERPRETATION\text{-}COST)) + (ACCESS\text{-}COST + RELATIONAL\text{-}OPERATION\text{-}COST) + TRANSMISSION\text{-}COST \quad (8)$$

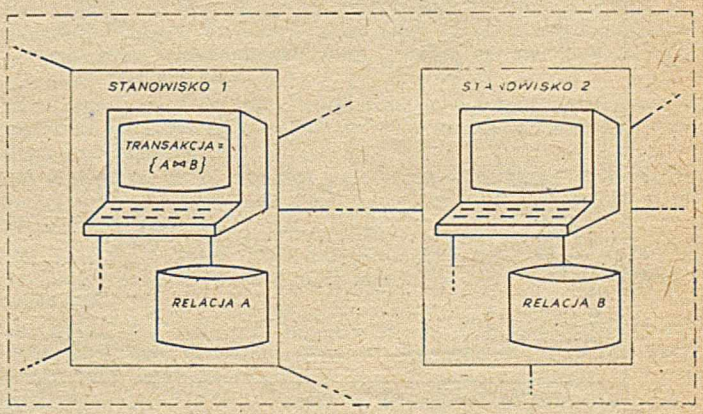
W przedstawionym modelu uwzględniono wszystkie istotne koszty wykonywania transakcji w SRBD. Do takich kosztów zaliczono koszty zależne od sprzętu i oprogramowania (SZRBD), w szczególności zależne od liczby stanowisk komputerowych, biorących udział w wykonywaniu transakcji, tj. koszty związane: z zarządzaniem współbieżnością (SYNCHRO-COST), rozproszonym wykonywaniem (LOCAL-ACTION-COST), dekompozycją i optymalizacją transakcji (INTERPRETATION-COST) oraz koszty dodatkowo zależne od rozmiarów danych, tj. koszty dostępu do pamięci zewnętrznej komputera (ACCESS-COST), koszty wykonywania operacji relacyjnych (RELATIONAL-OPERATION-COST) i koszty transmisji danych (TRANSMISSION-COST).

Wszystkie parametry występujące w modelu są możliwe do określenia. Wynikają z charakterystyk systemów komputerowych i mogą być uzyskane dzięki odpowiednim pomiarom. Koszty poszczególnych funkcji SZRBD są wyrażone liczbą instrukcji kodu maszynowego. Zastosowanie takich jednostek kosztu daje możliwość stosowania modelu dla różnych komputerów i języków programowania i ostatecznie wyrażenie kosztów w jednostkach czasu.

PORÓWNANIE STRATEGII POŁĄCZENIA I PÓŁPOŁĄCZENIA

W pierwszej części artykułu wykazano, że operacja połączenia może być wykonywana w SRBD zgodnie z jedną z dwóch strategii: jako połączenie naturalne lub jako sekwencja operacji półpołączenia. Na początku tej części artykułu zasygnalizowano zachodzącą zmianę proporcji między poszczególnymi składnikami kosztu wykonywania transakcji w SRBD. Koszty wykonywania transakcji w SRBD przestają być zdominowane przez koszty wykonywania operacji połączenia. Zmiana taka ma silny wpływ na efektywność poszczególnych strategii wykonywania operacji połączenia. Zaszła więc potrzeba porównania kosztów wykonywania transakcji stosujących jedną lub drugą strategię.

Przedstawiamy obecnie takie porównanie, w którym wykorzystano model kosztów wykonywania transakcji wyprowadzony powyżej. W analizie efektywności strategii połączenia i półpołączenia wykorzystano przykładowy zestaw wartości parametrów występujących w modelu kosztów [11], opracowany na podstawie analizy pomiarów dokonanych w eksperymentalnych i komercyjnych, scentralizowanych i rozproszonych systemach baz danych oraz na podstawie analizy charakterystyk systemów komputerowych i sieci komunikacyjnych [1, 2, 4, 5, 12, 13, 14, 15]. Analizę efektyw-

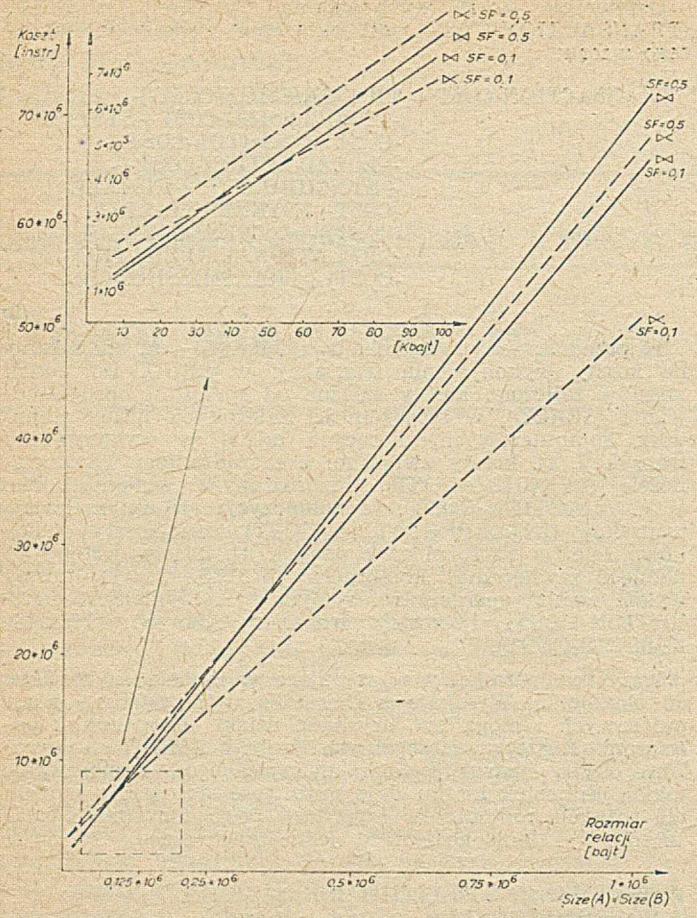


Rys. 1. Schemat SRBD wykorzystany do analizy efektywności strategii połączenia i półpołączenia

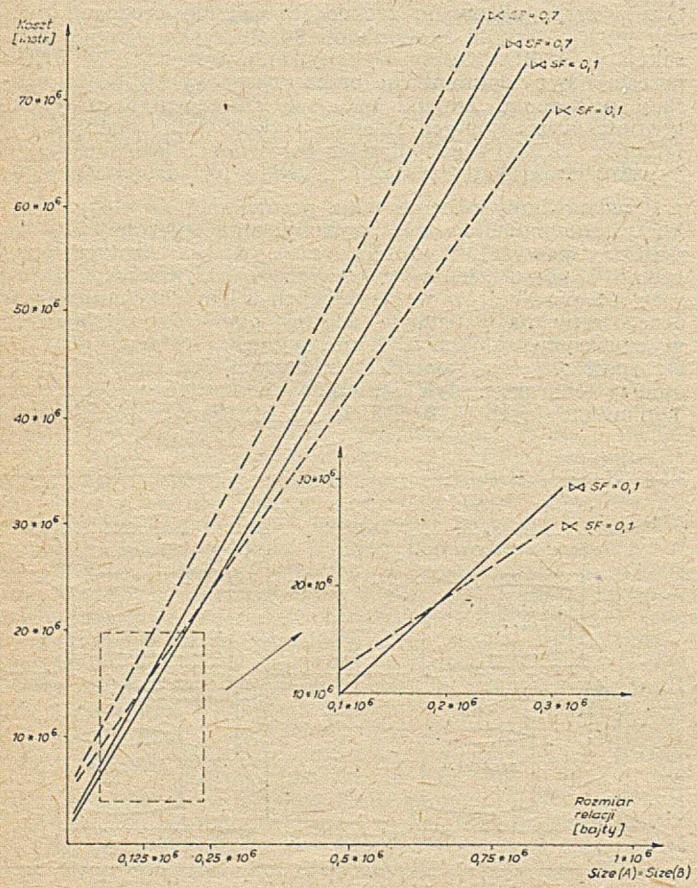
---

[1]) $R_i \overset{\bowtie}{A} R_k$ oznacza tu połączenie naturalne R_i z R_k wykonywane na stanowisku, na którym znajduje się relacja R_k; strzałka wskazuje kierunek przesłania relacji R_i; A jest atrybutem połączeniowym (przyp. red.).

[2]) Zgodnie z wyjaśnieniem autora, zamieszczonym w pierwszej części artykułu: „Współczynnik selektywności (oznaczony $SF(R_i \cdot A_j)$, gdzie A_j jest atrybutem relacji R_i) — jest stosunkiem rozmiaru atrybutu bez duplikatów do rozmiaru dziedziny tego atrybutu" (przyp. red.).



Rys. 2. Wykres zależności kosztów strategii półpołączenia i połączenia dla kryterium czasu odpowiedzi



Rys. 3. Wykres zależności kosztów strategii półpołączenia i połączenia dla kryterium sumarycznego obciążenia systemu

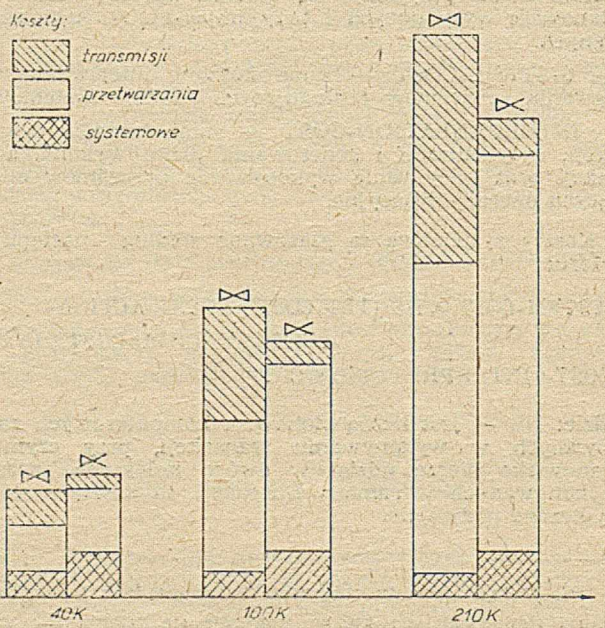
ności strategii połączenia i półpołączenia wykonano w funkcji rozmiarów relacji i współczynika selektywności atrybutów połączeniowych. Parametry te mają decydujący wpływ na efektywność tych strategii. Celem analizy była odpowiedź na pytanie, dla jakich wartości wymienionych parametrów strategia połączenia lub półpołączenia jest bardziej efektywna. Rozważano sytuację, gdy transakcja wymaga wykonania operacji połączenia dwóch relacji A i B, pamiętanych na różnych stanowiskach systemu rozproszonego (rys. 1 — relacja A na stanowisku 1 i relacja B na stanowisku 2). Taka sytuacja jest najczęściej spotykana w praktyce.

Zakładano, że schemat przetwarzania jest taki, jak w tabeli. Zgodnie z tym schematem oszacowano koszty obu strategii przy użyciu modelu prezentowanego powyżej. Obliczenia wykonano dla zmieniających się rozmiarów relacji w zakresie od 10 KB do 1 MB i wartości współczynnika selektywności SF atrybutów połączeniowych od 0,1 do 0,9. Założono, że rozmiary obu relacji i atrybutów połączeniowych oraz ich współczynnik SF są jednakowe. Analizy kosztów obu strategi dokonano dla kryterium sumarycznego obciążenia systemu oraz kryterium czasu odpowiedzi (wówczas, w przedstawionym schemacie przetwarzania, zakłada się równogłe wykonywanie operacji wyszczególnionych w punktach 1 i 3 oraz 1′ i 3′). Otrzymane wyniki przedstawiono na rys. 2 i 3.

Wnioski z porównania strategii

Jak wynika z wykresów przedstawionych na rys. 2 i 3, dla danej wartości współczynnika selektywności atrybutu połączeniowego istnieje pewien progowy rozmiar relacji, powyżej którego strategia półpołączenia jest bardziej efektywna niż strategia połączenia naturalnego. Poniżej tej wartości progowej koszty strategii półpołączenia przewyższają koszty strategii połączenia (patrz — rys. 2). Dla danych przyjętych w badaniach, przy SF = 0,1 wartością progową jest 60 Kbajtów.

Dla rosnących wartości współczynnika SF wartość progowa przesuwa się wraz ze wzrostem rozmiarów łączonych relacji. Dla wartości współczynników selektywności atrybutów połączeniowych od 0,7 do 1, strategia półpołączenia staje się nieefektywna — w rozważanej sytuacji jej koszty są zawsze wyższe niż koszty strategii połączenia. Wynika to z istnienia dużej różnicy w kosztach przetwarzania dla krotki „trafionej"[3]) i „nietrafionej" w trakcie wykonywania operacji połączenia. Koszty obsługi krotek trafionych są w rzeczywistych systemach baz danych prawie dwukrotnie wyższe niż koszty obsługi krotek nietrafionych. Wynika



Rys. 4. Procentowy rozkład kosztów wykonywania strategii półpołączenia i połączenia dla kryterium czasu odpowiedzi przy wartości SF = 0,1

[3]) Krotka „trafiona" oznacza wiersz łącznej relacji, który wystąpi w wyniku operacji połączenia (przyp. red.).

to z konieczności wykonywania, w przypadku krotek trafionych, pewnych dodatkowych operacji, w tym m.in. tworzenie dodatkowych relacji tymczasowych, wykonywanie dodatkowych operacji zapisu itp. Dla wartości SF powyżej 0,7 (tj. duża liczba trafień) zaczynają przeważać właśnie koszty obsługi krotek trafionych.

Powyższe uwagi dotyczą obu kryteriów, zarówno czasu odpowiedzi, jak i sumarycznego obciążenia systemu, przy czym w wypadku kryterium sumarycznego obciążenia systemu wartość progowa, przy danej wartości współczynnika SF, występuje dla większych rozmiarów relacji niż w wypadku kryterium czasu odpowiedzi.

Na rys. 4 przedstawiono procentowy rozkład kosztów wykonywania operacji półpołączenia i połączenia dla kilku wybranych rozmiarów łączonych relacji, przy wartości współczynnika selektywności atrybutu połączeniowego równej 0,1, a więc dla wypadku szczególnie korzystnego dla strategii półpołączenia.

Dokonana analiza porównawcza strategii połączenia i półpołączenia wykazała, że:

(i) — procentowy udział kosztów transmisji danych w kosztach wykonywania transakcji osiąga około 30% (patrz rys. 4); strategia półpołączenia redukuje koszty transmisji danych, jednakże równocześnie wywołuje wzrost kosztów przetwarzania i kosztów systemowych.

(ii) — koszty przetwarzania transakcji silnie zależą od takich parametrów jak rozmiary łączonych relacji i współczynniki selektywności ich atrybutów połączeniowych.

Generalny wniosek, nasuwający się na podstawie przeprowadzonych w [6, 7, 11] rozważań jest następujący: **dotychczasowe algorytmy generowania planów wykonywania transakcji w SRBD, stosujące operację półpołączenia, w pewnych sytuacjach mogą okazać się nieefektowne, a co więcej mogą powodować wzrost kosztów wykonywania transakcji.**

Algorytm Z

Wnioski z przedstawionej analizy były podstawą opracowania nowego algorytmu generowania planów wykonywania transakcji w SRBD, zwanego algorytmem Z [11]. Podstawową ideą tego algorytmu jest stosowanie zmiennej strategii wykonywania operacji połączenia. Algorytm Z dokonuje dekompozycji transakcji do zbioru podtransakcji, które operują na odpowiednio uporządkowanych porach relacji. Dekompozycja transakcji uwzględnia możliwość szeregowego i współbieżnego wykonywania podtransakcji. Następnie, dla każdej podtransakcji znajdowana jest efektywna strategia jej wykonywania. Ostatecznie ze zbiorów wygenerowanych podtransakcji algorytm wybiera ten zbiór, który dla danego kryterium charakteryzuje się minimalnym kosztem.

Schemat przetwarzania transakcji

Strategia półpołączenia	Strategia połączenia
1. Projekcja relacji A	1'. Lokalne przetwarzanie krotek relacji B
2. Transmisja [A] (gdzie [A] jest wynikiem operacji projekcji relacji A) na stanowisko 2	2'. Transmisja relacji B na stanowisko 1
3. Lokalne przetwarzanie krotek relacji B	3'. Lokalne przetwarzanie krotek relacji A
4. Połączenie [A] $\rhd\lhd$ B	4'. Połączenie A$\rhd\lhd$ B
5. Transmisja zredukowanej relacji B (tj. B') na stanowisko 1	
6. Połączenie [A] $\rhd\lhd$ B'	

W [10, 11] dokonano oceny efektywności zaproponowanego algorytmu. Głównym celem tej oceny było porównanie efektywności algorytmu Z z algorytmami SDD-1, AHY oraz S (wybór tych algorytmów odniesienia jest wynikiem ich szczególnego znaczenia w problematyce optymalizacji planów wykonywania transakcji w SRBD, na co zwrócono uwagę w drugiej części niniejszego artykułu).

Przeprowadzony eksperyment [11] dowiódł, że zastosowanie algorytmu ze zmienną strategią wykonywania operacji połączenia w SRBD implementowanych na szybkich sieciach lokalnych z mikrokomputerami w roli stanowisk komputerowych znacznie poprawia efektywność działania tych systemów z punktu widzenia obu kryteriów oceny ich działania.

* * *

Dotychczas uzyskane wyniki w problematyce optymalizacji planów wykonywania transakcji w SRBD wskazują na pewne nowe kierunki badań w tej dziedzinie.

Pierwszy z nich dotyczy duplikatów relacji na różnych stanowiskach komputerowych SRBD. W literaturze powszechnie akceptowane jest założenie, że w ramach transakcji rozpatruje się tylko jedną kopię każdej relacji (patrz część 1 artykułu). Celowe jest zbadanie czy — biorąc pod uwagę nowy model kosztów wykonywania transakcji — uwzględnienie faktu istnienia wielu kopii relacji w SRBD może doprowadzić do dalszej poprawy efektywności algorytmów generowania planów wykonywania transakcji.

Innym zagadnieniem wartym dalszych badań jest problem wpływu współbieżnego wykonywania transakcji w SRBD na efektywność algorytmów generowania planów ich wykonania. W przyszłości celowe wydaje się uwzględnienie w alogrytmach generowania planów wykonywania transakcji ewentualnych nierównomierności obciążenia SRBD.

Jeszcze innym kierunkiem badawczym w omawianej dziedzinie jest zastosowanie podejścia wielokryteryjnego i poszukiwanie algorytmów, zapewniających właściwy kompromis między różnymi kryteriami oceny działania systemu.

LITERATURA

[1] Bitton D., DeWitt D.J., Turbyfill C.: Benchmarking database systems a systematic approach. Computer Sciences Technical Report No. 526, University of Winconsin-Madison, 1983

[2] Bouchet P. et al.: Mesures de comportement en environment centralize et reparti sur SGBD relational-PEPIN. Rapport de recherche No. 08, ISEM, Université Paris-Sud, 1983

[3] Cellary W., Królikowski Z., Morzy T.: Comparison of semi-join and join distributed strategies using a new analytical cost model. Proc. Pacific Computer Communication Symposium. Seoul, 1985

[4] Ceri S., Pelagatti G.: Distributed Databases-Principles and Systems. McGraw-Hill Book Company, New York, 1984

[5] Gardarin G., Simon E., Berlaine L.: Querying real time relational database. Proc. ICC Conf. Amsterdam, 1984

[6] Królikowski Z.: Towards improved strategies of query processing in distributed database systems. Proc. 9th Symposium in Informatics Jahorina '85, Sarajevo, 1985

[7] Królikowski Z.: Query processing in distributed database systems. Proc. 7th International Symposium — Computer at the University, Cavtat, Zagreb, 1985

[8] Królikowski Z.: Some remarks on query processing in distributed database management systems. Proc. 8th International Seminar on DBMS, Piestany, Czechosłowacja, 1985

[9] Królikowski Z., Szymańska M.: Distribution techniques and performance relational data bases in decentralized health care environments. Proc. 6th International Congress of the European Federation for Medical Informatics, Helsinki, 1985

[10] Królikowski Z., Morzy T., Szulczyński J.: Modelling and simulation analysis of query processing in distributed database systems. Proc. International Conf. „Modelling and Simulation", Monastir, Tunezja, 1985

[11] Królikowski Z.: Optymalizacja planów wykonywania transakcji w systemach rozproszonych baz danych. Rozprawa doktorska, Politechnika Gdańska, 1985

[12] Litwin W. et al.: SIRIUS System for distributed data management. Proc. 2nd International Symposium on Distributed Data Bases, Berlin, 1982

[13] Rolin P.: Rapport sur la campaque de mesure de performance du prototype SIRIUS-DELTA. Rapport No. 175, INRIA, Rocquencourt, 1982

[14] Sarfati J., Rolin P.: Measure on the SIRIUS-DELTA distributed data base prototype. Proc. 7th International Conf. on Very Large Data Bases, Cannes, 1981

[15] Tanenbaum A. S.: Computer Networks. Prentice-Hall Inc., Englewood Cliffs (NJ), 1981

[16] Valduriez P.: Semi-join algorithms for multiprocessor systems. Pp. 225—233, Proc. SIGMOD Conf. on Management of Data, 1982.

Z początkiem roku akademickiego 1986—1987 Wydział Elektroniki Politechniki Warszawskiej obchodzi XXXV-lecie swego powstania. Korzystając z dogodnej okazji, w bieżącym numerze przedstawimy Instytut Informatyki — będący jednym z sześciu instytutów tego Wydziału (Red.).

Instytut Informatyki Politechniki Warszawskiej

Instytut Informatyki został formalnie utworzony w Politechnice Warszawskiej w 1975 r. Była to jednak tylko kolejna metamorfoza organizacyjna zespołu, który w swym zasadniczym kształcie powstał w 1951 r., a następnie — kierowany przez prof. Antoniego Kilińskiego — stanowił kolejno: Zakład Konstrukcji Telekomunikacyjnych i Radiofonii (do 1963 r.), Katedrę Budowy Maszyn Matematycznych (1963—1970), Instytutem Maszyn Matematycznych (1970—1975), wreszcie — Instytut Informatyki.

RZUT OKA WSTECZ

Prace nad urządzeniami cyfrowymi podjęto w tym zespole w latach pięćdziesiątych, rozpoczynając od serii przeliczników lampowych. Wkrótce przedmiotem zainteresowania zespołu stały się programowane maszyny cyfrowe. W latach 1957—1959 powstał prototyp uniwersalnej maszyny cyfrowej UMC-1 — jednej z kilku maszyn pierwszej generacji, konstruowanych wówczas w polskich instytucjach badawczych, lecz pierwszej, która po przekazaniu do zakładów ELWRO była produkowana w naszym kraju na — ówczesną — skalę przemysłową. Dzisiejszemu czytelnikowi warto przypomnieć, że była to maszyna lampowa, z pamięcią operacyjną bębnową o pojemności 4K słów i o szybkości 300 operacji na sekundę. Wkrótce też, w pierwszej połowie lat sześćdziesiątych opracowano kolejne konstrukcje: maszyny pierwszej generacji — AMC-1 (do przetwarzania danych administracyjnych), ANOPS-1 (do przetwarzania sygnałów biomedycznych) i GEO-1 (do obliczeń geodezyjnych) oraz drugiej generacji UMC-10 i ANOPS-10. Rzut oka na maszyny cyfrowe tych czasów, gdy informatyka nie nazywała się nawet jeszcze informatyką, na problemy techniczne, badawcze i dydaktyczne, z którymi wówczas zespół Katedry miał do czynienia, pozwala uprzytomnić sobie wielkość zmian, które były udziałem Instytutu przez następne dwadzieścia pięć lat. Niemniej, w toku tego długiego marszu przez cztery generacje maszyn cyfrowych, przy wszystkich dramatycznych zmianach, zarówno samej techniki cyfrowej, jak i społecznych oczekiwań wobec informatyki, przy towarzyszących zmianach modelu kształcenia w szkołach wyższych i zmianach personalno-organizacyjnych, można w działalności Instytutu wyróżnić pewne stosunkowo stałe tendencje, zarysowane już w początkach pracy i kształtujące obraz Instytutu w całym omawianym okresie:

1. Specjalnością Instytutu są prace badawcze i konstrukcyjne z dziedziny budowy sprzętu i oprogramowania maszyn cyfrowych, a także — projektowanie i małoseryjna produkcja specjalistycznej, profesjonalnej aparatury cyfrowej, zwłaszcza do cyfrowego przetwarzania sygnałów (m.in. w biologii i medycynie, pomiarach geofizycznych), lecz także do innych zastosowań (jak np. obliczenia geodezyjne, zbieranie i rejestracja danych).

2. Prowadzone w Instytucie prace badawcze mają w większości wypadków charakter praktyczny i są przewidziane do wdrożenia w (małoseryjnej) produkcji w Zakładzie Doświadczalnym Instytutu.

3. Zakład Doświadczalny i — od 1975 roku — Ośrodek Obliczeniowy grupują około 75% ogólnej liczby pracowników (np. w roku 1985 — łącznie 110 osób wobec 35 nauczycieli akademickich). Prace badawcze i konstrukcyjne są prowadzone w zespołach mieszanych.

4. Równolegle ze wspomnianymi pracami projektowymi prowadzone są badania o charakterze teoretycznym, dotyczące wybranych działów informatyki. Ich tematyka ewoluowała na przestrzeni lat. Najważniejsze tematy dotyczą zagadnienia testowania i diagnostyki urządzeń cyfrowych, projektowania układów i urządzeń o zwiększonej niezawodności i łatwej testowalności, teorii i praktyki mikroprogramowania, budowy narzędzi wspomagających proces projektowania urządzeń cyfrowych, modelowania systemów cyfrowych, projektowania systemów operacyjnych oraz — w ostatnich latach — techniki projektowania i tworzenia oprogramowania systemów mikrokomputerowych.

DZIAŁALNOŚĆ DYDAKTYCZNA INSTYTUTU

Nauczanie na poziomie akademickim z dziedziny informatyki prowadzi się w Instytucie od 1960 roku. Głębokie i niezliczone zmiany, jakie zachodziły zarówno w samym modelu nauczania i programie studiów oraz w treści poszczególnych przedmiotów, stanowiły odbicie starań o nadążanie za rozwojem dziedziny i zwiększaniem się społecznych oczekiwań co do kształcenia informatyków. Początkowo, do roku akademickiego 1965—1966 informatyczne przedmioty nauczania stanowiły treść specjalizacji „Maszyny Matematyczne" w ramach specjalności „Automatyka i Telemechanika" na ówczesnym Wydziale Łączności, potem — specjalności „Automatyka i Maszyny Matematyczne". Wreszcie, od roku akademickiego 1975—1976 Informatyka stała się jednym z trzech (obok Elektroniki i Telekomunikacji) kierunków studiów na Wydziale Elektroniki Politechniki Warszawskiej.

Studia na tym kierunku podejmuje co roku ok. 50 osób. Zgodnie z obowiązującym planem studiów, po pierwszych sześciu semestrach nauczania, wspólnych dla wszystkich studentów, na tym kierunku (gdzie wykłada się podstawowe przedmioty informatyczne obok ogólnych, jak matematyka i fizyka, typowo „elektronicznych" i społecznych) każdy ze studentów wybiera opiekuna naukowego, który najczęściej prowadzi go później aż do dyplomu. Pod kierunkiem opiekuna, zwykle w małym zespole, student rozpoczyna (w ramach tzw. pracowni problemowej) pracę nad pewnym wybranym problemem z zakresu konstrukcji sprzętu i oprogramowania. Równolegle uczestniczy w zajęciach z przedmiotów obieralnych, które z semestru na semestr zajmują coraz większą część planu studiów. Przedmioty obieralne umożliwiają studiowanie zgodne z predyspozycjami i zainteresowaniami studenta. Jednocześnie — ułatwiają ciągłe dostosowywanie treści programu nauczania do rozwoju dziedziny. Instytut Informatyki oferuje swym studentom corocznie ok. 35 przedmiotów obieralnych. Do ich prowadzenia zapraszani są także specjaliści spoza Politechniki.

Zespół nauczycieli akademickich Instytutu Informatyki liczy obecnie 36 osób, w tym 3 samodzielnych pracowników naukowych. Oprócz nauczania na macierzystym kierunku, nauczyciele akademiccy Instytutu prowadzą również zajęcia na innych kierunkach (Podstawy programowania na kierunku Elektronika, Elektroniczną Technikę Obliczeniową dla Studiów Wieczorowych) i uczelni (Uniwersytet Warszawski). Corocznie dyplomy magisterskie uzyskuje około 40 osób.

Od roku 1983 w Instytucie prowadzone jest Studium Podyplomowe „Systemy Mikroprocesorowe". W każdej edycji studium uczestniczy 18 słuchaczy.

Łącznie pracownicy Instytutu prowadzą prawie 10 tys. godzin zajęć dydaktycznych rocznie. Warto dodać, że na przestrzeni ostatnich 10 lat pracownicy Instytutu wydali ok. 20 książek i podręczników akademickich, ponad 40 skryptów i tłumaczeń podręczników,

nie licząc pomocniczych materiałów dydaktycznych o znaczeniu lokalnym.

Przedmiotem wielu starań Instytutu jest rozbudowa, ciągłe doskonalenie i unowocześnianie bazy laboratoryjnej zajęć dydaktycznych. Laboratoria sprzętowe są wyposażone w znacznej części w sprzęt dydaktyczny zaprojektowany i wykonany w Instytucie, w tym — 5 zestawów Modułowego Systemu Mikrokomputerowego (MSM) oraz dwa stanowiska uruchomieniowe i pulę pakietów dostosowanych do standardowych kaset produkcji POLONU, wykorzystywanych w zajęciach z zakresu projektowania i oprogramowania systemów mikrokomputerowych. Na zajęciach z przedmiotów związanych z oprogramowaniem wykorzystuje się zasoby Ośrodka Obliczeniowego: system ODRA 1305 w rozbudowanej konfiguracji (z pamięcią operacyjną 256 K słów i 16 terminalami), minikomputer SM4 (128 KB pamięci, ze stacjami taśm magnetycznych i dysków: sztywnych i elastycznych) oraz trzy instalacje mikrokomputerów kompatybilnych z IBM PC.

WAŻNIEJSZE PRACE BADAWCZE WYKONANE W OSTATNIM DZIESIĘCIOLECIU

Wspomniane, pierwsze konstrukcje maszyn cyfrowych mają swoje trwałe miejsce w tradycji i historii polskiej informatyki. Dzisiejszego czytelnika mogą jednak interesować prace, wykonane w Instytucie Informatyki w ostatnim okresie. Dlatego poniżej omówiono w skrócie ważniejsze prace badawcze, prowadzoné w latach 1976—1985.

Prace nad cyfrowym przetwarzaniem sygnałów w biologii i medycynie

Rodzina maszyn ANOPS. Kolejnymi wersjami specjalizowanych komputerów do analizy sygnałów bioelektrycznych ANOPS były modele ANOPS-101 i ANOPS-105. Produkowano je na zamówienie licznych ośrodków medyczno-badawczych i klinicznych w kraju oraz za granicą (głównie w ZSRR i CSRS, lecz pojedyncze egzemplarze sprzedano także do USA, RFN i Kanady). W kolejnych latach maszyny te były sukcesywnie wzbogacane o nowe urządzenia zewnętrzne i nowe funkcje (programy). W latach 1981—1982 opracowano nową metodę automatycznej analizy czynności bioelektronicznej mięśni (we współpracy z Kliniką Neurologiczną Akademii Medycznej w Warszawie). Badania te zostały nagrodzone w 1982 r. zespołową Nagrodą Ministra NSzWiT, za prace badawcze i wdrożeniowe wymagające szczególnego wysiłku twórczego" oraz nagrodą Ministra Handlu Zagranicznego za przyrost eksportu w 1982 r.

W latach 1981—1985 prowadzono też prace nad nową generacją maszyn ANOPS, opartą na systemie mikroprocesorowym MSM. W szczególności, opracowano projekt specjalizowanego urządzenia ANOPS-205 do analizy sygnałów elektromiograficznych i przygotowano małoseryjną produkcję tego urządzenia w Zakładzie Doświadczalnym Instytutu.

Urządzenia do nieinwazyjnych badań układu przewodzącego serca. Prace nad takim urządzeniem rozpoczęto w 1977 roku. W rok później wykonano model urządzenia KARDIO-78, który został skierowany do badań techniczno-medycznych w Szpitalu Wolskim w Warszawie. W wyniku prób opracowano ulepszony model urządzenia, którego prototyp (KARDIO-80) został wykonany w 1980 r. Od tego czasu wykonano kilka dalszych egzemplarzy KARDIO-80. Równocześnie w 1982 r. podjęto prace nad nową generacją urządzeń do nieinwazyjnych badań układu przewodzącego serca, opartych na zastosowaniu systemu MSM. Obecnie trwają prace nad algorytmami analizy przebiegów EKG oraz nad wdrożeniem urządzeń do produkcji.

Rodzina maszyn UMB. W 1979 r. podjęto pracę nad projektem urządzeń UBM-10 do przetwarzania sygnałów analogowych (zwłaszcza biomedycznych) w czasie rzeczywistym. Obecnie są prowadzone prace nad oprogramowaniem tego komputera.

Maszyny GEO do celów obliczeniowych w geodezji i kartografii

Projekt GEO-20 był największym przedsięwzięciem badawczym Instytutu Informatyki, w którym (w różnych okresach i w różnym wymiarze czasu) brało udział ok. 40 osób spośród nauczycieli akademickich i pracowników inżynieryjnych. Przedsięwzięcie to polegało na zaprojektowaniu i wykonaniu całkowicie oryginalnego systemu minikomputerowego (UMC-20) w bogatej konfiguracji sprzętowej i z bogatym oprogramowaniem, zawierającym m.in. wieloprocesorowy system operacyjny MISS, kilka wersji asemblera, translator pełnego języka FORTRAN IV oraz bibliotekę testów i programów specjalistycznych do obliczeń geodezyjno-kartograficznych.

Systemy GEO-20 wykorzystujące komputer UMC-20, wykonywane na zlecenie Zjednoczenia GEOKART, zostały zainstalowane w wytypowanych Okręgowych Przedsiębiorstwach Geodezyjno-Kartograficznych na terenie kraju. Pierwszy system GEO-20 zainstalowano w OPGK w Lublinie, w 1978 roku. Do 1983 roku wykonano i zainstalowano dalszych sześć systemów tego typu. Zespół uczestniczący w projektowaniu i wykonywaniu maszyn GEO-20 został nagrodzony w 1979 roku nagrodą wdrożeniową za wprowadzenie tych maszyn do praktycznej eksploatacji w jednostkach zjednoczenia GEOKART.

W 1983 roku rozpoczęto i znacznie zaawansowano prace nad projektem nowego urządzenia do celów obliczeń geodezyjno-kartograficznych. W roku 1984 opracowano system programowania SIGMA [1]) oraz język oprogramowania w tym systemie SIGMA-L; na podstawie tej koncepcji zaprojektowano i zbudowano model mikrokomputera GEO-3 z przygotowaniem wdrożenia do produkcji w Zakładzie Doświadczalnym Instytutu.

Cyfrowe przetwarzanie sygnałów analogowych w zastosowaniach przemysłowych

Urządzenie WEGA. W roku 1976 rozpoczęto wstępne badania nad cyfrowym urządzeniem pomiarowym WEGA, służącym do zbierania i rejestracji na nośniku magnetycznym danych generowanych w procesie poszukiwania złóż bitumitów, zgodnie z metodą opracowaną przez Instytut Górnictwa Naftowego i Gazownictwa. W roku 1980 urządzenie WEGA przeszło badania terenowe i znalazło zastosowanie w pracach poszukiwawczych. W rok później rozpoczęto prace nad nowym urządzeniem tego typu (WEGA D-02), zbudowanym w technologii mikroprocesorowej. W 1985 roku mikrokomputer ten przechodził badania terenowe.

Koncentrator KD02. W 1982 roku rozpoczęto prace nad mikroprocesorowymi koncentratorami do zbierania, przesyłania i rejestracji danych pomiarowych, przewidzianymi do zastosowania w procesie produkcyjnym Kopalni i Zakładów Przetwórczych Siarki SIARKOPOL w Tarnobrzegu.

Opierając się na zdobytych doświadczeniach, opracowano prototyp koncentratora danych oraz bufora odbiorczego o znacznie rozszerzonych możliwościach przetwarzania informacji (struktura wielomikrokomputerowa). Istotną cechą tych urządzeń jest ich przystosowanie do pracy w trudnych warunkach przemysłowych (odporność na zakłócenia, zwiększona niezawodność, autodiagnostyka itd.). Zespół prototypowy został przekazany KiZPS SIARKOPOL i obecnie jest wdrażany w jednej z kopalń. W roku 1986 planuje się wykonanie serii ośmiu koncentratorów, które zainstalowane w KiZPS będą tworzyły lokalną sieć przemysłową o strukturze gwiazdy. Przesyłanie informacji między koncentratorami oraz komputerem centralnym realizuje się za pomocą standardowych kabli telefonicznych (3—15 km). Jednocześnie trwają prace nad dalszym udoskonaleniem systemu oraz włączeniem do niego funkcji sterowania procesami technologicznymi.

Inne prace naukowe i badawcze

Do tematów o długiej tradycji należą badania nad techniką mikroprogramowania, testowaniem i diagnostyką układów cyfrowych, a także — projektowaniem układów cyfrowych z uwzględnieniem niezawodności.

Badania nad techniką mikroprogramowania koncentrowały się na matematycznych metodach opisu, projektowania i weryfikacji poprawności mikroprogramów oraz na narzędziach do projektowania i implementacji mikroprogramów. W latach 1977—1978 opracowano język MIDDLE do opisu i ana-

[1]) Por. artykuł Janusza Skolimowskiego w jednym z najbliższych numerów INFORMATYKI (przyp. red.).

lizy mikroprogramów, a po okresie sprawdzenia i doskonalenia tego narzędzia projektowego, w latach 1981—1982 wykonano symulator języka MIDDLE.

W związku z tymi pracami powstała koncepcja zbudowania urządzenia wspomagającego prace projektowe i uruchomieniowe mikroprogramowanych układów cyfrowych. Zaprojektowano i wykonano model urządzenia SUMUS, które pracuje w laboratorium dydaktycznym.

Badania nad diagnostyką i testowanie układów cyfrowych koncentrowały się wokół teorii i metod praktycznych generowania testów diagnostycznych dla asynchronicznych struktur cyfrowych oraz projektowania układów łatwotestowalnych, z naciskiem na testowanie i projektowanie układów scalonych. Rezultaty teoretyczne i opracowane metody były przedmiotem pracy habilitacyjnej, prac doktorskich oraz licznych publikacji naukowych. W roku 1981 prace zespołu zostały nagrodzone Nagrodą Sekretarza PAN. Są one obecnie kontynuowane i dotyczą zagadnień diagnostyki, jak również projektowania urządzeń i systemów odpornych na uszkodzenia.

Równolegle prowadzono badania nad projektowaniem układów elektronicznych z uwzględnieniem rozrzutu parametrów. Wyniki tych badań wykorzystano w pracy habilitacyjnej oraz w pracach doktorskich. Następnie, zespół opracował i przekazał do stosowania w CEMI metodykę testowania mikroprocesora 8080 (1978 rok) i innych układów LSI (8251, 8155A, 2716).

Spośród tematów, które jako nowe pojawiły się w omawianym dziesięcioleciu, należy wymienić przede wszystkim badania nad projektowaniem, konstrukcją i oprogramowaniem systemów mikrokomputerowych.

W szczególności, w latach 1979—1981 opracowano i wykonano Modułowy System Mikrokomputerowy (MSM), który jest od 1982 roku wytwarzany w Zakładzie Doświadczalnym Instytutu. Od chwili powstania prototypu trwa rozbudowa sprzętu i oprogramowania systemu o nowe urządzenia zewnętrzne (w szczególności — nowa klawiatura i monitor, 1983), elementy oprogramowania systemowego (systemy CP/M, ISIS, 1983) oraz urządzenia towarzyszące (np. stanowiska uruchomieniowe z monitorem szyny MSM) czy nowe funkcje (połączenia w 1984 roku z systemem ODRA-1305). W roku 1983 rozpoczęto praktyczne prace nad systemami z mikroprocesorami 16-bitowymi. System MSM stanowi podstawę dydaktycznego laboratorium mikrokomputerowego Instytutu, jak również (o czym wspomniano wcześniej) stanowi podstawę nowej generacji urządzeń projektowanych w Instytucie Informatyki do innych celów.

Do kręgu prac nad projektowaniem systemów mikrokomputerowych należy również projekt analizatora stanów logicznych ASL-80, zakończony wykonaniem modelu w 1983 roku. Równolegle, opracowano programator do programowania zawartości pamięci ROM. Kolejne wersje programatora są sukcesywnie uzupełniane o wkładki, umożliwiające programowanie nowych typów pamięci stałych [2]).

W omawianym okresie zainicjowano również i prowadzono prace nad **projektowaniem oprogramowania (w tym — systemów operacyjnych) dla małych maszyn cyfrowych.** Prace prowadzono na zlecenie Zakładu Systemów Automatyki Kompleksowej PAN w Gliwicach. Początkowo (lata 1976—1978) dotyczyły one organizacji współpracy z pamięcią dyskową, następnie (lata 1979—1980) — języka opisu systemów operacyjnych. W latach 1982—1984 wykonano oprogramowanie narzędziowe dla języka FORTH ułatwiające produkcję programów w tym języku.

[2]) Por. artykuł Marka Pawłowskiego w jednym z najbliższych numerów INFORMATYKI (przyp. red.).

W latach 1979—1980, rozpoczęto prace nad **modelowaniem wydajności systemów cyfrowych.** W szczególności sformułowano nową metodę modelowania (tzw. sieci sterowane zdarzeniami). Badania nad tą tematyką są kontynuowane.

Znaczną część wysiłku pochłaniało w omawianym okresie doskonalenie systemu ODRA-1305, będącego w wyposażeniu Ośrodka Obliczeniowego Instytutu. System ten, zainstalowany w 1979 roku podlegał nieustannej rozbudowie, zarówno sprzętowej (rozbudowa pamięci operacyjnej, pamięci dyskowej, liczby i typów terminali) jak i oprogramowania systemowego i użytkowego (uruchomienie systemu operacyjnego GEORGE-3, systemy obsługi dydaktyki — D1, a następnie D2, opracowanie, wdrożenie i eksploatacja oryginalnych elementów oprogramowania użytkowego, jak system ewidencji pracowników PW — ISEP, system ewidencji aparatury, system obsługi rekrutacji na pierwszy rok studiów, wdrożenie i eksploatacja innych systemów użytkowych, jak np. system finansowo-księgowy dla potrzeb administracji uczelni). Prace te zespół Ośrodka Obliczeniowego realizował ze znacznym, kierowniczym udziałem grupy nauczycieli akademickich. Miały one ogromne praktyczne znaczenie, zarówno dla dydaktyki i obliczeń naukowych w wielu instytutach Politechniki, jak i dla funkcjonowania administracji uczelni.

* * *

W rozpoczynającym się nowym pięcioleciu działalności Instytut Informatyki zamierza kontynuować prace tradycyjnie należące do zakresu jego specjalności, podejmując jednocześnie kilka nowych tematów badawczych, teoretycznych i praktycznych, m.in. związanych z cyfrowym przetwarzaniem sygnałów oraz zaawansowaną grafiką komputerową.

JERZY MIEŚCICKI

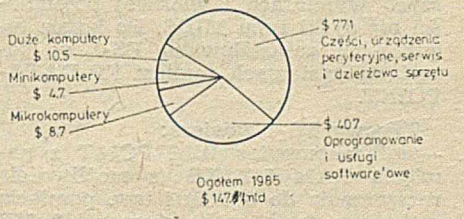
DATAPOINT i co dalej?

Kiedy na początku lat osiemdziesiątych firma DATAPOINT Corporation (DP) zajmowała pewne miejsce w światowej czołówce producentów systemów minikomputerowych, trudno było spodziewać się, że kilka lat później będzie walczyć o przetrwanie. W roku 1985 nowojorski finansista Asher B. Edelman zgromadził w swoim ręku największy pakiet akcji tej firmy, dokonując tzw. nieprzyjaznego przejęcia (ang. nostile take-over) [1]. Edelman jest znany z tego, że przejęte przedsiębiorstwa dzieli na niezależne części, po czym sprzedaje z zyskiem. Podobne działania podjął również w stosunku do DATAPOINT, z którego wydzielił najbardziej efektywną i dochodową gałąź — serwis techniczny — jako niezależne przedsiębiorstwo INTELOGIC TRACE. Pozostała część firmy działa pod starą nazwą i nowym zarządem, który podjął energiczne działania restauracyjne. Zmieniono m.in. organizację w pionie obsługi użytkowników, zarówno w USA, jak i za granicą, zreorganizowano pion badawczo-rozwojowy i dokonano redukcji personelu, głównie administracyjnego. Aktualny (w trakcie pisania artykułu) poziom akcji INTELOGIC TRACE na giełdzie nowojorskiej w pełni kompensuje Edelmanowi koszty przejęcia, a ponadto kontroluje on nowy DATAPOINT, z jego znacznymi środkami kapitałowymi w bankach oraz w ziemi i nieruchomościach. Wysokość akcji DP ustabilizowała się na poziomie jednej trzeciej ich poprzedniej wartości [2].

Przyczyny kryzysu

Przyczyny kryzysu są różnorodne. Do obiektywnych należy zmniejszenie popytu na tradycyjne systemy minikomputerowe. Mikroprocesorowa rewolucja technologiczna spowodowała przewartościowanie pojęć mikro- i minikomputer. Współczesne mikrokomputery są szybsze i mają większą moc obliczeniową od minikomputerów z lat siedemdziesiątych i zajęły ich miejsce na rynku. Według [3] spadająca stopa wzrostu zysków wielkich producentów minikomputerów w Nowej Anglii (USA) — WANG, DEC, DG, PRIME, APOLLO — (1982 — 30%, 1984 — 10%, 1985 — 4,6%) grozi odpływem kapitałów do innych, bardziej dochodowych gałęzi, a prognozy również nie są najlepsze. Rok 1985 był przełomowy nawet dla IBM, APPLE czy DEC, które zanotowały spadek obrotów i zysków w stosunku do roku poprzedniego. W czerwcu APPLE wykazało po raz pierwszy stratę i w rezultacie zmniejszono liczbę pracowników o 21% (1200 osób) oraz zamknięto trzy fabryki. Również DATA GENERAL dołączyła do grona firm poniżej kreski, zmniejszając personel o 1050 osób. Redukcje dotknęły również WANG (1600 pracowników), APOLLO (300) i COMPUTERVISION CORP. (1300). Wiele firm zamroziło lub zmniejszyło płace i inne świadczenia. Udział minikomputerów w całości rynku komputerowego USA zmalał do 3,3% [4] (rys. 1)

„Być albo nie być" przemysłu minikomputerowego zależy od nowych rozwiązań technologicznych. Stąd wyłania się dążenie do wąskiej specjalizacji, np. w obliczeniach naukowych i inżynierskich, własne serie mikrokomputerów, sieci mikro- i minikomputerowe oraz systemy wieloprocesorowe do przetwarzania dużych baz danych, ukierunkowane na przejęcie części rynku zastrzeżonego dotychczas dla dużych maszyn IBM. Kto tego nie uczynił lub zrobił zbyt późno i nie wytrzymał konkurencji — musiał za to zapłacić.



Rys. 1. Przychody w przemyśle komputerowym USA w mld dol. (wg [4])

DATAPOINT w zgodnej opinii użytkowników uchodziła za firmę ze złym zarządem, podejmującym kiepskie decyzje marketingowe, ponieważ oprogramowanie standardowe, systemy operacyjne i sieci lokalne DP są lepsze od innych dostępnych na rynku [1]. Właśnie zarząd firmy sprokurował w roku 1982 aferę, polegającą na zawyżaniu wyników sprzedaży sprzętu i usług. Zakwestionowana polityka sprzedaży znalazła natychmiastowe odbicie w spadku zysków, z 48,8 mln dolarów w roku 1981 do 2,4 mln w roku 1982 [5]. Utrata zaufania użytkowników, a przede wszystkim udziałowców to przysłowiowy „gwóźdź do trumny". Wskutek tego — pomimo reorganizacji i wprowadzenia na rynek nowych produktów — DP wykazuje w latach 1985—1986 wyraźne, choć malejące, straty i co jest bardzo symptomatyczne, dużo mniejszą sprzedaż w porównaniu z okresem 1981—1984.

Wprawdzie DATAPOINT rozpoczął intensywną akcję reklamową, przedstawiając siebie jako firmę silną, zreformowaną, prowadzącą ekspansywną politykę sprzętową i programistyczną [6], ale grono użytkowników (15 000 instalacji na całym świecie, w tym 6000 sieci) topnieje.

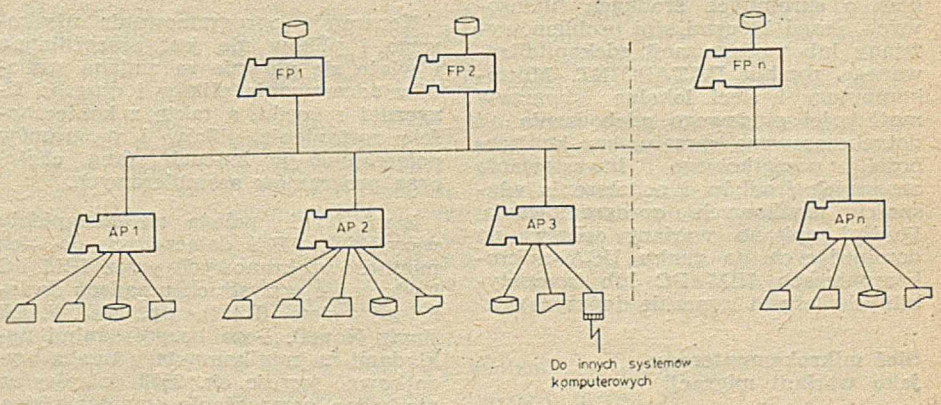
W roku 1985 firma spadła na 440 miejsce na liście FORTUNE TOP 500, daleko odstając np. od HEWLETT-PACKARDA czy WANGA.

Istotną przyczyną kryzysu jest również nieumiejętność zdyskontowania głównych atutów oprogramowania firmowego:

- języka DATABUS, który wraz z interpreterem DATASHARE tworzy doskonałe narzędzie do tworzenia systemów wielodostępnych i konwersacyjnych, wprowadzone na rynek w roku 1972 wraz z komputerem DP2200 i stale rozwijane;
- systemu operacyjnego ARC (Attached Resource Computer), umożliwiającego powiększenie mocy obliczeniowej przez tworzenie konfiguracji wieloprocesorowych; sprzedawany od roku 1977 ARC wyprzedził o wiele lat tzw. LAN (Local Area Networks) mając przy tym większe możliwości;
- systemu operacyjnego RMS (Resource Managament System), zapewniającego m.in. alokację wszystkich rodzajów zasobów w systemie dla dowolnego interaktywnego terminala (ang. workstation); idea ta stała się kilka lat później popularna i znalazła zastosowanie w sieciach z komputerami osobistymi jako końcówkami.

Aktualnie DATAPOINT oferuje wiele nowych produktów, opartych m.in. na 32-bitowym mikrokomputerze firmy Charles River:

— system zautomatyzowanego biura PRO-VISTA,
— MINX (Multimedia Information Network Exchange), system biurowy, w którym komputer osobisty jest również końcówką systemu przesyłania obrazu i dźwięku,
— STARSHIP, system integrujący konfigurację z różnymi systemami operacyjnymi, oparty na procesorze obsługi plików (ang. file server) STARSHIP1,
— nowe procesory, DP3200 z systemem operacyjnym UNOS, będącym



Rys. 2. Uproszczony schemat sieci ARC (FP — File Processor, AP — Application Processor)

wersją UNIXA, DP1200 z kolorowym monitorem i STARPORT kompatybilny z IBM PC/AT.

Pomimo starych i nowych osiągnięć, prognozy dla firmy są pesymistyczne. Obok utraty zaufania decydującym elementem jest konkurencja, przy której ceny sprzętu, oprogramowania i serwisu DP są zbyt wysokie. Trudno wróżyć powodzenie firmie oferującej monitory ekranowe w cenie tańszych mikrokomputerów kompatybilnych z IBM PC/XT czy nawet PC/AT. Niektórzy użytkownicy rozważają rezygnację z kontraktu serwisowego z INTEOGIC TRACE, na rzecz zakupu większej liczby bardzo taniego sprzętu używanego, zarówno DP jak i innych producentów, jako własnego zabezpieczenia w razie awarii.

Drogi migracji od sprzętu DATAPOINT

Systemy DATAPOINT znalazły swe zastosowanie w wielu mniejszych firmach, które nie miały środków lub też nie widziały sensu w inwestowaniu w duże komputery oraz w przedsiębiorstwach dużych (np. w bankach), gdzie współpracowały z dużym centralnym komputerem. Większość tych firm poniosła spore nakłady na oprogramowanie użytkowe i nie jest skłonna tworzyć go od nowa na innym sprzęcie. Zarysowały się więc dwie dwie drogi migracji od sprzętu DP:
— konwersja oprogramowania na inny język np. COBOL, akceptowany przez inne maszyny,
— pozostawienie oprogramowania w języku DATABUS bez zmian lub dokonanie niewielkich modyfikacji i przetwarzanie go na innym sprzęcie z odmiennym systemem operacyjnym.

Z możliwości pierwszej żyje szereg firm programistycznych w USA, stosujących zarówno automatyczne translatory, jak i tłumaczenie linia po linii. Koszty konwersji są nie do uniknięcia dla tych użytkowników, którzy zdecydowali się na sprzęt producentów nie oferujących aktualnie kompilatora DATABUS. W wypadku zainwestowania w całkowicie nowy sprzęt, np. DEC czy IBM, takie rozwiązanie jest na dłuższą metę celowe ze względu na jednorodność początkowego i późniejszego oprogramowania.

Druga droga pozostaje użytkownikom o skromnych środkach finansowych, przed którymi stoi problem wymiany lub modernizacji zdekapitalizowanego sprzętu DATAPOINT. Mikrokomputery i sieci lokalne stwarzają możliwości etapowego odchodzenia od dotychczasowego sprzętu, bez konieczności początkowego inwestowania znacznych środków i pogorszenia własności użytkowych oprogramowania. Poniżej opisano wariant oparty na dominujących na rynku USA mikrokomputerach IBM PC lub kompatybilnych (często znacznie tańszych).

Sieć mikrokomputerowa jako wariant migracji

Uproszczony schemat typowej wieloprocesorowej (ARC) konfiguracji firmy DATAPOINT przedstawiono na rys. 2.

Całkowite przejście do sieci mikrokomputerów IBM PC może być przeprowadzone w dwóch etapach:
— zastąpienie procesorów plików, (ang. File Processors — FP) z pamięciami dyskowymi
— zastąpienie procesorów użytkowych (ang. Application Processors — AP).

Procesory FP mogą być zastąpione przez tzw. procesory obsługi plików (ang. File Servers — FS), oparte na IBM PC/AT i systemie operacyjnym MS-DOS. W lipcu 1985 roku firma Network Standards rozpoczęła sprzedaż FS-1, który stał się silnym konkurentem dla FP DATAPOINT, dzięki większej szybkości i przepustowości oraz niższej cenie (także usług serwisowych). FS-1 umożliwia również dołączenie pamięci dyskowej o mniejszych wymiarach i większej pojemności niż dyski typu MIDS. Do końca tegoż roku zainstalowano FS-1 w ponad 100 systemach zamiast FP DATAPOINT.

Typowy FS-1 zawiera:
— mikrokomputer IBM PC/AT z pamięcią RAM 512 KB (z możliwością rozszerzenia do 8 MB) i tzw. pamięcią podręczną (ang. cache memory) 8 MB
— kolorowy monitor ekranowy
— urządzenie kopiujące pliki na taśmie magnetycznej 60 MB (ang. streamer tape backup)
— awaryjny zasilacz (ang. uninterrupted power supply), zapobiegający utracie danych z pamięci podręcznej (w razie przerwy w zasilaniu)
— pamięć dyskową 120 MB z opcjami 60 MB, 180 MB i 240 MB, instalowaną w obudowie mikrokomputera
— interface ARC.

Cena podstawowego zestawu (RAM 512 KB, dysk 120 MB, zegar 6,8 MHz) wraz z niezbędnym oprogramowaniem wynosi ok. 20 tys. dolarów. Ze względu na wielkość alokowanych plików w standardzie IBM PC, FS-1 może być stosowany w systemach DATAPOINT pracujących pod nadzorem systemu DOS, a takich w USA jest zdecydowana większość.

Aktualnie oferowane są dwie wersje FS-1 — starsza, kompatybilna z DOS.D (z wyjątkiem komendy DSKCHECK;F) i nowsza, oparta na komendach MS-DOS.

Konwersja oprogramowania użytkowego i plików nie jest pracochłonna i wiąże się głównie ze zmianą sposobów adresowania plików i długości fizycznej rekordu, a także z koniecznością zastąpienia plików i programów połączonych w łańcuch (ang. chain) oraz programów asemblerowych.

Etap drugi polega na zastąpieniu wszystkich AP i dołączonych do nich monitorów ekranowych przez IBM PC i włączenie wszystkich urządzeń do lokalnej sieci (LAN), np. NETWARE firmy Novell. Poza bezpośrednimi nakładami na minikomputery trzeba wydatkować jeszcze ok. 2500 dolarów na niezbędne wyposażenie sprzętowe sieci i ok. 1500 dolarów na oprogramowanie. Każda końcówka sieci NETWARE ma swój adres i może pracować jako stanowisko robocze (ang. workstation), mając dostęp do wszystkich zasobów systemu.

Istnieje również wiele wariantów pośrednich migracji od sprzętu DP, np. przez zastępowanie monitorów ekranowych DP8200 przez IBM PC i zastosowanie odpowiedniego emulatora (np. TERM82 firmy Sunbelt) lub też zastępowanie procesorów DP5500 przy użyciu pakietu INX PC.

* * *

Autor nie ma najmniejszej wątpliwości, że pozycja użytkowników firmy DATAPOINT w Polsce jest całkowicie odmienna niż w układach amerykańskich. Inne są możliwości wymiany tego sprzętu w sytuacji, gdy nawet niewielkie wydatki dewizowe stają się problemem. Również relatywnie inne są koszty serwisu technicznego, którego koszty w USA są porównywalne z cenami nowego sprzętu. Ponadto nie można ze stuprocentową pewnością założyć, że DATAPOINT zniknie całkowicie z rynku, ponieważ wariant wchłonięcia jej przez inną firmę i kontynuacja produkcji sprzętu kompatybilnego z dotychczasowym po niższych cenach, jest również możliwy.

Przedstawiona sytuacja świadczy również o tym, że rynek komputerowy nie znosi próżni i z każdego potknięcia korzystają nie tylko potentaci. Duże pieniądze zarabiają także małe firmy programistyczne, których główny kapitał stanowią „szare komórki".

Niniejszy artykuł stanowi odzwierciedlenie osobistych poglądów autora i w żadnym przypadku nie może on być interpretowany jako wykładnia polityki instytucji, z którą autor jest związany.

ANDRZEJ ELEK

LITERATURA

[1] Frieden K.: Datapoint Agony Grinds On — Users Trapped In Squeeze. MIS Weel, 27 March 1985
[2] Wall Street Journal: New York Times — New York Stock Exchange Issues. March 1986
[3] Carlyle R. E.: Year of The Multi. Datamation, No 2, 1986
[4] Goldstein M. L.: Slump Or Shakeout — U.S. Computer Industry Assesses Its Future. Industry Week, 6 January 1980
[5] Datapoint Seeks Turnaround When Takeover Artist Stalks Firm. Electronics Week, 7 January 1986
[6] Datapoint — A Strong Company With New Charter. Advertisement. Modern Office Technology, No 1, 1986

Kto jest kim w IFIP

Kaoru Ando (Japonia)

Dr Kaoru Ando został mianowany prezydentem IFIP podczas uroczystej ceremonii zamykającej Kongres IFIP we wrześniu 1983 r.

Dr Ando urodził się w Tokio. Studiował na Uniwersytecie Indiana (USA), gdzie w 1936 roku ukończył Wydział Ekonometrii. Otrzymał tytuł doktora praw w 1979 r., a Uniwersytet Indiana przyznał mu Specjalną Nagrodę dla Wybitnych Absolwentów w 1975 r.

W latach 1937—1966 był związany z japońską filią IBM, gdzie został dyrektorem ds. kształcenia, kadr i marketingu, a później — dyrektorem naczelnym. Pracował także jako doradca sił okupacyjnych w latach 1945—1950. Dr Ando opuścił IBM w 1966 roku, aby zostać prezydentem, a następnie prezesem Fujitsu FACOM Ltd. W 1970 r. został dyrektorem naczelnym Fujitsu Ltd., a następnie samodzielnym doradcą i specjalnym asystentem prezydenta, którym jest do dziś. Od 1961 r. jest przewodniczącym Instytutu Nauk Komputerowych Fujitsu.

Od połowy lat czterdziestych do połowy lat sześćdziesiątych był wykładowcą uniwersytetów w Kobe, Sofii, Keio i Konan. W czasie swojej kariery dr Ando był aktywnym członkiem wielu organizacji zawodowych, m.in. honorowym członkiem Data Processing Management Association (USA), dyrektorem wykonawczym japońskiego Towarzystwa Przetwarzania Informacji, członkiem komitetów normalizacyjnych, doradcą Agencji Naukowej Przemysłu i Technologii, Ministerstwa Handlu Międzynarodowego i Przemysłu, Komitetu Doradczego ds. Kształcenia Komputerowego, Japońskiego Towarzystwa Promocji Nauki i Ministerstwa Oświaty.

Dr Ando otrzymał wiele nagród, m.in. Order Błękitnej Szarfy od rządu japońskiego, trzykrotnie wysokie odznaczenia od różnych ministerstw oraz wysoką nagrodę z dziedziny marketingu od Japońskiego Zrzeszenia Dyrektorów. (M.K.)

KONFERENCJE

RELCOMEX'87

W dniach 22—25.09.1987 w Zamku Książ k. Wałbrzycha odbędzie się IV Międzynarodowa Konferencja Naukowo-Techniczna **„Niezawodność i Eksploatacja Systemów Komputerowych RELCOMEX'87".**

Tematyka Konferencji:

- modele matematyczne niezawodności
- niezawodność i eksploatacja systemów cyfrowych
- niezawodność i eksploatacja systemów mikrokomputerowych i mikroprocesorów
- testowanie i diagnostyka układów logicznych i systemów cyfrowych
- praktyczne aspekty sterowania niezawodnością i eksploatacją systemów cyfrowych
- niezawodność i technologia przetwarzania w systemach komputerowych.

Języki Konferencji: angielski lub rosyjski (teksty). Dopuszcza się wygłaszanie referatów w języku polskim.

Organizatorzy zapraszają do nadsyłania zgłoszeń uczestnictwa oraz streszczeń referatów do 20.12.1986, a pełnych tekstów przyjętych referatów — do 25.2.1987.

Dalsze informacje można otrzymać pod adresem: **RELCOMEX'87 — Politechnika Wrocławska, Instytut Cybernetyki Technicznej, ul. Janiszewskiego 11/17, 50-372 Wrocław, tel. 21-26-77, 20-32-04, 20-27-59, teleks 0712254 lub 0712539, pwr pl.**

Terminologia

Polskie nazewnictwo znaków kodu ASCII (ISO) — cz. I

Artykuł dotyczący możliwości przygotowywania tekstów w języku polskim na komputerze IBM PC (str. 9) i związanego z tym przyporządkowania znakom ciągów kodowych, jest dobrą okazją do omówienia polskiego nazewnictwa znaków powszechnie stosowanego kodu ASCII (mającego swój odpowiednik w normie ISO). Polską terminologię w zasadzie ustalono już w Polskiej Normie PN-78/T-42108 — „Przetwarzanie informacji i komputery. Znaki alfanumeryczne. Klasyfikacja, nazwy i symbole". Już sam tytuł normy razi jednak niekonsekwencją. W powszechnym rozumieniu, do znaków alfanumerycznych zalicza się tylko litery i cyfry. Według Polskiej Normy PN-71/T-01016 — „Przetwarzanie danych i komputery. Podstawowe nazwy i określenia". — przymiotnik „alfanumeryczny" oznacza tyle, co „obejmujący litery, cyfry dziesiętne oraz znaki specjalne", tj. inne znaki mające reprezentację graficzną.

W normie PN-78/T-42108 ustalono natomiast podział całego zbioru znaków na dwie klasy: znaki funkcyjne i znaki graficzne, tj. alfanumeryczne według normy PN-71/T-01016. Według tej normy znakami funkcyjnymi nazywa się znaki służące do inicjowania, modyfikowania lub zatrzymywania czynności zapisu, przetwarzania, transmisji lub interpretacji danych, jeżeli sterowanie tymi czynnościami jest realizowane przez dane. Określenie to podaję tylko dla porządku, ponieważ jest bardzo niejasne i nieprecyzyjne. Potocznie znaki funkcyjne nazywa się znakami sterującymi. Określiłbym je jako znaki nie wymagające reprezentacji graficznej (i rzadko mające taką reprezentację), gdyż sterują interpretacją danych. Znakami graficznymi nazywa się w tej normie „znaki przeznaczone do wizualnego przedstawiania danych". Obecnie, w związku z rozwojem technik graficznych, wygodniej byłoby nazywać tę klasę znaków znakami drukowalnymi, z podziałem na podklasę znaków tekstowych i podklasę właściwych znaków graficznych.

W klasie znaków graficznych, mających interpretację tekstową, wyróżnia się: cyfry dziesiętne, wielkie i małe litery łacińskie, wielkie i małe litery cyrylicy (rosyjskie), wielkie i małe litery polskie oraz znaki specjalne. Ze względów terminologicznych ciekawe są jedynie znaki specjalne. Większość z nich ma nazwy ustalone zgodnie z tradycją języka polskiego lub przyjęte w technice komputerowej. Należą do nich m.in. następujące znaki (dla każdego znaku podano jego reprezentację w kodzie szesnastkowym):

- **spacja** (ang. space), 20
- **wykrzyknik** (ang. exclamation mark), 21
- **cudzysłów** (ang. quotation mark), 22
- **procent** (ang. percent sign), 25
- **apostrof** (ang. apostrophe), 27

- **lewy nawias okrągły** (ang. left perenthesis), 28
- **prawy nawias okrągły** (ang. right perenthesis), 29
- **gwiazdka** (ang. asterisk), 2A
- **plus** (ang. plus sign), 2B
- **przecinek** (ang. comma), 2C
- **minus, myślnik** (ang. minus sign, hyphen), 2D
- **kropka** (ang. full stop), 2E
- **dwukropek** (ang. colon), 3A
- **średnik** (ang. semicolon), 3B
- **znak zapytania, pytajnik** (ang. question mark), 3F
- **lewy nawias kwadratowy** (ang. left square bracket), 5B
- **prawy nawias kwadratowy** (ang. right square bracket), 5D

W kilku wypadkach nazwy podane w normie warto skorygować, np.:

- **znak dzielenia, znak łamania** (ang. solidus), 2F, zamiast kreska ukośna prawa
- **znak mniejszości, lewy nawias kątowy** (ang. less-than sign), 3C, zamiast „mniejsze od"
- **znak równości** (ang. equals sign), 3D, zamiast „równe"
- **znak większości, prawy nawias kątowy** (ang. greater-then sign), 3E, zamiast „większe od"
- **lewy nawias klamrowy** (ang. left curly bracket), 7B, zamiast lewy nawias sześcienny

Tabela reprezentacji graficznej znaków specjalnych kodu ASCII

Kod	0	1	2	3	4	5	6	7
0			SP		@		`	
1			!					
2			"					
3			#					
4			¤					
5			%					
6			&					
7			'					
8			(					
9			)					
A			*	:				
B			+	;		[		{
C			,	<		\		|
D			-	=		]		}
E			.	>		^		~
F			/	?		_		

Znaki sterujące — Cyfry dziesiętne — Duże litery łacińskie — Małe litery łacińskie

- **prawy nawias klamrowy** (ang. right curly bracket), 7D zamiast nawias sześcienny prawy.

Nazwy niektórych znaków podane w normie wymagają bardziej szczegółowego wyjaśnienia, gdyż nie są to typowe znaki występujące w polskiej pisowni, a związane wyłącznie z wprowadzeniem notacji matematycznej lub wręcz techniki komputerowej. Niektórym z tych nowych znaków można nadać lepsze nazwy niż zaproponowano w normie.

Należą do nich następujące znaki:

- **znak numeru** (ang. number sign), 23
- **znak waluty**, popularne „słonko" (ang. currency sign), 24. zamiast znak ogólny jednostki monetarnej (w normie ASCII jest to znak dolara — $)
- **handlowe „i", and** (ang. ampersand), 26
- **handlowe „przy", at** (ang. commercial at), 40
- **ukośnik** (ang. reverse solidus), 5C, zamiast kreska ukośna lewa
- **grot** (ang. upward arrowhead), 5E, zamiast grot strzałki do góry (w niektórych implementacjach znakiem tym jest pełna strzałka)
- **znak podkreślenia** (ang. underline), 5F
- **znak akcentu** (ang. grave accent), 60, zamiast górna kreseczka ukośna lewa
- **belka** (ang. vertical line), 7C zamiast kreska pionowa
- **tylda** (ang. tilde), 7E przy czym w tym wypadku norma krajowa przewiduje znak górnej kreski poziomej (ang. overline), lecz w normach międzynarodowych używa się wężyka (p. tabela).

Dla handlowego „i" i „przy" proponuję używać nazw angielskich — and i at.

Niektórych znaków graficznych można używać jako tzw. znaków diakrytycznych. **Znaki diakrytyczne** są to znaki graficzne dodawane do liter, w celu oznaczenia brzmień odmiennych od brzmień odpowiadających danym literom, np. **kropka nad z** w literze **ż, kreska nad n** w literze **ń, ogonek pod e, a** w literach **ę, ą** (Słownik języka polskiego, pod redakcją M. Szymczaka, PWN, Warszawa, 1983). Często używanymi znakami diakrytycznymi są apostrof i akcent, czyli górne kreski ukośne o kodach szestnastkowych 27 i 60 (ang. accute accent, grave accent). Spośród innych znaków graficznych, **cudzysłowu** używa się jako tzw. **dierezy** (ang. diaeresis) w znakach **ä, ë** itd., **przecinka** — jako tzw. **cedilli** (ang. cedilla), czyli ogonka w znakach **ą, ę** itp., **grotu** — jako tzw. **cirkumfleksu** (ang. circumflex), np. **â, ê** itd., a **tyldy** — jako **wężyka** nad literami **n** i **o.**

Reprezentację graficzną wszystkich omówionych znaków przedstawiono w tabeli. Marzeniem informatyków jest, aby w pełny ich zbiór były wyposażone wszystkie drukarki i drukarnie. Polskie nazewnictwo drugiej klasy znaków, tzw. znaków funkcyjnych (tj. sterujących) omówimy w jednym z następnych numerów INFORMATYKI.

JANUSZ ZALEWSKI

Dokończenie artykułu ze str. 8

LITERATURA

[1] Deminet J.: Experience with multiprocessor algorithms. IEEE Trans. on Computers, Vl. C-31, April 1982

[2] Fuller S. H. et al.: Multi-microprocessors — An overview and working example. Proc. IEEE, Vol. 66, No. 2, February 1978

[3] Fuller S. H., Jones A. K., Durham I. (eds.): Cm* Review. Carnegie-Mellon University, Pittsburgh (PA). June 1977

[4] Jones A. K. et al.: Programming issues raised by a multiprocessor. Proc. IEEE, Vol. 66, No. 2, February 1978

[5] Jones A. K., Gehringer T. F.: The Cm* multiprocessor project — a research review. Carnegie-Mellon University, Pittsburgh (PA), July 1980

[6] Jones A. K., Schwarz P.: Experience using multiprocessor systems — a status report. Computing Surveys, Vol. 11, No. 2, June 1980

[7] Ousterhout J. K., Scelza D. A., Sindhu P. S.: Medusa — An experiment in distributed operating system structure. Communication of the ACM. Vol. 23, No. 2, February 1980.

Zakharov V.: Lokalna sieć komputerowa ETHERNET (1)
INFORMATYKA 1986, nr 9—10, s. 1
Pierwsza część charakterystyki obecnego stanu rozwoju sieci komputerowej ETHERNET z podkreśleniem prac normalizacyjnych.

Захаров В.: Локальная компьютерная сеть ETHERNET (1)
INFORMATYKA 1986, № 9—10, с. 1
Первая часть характеристики настоящего состояния развития компьютерной сети ETHERNET с особым учетом работ в области стандартизации.

Deminet J.: Cm* — przykład komputera wieloprocesorowego o strukturze hierarchicznej
INFORMATYKA 1986, nr 9—10, s. 4
Omówienie rozwiązań sprzętowych i programowych eksperymentalnego komputera Cm*, zbudowanego i eksploatowanego na Uniwersytecie Carnegie-Mellon w USA.

Деминет Я.: См* — Пример многопроцессорной ЭВМ с иерархической структурой
INFORMATYKA 1986, № 9—10, с. 4
Описание аппаратных и программных средств экспериментальной ЭВМ См*, построенной и используемой в Университете Carnegie-Mellon в США

Gecow A.: Litery polskie dla IBM PC
INFORMATYKA 1986, nr 9—10, s. 9
Propozycja rozwiązania problemu wyprowadzania na drukarkę znaków polskich w komputerach osobistych typu IBM PC.

Гецов А.: Польские буквы для IBM PC
INFORMATYKA 1986, № 9—10, с. 9
Предложение относительно решения вопроса введения в печатающее устройство польских букв в персональных компьютерах типа IBM PC.

Grabowicz R., Zalewski J.: System operacyjny PC-DOS (2)
INFORMATYKA 1986, nr 9—10, s. 12
Druga część charakterystyki systemu operacyjnego PC-DOS, zawierająca omówienie poleceń systemowych.

Грабович Р., Залевский Я.: Операционная система PC-DOS (2)
INFORMATYKA 1986, № 9—10, с. 12
Вторая часть характеристики операционной системы PC-DOS, содержащая описание системных команд.

Dworakowski W.: Komputery osobiste IBM PC (2)
INFORMATYKA 1986, nr 9—10, s. 15
Druga część charakterystyki komputerów osobistych typu IBM PC, zawierająca przegląd niektórych dodatkowych układów sterowników urządzeń we-wy rozszerzających możliwości funkcjonalne tych komputerów.

Двораковский В.: Персональные компьютеры IBM PC (2)
INFORMATYKA 1986, № 9—10, с. 15
Вторая часть характеристики персональных компьютеров типа IBM PC, содержащая обзор некоторых дополнительных схем управляющих устройств вводом-выводом информации, расширяющих функциональные возможности этих компьютеров.

Pierzchała E.: Interakcyjne środowisko języka Lisp (2)
INFORMATYKA 1986, nr 9—10, s. 18
Dokończenie charakterystyki języka Lisp z punktu widzenia środowiska interakcyjnego. Omówiono środowiska najlepiej rozwinięte i najpowszechniej używane.

Пешхала Э.: Интерактивная среда языка Lisp (2)
INFORMATYKA 1986, № 9—10, с. 18
Окончание характеристики языка Lisp с точки зрения интерактивной среды. Описание лучше всего развитой и наиболее широко применяемой среды.

Królikowski Z.: Generowanie planów wykonywania transakcji w systemach rozproszonych baz danych (3)
INFORMATYKA 1986, nr 9—10, s. 20
Dokończenie artykułu, zawierające omówienie najnowszych tendencji rozwoju algorytmów optymalizacji planów wykonywania transakcji oraz aktualnych kierunków prac badawczych w tej dziedzinie.

Круликовский З.: Формирование планов выполнения сделок в рассеянных системах баз данных (3)
INFORMATYKA 1986, № 9—10, с. 20
Окончание статьи, содержащей описание новейших тенденций развития алгоритмов оптимизации планов выполнения сделок, а также существующих направлений исследовательских работ в этой области.

Zakharov V.: The Ethernet — a local area network (1)
INFORMATYKA 1986, No. 9—10, p. 1
First part of characteristics of the Ethernet network actual development with emphasis on its standardization.

Zakharov V.: Das lokale Ethernet-Netz (1)
INFORMATYKA 1986, Nr. 9—10, S. 1
Erster Teil einer Charakteristik von jetzigen Entwicklungsstand des Ethernet-Netzes mit Betonung der Standardisierungsarbeit.

Deminet J.: Cm* — an example of multiprocessorcomputer with hierarchic structure
INFORMATYKA 1986, No. 9—10, p. 4
Presentation of hardware and software solutions of the Cm* experimental computer, which was build and operated at the american Carnegie-Mellon University.

Deminet J.: Cm* — ein Beispiel von Multiprozessorcomputer mit Hierarchiestruktur
INFORMATYKA 1986, Nr. 9—10, S. 4
Eine Besprechung von Hard- und Softwarelösungen des Cm-Experimentalcomputers, der an der amerikanischen Carnegie-Mellon-Universität gebaut und angewendet wird.

Gecow A.: Polish letters for IBM PC
INFORMATYKA 1986, No. 9—10, p. 9
A proposal for solution of polish national characters printing on IBM PC computers.

Gecow A.: Polnischen Buchstaben für IBM PC
INFORMATYKA 1986, Nr. 9—10, S. 9
Ein Vorschlag für das Drucken der polnischen nationalen Alphabetszeichen auf Personal Computer der IBM PC--Klasse.

Grabowicz R., Zalewski J.: PC-DOS operating system (2)
INFORMATYKA 1986, No. 9—10, p. 12
Second part of the PC-DOS operating system characteristics, which includes presentation of system commands.

Grabowicz R., Zalewski J.: Das PC-DOS-Betriebssystem (2)
INFORMATYKA 1986, Nr. 9—10, S. 12
Zweiter Teil einer Charakteristik von PC-DOS-Betriebssystem, der eine Besprechung von Systemanweisungen umfasst.

Dworakowski W.: IBM personal computers (2)
INFORMATYKA 1986, No. 9—10, p. 15
Second part of IBM personal computer characteristics, which includes a survey of some additional input/output controller circuits for extending its functional possibilities.

Dworakowski W.: IBM PC (2)
INFORMATYKA 1986, Nr. 9—10, S. 15
Zweiter Teil einer Charakteristik von den IBM PC-Klasse Personal Computer, der eine Übersicht der manchen zusätzlichen Schaltkreise der Ein- und Augabesteuerungen zur Erweiterung von funktionellen Möglichkeiten dieser Computer, umfasst.

Pierzchała E.: Interactive environment of the Lisp language (2)
INFORMATYKA 1986, No. 9—10, p. 18
Termination of the Lisp characteristics from interactive environment poit of view. The most developed and applied environments are discussed.

Pierzchała E.: Interaktive Umwelt der Lisp-Sprache (2)
INFORMATYKA 1986, Nr. 9—10, S. 18
Beendigung einer Charakteristik der Lisp-Sprache vom Standpunkt der interaktiven Umwelt. Es wurden die am besten entwickelten und am meisten angewendeten Umwelte besprochen.

Królikowski Z.: Generation of transaction plans in distributed data base systems (3)
INFORMATYKA 1986, No. 9—10, p. 20
Termination of the paper, which includes discussion of the newest development trends of algorithms for transaction plans optimalization, as well as actual directions of research work in this area.

Królikowski Z.: Generierung der Transaktionsausführungspläne in verteilten Datenbanksystemen (3)
INFORMATYKA 1986, Nr. 9—10, S. 20
Beendigung des Beitrages, die eine Besprechung von neuesten Entwicklungstrends im Bereich der Algorithmen für Planoptimierung, sowie von aktuellen Richtungen der Forschungsarbeit zu diesem Thema, umfasst.

KOŃIEC TWOICH PROBLEMÓW

Nareszcie wszystkie potrzebne Ci dane dotrą na czas
w przejrzystej formie tabel i wykresów.

Pakiet oprogramowania na mikrokomputery 16-bitowe

MEGA — BANK

to nowe MEGA możliwości jakie otwierają się przed
Twoim przedsiębiorstwem.

Wyobraź sobie

MILIARD

rekordów, które możesz
zapełnić według własnych potrzeb i uznania.

Miliard kooperantów, miliard pracowników, miliard produktów
— z tym wszystkim nasz MEGA-BANK poradzi sobie bez trudu.

System jest łatwy w obsłudze i opracowany w języku polskim.

Gwarantujemy satysfakcję!

COMPUTER STUDIO KAJKOWSCY
PROFESJONALNE OPROGRAMOWANIE MIKROKOMPUTERÓW
ul. Balladyny 3B, 81-524 Gdynia, tel.: 29-00-18, 24-01-50

EO/81/K/85