7-8 1986

informatyka

Prof. Vasilii Zakharov o komputerach osobistych

Perspektywy rozwoju układów LSI i VLSI

Komputery osobiste IBM PC

System operacyjny PC-DOS

WYDAWNICTWO NOT SIGMA

Cena 240 zł

informatyka

7-8 1986

Nr 7-8

Miesięcznik Rok XXI

Lipiec — Sierpień 1986

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

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

Zakł. Graf. „Tamka". Zam. 0619-1300/86. Obj. 7,5 ark. druk. Nakład 8300 egz. P-72.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 240 zł
Prenumerata roczna 1440 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA CZASOPISM I KSIĄŻEK TECHNICZNYCH

SIGMA

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

W NAJBLIŻSZYCH NUMERACH:

- Waldemar Dworakowski o dodatkowych sterownikach dla komputerów IBM PC
- Vasilii Zakharov o lokalnej sieci komputerowej Ethernet
- Jarosław Deminet o Cm* — wielomikroprocesorowym komputerze o strukturze hierarchicznej
- Andrzej Gecow o polskich literach dla IBM PC
- 35 lat Instytutu Informatyki Politechniki Warszawskiej

POLSKIE TOWARZYSTWO INFORMATYCZNE

Pierwsza wersja artykułu została przedstawiona podczas Szkoły Jesiennej PTI „Współczesne kierunki rozwoju informatyki" — Rydzyna, 9—13 października 1984 r.

JAN ZABRODZKI
Instytut Informatyki
Politechnika Warszawska

Stan obecny i perspektywy rozwoju układów LSI i VLSI

Począwszy od końca lat pięćdziesiątych, kiedy pojawiły się pierwsze monolityczne układy scalone, stale wzrasta liczba elementów scalanych w jednej strukturze. Stopień scalenia układu charakteryzuje się często określając liczbę bramek sieci logicznej równoważnej pod względem funkcjonalnym strukturze scalonego układu. Pierwsze układy scalone o złożoności kilku czy kilkunastu bramek określane były jako układy małej skali integracji (SSI). Układy scalone o złożoności do stu bramek przyjęto zaliczać do grupy układów średniej skali integracji (MSI). Układy dużej skali integracji (LSI) o złożoności powyżej stu bramek pojawiły się na początku lat siedemdziesiątych.

Na przełomie lat siedemdziesiątych i osiemdziesiątych, w ślad za pojawieniem się układów cyfrowych o złożoności kilku tysięcy bramek, pojawiło się określenie — układy bardzo wielkiej skali integracji (VLSI). W odniesieniu do układów analogowych, dla których stopień scalenia wzrasta wolniej niż w układach cyfrowych, nazwy układy VLSI używa się do określenia układów o złożoności funkcjonalnej porównywalnej ze złożonością 20 wzmacniaczy operacyjnych.

Zaliczenie układu do jednej z grup SSI, MSI, LSI czy VLSI ma oczywiście charakter całkowicie umowny i orientacyjny. Odzwierciedla to jednak w pewnym stopniu złożoność procesów technologicznych wykorzystywanych do wytworzenia układu. Przykładowo, układy VLSI zawierające kilkadziesiąt i więcej tysięcy tranzystorów są wykonywane w materiale półprzewodnika o powierzchni kilkudziesięciu mm², a najmniejsze wymiary poszczególnych elementów struktury są rzędu 1 μm. Osiągnięcie takiej precyzji wykonywania struktur scalonych wymagało rozwiązania wielu problemów natury czysto technologicznej. Równocześnie jednak, uzyskanie stopnia scalenia cechującego układy VLSI wymaga rozwiązania wielu innych problemów związanych z projektowaniem takich układów oraz ich testowaniem.

Obecnie produkuje się tak wiele różnych typów układów LSI i VLSI, że jakakolwiek próba ich wymieniania czy klasyfikacji nie miałaby sensu. Dla scharakteryzowania możliwości współczesnej technologii półprzewodnikowej można ograniczyć się do wymienienia kilku wybranych rozwiązań. Najbardziej spektakularne osiągnięcia wiążą się z rozwojem układów mikroprocesorowych oraz pamięci półprzewodnikowych.

Przegląd układów LSI i VLSI

W dziedzinie mikroprocesorów największy stopień scalenia osiągnięto przy produkcji układów mikroprocesorowych 32-bitowych. Opracowany w 1983 r. mikroprocesor firmy Hewlett-Packard, wykorzystywany w komputerze HP-9000, zawiera około 450 tysięcy tranzystorów na powierzchni 30,2 mm² i umożliwia pracę z częstotliwością 18 MHz. Inny 32-bitowy mikroprocesor Bellmac-32A firmy Bell Laboratories zawiera 146 tysięcy tranzystorów na powierzchni 100 mm². Dla porównania warto dodać, że 16-bitowy mikroprocesor 8086 z 1978 r. zawierał 29 tys. tranzystorów na powierzchni 30 mm², a pierwszy mikroprocesor 4004 z 1971 r. zawierał około 2300 tranzystorów (układ 8080 zawiera 4500 tranzystorów na powierzchni 15 mm²).

Jeśli chodzi o pamięci półprzewodnikowe, to laboratoryjnie wytwarza się układy dynamicznej pamięci RAM o pojemności 4 Mb; anonsowana jest pamięć DRAM CMOS o pojemności 1 Mb i czasie dostępu 65 ns; handlowo są dostępne pamięci 512 Kb i 256 Kb o czasach dostępu rzędu 90 ns. W zakresie pamięci nieulotnych dostępne są pamięci ROM o pojemności 1 Mb i czasie dostępu 350 ns, pamięci PROM 64 Kb (40 ns), pamięci EPROM 256 Kb (170 ns), pamięci E² PROM 64 Kb (200 ns). Produkuje się również statyczne pamięci RAM o pojemności 64 Kb i czasie dostępu 35 ns, 16 Kb o czasie dostępu 15 ns, 4 Kb o czasie dostępu 4 ns oraz 1 Kb o czasie dostępu 2 ns (jest to pamięć wykonana w technologii opartej na arsenku galu).

Spośród innych typów układów LSI i VLSI warto wymienić specjalizowane układy mikroprocesorowe, takie jak układ mikroprocesora analogowego 2920 firmy Intel, umożliwiający konwersję na postać cyfrową sygnałów o częstotliwości do 6,5 kHz (układ zawiera 9-bitowe przetworniki AD, DA, układy multipleksera analogowego, układ próbkująco-pamiętający, jednostkę ALU pozwalającą wykonywać operacje arytmetyczne na słowach 25-bitowych, pamięci RAM i EPROM), układ TMS-320 firmy Texas Instruments przeznaczony do przetwarzania sygnałów cyfrowych, układ PF-474 firmy Proximity Technology do przeglądu baz danych, układy do realizacji złożonych systemów mikroprogramowanych (AMD 29116), układ rozpoznawania i syntezy mowy (SP 1000, General Instruments), układy procesorów graficznych (Texas Instruments) itd.

Interesujące podejście zaprezentowała firma Nippon Telegraph, która na początku 1984 r. opublikowała dane o pamięci o pojemności użytecznej 1,5 Mb, zorganizowanej w 256 K słów 6-bitowych i przeznaczonej do pamiętania informacji o matrycy punktów 512 × 512, z których każdy niesie informację o jednym z 64 kolorów. Pamięć ta została wykonana na całej powierzchni płytki krzemu o średnicy 102 mm, takiej, na której w tradycyjnej technologii wykonuje się kilkadziesiąt lub kilkaset identycznych struktur, które po podzieleniu i testowaniu stanowią niezależne, identyczne układy VLSI. Odpowiednio duża obudowa układu ma 36 końcówek. W celu zapewnienia rozsądnego poziomu uzysku [1]) faktycznie wykonuje się pamięć o pojemności 3,7 Mb, co oznacza ponad 100% redundancję. Po pierwszym włączeniu pamięci jej uszkodzone elementy są wyszukiwane i eliminowane. Ponadto pamięć ta ma wbudowane mechanizmy kontroli parzystości i inne układy zapewniające jej niezawodną pracę.

Wymienione przykładowe układy LSI i VLSI reprezentują grupę układów scalonych dostępnych handlowo, o strukturze i właściwościach funkcjonalnych określanych przez producenta, który decyduje się na poniesienie wysokich kosztów projektu wtedy, gdy przewidywane zapotrzebowanie rzędu setek tysięcy sztuk układów gwarantuje odpowiedni zysk. Obok takich układów, technologia półprzewodnikowa umożliwia wytwarzanie układów na zamówienie — są to układy produkowane na podstawie dokumentacji, dostarczonej w tej lub innej postaci przez ich wyłącznego odbiorcę. Uproszczenie niektórych faz projektu kosztem niewykorzystania wszystkich możliwości technologicznych, istotnie obniża koszty projektu nowego układu.

Dla uproszczenia i skrócenia fazy projektowania wykorzystuje się dwa rodzaje rozwiązań. W jednym z nich dopuszcza się możliwość wykorzystywania w procesie projektowania jedynie opracowanych wcześniej bramek czy zespołów funkcjonalnych. Rozwiązanie takie skraca cykl projektowy i zmniejsza granicę opłacalności podjęcia produkcji z około 100 000 do 10 000 sztuk układów. Dalsze skrócenie czasu i zmniejszenie kosztów zapewniają sieci tranzystorowe. Przy tym podejściu adaptacja projektu użytkownika wymaga jedynie wykonania projektu masek dla końcowego połączenia gotowych struktur tranzystorów. Opłacalność tego rozwiązania jest na poziomie 1000 sztuk układów. Czas realizacji zamówień na te układy dużej skali integracji jest rzędu pojedynczych tygodni. Dla przykładu, układy firmy Fujitsu umożliwiają realizację sieci logicznych o złożoności równoważnej 8000 bramek NAND o czasach propagacji 2,5 ns.

Wybrane problemy wytwarzania układów LSI i VLSI

Układy LSI i VLSI są układami monolitycznymi, w których elementy czynne i bierne są wykonywane w materiale półprzewodnika. Materiałem podstawowym dla współczesnej technologii półprzewodnikowej jest krzem i przewiduje się, że będzie on dominował do końca lat osiemdziesiątych. Za ewentualnego następcę krzemu uważa się obecnie arsenek galu — materiał, który teoretycznie umożliwia uzyskanie większych szybkości działania układów (ruchliwość elektronów jest pięciokrotnie większa w arsenku galu niż w krzemie).

Wśród monolitycznych układów scalonych wyróżnia się układy bipolarne i układy unipolarne zależnie od tego, czy podstawowymi elementami czynnymi użytymi do budowy tych układów są tranzystory bipolarne czy też tranzystory unipolarne typu MOS. Nie wnikając w szczegóły związane z budową i zasadą działania tych elementów, można powiedzieć, że układy bipolarne charakteryzują się większą szybkością działania niż układy unipolarne, w tych ostatnich natomiast osiąga się większe gęstości upakowania. W ramach technologii bipolarnej dominują obecnie układy TTL i ECL, natomiast w technologii unipolarnej — układy nMOS przy czym coraz większego znaczenia nabierają układy CMOS i z nimi wiąże się nadzieje na dalszy wzrost stopnia scalenia. Wynika to z bardzo małego poboru mocy przez układy CMOS oraz osiągniętego w ostatnich latach znacznego wzrostu szybkości działania.

Zagadnienie odprowadzania mocy wydzielanej przez strukturę VLSI jest jednym z istotniejszych zagadnień ograniczających wzrost stopnia scalenia. Dla zasygnalizowania istniejących trudności wystarczy zauważyć, że jeśli typowa obudowa struktury scalonej może odprowadzić 1 W mocy, a struktura scalona zawiera kilkaset tysięcy elementów aktywnych, to dopuszczalna moc wydzielana przez jeden element jest rzędu pojedynczych μW. Dla porównania — moc wydzielana przez bramkę ALS TTL jest większa od 1 mW.

W celu charakteryzowania możliwości, jakie stwarza technologia pod względem dokładności wykonywania struktur scalonych, podaje się często wartość minimalnej szerokości ścieżki w strukturze bądź minimalnego odstępu między dwoma sąsiednimi ścieżkami, albo też wartość wielkości oznaczonej przez λ, która określa tolerancję, z jaką mogą być wykonywane poszczególne elementy struktury. Przy określaniu wartości λ dla danej technologii uwzględnia się zarówno tolerancje wykonania elementów dla jednej maski, jak i tolerancje związane z centrowaniem kolejnych masek względem siebie. W praktyce minimalna szerokość ścieżki jest równa około 4 λ. Obecnie przemysłowo stosowane technologie pozwalają uzyskiwać szerokość ścieżek poniżej 2 μm, natomiast laboratoryjnie osiąga się wartość poniżej 1 μm.

Przy wartościach rozdzielczości, porównywalnych z długością fali świetlnej (0,76 μm dla czerwieni), zaczynają zawodzić dotychczas powszechnie stosowane metody fotolitograficzne. Uzyskanie procesów technologicznych o większych rozdzielczościach powinny zapewnić badane obecnie metody wykorzystujące wiązki elektronów albo promieni X. Zakładając, że metody te zostaną opanowane na tyle, iż będą mogły być stosowane w praktyce produkcyjnej i że zostanie pokonana kolejna bariera technologiczna, można postawić pytanie, czy i kiedy pojawią się bariery, których już nie da się pokonać i osiągnie się kres możliwości zwiększenia gęstości upakowania. Odpowiedź na to pytanie dają prace, które na podstawie analizy zjawisk fizycznych, związanych z budową i działaniem układów cyfrowych, określają bariery teoretyczne, przynajmniej dla obecnej technologii półprzewodnikowej.

Do zmiany stanu dowolnego elementu przełączającego jest potrzebna pewna minimalna energia. Można próbować je zmniejszać przez minimalizację wymiarów elementów i zmniejszanie wartości napięć zasilających. Okazuje się jednak, że w obu wypadkach występują różnego rodzaju ograniczenia, które powodują, na przykład, że długość kanału tranzystora MOS nie może być mniejsza niż 0,4 μm, a napięcie zasilania nie powinno być mniejsze niż 0,7 V. Ograniczenia te są związane z różnego rodzaju zjawiskami fizycznymi, które zaczynają odgrywać dominującą rolę przy zbyt małych wartościach wymienionych parametrów (wpływ elementów pasożytniczych, efekt tunelowania, efekt elektromigracji atomów itd.). W rezultacie okazuje się, że po przekroczeniu pewnej granicy, zmniejszaniu wartości elementów zaczyna towarzyszyć zwiększanie wartości energii, potrzebnej dla przełączania układów.

Porównując podane wyżej przykładowe wartości graniczne z wartościami charakteryzującymi obecny stan technologii (długość kanału rzędu 1,5 μm, typowe napięcie zasilania 5 V) łatwo zauważyć, że różnice między nimi są jeszcze bardzo duże; można się zatem spodziewać, że istniejąca od ponad 20 lat tendencja do dwukrotnego zwiększenia skali scalenia co około dwa lata będzie się nadal utrzymywała.

Inną drogę do zwiększenia skali scalenia można by widzieć w zwiększaniu powierzchni, na której jest wykonywana struktura scalona. Jednak i w tym wypadku występują bardzo silne ograniczenia spowodowane względami technologicznymi. Podstawowym warunkiem uzyskania poprawnie działającej struktury scalonej jest czystość i jednorodność wyjściowego materiału półprzewodnikowego. Każde zanieczyszczenie powierzchni krzemu, w którym wykonuje się układ scalony, oznacza praktycznie jego niesprawność. Ponieważ uzyskanie materiału o stuprocentowej czystości nie jest możliwe, zawsze część wytworzonych struktur scalonych jest niesprawna. Jeżeli prawdopodobieństwo wytworzenia struktury na powierzchni wolnej od zanieczyszczeń jest odwrotnie proporcjonalne do powierzchni zajmowanej przez układ, to chcąc otrzymać uzysk różny od zera, trzeba ograniczyć powierzchnię zajmowaną przez pojedynczą strukturę. Być może pewną drogą obejścia problemu niejednorodności materiałów wyjściowych będzie rozwinięcie metody zastosowanej przy omówionej wyżej pamięci firmy Nippon Telegraph.

Projektowanie układów VLSI

Technologia półprzewodnikowa daje możliwość scalania coraz bardziej złożonych sieci logicznych. Równocześnie jednak wzrost złożoności sieci stwarza wiele nowych problemów projektowych. Obecnie wyraża się opinię, że rozwój metod projektowania jest wyraźnie opóźniony w stosunku do rozwoju technologii. Koszty i czas opracowania nowego układu VLSI są ograniczone przez fazę projektowania układu, a nie przez fazę jego wytwarzania. Stąd też prowadzi się intensywne prace nad nowymi metodami projektowania i narzędziami wspomagającymi proces projektowania. We wszystkich fazach projektowania układów VLSI powszechnie wykorzystuje się komputery. Bez ich udziału, zarówno czas wykonania projektu, jak i jego jakość wyrażona w postaci liczby błędów projektowych, byłyby nie do przyjęcia.

---

[1]) Uzysk — stosunek liczby dobrych struktur do wszystkich wytworzonych struktur.

Złożoność problemu projektowania układów VLSI niejako wymusza wyróżnienie w procesie projektowania etapów realizowanych przez różnych specjalistów. W szczególności występuje wyraźna tendencja do oddzielenia fazy projektowania na poziomie struktury funkcjonalnej od fazy wykonania układu scalonego. Uważa się, że podanie kilku podstawowych reguł czy ograniczeń wynikających z możliwości technologii i ich przestrzeganie w fazie projektowania funkcjonalnego powinno wystarczać do zapewnienia możliwości wytworzenia układu. Reguły te dotyczą najczęściej ograniczeń gęstości upakowania układów i określają minimalne wymiary elementów struktury oraz odległości między nimi, dopuszczalną powierzchnię struktury oraz liczbę poziomów masek. (Przy rosnącej złożoności struktur logicznych ograniczającym czynnikiem stają się połączenia; zajmowana przez nie powierzchnia może przekraczać powierzchnię zajmowaną przez łączone elementy. Jedną z dróg obejścia problemu jest wykonywanie połączeń na kilku warstwach, co powoduje zwiększenie liczby poziomów masek).

Taka organizacja procesu projektowania układu VLSI, w której faza projektowania struktury logicznej jest związana z technologią wytwarzania układu jedynie przez kilka reguł, ma istotną zaletę przy zmianie technologii bądź przy zmianie producenta wytwarzającego układ. Przez wprowadzenie poprawek do projektu wynikających ze zmiany reguł technologicznych można stosunkowo szybko dokonywać adaptacji gotowego projektu struktury logicznej, zwłaszcza jeżeli może to być zrealizowane automatycznie.

Pomijając problemy związane z technologią wykonania układów, warto zwrócić uwagę na skalę trudności, które wiążą się z opracowaniem projektu układu VLSI przygotowanego do wytwarzania produkcyjnego. Scaleniu podlegają sieci zawierające setki tysięcy tranzystorów, równoważne pod względem złożoności funkcjonalnej wcześniejszym, dużym komputerom czy minikomputerom. Na przykład, scalone zostały jednostki centralne maszyn PDP-11, Eclipse czy też IBM 370. Opracowanie końcowej wersji mikroprocesora Bellmac-32A trwało 14 miesięcy i wymagało 100 „osobolat" pracy projektantów, przy ogromnym udziale środków wspomagania komputerowego. Dla porównania — układ 4004 był opracowany przez jednego projektanta w ciągu 9 miesięcy.

W projektowaniu układu VLSI można wyróżnić trzy etapy: etap opracowania założeń funkcjonalnych i koncepcji układu, zajmujący około 30% czasu; etap projektowania struktury logicznej, zajmujący około 25% czasu, oraz ostatni etap — zajmujący około 45% czasu — poświęcony określeniu rozplanowania geometrii układu, a więc rozmieszczenia tranzystorów z uwzględnieniem elementów składających się na ich konstrukcję, sieci połączeń między tranzystorami, pól pod łączenie końcówek itd., z rozbiciem na poszczególne warstwy odpowiadające w późniejszym procesie kolejnym maskom.

Podane przykłady scalania struktur komputerów w pewnym sensie charakteryzują stan istniejący w dziedzinie architektury układów VLSI. Dotychczas nie pojawiły się jakościowo nowe pomysły, które pozwoliłyby odejść od podstawowych koncepcji von Neumanna. Projektanci, wzorując się na wcześniejszych rozwiązaniach, wykorzystują znane koncepcje, z których być może nie wszystkie były wcześniej stosowane praktycznie. Istotnym elementem ułatwiającym realizację niektórych koncepcji jest zmniejszenie różnicy czasów wykonywania instrukcji i czasów dostępu do pamięci dzięki scalaniu pamięci i układów przetwarzania w jednej strukturze. W układach VLSI występuje tendencja do wprowadzania równoległości działania, zarówno w sensie równoczesnego wykonywania różnych czynności przez różne, niezależnie działające bloki, jak i w sensie równoczesnego wykonywania kilku instrukcji przez kolejne, szeregowo połączone bloki. Stosuje się układowe rozwiązania systemów zarządzania pamięcią do realizacji pamięci wirtualnych i ochrony pamięci systemów wieloprocesorowych. Występuje tendencja do przejmowania przez sprzęt funkcji dawniej realizowanych programowo — ma to na celu zwiększenie szybkości wykonywania programów i ułatwienia tworzenia oprogramowania i programów użytkowych oraz zabezpieczenie przed błędami programowymi. Rozwijane są koncepcje scalania systemów operacyjnych (całych, jak np. w układzie 80150 zawierającym kompletny system CP/M-86, bądź zasadniczych fragmentów, które nie ulegają zmianie przy ewentualnych modyfikacjach czy rozbudowie systemu). Stosowane są również architektury przystosowane do realizacji określonych algorytmów (jak np. FFT, mnożenie macierzy, przeszukiwanie).

Po ustaleniu koncepcji funkcjonalnej układu i jego organizacji następuje faza określenia struktury logicznej układu. Również i na tym poziomie, podobnie jak i poprzednio, trudno jest mówić o jednolitej metodyce postępowania. Można w zasadzie wymienić jedynie kilka podstawowych tendencji. Przede wszystkim należy zwrócić uwagę na inne kryteria projektowe niż w klasycznej teorii układów logicznych. W wypadku układów VLSI istotne są takie elementy, jak minimalizacja powierzchni zajmowanej przez całą sieć łącznie z połączeniami, minimalizacja mocy wydzielanej przez układy wchodzące w skład sieci oraz szybkość pracy układu. Wychodząc z tych przesłanek przyjmuje się tezę o przewadze rozwiązań, które zapewniają regularność sieci logicznej. Stąd też m.in. preferuje się rozwiązania wykorzystujące sieci PLA. Dążąc do skracania czasu wykonywania projektów wykorzystuje się metody standardowych komórek oraz sieci tranzystorów.

Najbardziej pracochłonny jest etap wyznaczania geometrycznego rozmieszczenia struktury logicznej układu, co przy obecnym stopniu scalania nie może być wykonywane ręcznie ze względu na czas wykonania projektu, zbyt wielką liczbę danych, których człowiek nie jest w stanie zapamiętać i kontrolować, oraz wiążący się z tym problem błędów, których przy ręcznym projektowaniu nie można uniknąć.

Wśród metod komputerowego wspomagania procesu projektowania układów VLSI można wyróżnić dwie grupy: metody analizy układu i metody syntezy. Metody analizy są związane głównie z problemami symulacji i weryfikacji projektu na wszystkich jego etapach. Metody syntezy obejmują zarówno te, które wspomagają projektanta pracującego interakcyjnie z systemem komputerowym, jak i te, które zastępują projektanta przy realizacji określonych zadań. Ostatnio szczególnie dużo uwagi poświęca się tej ostatniej grupie metod. Powszechnie stosuje się programy automatycznego rozmieszczania elementów, prowadzenia połączeń czy też ścieśniania. Prowadzone są prace nad metodami automatycznej syntezy sieci PLA, pozwalającymi na przejście od równań boolowskich do planu geometrycznego układu. Realizowane są koncepcje automatyzacji procesu generowania planu geometrycznego na podstawie szkicu topologii układu przedstawianej w postaci symbolicznej. Prowadzone są próby z tzw. kompilatorami krzemu, które na podstawie opisu właściwości funkcjonalnych w języku wysokiego poziomu, wytwarzają plan geometryczny układu. Rozważane są również możliwości wykorzystania koncepcji sztucznej inteligencji, rozwijanych ostatnio w ramach systemów ekspertowych wykorzystujących bazy wiedzy.

Testowanie układów VLSI

Jedna z najpoważniejszych trudności pojawiających się wraz ze wzrostem stopnia scalenia wiąże się z zagadnieniem testowania wytworzonych struktur i układów. Przy scaleniu kilkudziesięciu tysięcy elementów w pojedynczej strukturze i dostępie jedynie do kilkudziesięciu końcówek wejściowych i wyjściowych, tradycyjne metody testowania stają się nieefektywne. Różne próby szacowania czasu pełnego testowania układu VLSI dają wyniki na poziomie setek czy milionów lat. W związku z tym poszukuje się nowych rozwiązań tego problemu. Najprostsze metody sprowadzają się do ograniczenia zakresu testowania, na przykład, tylko do sprawdzenia podstawowych właściwości funkcjonalnych, ewentualnie w połączeniu z badaniem właściwości pomocniczych wbudowanych struktur próbnych. Jednak w większości wypadków rozwiązanie takie nie jest zadowalające.

Obecnie uważa się, że problem testowania można rozwiązać jedynie przez włączenie go do zadań realizowanych w czasie projektowania struktury funkcjonalnej i logicznej. Realizuje się to przez przyjęcie takich rozwiązań, które ułatwią testowanie; na przykład, przyjmując sterowanie synchroniczne można uniknąć konieczności sprawdzania, czy w sieci nie występują hazardy. Można też zadanie testowania rozbić na wiele prostszych podzadań, wydzielając w sieci logicznej mniejsze bloki niezależne z punktu widzenia testowania.

W wielu wypadkach dopuszcza się rozbudowę sieci logicznej o pomocnicze układy przeznaczone wyłącznie do celów testowania. Stopień rozbudowy może sięgać nawet

50% pierwotnej sieci, najczęściej jednak przyjmuje się, że rozsądny kompromis między zyskami płynącymi z uproszczenia testowania a stratami wynikającymi z pogorszenia uzysku na skutek zwiększenia powierzchni krzemu, jest na poziomie 10—20%.

Wprowadzenie pomocniczych układów umożliwia realizację różnych metod testowania. Wykonanie dodatkowej pamięci ROM umożliwia przechowywanie programów testowania wewnętrznych podzespołów. Dodatkowe układy umożliwiają stosowanie metod analizy sygnatur w stosunku do wewnętrznych podzespołów. W niektórych metodach dąży się do zredukowania problemu do testowania sieci kombinacyjnych. W tym celu tak organizuje się strukturę logiczną, by móc w czasie testowania zapisać stan wszystkich elementów pamiętających struktury do specjalnego rejestru szeregowego, a następnie zawartość tego rejestru wyprowadzić na zewnątrz. Znana jest koncepcja, która zakłada, że wszystkie elementy pamiętające struktury logicznej są dostępne dla celów testowania, tak jak komórki pamięci RAM.

Rozwinięcie koncepcji stosowania dodatkowych układów do testowania prowadzi do realizacji idei samotestowania. Z zagadnieniem tym wiążą się z kolei problemy związane z możliwościami restrukturalizacji układów bądź to bezpośrednio po wytworzeniu układu, bądź też w trakcie jego eksploatacji po wystąpieniu niesprawności pewnych elementów struktury.

* * *

Większość problemów poruszonych w artykule została jedynie zasygnalizowana — jednak powinno to dać pewien obraz złożoności zagadnień związanych z produkcją współczesnych układów scalonych, a także ułatwić określenie kierunku prac, które są i będą rozwijane w ślad za ciągłym postępem technologii półprzewodnikowej.

Ograniczając do minimum zagadnienia technologiczne, zwrócono uwagę na te problemy, które mogą interesować informatyków bądź jako użytkowników współczesnego sprzętu budowanego z wykorzystaniem układów LSI i VLSI, bądź też jako współuczestników procesu projektowania takich układów. Rozwój procesów wytwarzania, metody projektowania i testowania nie może odbywać się bez udziału komputerów. O ile aspekty związane z technologią wytwarzania układów VLSI leżą poza obszarem działalności informatyków, to pozostałe fazy procesu wytwarzania takich układów nie mogą rozwijać się bez udziału specjalistów z dziedziny organizacji systemów cyfrowych, projektowania sieci logicznych, testowania, jak i programistów specjalizujących się w systemach operacyjnych, językach programowania, konstrukcji narzędzi wspomagających projektanta itp. Zapotrzebowanie na specjalistów z tych dziedzin szybko wzrasta, co jest wynikiem m.in. tego, iż postęp technologii wyprzedza rozwój teorii i metod projektowania układów VLSI — metod, które pozwoliłyby pokonać barierę kosztów i czasu opracowywania nowych projektów. Przełamanie tej bariery może oznaczać przejście z obecnego etapu rozwoju określanego jako rewolucja mikroprocesorowa, do etapu rewolucji na poziomie systemów komputerowych.

Analizując problemy rozwoju układów VLSI nie można pominąć zagadnień związanych z wykorzystaniem tych układów. Modyfikacje układów muszą powodować zmiany w konstrukcji urządzeń, a równocześnie problemy wynikające w trakcie projektowania i eksploatacji urządzeń stanowią istotny bodziec do dalszego rozwoju układów.

Stały wzrost stopnia scalania podstawowych układów stwarza z jednej strony możliwość realizacji systemów o coraz większej złożoności i coraz lepszych parametrach, a z drugiej strony — konstrukcję tanich, przenośnych urządzeń zarówno uniwersalnych, jak np. komputery osobiste, jak i specjalizowanych, wykorzystywanych np. w samochodach. Jest to wynikiem zmniejszania ceny pojedynczego układu zastępującego wiele układów o mniejszym stopniu scalenia, przy równoczesnym zmniejszeniu wymiarów i poboru mocy, zwiększeniu szybkości i niezawodności.

Możliwość korzystania z układów LSI i VLSI stawia konstruktorów sprzętu cyfrowego przed koniecznością zmiany stylu projektowania: projektowanie na poziomie logiki zostaje coraz częściej zastępowane projektowaniem na poziomie systemowym — problem łączenia bramek ewoluuje w kierunku problemu łączenia mikroprocesorów.

LITERATURA

[1] Ahdoot K. et al.: IBM FSD VLSI chip design methodology. 20th Design Automation Conference Proceedings. pp. 39—45, June 1983

[2] Avenier J. P.: Digitizing, layout, rule checking — the everyday tasks of chip designers. Proc. of the IEEE, No. 1, pp. 49—56, 1983

[3] Beyers J. W., Zeller E. R., Seccombe S. D.: VLSI technology packs 32-bit computer system into a small package. Hewlett-Packard Journal, No. 8, pp. 3—6, 1983

[4] Burkhart K. P. et al.: An 18-MHz, 32-bit VLSI microprocessor. Hewlett-Packard Journal, No. 8, pp. 7—11, 1983

[5] Bursky D.: Innovative chip design leads to dense, superfast RAM's. Electronic Design, No. 17, pp. 97—112, 1983

[6] Cole B. C.: CAD, CMOS and VLSI are charging analog world. Electronics, pp. 35—39, December 23, 1985

[7] Cole B. C.: The big news at ISSCC — digital signal processors. Electronics, pp. 50—52, December 23, 1985

[8] Cushman R. H.: Digital signal processing advances slowly but steadily. EDN, No. 16, pp. 60—72, 1983

[9] Donze R. L., Sporzyński G.: Masterimage approach to VLSI design. Computer, No. 12, pp. 18—25, 1983

[10] Fischetti M. A.: Solid state, IEEE Spectrum, No. 1, pp. 58—63, 1984

[11] Gupta A., Toong H. D.: Microprocessors — the first twelve years. Proc. of the IEEE, No. 11, pp. 1236—1256, 1983

[12] Guterl F.: Chip architecture — a revolution brewing. IEEE Spectrum, No. 7, pp. 30—37, 1983

[13] Mead C., Convay L.: Introduction to VLSI systems. Addison-Wesley, Reading (MA), 1980

[14] Murphy B. T.: Microcomputers — trends, technologies and design strategies. IEEE Journal of Solid-State Circuits, No. 5, pp. 236—244, 1983

[15] Noyce R. N., Hoff M. E.: A history of microprocessor development at Intel. IEEE Micro, No. 1, pp. 8—20, 1981

[16] Reisman A.: Device, circuit, and technology scaling to micron and submicron dimensions, Proc. of the IEEE, No. 5, pp. 550—565, 1983

[17] Schwettmann F. N., Moll J. L.: IC process technology — VLSI and beyond. Hewlett-Packard Journal, No. 8, pp. 3—4, 1982

[18] Slamp R., Person J.: Software that resides in silicon. VLSI Design, pp. 24—28, March — April 1983

[19] Southard J. R.: MacPitts — an approach to silicon compilation. Computer, No. 12, pp. 74—82, 1983

[20] Tarr M., Boudreau D., Murphy R.: Defect analysis system speeds test and repair of redundant memories. Electronics, No. 1, pp. 175—179, 1984

[21] Toong H. D., Gupta A.: An architectural comparison of contemporary 16-bit microprocessors. IEEE Micro, No. 2, pp. 26—37, 1981

[22] Williams T. W., Parker K. P.: Design for testability — a survay. Proc. of the IEEE, No. 1, pp. 98—112, 1983

[23] Yanilos P. N.: A dedicated comparator matches symbol strings fast and intelligently. Electronics, pp. 113—117, December 1, 1983.

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

Interakcyjne środowisko języka Lisp (I)

W latach sześćdziesiątych w informatyce nastąpił przede wszystkim dalszy rozwój języków programowania wysokiego poziomu, zapoczątkowano też rozwój systemów operacyjnych. Upowszechnił się wówczas model pracy wsadowej w wielodostępnym systemie komputerowym, polegający na cyklicznym powtarzaniu procesu edycja-kompilacja-wykonanie. Pojawienie się terminali ekranowych do pracy interakcyjnej dało możliwość wykonywania tego samego procesu w sposób znacznie wygodniejszy, bo umożliwiający ciągłą obserwację i ingerencję w dowolnie wybranym momencie. Okazało się jednak, że terminal ekranowy pozwala komunikować się z komputerem w zupełnie nowy jakościowo sposób. Miejsce dawniej używanych narzędzi wsadowych zaczęły zajmować ich interakcyjne odpowiedniki — edytory ekranowe, edytory wrażliwe na składnię, interakcyjne narzędzia do testowania i usuwania błędów. Programy te zaczęto scalać w środowiska interakcyjne, które wyróżniają się następującymi cechami [1]:

- **są dużymi zestawami narzędzi wzajemnie powiązanych ze sobą i tworzącymi niepodzielną całość**
- **są ukierunkowane na konkretny język programowania, dzięki czemu programy w nim napisane są czymś więcej niż ciągami znaków, tzn. mają one strukturę mogącą służyć za podstawę organizacji środowiska**
- **pomagają w stopniowym rozwijaniu programu, zarówno w fazie projektowania i uruchamiania, jak i użytkowania**
- **w dużym stopniu wykorzystują sprzętowe możliwości interakcji z użytkownikiem.**

W pierwszej części artykułu omówiono podstawowe cechy Lispu, które zadecydowały o jego szczególnej przydatności do budowy środowiska interakcyjnego. Skrótowo można powiedzieć, że najważniejsza jest tutaj interakcyjna natura Lispu jako języka i składniowa równoważność danych i programów, a także, co poniekąd jest tego konsekwencją, historyczne pierwszeństwo lispowych środowisk interakcyjnych.

Wprowadzenie niniejsze ma na celu przedstawienie jedynie podstawowych cech języka, dlatego pominięto w nim mniej istotne szczegóły tej czy innej jego implementacji. Zamieszczone przykłady odwołują się do wersji Lispu dla komputerów CDC, opracowanej na uniwersytecie teksaskim.

Mgr inż. EDMUND PIERZCHAŁA ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej, broniąc pracę dyplomową z zakresu rozpoznawania obrazów. Obecnie pracuje w Środowiskowym Centrum Obliczeniowym CYFRONET, Instytut Energii Atomowej, zajmuje się zastosowaniem techniki sztucznej inteligencji.

STRUKTURY DANYCH

Podstawowym obiektem Lispu jest tzw. **s-wyrażenie.** Szczególnym przypadkiem s-wyrażenia, do którego można się ograniczyć jest **lista,** tzn. ciąg elementów złożonych z atomów (literowych, numerycznych lub specjalnych) i (lub) list, np.:

(A B C)
(A (B C) D (E (F G)))
(CAR (QUOTE (23 A 51)))

Jest to zewnętrzna notacja dla list, czyli to, co translator Lispu czyta lub drukuje. Listy te mają swoją reprezentację wewnętrzną w pamięci komputera (rys. 1). Podwójna kratka oznacza tzw. komórkę lispową, reprezentowaną przez słowo maszynowe lub grupę słów. Strzałki oznaczają wskaźniki (adresy), natomiast kreska ukośna oznacza wskaźnik do atomu specjalnego NIL, będącego synonimem listy pustej, zapisywanej wobec tego w notacji zewnętrznej () lub NIL.

S-wyrażeniu przypisuje się wartość według następujących reguł:

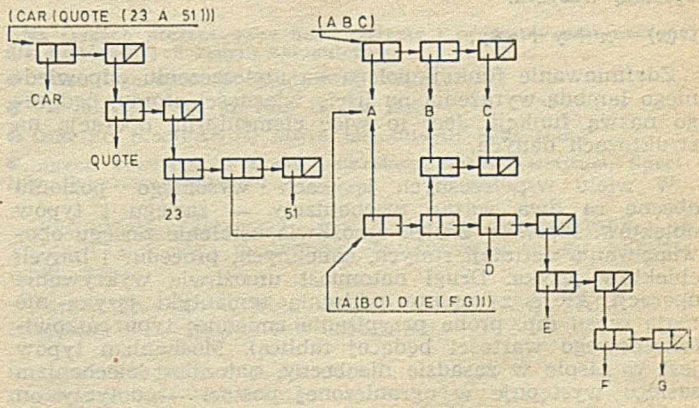
1. Wartością atomu literowego jest ostatnio przypisane mu s-wyrażenie; wartością atomu numerycznego jest ten sam atom; wartością atomu specjalnego jest s-wyrażenie przypisane mu (niekiedy na stałe) w danej wersji języka.
2. Wartością wyrażenia listowego jest wynik udostępniany przez funkcję reprezentowaną przez pierwszy element listy i zastosowaną do pozostałych jej elementów, jako argumentów.

Funkcję może reprezentować atom literowy (nazwa funkcji) lub tzw. lambda-wyrażenie.

Gdy chcemy posłużyć się samym s-wyrażeniem, nie zaś jego wartością, poprzedzamy je znakiem '.

Atom literowy ma tzw. listę własności, którą można określić jako zbiór par INDYKATOR-WŁASNOŚĆ. Indykatorem może być atom literowy, własnością natomiast — dowolne s-wyrażenie. Funkcje PUT i GET służą do manipulowania listą własności:

*(PUT 'KWADRAT 'LICZBA-KĄTÓW 4)
=4

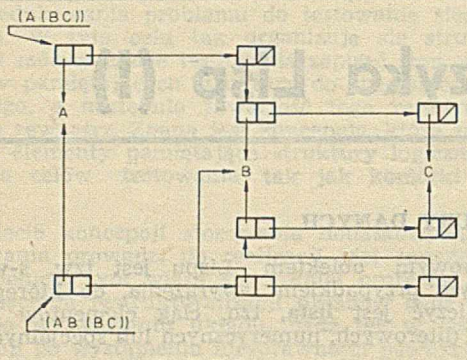


Rys. 1. Reprezentacja wewnętrzna list w pamięci komputera

Na liście własności atomu KWADRAT umieszczono własność 4 pod indykatorem LICZBA-KĄTÓW.

```
*(GET 'KWADRAT 'LICZBA-KĄTÓW)
=4
```

Znakiem „*" pętla główna interpretera sygnalizuje swą gotowość wczytania kolejnego s-wyrażenia, natomiast znakiem „=" poprzedza wyprowadzenie jego obliczonej wartości.



Rys. 2. Ilustracja reprezentacji list

Atomy w Lispie są reprezentowane niepowtarzalnie, tzn. wystąpienie w dwóch różnych miejscach programu tego samego atomu oznacza zawsze odwołanie się do tej samej struktury danych reprezentujących atom, a więc w szczególności do tej samej listy własności. Listy natomiast na ogół nie są reprezentowane niepowtarzalnie, np. reprezentacja (A (B C)) i (A B (B C)) może być taka jak na rysunku 2.

FUNKCJE

Funkcje Lispu, będące jedynymi obiektami proceduralnymi tego języka, komunikują się ze swoim otoczeniem otrzymując (jako argumenty) i udostępniając (jako wartość) s-wyrażenia. Funkcja może wywoływać efekty uboczne w swoim środowisku, takie jak zmiany wartości zmiennych wolnych czy komunikacja z urządzeniami wejścia-wyjścia. Wówczas jest ona nazywana **pseudofunkcją.**

Lambda-wyrażenie jest listą postaci:
(LAMBDA ⟨lista argumentów⟩ ⟨ciało funkcji⟩)
gdzie ⟨ciało funkcji⟩ jest ciągiem s-wyrażeń, kolejno obliczanych po wywołaniu funkcji; wartość ostatniego s-wyrażenia jest udostępniana jako jej wartość (tzn. jako wartość s-wyrażenia będącego wywołaniem funkcji).
Przykładowo, lambda-wyrażenie:

```
(LAMBDA (x y) (PLUS x y (TIMES x y)))
```

określa funkcję dwóch zmiennych x i y, opisaną wzorem:

$$f(x,y) = x + y + x*y$$

natomiast wyrażenie:

```
(LAMBDA (x) (PLUS x y (TIMES x y)))
```

określa funkcję jednej zmiennej x, zależną od zmiennej nielokalnej (zmiennej wolnej, albo inaczej parametru) y, opisaną wzorem:

$$f(x) = x + y + x*y.$$

Zdefiniowanie funkcji polega na umieszczeniu odpowiedniego lambda-wyrażenia na liście własności atomu, będącego nazwą funkcji. Jest to więc elementarna operacja na strukturach danych.

W wielu współczesnych językach wysokiego poziomu obecne są dwa ważne mechanizmy — zasięgu i typów obiektów. Pierwszy z nich umożliwia ustalenie zasięgu obowiązywania definicji stałych, zmiennych, procedur i innych obiektów języka. Drugi natomiast umożliwia wykrywanie operacji, które z punktu widzenia semantyki języka nie mają sensu (np. próba przypisania zmiennej typu całkowitoliczbowego wartości będącej tablicą). Mechanizm typów jest w Lispie w zasadzie nieobecny, natomiast mechanizm zasięgu występuje w ograniczonej postaci — dotyczy on zmiennych występujących w lambda-wyrażeniach i wyrażeniach typu PROG (p poniżej). Skoro mechanizm zasięgu nie obejmuje definicji funkcji (inaczej niż w Pascalu, a podobnie jak w Fortranie), program lispowy ujawnia swą strukturę jedynie przez wzajemne wywołania funkcji. Program lispowy jest więc zbiorem funkcji równoprawnych z punktu widzenia translatora.

STEROWANIE

W konwencjonalnych językach programowania kolejność wykonywania instrukcji jest wyznaczona przez ich sekwencję i może być zmieniana za pomocą instrukcji sterujących. W Lispie zasadniczo kolejność akcji wyznaczają zagnieżdżenia wywołań funkcji i rekursja, natomiast rolę instrukcji warunkowej, pozwalającej tworzyć konstrukcje sterujące, pełni funkcja COND.

Wywołanie funkcji COND ma następującą postać:

```
(COND ((⟨war-1⟩ ⟨s-wyr-11⟩ ⟨s-wyr-12⟩ ... ⟨s-wyr-1i⟩)
       (⟨war-2⟩ ⟨s-wyr-21⟩ ⟨s-wyr-22⟩ ... ⟨s-wyr-2j⟩)
       ...
       (⟨war-n⟩ ⟨s-wyr-n1⟩ ⟨s-wyr-n2⟩ ... ⟨s-wyr-nk⟩)))
```

Obliczając wartość takiego s-wyrażenia, interpreter znajduje wartości kolejnych warunków ⟨war⟩. Z chwilą napotkania warunku o wartości różnej od NIL (w konwencji przyjętej w Lispie oznacza to prawdę logiczną), obliczanie warunków zostaje zakończone. Umieszczone obok warunku s-wyrażenia zostają obliczone; wartość ostatniego jest udostępniana jako wartość całego wywołania COND.

Wykonanie sekwencji operacji w klasycznym sensie oraz interacji wymaga użycia konstrukcji PROG. Wywołanie PROG ma następującą składnię:

(PROG ⟨lista zmiennych lokalnych⟩ ⟨ciało⟩)

gdzie ⟨ciało⟩ jest ciągiem s-wyrażeń, które (o ile nie są to atomy) będą obliczane w takiej kolejności, w jakiej zostały wymienione. Atomy pełnią rolę etykiet dla s-wyrażeń, występujących bezpośrednio po nich. Funkcja GO realizuje skok do podanej etykiety, natomiast funkcja RETURN powoduje zakończenie wykonania PROG-u i udostępnienie przezeń wartości wyrażenia będącego argumentem RETURN.

Prostym przykładem wykorzystania konstrukcji PROG jest funkcja obliczająca silnię:

```
(DEF (SILNIA (n)
  (PROG (x)
     (SETQ x 1)
loop (COND ((ZEROP n) (RETURN x)))
     (SETQ x (TIMES x n))
     (SETQ n (SUB1 n))
     (GO loop]
```

Nawias kwadratowy domyka wszystkie niedomknięte woduje zakończenie wykonania PROG-u i udostępnienie wartość prawdy logicznej, o ile wartością jego argumentu jest liczba (atom numeryczny) 0. SUB 1 udostępnia liczbę o 1 mniejszą od wartości argumentu. Pseudofunkcja SETQ udostępnia wartość drugiego argumentu, przypisując ją też (efekt uboczny) pierwszemu argumentowi.

Rekurencyjna definicja funkcji SILNIA (bardziej naturalna dla Lispu) ma następującą postać:

```
(DEF (SILNIA (n)
(COND ((ZEROP n) 1)
      (T (TIMES n (SILNIA (SUB1 n]
```

gdzie T jest atomem zwykle używanym do oznaczenia prawdy logicznej.

W innym przykładzie definicji:

```
(DEF (OSTATNI (l)
(COND ((ATOM l) (ERROR '(niewłaściwy argument)))
      ((NULL (CDR l)) (CAR l))
      (T (OSTATNI (CDR l]
```

funkcja OSTATNI udostępnia ostatni element listy. Sprawdza ona, czy jej argument nie jest przypadkiem atomem (w szczególności atomem NIL, a więc listą pustą) — wówczas kończy działanie, generując błąd sygnalizowany przez translator Lispu komunikatem zawierającym podany tekst. NULL sprawdza, czy jego argument jest listą pustą (atomem NIL), udostępniając w takim wypadku wartość prawdy logicznej. CAR udostępnia pierwszy element listy, nato-

miast CDR — „ogon" listy, tzn. listę powstałą przez usunięcie pierwszego elementu.

PĘTLA GŁÓWNA INTERPRETERA

Interpreter Lispu jest używany znacznie powszechniej niż kompilator tego języka. Główna pętla interpretera składa się z następujących po sobie operacji: READ, EVAL, PRINT. Interpreter czyta s-wyrażenie podane przez programistę, oblicza jego wartość, po czym wyprowadza ją:

```
*(CAR '(a b c))
=a
*(CDR '(x (y z)))
= ((y z))
```

Po wyprowadzeniu wartości interpreter czyta następne wyrażenie, oblicza je, wyprowadza wartość itd. Tak więc pętlę główną Lispu można sobie wyobrażać jako:

```
(PROG(exp)
pętlagłówna (PRINT '*)
            (SETQ exp (READ))
            (PRINT '=)
            (PRINT (EVAL exp))
            (GO pętlagłówna]
```

Ponieważ READ, EVAL i PRINT są funkcjami Lispu (służącymi do czytania, obliczania wartości i drukowania s-wyrażenia), programista może zaimplementować własną pętlę główną o pożądanych przez niego cechach. Zwykle zestaw funkcji standardowych interpretera jest tak zaprojektowany, aby ułatwić to zadanie (zawiera on np. funkcje przechwytujące sterowanie w razie wystąpienia błędu, symulujące błąd itp.).

Brak fazy generacji i wykonywania kodu wynikowego pozwala — w wypadku pracy interakcyjnej — na stopniowe wpisywanie i uruchamianie programu. Sprzyja temu fakt, że wymagania języka nie narzucają programowi żadnej struktury — poszczególne funkcje, składające się na program, mogą być kolejno prezentowane interpreterowi i testowane. W miarę postępowania tego procesu programista dysponuje coraz większą liczbą uruchomionych podstawowych „cegiełek" i buduje z nich nowe funkcje, realizujące coraz bardziej skomplikowane zadania. Ten sposób uruchamiania programu jest bardzo niewygodny w wypadku kompilowanego języka, stawiającego wysokie wymagania strukturze programu. W Pascalu, na przykład, testowanie pojedynczej procedury wymagałoby „obudowania" jej instrukcjami, czytającymi dane testowe i drukującymi wyniki. Połączenie kilku przetestowanych procedur wymagałoby oczywiście dodatkowego przebiegu kompilacji. W interakcyjnie interpretowanym Lispie wystarczy dopisać następną definicję funkcji. Pętla READ-EVAL-PRINT interpretera sama zajmuje się czytaniem s-wyrażeń i wydrukiem ich wartości, uwalniając programistę od konieczności oprogramowania wejścia-wyjścia dla danych testowych i wyników testów. Dostępność definicji funkcji (zapisanej na liście własności atomu, będącego jej nazwą) pozwala modyfikować treść funkcji bez potrzeby opuszczania interpretera.

INTERAKCYJNOŚĆ LISPU

Poniżej uporządkowano informacje na temat tych cech Lispu, które mają znaczenie przy budowie środowiska interakcyjnego dla tego języka [2]:

- **Samowystarczalność**

Środowisko interakcyjne Lispu może być zrealizowane w samym Lispie. Daje to korzyść polegającą na możliwości pracy z jednym językiem; ponadto tworzące się środowisko może wspomagać swój własny rozwój.

- **Możliwość przyrostowego uruchamiania programu**

Wprowadzenie definicji nowej funkcji lub wykonanie dowolnego innego s-wyrażenia, pozostawiającego trwały efekt w bazie danych interpretera, nie jest okupione koniecznością wywoływania edytora, kompilacji całego programu itd. Lisp realizuje takie zadanie w jednym cyklu pętli READ-EVAL-PRINT interpretera.

- **Istnienie wewnętrznej reprezentacji programów i łatwy dostęp do niej**

W Lispie program jest reprezentowany w postaci s-wyrażeń, mających dobrze określoną reprezentację wewnętrzną i zewnętrzną. Pozwala to łatwo operować programami przy użyciu innych programów. Jądro środowiska interakcyjnego powinno zawierać:

— funkcję tłumaczącą reprezentację zewnętrzną programów na reprezentację wewnętrzną (funkcja READ)
— funkcję tłumaczącą reprezentację wewnętrzną na zewnętrzną (funkcja PRINT)
— kompilator tłumaczący wewnętrzną reprezentację programu na kod maszynowy.

Należy zauważyć, że konwencjonalny kompilator spełnia funkcje podzielone tutaj pomiędzy punkty 1 i 3.

- **Wejście-wyjście dla struktur danych**

Interpreter Lispu zapewnia wczytywanie i drukowanie dowolnych struktur danych (s-wyrażeń) za pomocą pojedynczych operacji — READ i PRINT. Inaczej jest np. w Pascalu, który nakłada na programistę obowiązek obsługi wejścia-wyjścia w wypadku takich struktur danych, jak rekordy, tablice itp.

- **Obsługa błędów kontrolowana przez użytkownika**

Interpreter Lispu pozwala na interakcyjny dostęp do swych atomów systemowych (decydujących o podstawowych jego parametrach) oraz na programowanie mechanizmu przerwań (generowanych m.in. w wypadku wykrycia błędu). Daje to możliwość elastycznego kształtowania takiego sposobu reagowania interpretera na błędy, jaki najbardziej odpowiada programiście.

* * *

Na zakończenie warto podkreślić, że takie konwencjonalne języki programowania, jak Fortran, PL/I, Pascal, Simula itp. nie spełniają wymagań dotyczących wejścia-wyjścia dla struktur danych, nie zapewniają dostępu do reprezentacji wewnętrznej programów w takim sensie jak Lisp, jak również trudno jest w nich pisać programy metodą stopniowego dodawania fragmentów. Natomiast Snobol i APL spełniają wyżej wymienione wymagania.

LITERATURA

[1] Barstow D. R., Shrobe H. E., Sandewall E.: Interactive Programming Environments. McGraw-Hill, New York, 1984
[2] Sandewall E.: Programming in an Interactive Environment — The LISP Experience. ACM Computing Surveys, Vol. 10, No. 1, pp. 35—71, March 1978.

JACEK OWCZARCZYK
MACIEJ STOLARSKI
Instytut Podstaw Informatyki PAN
EWA WOŹNIAK
Instytut Maszyn Matematycznych

Architektura współbieżnych systemów przetwarzania obrazów

Wszystkie nowe tendencje pojawiające się w architekturze sprzętu komputerowego, a pozwalające na zwiększenie jego mocy obliczeniowej, znajdują natychmiast odbicie w architekturze systemów przetwarzania obrazów. Komputerowe przetwarzanie obrazów jest bowiem dobrym „poligonem doświadczalnym" dla weryfikacji nowych rozwiązań, gdyż wymaga ono z reguły bardzo dużych mocy obliczeniowych, a ponadto różnorodność jego zastosowań tworzy sprzyjające warunki dla szerokiej w tej dziedzinie bazy na świecie (rozumianej jako liczba ośrodków naukowo-badawczych zajmujących się tą problematyką).

Kluczem do uzyskania dużych mocy obliczeniowych w komputerach nowych generacji jest szeroko rozumiana współbieżność procesów obliczeniowych. Postępy, jakie osiągnięto w technologii, umożliwiły konstrukcję systemów wieloprocesorowych, odbiegających od klasycznych architektur systemów komputerowych. Celem niniejszego artykułu jest przedstawienie zasadniczych tendencji występujących w architekturze systemów przetwarzania obrazów opartych na współbieżnej (równoległej) realizacji obliczeń.

Stosowanie techniki cyfrowej w przetwarzaniu obrazów wymaga ich uprzedniej dyskretyzacji. Proces ten coraz częściej wykonuje się już w momencie pozyskiwania danych, gdyż umożliwia to stosowanie techniki cyfrowej zarówno do rejestracji, jak i do transmisji obrazów. Przykładem mogą tu być obrazy satelitarne, rejestrowane i przesyłane do naziemnych stacji odbiorczych już w postaci cyfrowej. Bez względu jednak na to, czy proces dyskretyzacji przebiega w momencie pozyskiwania danych, czy też dyskretyzuje się dopiero utrwaloną uprzednio kopię obrazu (np. zdjęcie fotograficzne), zasada tego procesu jest taka sama. Zawiera on dwa następujące etapy:

— **rastrowanie**, czyli podział obrazu na elementarne pola (punkty) i przyporządkowanie każdemu z nich pewnej liczby — najczęściej średniej wartości rejestrowanej wielkości fizycznej (dla elementarnego pola);

— **kwantowanie**, czyli przekształcenie otrzymanego w etapie pierwszym zbioru liczb w zbiór przeliczalny (dyskretny).

Uzyskana w ten sposób tablica będzie nazywana obrazem cyfrowym, a jej elementy — elementami lub punktami obrazu. W szczególności obraz cyfrowy, którego poszczególnym elementom przyporządkowano liczbę binarną, będziemy nazywać obrazem binarnym.

W typowych zastosowaniach liczba bitów, potrzebna do kodowania poszczególnych elementów, jest stosunkowo niewielka (najczęściej wystarcza osiem bitów), lecz jednocześnie występują tu względnie duże wymiary tablic (rzędu kilkuset lub nawet kilku tysięcy).

STRUKTURY OBLICZEŃ W KOMPUTEROWYM PRZETWARZANIU OBRAZÓW

Komputerowe przetwarzanie obrazów może być traktowane jako wykonywanie pewnych operacji na obrazach cyfrowych. Typowe operacje występujące w komputerowym przetwarzaniu obrazów można podzielić na trzy podstawowe grupy [14]:

— operacje punktowe (jednoelementowe),
— operacje lokalne (na otoczeniach),
— globalne transformacje obrazu.

Podstawą takiej klasyfikacji jest liczba elementów, z jakich korzysta się przy obliczeniach pojedynczego elementu przetworzonego (wyjściowego) obrazu. Jak zostanie wykazane w dalszej części artykułu, szybka realizacja każdej operacji z wymienionych grup wymaga innych zasobów obliczeniowych.

Operacje punktowe

W operacjach punktowych obliczana wartość obrazu wyjściowego w dowolnym jego punkcie zależy jedynie od wartości punktu o tych samych współrzędnych obrazu wejściowego. Operacje punktowe są używane m.in. przy zwiększaniu kontrastu obrazu [13], segmentacji, a także w wielu przekształceniach elementów obrazu zarówno liniowych, jak i nieliniowych.

W praktycznych zastosowaniach, do kodowania poszczególnych elementów obrazów cyfrowych potrzebna jest stosunkowo niewielka liczba różnych wartości, a tym samym nieduża również liczba argumentów przekształceń punktowych. Umożliwia to względnie prostą realizację operacji

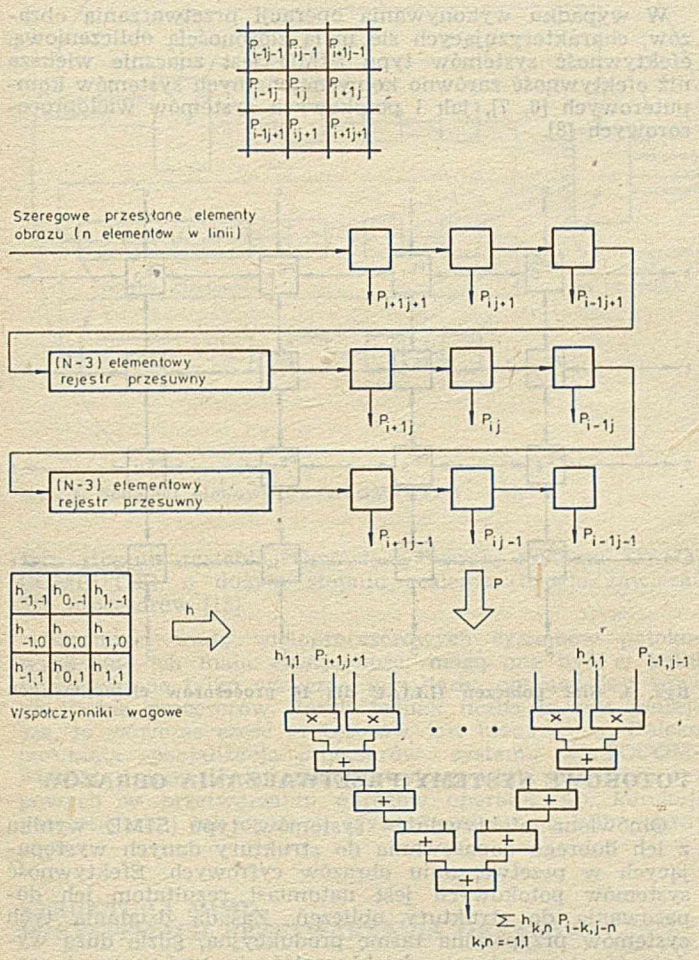
Dr JACEK OWCZARCZYK ukończył studia na Uniwersytecie Warszawskim (matematykę) i Uniwersytecie w Durham, Wlk. Brytania (informatykę). W 1977 r. otrzymał stopień doktora nauk technicznych w Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej. Pracował m.in. na uniwersytetach w Lagos i Londynie. Obecnie jest adiunktem w Instytucie Podstaw Informatyki PAN. Prowadzi również LABORATORIUM SYSTEMÓW KOMPUTEROWYCH. Zajmuje się cyfrowym przetwarzaniem obrazów i grafiką komputerową.

Prof. dr MACIEJ STOLARSKI pracuje w Instytucie Podstaw Informatyki PAN. Zajmuje się architekturą systemów komputerowych i grafiką komputerową. Pod jego kierunkiem został opracowany w 1970 roku pierwszy w kraju wektorowy system graficzny.

punktowych, np. za pomocą dodatkowej, odpowiednio wypełnionej tablicy, tzw. tablicy pośredniej. Sprzętową jej realizacją może być np. dodatkowy blok pamięci RAM, wypełnianej w zależności od wykonywanej operacji, a adresowany elementami obrazu wejściowego.

Operacje lokalne

W operacjach lokalnych wartość w dowolnym punkcie obrazu wyjściowego zależy nie tylko od wartości odpowiadającego mu punktu obrazu wejściowego, lecz także od wartości otoczenia tego punktu. Operacje tego typu są używane np. przy zwiększaniu ostrości obrazu, detekcji brzegów, wygładzaniu obrazu i filtrowaniu szumów.



Rys. 1. Sprzętowa realizacja operacji lokalnych dla otoczenia 3 × 3

Operacje lokalne stanowią obecnie najliczniejszą klasę obliczeń w komputerowym przetwarzaniu obrazów i dlatego sposób ich realizacji decyduje o efektywności całego systemu. Złożoność obliczeniowa operacji lokalnych zależy w oczywisty sposób od wielkości otoczenia. Dla ustalonego typu tego otoczenia szybkość wykonywania operacji w dużym stopniu zależy od czasu pobierania argumentów. Najlepsza zatem jest organizacja systemu, umożliwiająca jednoczesny dostęp do wszystkich wartości z otoczenia każdego punktu obrazu.

Realizacja jednoczesnego dostępu dla prostego, najczęściej stosowanego typu otoczenia 3×3, jest możliwa przez zastosowanie odpowiednich rejestrów opóźniających i szeregowe przesyłanie obrazu cyfrowego (rys. 1). Dalsze zwiększenie szybkości wykonywania operacji (dla liniowych kombinacji wartości z otoczenia) można uzyskać przez zastosowanie układów mnożąco-sumujących.

Dla obrazów binarnych realizacja operacji lokalnych dla otoczenia typu 3×3 znacznie się upraszcza, ponieważ liczba wszystkich przekształceń dla dziewięcioelementowego otoczenia wynosi jedynie 512, można więc tu wykorzystać omówioną wcześniej zasadę tablicy pośredniej.

Globalne transformacje obrazu

W transformacjach globalnych przy obliczaniu pojedynczego punktu obrazu wyjściowego uwzględnia się wszystkie elementy obrazu wejściowego. Przykładem tej klasy obliczeń może być transformacja Fouriera (analiza widmowa obrazu), a także transformacje Hadamarda, Karhunen-Loeve'go, stosowane przy filtrowaniu, kompresji i kodowaniu obrazów.

Struktura obliczeń w typowych operacjach stosowanych w przetwarzaniu obrazów określa organizację systemu obliczeniowego, a ponadto pozwala oszacować wymagane zasoby obliczeniowe. Na przykład, realizacja globalnych transformacji obrazu wymaga pamięci RAM o podwójnej pojemności (w stosunku do wymiarów obrazu), natomiast wykonywanie operacji punktowych lub lokalnych może odbywać się w tym samym bloku pamięci.

WSPÓŁBIEŻNOŚĆ W PRZETWARZANIU OBRAZÓW

Z dotychczasowych rozważań wynika, że przetwarzanie obrazów charakteryzuje się bardzo dużą złożonością obliczeniową. Już nawet wykonanie prostej operacji lokalnej dla obrazu o wymiarach 512×512 (w wielu zastosowaniach są to typowe wymiary obrazów cyfrowych) z najprostszym otoczeniem typu 3×3, wymaga wykonania aż dwóch milionów instrukcji maszynowych. Bardziej złożone obliczenia wymagają nawet kilkuset lub kilku tysięcy instrukcji dla każdego punktu obrazu. Należy się liczyć z tym, że w niektórych zastosowaniach, np. w przetwarzaniu obrazów satelitarnych, wymiary obrazów są rzędu kilku tysięcy. Szacuje się, że dla tych zastosowań wymagana moc obliczeniowa powinna wynosić 1—10 mld operacji na sekundę [2]. Podobne wymagania występują również w takich zastosowaniach, w których obrazy muszą być przetwarzane w czasie rzeczywistym. Klasyczne systemy komputerowe, oparte na architekturze Von Neumanna, nie są w stanie osiągnąć takiej mocy obliczeniowej, korzystając z istniejącej technologii. Główną barierę stanowi mała przepustowość kanału pamięć-procesor.

Zasadniczo istnieją dwie możliwe strategie udoskonalania architektury systemów przetwarzania obrazów. Pierwsza z nich polega na skróceniu czasu obliczeń przez wprowadzenie możliwie jak największej liczby równocześnie pracujących procesorów. W szczególności liczba procesorów może być równa nawet liczbie punktów obrazu i wówczas obliczenia wykonywane będą dla wszystkich punktów współbieżnie (równolegle). W tym wypadku czas wykonywania operacji będzie zależeć jedynie od jej złożoności obliczeniowej, a nie od liczby punktów obrazu.

Druga strategia opiera się na optymalnej organizacji specjalizowanych (dla typowych operacji przetwarzania obrazów) procesorów. Skrócenie czasu obliczeń zależy tu zarówno od wyeliminowania cyklu pobierania instrukcji, jak również od współbieżnej (nakładanej w czasie) realizacji poszczególnych operacji.

Należy dodać, że wspomniana organizacja obliczeń, w aktualnie konstruowanych systemach przetwarzania obrazów, jest w dużej mierze zdeterminowana przez szeregowy sposób przesyłania danych w urządzeniach wejściowych i wyjściowych tych systemów. Ten tryb transmisji ogranicza wypadkową efektywność systemów współbieżnych, trudno jest jednak dzisiaj wskazać inne rozwiązanie — ekonomicznie uzasadnione i technicznie możliwe — dla realizacji przyjętej w telewizyjnej technice dekompozycji i formowania obrazu.

Odpowiednio duża moc obliczeniowa systemów przetwarzania obrazów może być uzyskana przez różne formy współbieżności obliczeń, dlatego architektura tych systemów

Mgr EWA WOŹNIAK ukończyła w 1974 r. Wydział Matematyki Uniwersytetu Warszawskiego. Od 1974 roku pracuje w Instytucie Maszyn Matematycznych. Początkowo zajmowała się sterowaniem urządzeń graficznych, a obecnie oprogramowaniem systemów mikroprocesorowych.

nie jest jednolita. Dodatkowo różnorodność rozwiązań architektonicznych jest potęgowana przez różne cele stawiane przed systemami przetwarzania obrazów. Przedstawione w dalszej części artykułu rozwiązania charakteryzują główne tendencje istniejące w udoskonalaniu architektury systemów przetwarzania obrazów. Przy omawianiu ich organizacji przyjęto najczęściej stosowany podział wieloprocesorowych systemów komputerowych [9]:

— systemy wieloprocesorowe typu SIMD (Single Instruction stream — Multiple Data stream; pojedynczy strumień instrukcji — wielokrotny strumień danych),

— potokowe systemy wieloprocesorowe,

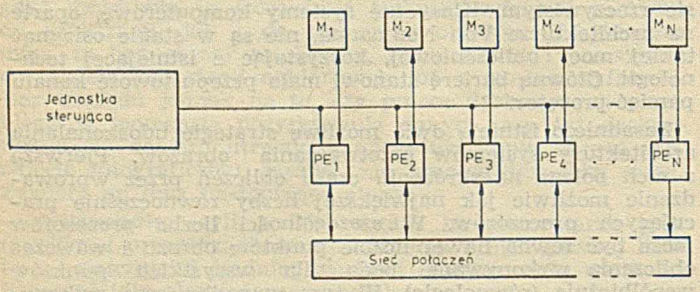
— systemy wieloprocesorowe typu MIMD (Multiple Instruction stream — Multiple Data stream; wielokrotny strumień instrukcji — wielokrotny strumień danych).

SYSTEMY PRZETWARZANIA OBRAZÓW WYKORZYSTUJĄCE STRUKTURĘ SIMD

Struktura SIMD jest siecią jednakowych procesorów elementarnych PE, wspólnie sterowanych przez jednostkę sterującą (rys. 2). Każdy procesor zawiera lokalną pamięć (M_i); procesory komunikują się między sobą za pomocą sieci połączeń. Poszczególne realizacje systemów o tej strukturze mogą się różnić:

- liczbą N procesorów elementarnych; obecnie najbardziej rozbudowanym systemem jest MPP (Massiwely Parallel Processor), przeznaczony do przetwarzania obrazów satelitarnych i zawierający 16 K (K = 1024) procesorów [2],
- parametrami procesorów takimi jak: liczba rejestrów (zazwyczaj jest ich od 4 do 8), pojemność pamięci lokalnej (od kilkudziesięciu do 4096 bitów), możliwość maskowania poszczególnych procesorów (tzn. wprowadzania ich w stan nieaktywny),
- rodzajem sieci połączeń.

Spośród wielu rodzajów sieci połączeń, które mogą być używane w systemie o strukturze SIMD [16, 17], w systemach służących do przetwarzania obrazów najczęściej spotyka się sieć, w której każdy z procesorów jest połączony z czteroma lub ośmioma sąsiednimi procesorami. Systemy z taką siecią połączeń są nazywane również procesorami tablicowymi. Wariant tej sieci (rys. 3), w którym skrajne procesory połączone są cyklicznie (co umożliwia zmiany konfiguracji tablicy procesorów), jest nazywany ILLIAC — od nazwy komputera, w którym został użyty po raz pierwszy.



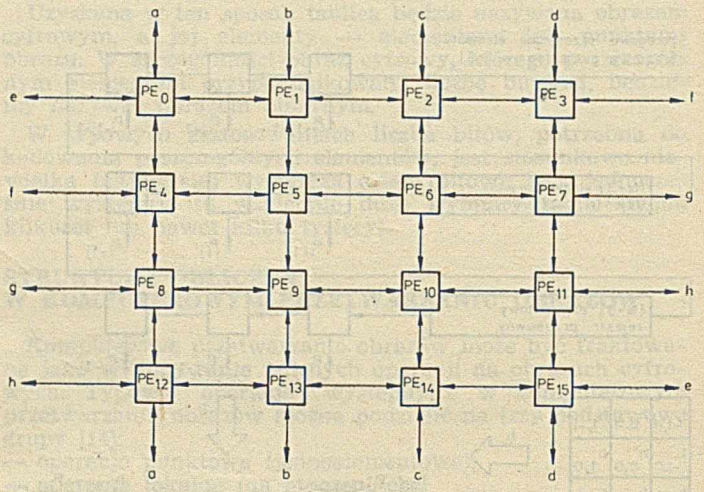
Rys. 2. System wieloprocesorowy SIMD

Zaletą procesorów tablicowych jest szybka i prosta realizacja operacji lokalnych, które w przetwarzaniu obrazów odgrywają szczególną rolę. Dzięki zaś regularnym połączeniom procesorów istnieje możliwość wykonania ich w postaci układu o dużym i bardzo dużym stopniu scalenia. Na przykład układ scalony GAPP (Geometric Arithmetic Parallel Processor) [4], produkowany przez firmę NCR, zawiera 72 procesory; każdy z nich jest połączony z czterema sąsiednimi, tworząc tablice 12×6. Każdy procesor ma cztery rejestry i lokalną pamięć o pojemności 128 bitów; podczas jednego cyklu, trwającego 100 ns, może wykonywać pięć różnych instrukcji.

W systemach przetwarzania obrazów wykorzystujących strukturę SIMD, procesory elementarne charakteryzują się jednobitową szyną danych, to znaczy, że dla grupy bitów odpowiadającej pojedynczemu punktowi obrazu, operacje są wykonywane sekwencyjnie [3]. Ten sposób realizacji obliczeń, jakkolwiek mniej efektywny w porównaniu z realizacją za pomocą arytmetyczno-logicznych układów równoległych, zwiększa elastyczność systemów przetwarzania obrazów (ze względu na różną, w różnych zastosowaniach liczbę bitów odpowiadającą pojedynczemu punktowi obrazu).

Podstawową wadą systemów o strukturze SIMD jest zbyt mała — w porównaniu z wymiarami obrazów cyfrowych w konkretnych zastosowaniach — liczba procesorów, co powoduje konieczność podziału obrazu podczas jego przetwarzania, a tym samym istotnie utrudnia organizację obliczeń. Dla dużej liczby procesorów elementarnych, istotnym problemem technicznym jest wytworzenie i przesyłanie bez opóźnień sygnałów zegarowych, umożliwiających synchroniczne działanie tablicy procesorów.

W wypadku wykonywania operacji przetwarzania obrazów, charakteryzujących się małą złożonością obliczeniową, efektywność systemów typu SIMD jest znacznie większa niż efektywność zarówno konwencjonalnych systemów komputerowych [6, 7], jak i potokowych systemów wieloprocesorowych [8].



Rys. 3. Sieć połączeń ILLIAC dla 16 procesorów elementarnych

POTOKOWE SYSTEMY PRZETWARZANIA OBRAZÓW

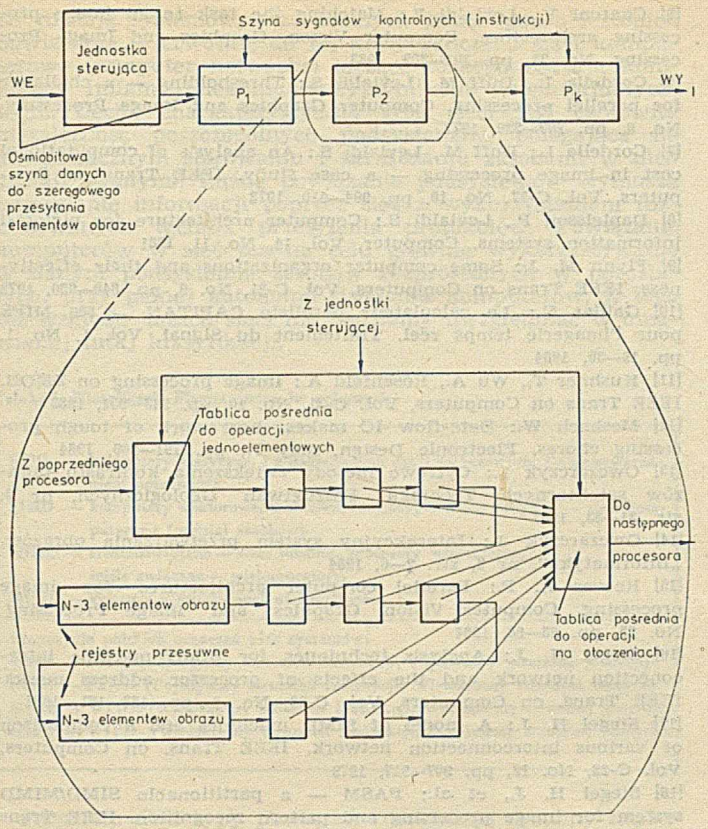
Omówiona efektywność systemów typu SIMD wynika z ich dobrego dopasowania do struktury danych występujących w przetwarzaniu obrazów cyfrowych. Efektywność systemów potokowych jest natomiast rezultatem ich dopasowania do struktury obliczeń. Zasada działania tych systemów przypomina taśmę produkcyjną, gdzie dużą wydajność otrzymuje się dzięki rozłożeniu wykonywania zadania globalnego na mniejsze etapy i współbieżnym ich wykonywaniu.

Zasada przetwarzania potokowego jest bardzo często stosowana we współczesnych systemach komputerowych. Może być ona stosowana przy realizacji pojedynczej instrukcji (nakładanie na siebie w czasie poszczególnych faz kolejnych instrukcji, np. pobranie rozkazu, dekodowanie rozkazu, pobranie argumentu, wykonywanie rozkazu), jak również przy współbieżnej (nakładanej na siebie w czasie) realizacji różnych operacji arytmetycznych.

Przykładem systemu przetwarzania obrazów, w którym wykorzystano tę zasadę, jest CYTOCOMPUTER [19]. Jego strukturę przedstawiono na rys. 4. System ten składa się ze 113 procesorów dwóch różnych typów. Pierwszy typ jest przeznaczony do realizacji operacji punktowych i lokalnych w obrazach binarnych, drugi — do analizy obrazów wielotonalnych.

Jak wspomniano, w obecnie spotykanych systemach przetwarzania obrazów najczęściej stosuje się szeregowe przesyłanie obrazów cyfrowych. Ten tryb transmisji wynika z naśladowania techniki telewizyjnej w formowaniu i dekompozycji obrazu. Dlatego też włączanie układu kaskadowo połączonych procesorów, takich jak w systemie CYTOCOMPUTER, jest w tych systemach szczególnie proste.

Zaletą procesorów potokowych jest również prosta struktura połączeń, dzięki czemu można stosować układy o du-



Rys. 4. Schemat blokowy CYTOCOMPUTER

żym stopniu scalenia. Omówiona wersja systemu CYTOCOMPUTER, o dużym stopniu scalenia, będzie zawierać 550 procesorów [15].

Natomiast wadą wieloprocesorowych systemów potokowych jest ich mała elastyczność; mogą one być w pełni wykorzystane tylko wówczas, gdy liczba operacji jest równa liczbie procesorów. Jeżeli jednak liczba ta jest mniejsza, to wówczas część procesorów jest nieużywana. Daleko posunięta specjalizacja procesorów systemu CYTOCOMPUTER uniemożliwia ponadto wykonywanie pewnych, typowych w przetwarzaniu obrazów operacji, np. korekcji geometrycznej lub operacji arytmetycznych na kilku obrazach.

STRUKTURA MIMD W SYSTEMACH PRZETWARZANIA OBRAZÓW

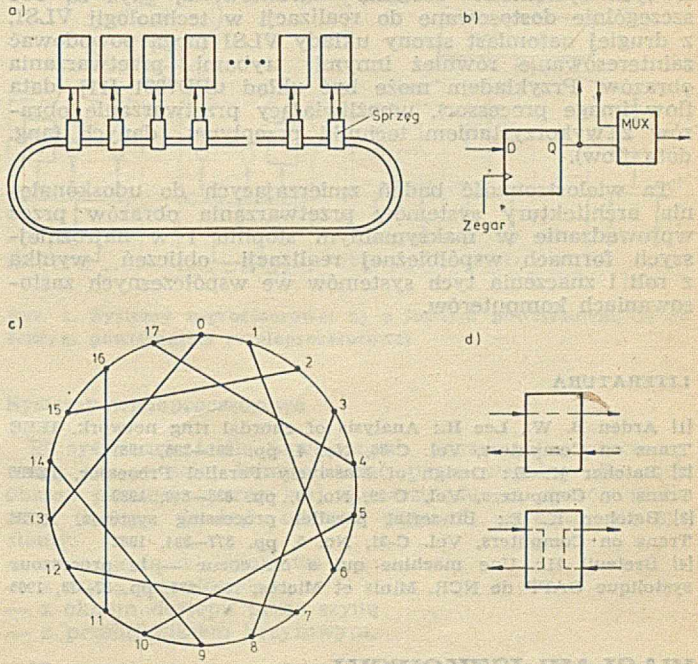
W wielu zastosowaniach, przetwarzanie obrazów stanowi pierwszy etap globalnego procesu obliczeniowego i struktura dalszych obliczeń może istotnie odbiegać od dotychczas omawianej. Na przykład analizę obrazu, obejmującą rozpoznawanie i klasyfikację obiektów, można zaliczyć do realizacji szeregu procesów obliczeniowych (procesem takim może być np. identyfikacja pojedynczego obiektu), przy czym każdy z nich może być wykonywany niemal niezależnie od pozostałych. Wspólną bazę danych stanowi tu przetworzony obraz. Współbieżną realizację takich obliczeń umożliwia system, w którym każdy z procesów jest wykonywany za pomocą oddzielnego procesora, a więc system o strukturze MIMD.

W niniejszym artykule, struktura MIMD opisuje sieć takich procesorów, które pracują współbieżnie, wykorzystując wspólne zasoby obliczeniowe, np. pamięć. Sieć połączeń jest tak zorganizowana, że umożliwia przepływ danych pomiędzy poszczególnymi procesorami, a transmisja danych odbywa się asynchronicznie i ma mniejsze natężenie i inny charakter niż w systemach typu SIMD.

W systemach przetwarzania obrazów, wykorzystujących strukturę MIMD, używa się najczęściej typowych mikroprocesorów, np. ZMOB (256×Z80A) [11], FLIP (16×Z80A) [8], PASM (1024×68010) [18]. (W nawiasach podano liczbę i typ używanych mikroprocesorów).

Stosuje się również specjalizowane procesory wykorzystujące układy VLSI, np. TMS320 czy uPD7720 (procesory do przetwarzania sygnałów), umożliwiające dalsze zwiększenie mocy obliczeniowej. Takie rozwiązanie zostało zastosowane np. w systemie CAPITAN [10], w którym 15 procesorów (z których każdy zawiera osiem układów TMS320) pozwala na przetwarzanie obrazów z szybkością 600 milionów operacji na sekundę.

W systemach MIMD można używać różnych rodzajów połączeń między poszczególnymi procesorami, przy czym w systemach przetwarzania obrazów regułą jest równoległe przesyłanie danych. Z porównania efektywności podstawowych rodzajów połączeń (np. „gwiazda", „pierścień", wspólna magistrala, „krzyżowe", (ang. crossbar) czy „każdy z każdym" wynika [21], że bardzo dobre parametry uzyskuje się dla połączenia pierścieniowego (pomijając połączenie „każdy z każdym", którego realizacja dla dużej liczby procesorów jest praktycznie niemożliwa).



Rys. 5a. System wieloprocesorowy MIMD z pierścieniową siecią połączeń
Rys. 5b. Sprzęg sieci pierścieniowej (dla jednej linii)
Rys. 5c. Ilustracja dodatkowych połączeń w sieci pierścieniowej
Rys. 5d. Węzeł pierścieniowej sieci połączeń z dwukierunkową szyną połączeń i mechanizmem umożliwiającym eliminację uszkodzonych elementów sieci

Połączenie pierścieniowe jest używane m.in. w systemach ZMOB, PASM, CAPITAN. Składa się ono z cyklicznych rejestrów przesuwnych (rys. 5a, b), co stanowi rozszerzenie jednoprzewodowego połączenia pierścieniowego używanego w sieciach lokalnych. Liczba linii pierścienia zależy więc od liczby procesorów, liczby bitów danych i protokołu komunikacyjnego. Jednocześnie (równolegle) przesyłane są: adres nadawcy, adres odbiorcy oraz dane i sygnały kontrolne.

W celu zwiększenia niezawodności połączenia pierścieniowego stosuje się różne jego modyfikacje, np. przez wprowadzenie dodatkowych połączeń wzdłuż cięciw [1] (rys. 5c), przez stosowanie podwójnego pierścienia przeciwnych kierunkach przesyłania danych lub mechanizmów „zwierania" pierścienia, w celu eliminacji uszkodzonego elementu sieci (rys. 5d). Ta ostatnia zasada może być również wykorzystywana do rekonfiguracji systemu w zależności od zadań obliczeniowych.

* * *

Omówione systemy charakteryzują zasadnicze tendencje występujące w architekturze współbieżnych systemów przetwarzania obrazów. Prace badawcze koncentrują się obecnie na koncepcji architektury umożliwiającej dynamiczne zmiany — rekonfigurację systemu w zależności od stawianych mu zadań obliczeniowych. Na przykład, do wstępnego przetwarzania obrazów celowe jest zastosowanie struktury SIMD, natomiast do analizy obrazu — systemu o strukturze MIMD. Dzięki dostosowaniu struktu-

ry systemu do aktualnie wykonywanych obliczeń, efektywność systemowo zmiennej konfiguracji wzrośnie.

Inną istotną tendencją występującą w architekturze systemów przetwarzania obrazów jest wprowadzanie w nich specjalizowanych procesorów umożliwiających sprzętową realizację pewnych typowych obliczeń, np. szybkiej transformaty Fouriera (FFT) czy iloczynu skalarnego wektorów [20]. Jakkolwiek konstrukcja takich modułów jest obecnie dużo łatwiejsza, dzięki odpowiednim układom o dużym scaleniu, to jednak zbyt duża liczba różnych specjalizowanych procesorów powoduje niepełne wykorzystanie sprzętu obliczeniowego i może utrudniać organizację współbieżnej realizacji procesów obliczeniowych.

Również postęp w technologii VLSI może mieć istotny wpływ na architekturę systemów przetwarzania obrazów. Z jednej strony wydaje się, że rola procesorów tablicowych w tych systemach zostanie ugruntowana, gdyż są one szczególnie dostosowane do realizacji w technologii VLSI, z drugiej natomiast strony układy VLSI mogą powodować zainteresowanie również innymi trybami przetwarzania obrazów. Przykładem może być układ uPD7821 [11] (data flow image processor), umożliwiający przetwarzanie obrazów z wykorzystaniem techniki przepływu danych (ang. data flow).

Ta wielostronność badań zmierzających do udoskonalenia architektury systemów przetwarzania obrazów przez wprowadzanie w maksymalnym stopniu i w najróżniejszych formach współbieżnej realizacji obliczeń wynika z roli i znaczenia tych systemów we współczesnych zastosowaniach komputerów.

LITERATURA

[1] Arden B. W., Lee H.: Analysis of chordal ring network. IEEE Trans on Computers, Vol. C-30, No. 4, pp. 291—295, 1981
[2] Batcher K. E.: Design of Massively Parallel Processor. IEEE Trans on Computers, Vol. C-29, No. 9, pp. 836—840, 1980
[3] Batcher K. E.: Bit-serial parallel processing systems. IEEE Trans on Computers, Vol. C-31, No. 5, pp. 377—384, 1982
[4] Breteuil H.: Une machine qui a du coeur — le processour systolique GAPP de NCR. Minis et Micros, No. 235, pp. 67—73, 1985
[5] Cantoni V., Levialdi S.: Matching the task to an image processing architecture. Computer Vision, Graphics and Image Processing, No. 22, pp. 301—309, 1983
[6] Cordella L., Duff M., Levialdi S.: Thresholding — a challange for parallel processing. Computer Graphics and Image Processing, No. 6, pp. 207—220, 1977
[7] Cordella L., Duff M., Levialdi S.: An analysis of computational cost in image processing — a case study. IEEE Trans on Computers, Vol. C-27, No. 10, pp. 904—910, 1978
[8] Danielsson P., Levialdi S.: Computer architecture for pictorial information systems. Computer, Vol. 14, No. 11, 1981
[9] Flynn M. J.: Some computer organizations and their effectivness. IEEE Trans on Computers, Vol. C-21, No. 9, pp. 848—960, 1972
[10] Gaillat G.: Le calculateur parallele CAPITAN — 600 MIPS pour l'imagerie temps reel. Traitement du Signal, Vol. 1, No. 1, pp. 19—30, 1984
[11] Kushner T., Wu A., Rosenfeld A.: Image processing on ZMOB. IEEE Trans on Computers, Vol. C-31, No. 10, pp. 943—951, 1982
[12] Meshach W.: Data-flow IC makes short work of tough processing chores. Electronic Design, May 17, pp. 191—206, 1984
[13] Owczarczyk J.: Cyfrowe metody zwiększenia kontrastu obrazów satelitarnych. Technika Poszukiwań Geologicznych, nr 2, str. 31—33, 1983
[14] Owczarczyk J.: Interakcyjny system przetwarzania obrazów. „Informatyka", nr 2, str. 2—6, 1984
[15] Reeves A. P.: Parallel computer architectures for image processing. Computer Vision, Graphics and Image Processing. No. 25, pp. 68—88, 1984
[16] Siegel H. J.: Analysis techniques for SIMD machine interconection network and the effects of processor address masks. IEEE Trans. on Computers, Vol. C-26, No. 2, pp. 153—161, 1977
[17] Siegel H. J.: A model of SIMD machines and a comparison of various interconnection network. IEEE Trans. on Computers, Vol. C-28, No. 12, pp. 907—917, 1979
[18] Siegel H. J., et al.: PASM — a partitionable SIMD/MIMD system for image processing and pattern recognition. IEEE Trans on Computers, Vol. C-30, No. 12, pp. 934—947, 1981
[19] Sternerg S. R.: Biomedical image processing. Computer, Vol. 16, No. 1, pp. 22—34, 1983
[20] Swartzlander E., Gilbert B., Reed I.: Inner product computers. IEEE Trans on Computers, Vol. C-27, No. 1, pp. 21—31, 1978
[21] Swartzlander E., Gilbert B.: Supersystems — technology and architecture. IEEE Trans. on Computers, Vol. C-31, No. 5, pp. 399—409, 1982.

WACŁAW ISZKOWSKI
Instytut Informatyki
Politechnika Warszawska

Architektury systemów n-mikroprocesorowych

W ślad za intensywnym rozwojem cyfrowych systemów jednoprocesorowych rozwijają się badania i następują praktyczne realizacje systemów cyfrowych, zwanych ogólnie systemami rozproszonymi lub wieloprocesorowymi. Dla uporządkowania pojęć oraz wprowadzenia odpowiedniej terminologii polskiej przedstawimy klasyfikację tych systemów ze względu na struktury ich architektur. Ograniczymy się do systemów składanych najczęściej z mikroprocesorów, wykorzystywanych dotychczas w systemach jednoprocesorowych. Podstawowym kryterium klasyfikacyjnym jest topologia połączeń tych n-mikroprocesorowych układów, wykorzystujących powiązania poprzez wspólną pamięć lub pewną strukturę sieci połączeń.

Klasyfikacja architektur systemów cyfrowych

W tabeli 1 przedstawiono klasyczną już klasyfikację architektur systemów cyfrowych; jej podstawą jest kryterium krotności równocześnie wykonywanych strumieni instrukcji i przetwarzanych przez nich krotnych strumieni danych.

Systemy cyfrowe klasy SISD (Single Instruction, Single Data) umożliwiają wykonywanie pojedynczego strumienia instrukcji dla pojedynczego strumienia danych. W celu zwiększenia szybkości przetwarzania architektury systemów tej klasy modyfikuje się przez wykorzystanie nakładania cykli rozkazowych i potokowego przetwarzania danych. Do tej klasy zalicza się też systemy cyfrowe ze zwielokrotnionymi specjalizowanymi procesorami przechwytującymi wybrane rozkazy do wykonania. Przykładem takiego systemu w kategorii systemów mikroprocesorowych może być układ trzech procesorów z serii Intel, opisany w tabeli 2. Systemów tego typu, pomimo istnienia kilku procesorów, nie będziemy zaliczać do grupy systemów n-procesorowych. W zasadzie wykonywany jest bowiem pojedynczy strumień instrukcji, nawet jeżeli niektóre z ciągów rozkazów mogą być wykonywane równolegle.

Systemy klasy SIMD (Single Instruction, Multiple Data) są systemami o niestandardowych architekturach, umożliwiających równoczesne przetwarzanie wielu strumieni danych, wykonując dla nich ten sam ciąg rozkazów, ewentualnie — lokalnie dla poszczególnych danych — modyfikowanych. Systemy te uzyskują duże szybkości przetwarzania dla specjalnych typów obliczeń. Są one podstawą konstrukcji wielu nowoczesnych superkomputerów.

Podstawą omawianej tutaj klasyfikacji jest klasa MIMD (Multiple Instruction, Multiple Data) obejmująca trzy róż-

ne rodzaje systemów, rozróżnianych przez sposób i stopień powiązania. Pierwotnie do tej klasy zaliczano sieci komputerowe (computer networks) łączące różnego rodzaju komputery liniami telekomunikacyjnymi lub teleinformatycznymi. Cechą charakterystyczną tych sieci jest całkowita niezależność poszczególnych podsystemów, polegająca na autonomicznym sterowaniu i zarządzaniu zasobami w nich zlokalizowanymi. Luźne powiązanie podsystemów wymusza przesyłanie informacji poprzez sieć z wykorzystaniem rozbudowanych technik przesyłania protokołów. Powiązanie komputerów w sieć ma na celu głównie wykorzystywanie w danym podsystemie zasobów rozproszonych w pozostałych, przy pełnej autonomii działania komputerów. Z tego też względu takich sieci nie będziemy uwzględniać w omawianej tutaj klasyfikacji.

Tabela 1. Klasyfikacja Flynna systemów cyfrowych

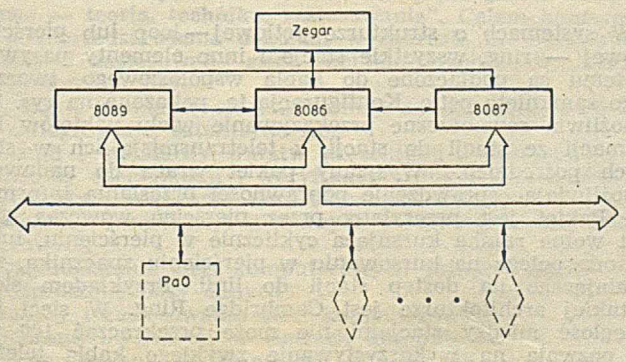
SISD — komputery klasyczne, i z nakładaniem cykli rozkazowych, i z przetwarzaniem potokowym
SIMD — komputery wektorowe, tablicowe, asocjacyjne, luźno związane — sieci komputerowe (podział zasobów)
MIMD — średniozwiązane — sieci lokalne, procesory wielokrotne, ściśle związane — wieloprocesory.
MISD — systemy o zwiększonej niezawodności.
Propozycja polskich oznaczeń klas systemów:
SISD — PIPD (Pojedynczy strumień Instrukcji, Pojedynczy strumień Danych)
SIMD — PIWD (Pojedynczy strumień Instrukcji, Wielokrotny strumień Danych)
MIMD — WIWD (Wielokrotny strumień Instrukcji, Wielokrotny strumień Danych)
MISD — WIPD (Wielokrotny strumień Instrukcji, Pojedynczy strumień Danych)

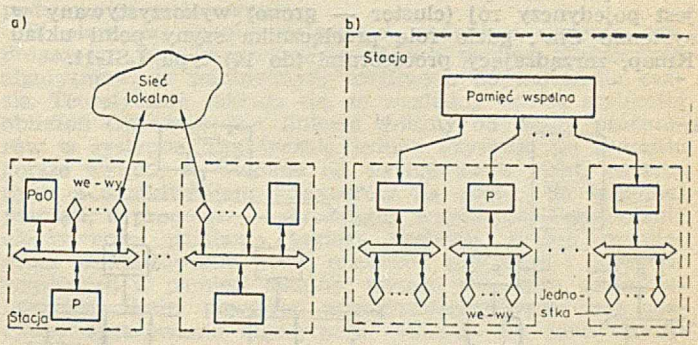
Pozostałe dwa typy powiązań rozróżniają systemy n-procesorowe pod względem metod ich połączeń. Luźne powiązanie (loosely coupled, niekiedy zwane też powiązaniem średnim — moderate coupled, gdy w klasyfikacji uwzględnia się sieci) polega na utworzeniu — przez połączenie procesorów liniami komunikacyjnymi — sieci lokalnych — LAN (Local Area Network) zwanych też procesorami wielokrotnymi (multiple processors). Architektura takiego systemu jest pokazana na rys. 1a. Pojedynczy element sieci, zwany stacją (station), zawiera procesor, moduł pamięci lokalnej oraz układy we-wy powiązane z procesorem przez szynę lokalną. Jeden z układów we-wy (czasem nawet i kilka) jest układem współpracującym z siecią, realizując fizyczne przesyłanie informacji. Niekiedy układy współpracujące są specjalizowanymi procesorami komunikacyjnymi.

Ścisłe powiązanie (tightly coupled) wiąże procesory przez umożliwienie im dostępu do wspólnej (common), współdzielonej (shared), globalnej (global) pamięci systemu. Ogólna architektura takiego systemu, zwanego prawidłowo systemem wieloprocesorowym (multiprocessor — wieloprocesor), jest pokazana na rys. 1b. Pojedynczy procesor z szyną lokalną wiążącą go z układami we-wy oraz ewentualnie modułem pamięci lokalnej, jest zwany jednostką (unit). Cały wieloprocesor, uzupełniony o pasywny procesor komunikacyjny lub układ współpracy z lokalną siecią, może być rozważany jako pojedyncza stacja. Dodatkowo w klasyfikacji systemów n-procesorowych wyróżnia się cechę jednorodności procesorów wchodzących w skład systemu. W systemie homogenicznym wszystkie procesory są tego samego typu, a w systemie heterogenicznym są różne.

Tabela 2. Przykład systemu 3-procesorowego klasy SISD

W systemie cyfrowym 3-procesorowym, zbudowanym z układów serii Intel (rys.) centralny mikroprocesor 8086 współpracuje bezpośrednio z dwoma mikroprocesorami specjalizowanymi. Jeden z nich — mikroprocesor 8089 — jest jednostką obsługi kanału we-wy typu DMA, drugi — 8087 — procesorem arytmetycznym, realizującym operacje zmiennoprzecinkowe. Oba koprocesory korzystają ze strumienia instrukcji i danych pobieranych z pamięci przez procesor centralny, wychwytując rozkazy dla nich przeznaczone. Wszystkie trzy procesory mogą pracować częściowo współbieżnie. Cechą charakterystyczną tego typu systemów jest wspólny układ zegara synchronizującego.





Rys. 1. Systemy n-procesorowe: a) z luźnym powiązaniem, b) ze ścisłym powiązaniem (wieloprocesorowe)

Systemy wieloprocesorowe

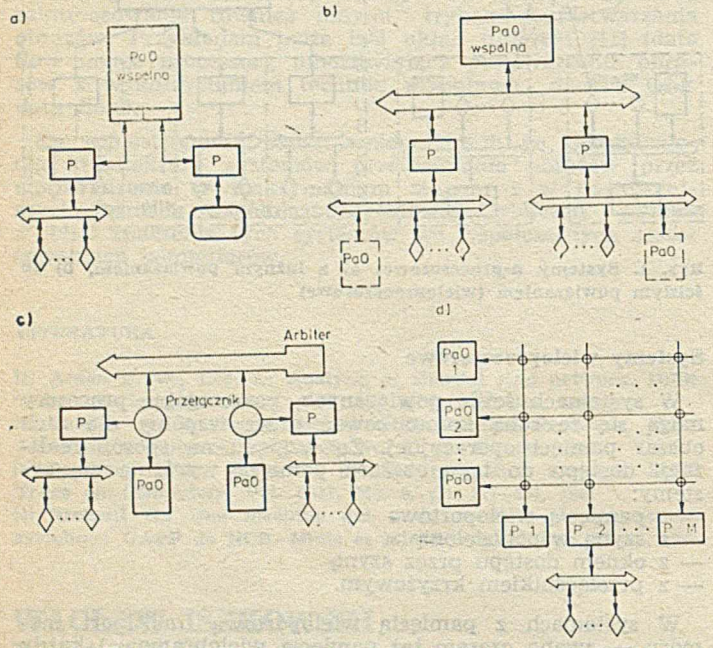
W systemach ściśle powiązanych, poszczególne procesory mogą się ze sobą komunikować przez wspólny dla nich obszar pamięci operacyjnej. Ze względu na sposób realizacji dostępu do tego obszaru pamięci, wyróżnia się systemy:
— z pamięcią wieloportową
— z szyną współdzieloną
— z oknem dostępu przez szynę
— z przełącznikiem krzyżowym.

W systemach z pamięcią wieloportową (multiport memory — zwaną czasem też pamięcią wielobramową) każdy z procesorów ma bezpośredni dostęp do całego obszaru pamięci. Względy technologiczne ograniczają krotność jednoczesnego, wzajemnie się wykluczającego, dostępu — co powoduje, że zastosowania systemów tej klasy są ograniczone. Jednym z przykładów zastosowań są systemy graficzne (VDU — Visual Display Unit), w których jeden z procesorów jest dedykowany do wyłącznej współpracy z ekranem graficznym, a pozostałe przekazują mu przez wspólną pamięć dane do wyświetlania po ich graficznej obróbce. Architekturę systemu z pamięcią wieloportową pokazano na rys. 2a.

Systemy z szyną współdzieloną (shared bus) są najbardziej popularnym rozwiązaniem w tej kategorii. Wiele standardowych realizacji szyny współdzielonej typu Multibus, S-100, Eurobus itp., umożliwia rozłączny przydział szyny żądającym jej procesorom oraz przesyłanie adresów i danych z/do pamięci wspólnej i innych sygnałów sterujących. Algorytmy przydziału szyny są oparte na jednej z zasad: podziału czasu, szeregowego lub równoległego przesyłania żądań szyny, z uwzględnieniem lub nie priorytetów przypisanych procesorom. Arbitraż przydziału szyny jest jej częścią zrealizowaną sprzętowo. Systemy tego typu są budowane ze standardowych układów danej serii systemu (Intel, Motorola, itp.) według określonych w katalogu standardów. W celu zwiększenia ogólnej efektywności działania systemu, każdy z procesorów ma dostęp przez szynę lokalną do własnej pamięci lokalnej oraz układów we-wy. Schemat architektury takiego systemu pokazano na rys. 2b. Szybkości przesyłania danych w szynie są rzędu 5 Msłów/sek. Ze względów technologicznych liczba procesorów podłączonych do szyny nie przekracza kilkunastu. Niekiedy do szyny współdzielonej mogą być podłączone inne niż pamięci wspólne, układy pracujące w trybie podrzędnym (slave).

Systemy z oknem dostępu do pamięci przez szynę (bus window) są funkcjonalnym rozszerzeniem możliwości szyny współdzielonej. Wykorzystując układy analogiczne jak dla zarządzania pamięcią wirtualną (memory management

unit), dodatkowe przełączniki zlokalizowane przy każdym procesorze rozdzielają odwołania do pamięci na: odwołania do pamięci lokalnej danego procesora i odwołania do pamięci lokalnych innych procesorów. Stąd też każdy procesor ma możliwość dostępu do wszystkich obszarów pamięci istniejących w systemie. Niekiedy możliwość ta jest ograniczana tylko do wybranych części tych obszarów. Architekturę takiego systemu pokazano na rys. 2c. W sprzętowym wyposażeniu szyny, zrealizowanej analogicznie jak poprzednio, znajduje się dodatkowy przełącznik, zawierający układy wyznaczania adresata odwołania na podstawie globalnej tabeli stron obszarów wspólnych dla całego systemu. Często przełącznik ten jest układem mikroprogramowanym lub wręcz dodatkowym specjalizowanym mikroprocesorem. Przykładem takiej realizacji jest pojedynczy rój (cluster — grono) wykorzystywany w systemie Cm*, gdzie rolę przełącznika szyny pełni układ Kmap, zarządzający procesorami (do 10) typu LSI-11.



Rys. 2. Systemy wieloprocesorowe: a) z pamięcią wieloportową, b) z szyną współdzieloną, c) z oknem dostępu przez szynę, d) z przełącznikiem krzyżowym

W systemach z przełącznikiem krzyżowym (crossbar switch) pamięć wspólna jest podzielona na moduły (banks) i każdy z procesorów ma możliwość bezpośredniego dostępu do każdego z modułów pamięci. Zależnie od bieżących adresów odwołań możliwy jest dostęp do pamięci kilku procesorów równocześnie. Architektura takiego systemu jest przedstawiona na rys. 2d. Rozwiązanie to jest dość kosztowne. Klasycznym, wielokrotnie opisywanym przykładem systemu, w którym zastosowano takie rozwiązanie, jest system C.mmp., złożony z 16 procesorów typu PDP 11/40 oraz 16 modułów pamięci. W systemie tym dodatkowo istnieje możliwość przekazywania przerwań między procesorami.

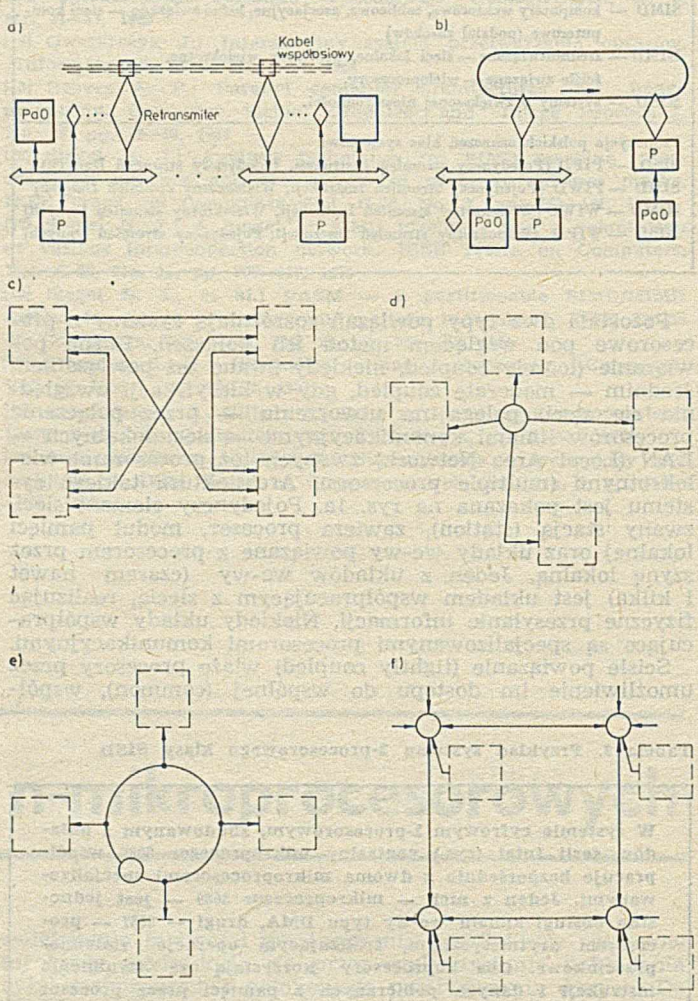
Cechą charakterystyczną wieloprocesorów jest możliwość operowania przez każdy z procesorów na dużych, wspólnych przestrzeniach adresowych pamięci, o pojemności rzędu 1 Mbajtów. Ograniczeniem tych systemów jest występowanie nasycenia szyny przy zbyt dużej liczbie równoczesnych odwołań z różnych procesorów do zasobów wspólnych. Drugim ograniczeniem jest konieczność geograficznego skupienia całego systemu.

Lokalne sieci procesorów wielokrotnych

W systemach luźno powiązanych, do komunikacji międzyprocesorowej wykorzystuje się metody cyfrowego przesyłania informacji w sieciach, tutaj lokalnie wyodrębnionych. Geograficzny zasięg sieci lokalnych zawiera się od 0,1 do 1 km, z wykorzystaniem kabla współosiowego (coaxial cable), przewodów skręconych (twisted pair) lub łączy optycznych czy światłowodowych, o szybkości szeregowego przesyłania informacji rzędu 0,1—10 Mbitów/sek. Ze względu na sposób realizacji konfiguracji sieci lokalnej wyróżnia się systemy:

— z linią wiążącą,
— pętlowe, pierścieniowe,
— o różnej topologii połączeń.

Najprostsza konfiguracja sieci lokalnej jest pojedynczą linią wiążącą (contention bus) kablem współosiowym określoną liczbę stacji. Architektura tego systemu jest pokazana na rys. 3a. Informacje między procesorami są przesyłane jako ramki (frame) zawierające oprócz samych danych, także identyfikator stacji nadającej, odbierającej, rodzaj danych i wartości kontrolne. W wypadku równoczesnego zgłoszenia z wielu stacji, każda z nich jest wstrzymywana na pewien niejednakowy okres, po czym następuje ponowienie zgłoszenia. Najbardziej rozpowszechnionym przykładem takiej sieci jest sieć Ethernet, w której podstawowym elementem jest segment sieci. W pojedynczym segmencie — do kabla o maksymalnej długości do 500 m może być podłączone do 100 stacji. Poszczególne segmenty przez układy retransmisji mogą być ze sobą połączone w sieci typu drzewiastego, mogące zawierać do 1024 stacji z maksymalnym rozproszeniem geograficznym do 2,5 km.



Rys. 3. Konfiguracje sieci lokalnej: a) pojedyncza linia wiążąca, b) pętla, c) pełne powiązanie, d) gwiazda z centralnym przełącznikiem, e) pętla z przełącznikiem, f) struktura regularna

W systemach o strukturze pętlowej — loop lub pierścieniowej — ring, wszystkie stacje i inne elementy pasywne systemu są podłączone do kabla współosiowego, tworzącego zamkniętą pętlę. Konfiguracja ta, pokazana na rys. 3b, umożliwia równoczesne przekazywanie wielu pakietów informacji ze stacji do stacji, z teletransmisją ich w stacjach pośrednich. Wysyłany pakiet wraca do nadawcy, umożliwiając sprawdzenie poprawności przesłania informacji. Pakiet jest przesyłany przez pierścień wówczas, gdy jest wolna ramka kursująca cyklicznie w pierścieniu. Inna metoda polega na kursowaniu w pierścieniu znacznika, zezwalającego na dostęp stacji do linii. Przykładem sieci o takiej architekturze jest Cambridge Ring. W sieci tej odległość między stacjami nie może przekraczać 100 m, co pozwala na wykorzystywanie zwykłego kabla telefonicznego.

Ostatni typ konfiguracji sieci lokalnej obejmuje różne topologie połączeń. Wyróżnia się tutaj: pełne połączenie (fully connected), w których każdy z procesorów jest połączony z każdym innym (rys. 3c); układ gwiazdy z centralnym przełącznikiem (star); pętlę również z centralnym przełącznikiem; sieć o strukturze regularnej (regular; rys. 3f) oraz o strukturze nieregularnej. Pokazane topologie połączeń procesorów są wykorzystywane jako całość systemu lub też stanowią podstawę do tworzenia różnego rodzaju sieci o strukturach drzewiastych, kaskadowych i hierarchicznych. Należy jednak stwierdzić, że większość tych rodzajów sieci znajduje się dopiero w fazie prób i badań modelowych.

Tabela 3. Klasyfikacja Andersena/Jensena topologii połączeń międzyprocesorowych

Strategia transferu	Sterowanie transferem	Ścieżka transferu	Topologia połączeń
Bezpośrednia		Ścieżka dedykowana	Pętla (rys. 3b)
			Pełne powiązanie (rys. 3c)
		Ścieżka współdzielona	Wspólna pamięć (rys. 2a)
			Wspólna szyna (rys. 2b)
Pośrednia	Zcentralizowane	Ścieżka dedykowana	Gwiazda (rys. 3d)
			Pętla z przełącznikiem (rys. 3e)
		Ścieżka współdzielona	Szyna z przełącznikiem (rys. 2c)
	Zdecentralizowane	Ścieżka dedykowana	Sieć regularna (rys. 3f)
			Sieć nieregularna
		Ścieżka współdzielona	Okno dostępu (rys. 2c)

Jednym z najczęściej przytaczanych w literaturze przykładów taksonomii topologii połączeń systemów n-procesorowych (ściśle i luźno powiązanych) jest taksonomia Andersena-Jensena pokazana w tabeli 3.

Systemy rozproszone

Omówione dwa rodzaje systemów n-procesorowych określa się wspólnym terminem **system rozproszony** (distributed system). Termin ten opisuje podstawową cechę funkcjonalną tego rodzaju systemów, jaką jest logiczna jednorodność oprogramowania systemu, a fizycznie rozmieszczona przypisana poszczególnym procesorom. Rozmieszczenie logiczne zwartego oprogramowania złożonego z systemu operacyjnego oraz programu użytkowego może być statyczne lub dynamiczne. W pierwszym wypadku każdy istniejący w oprogramowaniu proces ma na stałe przypisany jeden z procesorów, na którym jest wykonywany. Rozmieszczania dynamicznego dokonuje się przez rekonfigurację logiczną systemu. Rekonfiguracja logiczna może wynikać z konieczności dokonania rekonfiguracji fizycznej systemu cyfrowego (zmiana liczby procesorów) lub też z chęci uzyskania lepszego współczynnika wykorzystania procesorów Podstawą klasyfikacji systemów rozproszonych jest przeznaczenie systemu, determinujące sposób realizacji i strukturę oprogramowania, oraz kryteria oceny efektywności działania sytemu n-procesorowego. Wyróżnia się tutaj dwa podstawowe zastosowania.

W pierwszej grupie zastosowań, zadaniem systemu rozproszonego jest przetwarzanie danych według określonych algorytmów, w minimalnym możliwym do uzyskania czasie. Teoretycznie zakłada się, że względny wzrost szybkości obliczeń (speedup) jest liniowo zależny od liczby procesorów w systemie. Praktycznie jednak uzyskuje się znacznie gorsze wyniki, ze względu na występowanie strat, związanych z oczekiwaniem procesorów na dostęp do wspólnej pamięci, a procesów — do danych wspólnych. Poprawienie efektywności działania takiego systemu, można uzyskać przez zlokalizowanie kodu rozkazów procesów i procedur wspólnych w pamięci lokalnej danego procesora, zminimalizowanie liczby powiązań międzyprocesorowych oraz specjalne opracowania algorytmów obliczeń dla danej konfiguracji.

Druga grupa zastosowań obejmuje ogólnie rozumiane systemy pracujące w trybie czasu rzeczywistego. W tym wypadku system rozproszony musi jedynie spełnić warunek nieprzekraczania określonego czasu reakcji, na każde ze zdarzeń odebranych przez system. Rozproszenie systemu ma na celu dopasowanie go do rozproszenia elementów sterowanego obiektu oraz lokalnej obsługi układów we-wy. Również i w tym wypadku należy dążyć do minimalizacji wzajemnych powiązań międzyprocesorowych oraz lokalizacji kodu procesów w pamięciach lokalnych procesorów.

Obecne oprogramowanie systemów rozproszonych jest jeszcze na etapie badań modelowych. Pojedyncze konfiguracje z unikalnymi systemami zarządzającymi są w większym stopniu wykorzystywane do zbierania doświadczeń niż pełnej eksploatacji.

LITERATURA

[1] Cellary W.: Systemy wieloprocesorowe. Materiały Jesiennej Szkoły PTI, Rydzyna 1984

[2] Jones A. K., Schwarz P.: Experience using multiprocessor systems — A status report: ACM Computing Surveys, Vol. 12, No. 2, 1980

[3] Distributed Systems-Architecture and Implementation. Ed. Lampson S. W., Paul M., Siegert J., LCNS 105, Springer Verlag, 1981

4 Прондишвили И. В., Подлазов В. С., Сшецюра Ф.; Покальные процессорные вычислительные сети, Издательство Наука, 1984.

Konferencje

Sieci komputerowe

Centrum Obliczeniowe Politechniki Wrocławskiej i Instytut Komputerowych Systemów Automatyki i Pomiarów organizują w dniach 23—25 września 1987 roku we Wrocławiu konferencję naukowo-techniczną pn. **„Sieci komputerowe — teoria, technika, zastosowania"**. Celem konferencji jest przedstawienie dokonań w zakresie projektowania, budowy i zastosowań sieci komputerowych w kraju. Grupy tematyczne:

- usługi i standardy sieci transmisji danych (sieci publiczne, zintegrowane, satelitarne itp.)
- usługi i standardy warstw prezentacji i aplikacji
- metody specyfikacji i weryfikacji usług i protokołów
- projekty i realizacje sieci transmisji danych
- projekty i realizacje globalnych i lokalnych sieci komputerowych
- współdziałanie sieci komputerowych różnych typów
- rozproszone bazy danych
- zasoby i zastosowania sieci komputerowych
- ochrona informacji w sieciach komputerowych.

Zainteresowani udziałem w konferencji proszeni są o przesłanie zgłoszenia uczestnictwa w terminie do 15 listopada 1986 pod adresem organizatorów. Autorzy referatów proszeni są o równoczesne przesłanie wraz z kartą uczestnictwa pełnego tekstu referatu (nie przekraczającego 5 stron znormalizowanego maszynopisu) w tym samym terminie. Po zaakceptowaniu referatów przez Komitet Programowy autorzy otrzymają potwierdzenie uczestnictwa, matryce oraz szczegółowe informacje dotyczące warunków uczestnictwa w konferencji.

Na konferencji będą wygłoszone referaty programowe przez zaproszonych specjalistów i referaty uczestników (czas trwania 15 minut). Referaty zostaną wydane przed konferencją w postaci materiałów konferencyjnych. Ze względu na ograniczoną liczbę miejsc, przy akceptacji uczestnictwa będzie uwzględniana kolejność zgłoszeń.

Terminarz

- zgłoszenie uczestnictwa i przesłanie pełnego tekstu referatu do 15 listopada 1986
- potwierdzenie przez organizatorów uczestnictwa i akceptacja referatów do 15 stycznia 1987
- nadesłanie pełnego tekstu referatu na matrycach do 15 lutego 1987
- wniesienie opłaty za udział w konferencji do 31 marca 1987
- obrady konferencji 23—25 września 1987

Adres sekretariatu:
Centrum Obliczeniowe Politechniki Wrocławskiej
Pl. Grunwaldzki 9, 50-372 Wrocław
Telefony: 21-10-18, 20-34-31, 20-35-16.

LESŁAW BUDZIANOWSKI
JERZY WIETRZYCH
Instytut Cybernetyki Technicznej
Politechnika Wrocławska

JAN KWIATKOWSKI
Centrum Obliczeniowe
Politechnika Wrocławska

Komputer ODRA 1305 w Międzyuczelnianej Sieci Komputerowej

Międzyuczelniana Sieć Komputerowa MSK składa się z szeregu sprzężonych ze sobą systemów cyfrowych. W sieci wyróżnia się trzy podstawowe elementy — podsieć komunikacyjną, dołączone do niej komputery obliczeniowe oraz koncentratory terminali. Budowa wewnętrzna i własności podsieci komunikacyjnej zostały przedstawione w artykule [3]. W niniejszym artykule omówiony jest sposób dołączenia do podsieci komputera obliczeniowego typu ODRA 1305 i związanych z nim terminali.

Autorzy przedstawionego rozwiązania przystępując do jego realizacji przyjęli kilka założeń. Najważniejszym z nich było zachowanie i udostępnienie użytkownikom sieci komputerowej całości istniejącego już oprogramowania komputera ODRA 1305, a w szczególności systemu operacyjnego GEORGE 3.

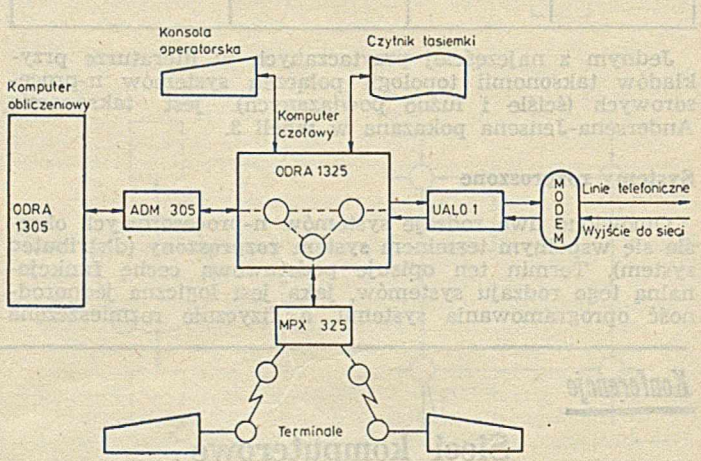
Dla nowego oprogramowania spełniającego funkcje sieciowe założono, że w pierwszej kolejności będzie tworzone oprogramowanie uzupełniające istniejący system operacyjny. Ze względu na eksperymentalny — w dużej mierze — charakter przedsięwzięcia, starano się tak konstruować to oprogramowanie, aby umożliwić przyszłą modernizację i rozwój sieci, a równocześnie dostarczyć użytkownikom uniwersalnych narzędzi programowych, pozwalających na tworzenie ich własnych sieciowych systemów użytkowych.

WŁĄCZENIE KOMPUTERA DO SIECI

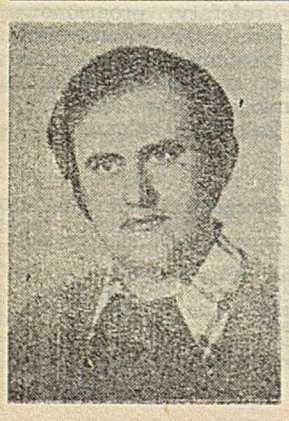
W celu włączenia komputera ODRA 1305 do MSK zastosowano środki typu sprzętowego, programowego oraz dodatkowe ustalenia organizacyjne. Przyjęto przy tym, że najważniejsze funkcje przystosowania systemu do współpracy w sieci będą realizowane poza komputerem obliczeniowym, tak aby nie zwiększać bez potrzeby jego obciążenia. W tym celu między komputer ODRA 1305 a podsieć wprowadzono komputer czołowy zrealizowany na ODRZE 1325. Pełni on funkcję koncentratora terminali dla współpracy z lokalnym komputerem obliczeniowym i z systemami dostępnymi przez podsieć oraz funkcję konwertera protokołów współpracy, obowiązujących w MSK i lokalnie w systemie ODRA. Taki zakres usług, udostępniony przez komputer czołowy, nie zaspokaja wszystkich potrzeb użytkowników i dlatego też przewidziano uzupełnienie oprogramowania komputera obliczeniowego o system transferu zbiorów i zadań.

Konfiguracja sprzętowa omawianego rozwiązania (przedstawiona na rys. 1) zawiera — obok komputerów ODRA 1305 i ODRA 1325 — adapter międzykanałowy ADM 305, multiplekser MPX 325, modemy, terminale DZM 180 KSRE oraz adapter liniowy UAL 01. Wymieniono tu tylko elementy charakterystyczne dla sieci MSK. Oba komputery współpracują także z pamięciami zewnętrznymi, drukarkami, czytnikami kart, tasiemkami itp. Adapter liniowy UAL 01 został specjalnie zaprojektowany i wykonany dla potrzeb MSK przez Zakład Aparatury Naukowej i Doświadczalnej Politechniki Wrocławskiej. Jego podstawowym celem jest umożliwienie współpracy z podsiecią komunikacyjną przy użyciu protokołu liniowego LAPB, należącego do grupy protokołów zorientowanych bitowo. Inne urządzenia są typowe dla maszyn serii ODRA i nie wymagają odrębnego przedstawienia.

Oprogramowanie komputera czołowego realizuje tworzenie, utrzymywanie i likwidowanie połączeń między trzema dostępnymi rodzajami wyjść: do komputera obliczeniowego,



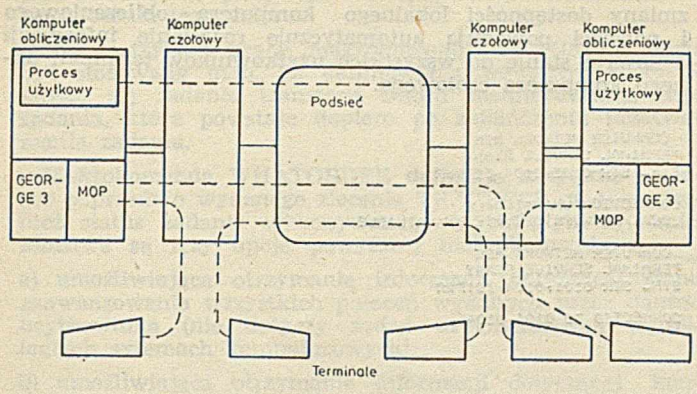
Rys. 1. Konfiguracje sprzęgowe i drogi przesyłania wewnątrz komputera czołowego



Mgr inż. LESŁAW BUDZIANOWSKI ukończył w 1974 r. studia na Wydziale Elektroniki Politechniki Wrocławskiej. Pracuje w Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej. Zajmuje się problematyką dotyczącą sieci komputerowych, a w szczególności badaniem i standaryzacją protokołów sieciowych oraz ich implementacją w sieci MSK.

Mgr inż. JERZY WIETRZYCH studia ukończył na Wydziale Elektroniki Politechniki Wrocławskiej. Od 1976 r. pracuje w Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej na stanowisku specjalisty projektanta systemów operacyjnych. Zajmuje się budową oprogramowania systemów teletransmisji danych.

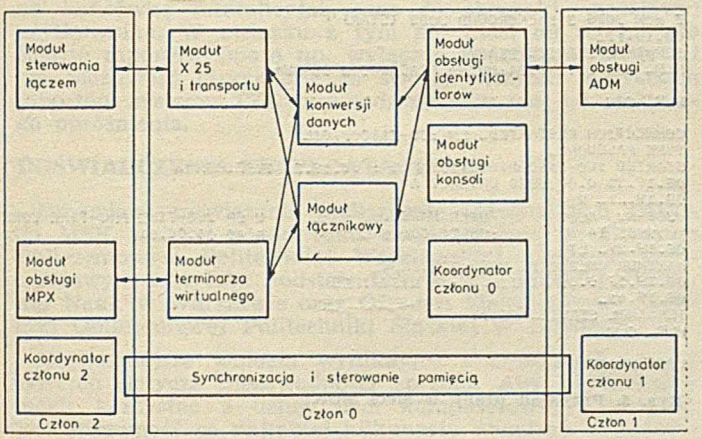
do sieci i na terminale. W praktyce umożliwia to tworzenie połączeń terminali z systemami, terminali między sobą oraz programów pracujących w komputerach obliczeniowych. We wszystkich tych wypadkach współpraca jest możliwa zarówno lokalnie, jak i poprzez podsieć (rys. 2).



Rys. 2. Rodzaje realizowanych połączeń

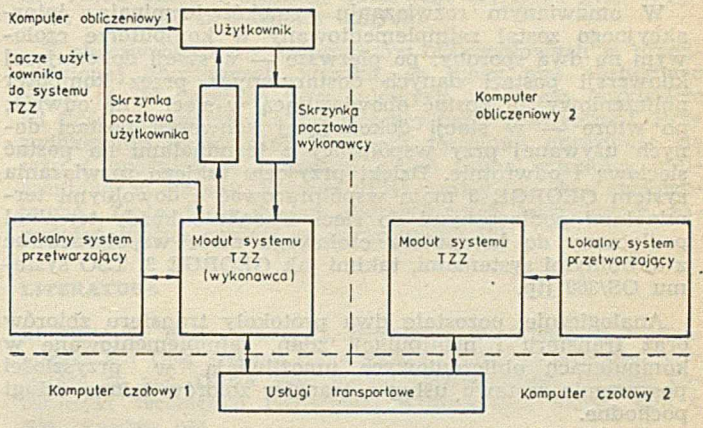
Ogólną strukturę oprogramowania komputera czołowego przedstawiono na rysunku 3. Pokazany tam zestaw modułów programowych obsługuje konfigurację ze wszystkimi trzema rodzajami wyjść. Przy użyciu tych samych modułów można tworzyć oprogramowanie konfiguracji pozbawionych jednego z rodzajów wyjść. Wykorzystuje się wtedy tylko część potencjalnych funkcji systemu, np. funkcję komputera czołowego do komputera obliczeniowego lub koncentratora terminali do współpracy z siecią.

W założeniach dotyczących funkcjonowania MSK jest także udostępnienie użytkownikom usługi transferu zbiorów i zadań między komputerami obliczeniowymi. Jakkolwiek ta część oprogramowania jest jeszcze w stadium testowania, pewne cechy i własności systemu można już podać.



Rys. 3. Moduły funkcjonalne systemu i ich powiązania z modułami jądra i członami programu

System Transferu Zbiorów i Zadań (TZZ) składa się z modułów umieszczonych w poszczególnych systemach. Do wykonania zlecenia potrzebna jest współpraca dwóch modułów, zlokalizowanych w różnych systemach komputerowych, komunikujących się ze sobą za pośrednictwem połączenia zrealizowanego za pomocą usług transportowych. Swoje polecenia użytkownik kieruje do modułu znajdującego się w komputerze, z którym bezpośrednio współpracuje. Moduł ten staje się wykonawcą zlecenia, steruje jego wykonaniem i przekazuje użytkownikom wszystkie informacje z tym związane. Ogólną strukturę systemu TZZ



Rys. 4. Struktura systemu Transferu Zbiorów i Zadań

przedstawiono na rysunku 4.

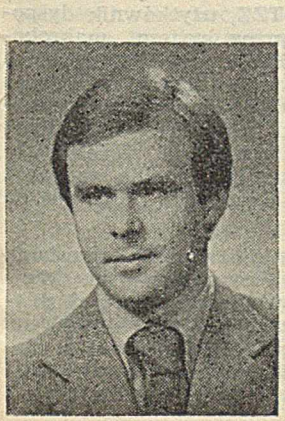
W wydzielonych kartotekach systemu operacyjnego gromadzone są informacje o zauważonych niesprawnościach oprogramowania i sprzętu, co ma znaczenie dla konserwatorów systemu. Prawo do umieszczenia tych wiadomości w systemie ma każdy użytkownik. Informacje o nowych usługach lub zmianach w ich świadczeniu są rozsyłane do użytkowników w chwili ustanowienia połączenia, za pomocą mechanizmu dostępnego w systemie GEORGE 3. Bieżące komunikaty o zdarzeniach w systemie są rozsyłane do współpracujących terminali przy użyciu komunikatu BROADCAST przez operatora komputera obliczeniowego. Docierają one wtedy do terminali, z którymi utrzymuje on połączenie logiczne. Analogicznie jest możliwe rozsyłanie wiadomości przez operatora komputera czołowego do wszystkich terminali z nim związanych. W relacjach między ośrodkami obliczeniowymi podłączonymi do MSK, informacje można przesyłać nawiązując połączenie między terminalami.

PROTOKOŁY WSPÓŁPRACY W MSK

Protokoły współpracy obowiązujące w MSK funkcjonują w sposób warstwowy, co wynika z przyjęcia jako nadrzędnej koncepcji rozwiązania modelu odniesienia ISO-OSI [5] Technicznymi środkami umożliwiającymi tworzenie połączeń są linie telefoniczne, zakończone modemami dupleksowymi, wyposażonymi w styk S2, zgodny z zaleceniem V24 CCITT. Przy użyciu tych środków, w warstwie łączy fizycznych jest realizowany sprzęg zgodny z zaleceniem X.21bis CCITT [11]. Jego realizatorem od strony komputera ODRA jest adapter typu UAL 01. W warstwie łączy logicznych zastosowano protokół liniowy LAPB, realizowany przez oprogramowanie komputera czołowego. To samo oprogramowanie obsługuje także współpracę z węzłami podsieci przy użyciu sprzęgu zgodnego z zaleceniem X.25, co jest funkcją warstwy łączy sieciowych. Pewne charakterystyczne dla wymienionych protokołów cechy przedstawiono w artykule [3].

Ponad warstwą łączy sieciowych funkcjonuje warstwa transportowa, w której zastosowano protokół transportowy zgodny z normą ISO [2, 6]. Zadaniem jego jest eliminacja błędów wynikających z chwilowych niesprawności podsieci komunikacyjnej. Stacje końcowe tego protokołu (tj. realizujące je oprogramowanie) umieszczone są także w komputerach czołowych.

W modelu odniesienia ISO, ponad warstwami o charakterze komunikacyjnym, określono jeszcze warstwy o charakterze funkcjonalnym. Są to odpowiednio: warstwa sesji, prezentacji i aplikacji. W chwili, gdy przystępowano do budowy MSK, nie istniały propozycje ISO czy CCITT, dotyczące protokołów dla tych warstw o własnościach nadających się do implementacji. Prace nad odpowiednimi standardami międzynarodowymi nie są zresztą do dzisiaj



Dr inż. JAN KWIATKOWSKI ukończył w 1977 r. Wydział Elektroniki Politechniki Wrocławskiej. Następnie był słuchaczem studium doktoranckiego przy Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej. Od 1980 roku jest zatrudniony na stanowisku adiunkta w Centrum Obliczeniowym Politechniki Wrocławskiej. Zajmuje się problemami przetwarzania równoległego oraz oprogramowaniem sieci komputerowych.

zakończone. Dlatego też na potrzeby MSK opracowano trzy protokoły funkcjonalne obejmujące łącznie warstwę sesji i prezentacji. Są to odpowiednio: protokół terminala interakcyjnego [2, 10], protokół transferu zbiorów [9] oraz protokół transferu i manipulacji zadaniami [4]. Protokoły te zostały określone w sposób umożliwiający budowę sieci o charakterze heterogenicznym, tzn. zawierającej komputery różnych typów (ODRA, RIAD itp.) oraz różne typy terminali (DZM 180 KSRE, dalekopisy, monitory ekranowe).

W omawianym rozwiązaniu protokół terminala interakcyjnego został zaimplementowany w komputerze czołowym na dwa sposoby: po pierwsze — w stacji dokonującej konwersji postaci danych dostarczanych przez komputer obliczeniowy na postać obowiązującą w sieci i na odwrót; po wtóre — w stacji dokonującej konwersji postaci danych używanej przy współpracy z terminalami na postać sieciową i odwrotnie. Dzięki przyjęciu takiego rozwiązania system GEORGE 3 może współpracować z dowolnymi terminalami podłączonymi do sieci. Podobnie każdy terminal podłączony do komputera czołowego może współpracować z dowolnymi systemami, takimi jak GEORGE 3, TSO systemu OS/360 itp.

Analogicznie, pozostałe dwa protokoły transferu zbiorów oraz transferu i manipulacji zdań, zaimplementowane w komputerach obliczeniowych umożliwiają w przyszłości poszerzenie zestawu usług o transfer zbiorów i inne usługi pochodne.

JĘZYK DOSTĘPU UŻYTKOWNIKÓW DO USŁUG SIECI

Użytkownicy terminali dołączonych do komputera czołowego mogą za pomocą specjalnego języka poleceń ustanawiać połączenia z wybranym komputerem obliczeniowym, ustalać parametry współpracy oraz likwidować połączenia. W odpowiedzi na wprowadzane polecenia, komputer czołowy udziela odpowiedzi informującej o zmianach stanów współpracy.

Połączenia tworzy się za pomocą polecenia STR, którego parametrami są: adres systemu docelowego (komputera obliczeniowego wraz z komputerem czołowym), adres procesu użytkowego wewnątrz systemu oraz nazwa użytkownika inicjującego tworzenie połączenia. W odpowiedzi komputer czołowy wyprowadza komunikat CONNECTED TO XXXX, YYYY, gdy połączenie ustanowiono lub komunikat CONNECTION UNAVAILABLE, gdy nie udało się utworzyć połączenia. W tym ostatnim wypadku jest wyprowadzany dodatkowy komunikat, określający przyczynę odrzucenia połączenia. Po utworzeniu połączenia, współpraca między terminalem a procesem użytkowym (np. GEORGE 3) odbywa się zgodnie z zasadami określanymi przez wybrany proces użytkowy, to jest w jego języku poleceń.

Do zmiany parametrów terminala bądź likwidacji połączenia użytkownik terminala musi zastosować specjalne polecenie DLE, powodujące zawieszenie pracy z procesem użytkowym. Od tego momentu wprowadzane polecenia są interpretowane przez komputer czołowy. W tym kontekście można używać poleceń: PAR, CON, CLR lub MSG. Polecenie PAR zmienia parametry terminala (np. szerokość wiersza), polecenie CON natomiast powoduje wznowienie pracy z procesem użytkowym. Polecenie CLR służy do likwidacji zawieszonego połączenia. Po jego zastosowaniu i zakończeniu zamykania połączenia, komputer czołowy wyprowadza własny komunikat potwierdzający zamknięcie.

Na podobnej zasadzie jak połączenie terminali z systemami można także tworzyć połączenia między terminalami. Reguły współpracy po ustanowieniu takiego połączenia ustalają między sobą użytkownicy terminali.

Między terminalami związanymi z danym komputerem czołowym można także wymieniać dane w trybie bezpołączeniowym. Służy do tego polecenie MSG, które przekazuje pojedynczy komunikat pod wskazany adres. Komunikat ten będzie dostarczony do odbiorcy niezależnie od tego, że może on być w danej chwili zaangażowany we współpracę z innym systemem. Przykład współpracy w przedstawionym języku poleceń pokazano na rys. 5.

W celu usprawnienia obsługi użytkowników komputer czołowy wyposażono w konsolę operatorską, z której można wprowadzać polecenia sterujące pracą systemu. Umożliwiają one zamykanie istniejących połączeń, wyjść do podsieci lub lokalnego komputera obliczeniowego, a także zawiadamianie użytkowników o zachodzących zdarzeniach. Działania takie operator może podejmować w sytuacjach awaryjnych. W stanie normalnej pracy konsola operatorska służy do monitorowania pracy systemu, co może się odbywać w sposób ciągły lub przez polecenia pytania o stan. Niezależnie od funkcjonowania procesu operatorskiego, zmiany dostępności lokalnego komputera obliczeniowego i podsieci powodują automatycznie rozesłanie informacji o nowym stanie do wszystkich użytkowników terminali danego komputera czołowego.

```
 COMPUTER NETWORK MSK
TERMINAL SERVICE READY
STR- STR 0103,0104,ICTPWR

CONNECTION UNAVAILABLE
LOCAL/NETWORK SERVICE UNAVAILABLE

 COMPUTER NETWORK MSK
TERMINAL SERVICE READY
STR- STR 0101,0104,ICTPWR

CONNECTED TO 0101,0104
```

```
POLITECHNIKA WROCLAWSKA G-3/B.65 13MAR85
08.17.29- WA
PROPERTY NAME EXCL? PERM? CONSOLE?
CENTRAL                Y
W120                   Y
W160                   Y
NIEMA          Y       Y
TESTOWANE      Y       Y
APIN                   Y
PAPIER2X       Y
TASIEMKA       Y
PAPIER3X       Y
PAPIER5X       Y
WSCAL          Y
W6CAL          Y
NIESPRAWNA     Y       Y

PAPIER4X       Y
W161                   Y
SDI            Y
CAFE           Y       Y
S71            Y       Y       Y
U13            Y
PS             Y
08.18.07- WJ

3 MOP JOBS 3 BACKGROUND JOBS (STAR)
08.18.13-
TERMINAL SERVICE READY
HLP- MSG 0101,0104,MOZNA PRACOWAC W CO-PWR

HLP- CON

CONNECTION RESTARTED- LN :CO-Z1600B,GFL
TYPE PASSWORD-
STARTED :CO-Z1600B,GIL,13MAR85 : 08.19.21 TYPE:MOP
08.19.21 0.00 USED URGENCY A
08.19.21- WE
 JOB-NO  MOP?       USER NAME.JOB NAME     U CP JOB-TIME PRU-TIME CORE
   9985 I4         :CO-Z1600B.GNL          A 40 00.00.01
08.19.29- LT
MAXIMUM ONLINE FS USED 1 KWORDS
08.19.37 0.01 FINISHED : 0 LISTFILES
08.19.41-
TERMINAL SERVICE READY
HLP- CLR

LOGGED OFF
```

Rys. 5. Przykład pracy w sieci MSK

Komunikacja użytkownika z systemem TZZ została włączona do Języka Opisu Zadań (JOZ) systemu GEORGE 3. Umożliwia ona użytkownikowi przekazywanie dyrektyw do systemu TZZ za pomocą makrokomend JOZ, niezależnie od trybu wykonania zadania (interakcyjnie, wsadowo) oraz otrzymywanie informacji od systemu TZZ w sposób analogiczny do innych informacji z systemu operacyjnego.

Definiując zlecenie dla systemu TZZ, użytkownik dysponuje czterema interpretowanymi przez system makrokomendami: TRANSFER, WHATORDER, MODIFYORDER i MAIL. Mogą one być wydawane w dowolnym czasie, niezależnie od siebie i niezależnie od wykonywanych transferów. Każda makrokomenda zawiera wiele opcjonalnych pozycji, których kombinacja stanowi treść zlecenia. Ze szczegółowymi formatami makrokomend można zapoznać się z dokumentu specyfikującego łącze użytkownika do systemu TZZ [8].

Makrokomenda TRANSFER definiuje zlecenie przesłania zbioru między dwoma systemami komputerowymi. Miejscem przeznaczenia zbioru może być lokalna pamięć zbiorów (transfer zbiorów), zdalne urządzenie zewnętrzne, np. drukarka (transfer dokumentów) lub lokalna kolejka zleceń systemu (transfer zadań). Użytkownik dysponuje dwoma trybami wykonania każdego ze zleceń. Są to:

Transfer bezwarunkowy, stosowany do przesyłania zbiorów istniejących w momencie wydania zlecenia.

Transfer oczekujący — przed rozpoczęciem wykonywania zlecenia jest sprawdzany warunek istnienia zbioru; fakt jego nieistnienia powoduje odłożenie transferu do momentu założenia tego zbioru.

Drugi z trybów pracy makrokomendy TRANSFER może być stosowany m.in. do definiowania wewnątrz wykonującego się zadania transferu zbioru monitorowania, tego zadania, które powstaje dopiero po zakończeniu przetwarzania zadania.

Makrokomenda WHATORDER definiuje zapytanie o status poprzednio wydanego zlecenia TRANSFER, w tym również status zadania wykonywanego w odległym systemie. Możliwe są trzy opcje powyższej makrokomendy:

a) umożliwiająca otrzymanie informacji dotyczącej stanu zaawansowania wszystkich poleceń wydanych przez danego użytkownika (nie dotyczy zadań uruchamianych w odległych sytemach komputerowych)

b) umożliwiająca otrzymanie informacji dotyczącej konkretnego polecenia na poziomie lokalnego systemu TZZ (stosowana dla makrokomend TRANSFER wykorzystywanych do przesłania zbioru)

c) umożliwiająca otrzymanie informacji o stanie zaawansowania konkretnego zlecenia (zadania uruchamianego w odległym systemie komputerowym) na poziomie lokalnego i odległego systemu TZZ.

Trzecia z makrokomend — MODIFYORDER — służy do skasowania uprzednio wydanej makrokomendy TRANSFER. Podobnie jak w wypadku makrokomendy WHATORDER polecenie może być wykonywane na poziomie lokalnego (w wypadku transferu zbiorów) bądź lokalnego i odległego (w wypadku transferu zadań) systemu TZZ.

Ostatnia makrokomenda — MAIL — służy do otrzymywania zawartości skrzynki pocztowej użytkownika. Generowane przez system TZZ raporty, dotyczące postępu w realizacji zlecenia, są w trakcie jego realizacji wyprowadzane na końcówkę interakcyjną oraz do skrzynki pocztowej użytkownika. W związku z tym nie musi on na bieżąco śledzić raportów (może np. wyłączyć końcówkę), ponieważ ma możliwość otrzymania ze swojej skrzynki wszystkich raportów systemu TZZ zgromadzonych w niej od ostatniego opróżnienia.

DOŚWIADCZENIA EKSPLOATACYJNE

Omawiane rozwiązanie podłączenia maszyn typu ODRA do MSK jest od ponad roku eksploatowane w: Centrum Obliczeniowym Politechniki Wrocławskiej, Centrum Obliczeniowym Instytutu Podstaw Informatyki Polskiej Akademii Nauk w Warszawie oraz Ośrodku Elektronicznej Techniki Obliczeniowej Politechniki Śląskiej w Gliwicach.

Najważniejsze wnioski wynikające z obserwacji pracy systemu dotyczą niezawodności sprzętu. Aby użytkownicy mogli korzystać z usług sieci komputerowej wymagana jest sprawność co najmniej kilkunastu urządzeń równocześnie. Wprowadzanie komputera czołowego na drodze połączeń terminalowych przy dostępie do lokalnego komputera obliczeniowego spowodowało pewne pogorszenie niezawodności tego dostępu.

Liczba awarii tego komputera jest jednak stosunkowo niewielka, co sprawiło, że systemy są eksploatowane w omawianej konfiguracji niezależnie od tego, czy wykorzystuje się pracę sieciową, czy też nie (to znaczy czy uruchomiona jest podsieć i możliwość tworzenia przez nią połączeń). Równocześnie wprowadzenie komputera czołowego do konfiguracji daje użytkownikowi możliwość tworzenia połączeń sieciowych do innych komputerów obliczeniowych, przy awarii komputera lokalnego.

Z obserwacji pracy sieciowej wynika, że możliwe jest uzyskiwanie połączeń przez sieć o czasie trwania rzędu kilku godzin. Liczba przerw w utrzymywaniu połączeń jest jednak wyraźnie większa niż przy pracy lokalnej. Wydaje się, że pewien wzrost komfortu pracy można uzyskać przez udostępnienie możliwości tworzenia dróg zastępczych w podsieci oraz przez wprowadzenie szybkiej, aktualnej informacji o stanie podłączonych do niej systemów. To ostatnie wymaga jednak rozszerzenia zestawu usług, dostarczonych przez podsieć, o możliwość przesyłania datagramów w sensie zalecenia X.25. Umożliwiłoby to przesyłanie informacji o stanie elementów składowych sieci bez potrzeby nawiązywania połączeń.

Podczas pracy przy użyciu terminali typu DZM 180 KSR nie zauważono istotnego wzrostu opóźnień w przesyłaniu danych spowodowanych realizowaniem połączeń przez komputery czołowe i sieć. Zastosowane mechanizmy sterowania przepływem pozwalają na dobre wykorzystanie własności podsieci, przy czym z badań szczegółowych wynika, że jej przepustowość jest większa niż sumaryczne obciążenie wnoszone przez terminale, które mogą być obsługiwane przez poszczególne komputery obliczeniowe. Ewentualne opóźnienia rzędu około 1 sekundy obserwuje się tylko przy zmianach kierunku przesyłania danych.

Z obserwacji pracy komputerów obliczeniowych w przedstawionej konfiguracji wynika, że mogą one obsługiwać około 15—18 połączeń terminalowych, co zależy oczywiście od ilości dostępnej pamięci operacyjnej i zewnętrznej. Przy zwiększeniu liczby obsługiwanych terminali obserwuje się wzrost opóźnień w realizacji zadań przez komputer obliczeniowy w stopniu, którego użytkownicy nie chcą zaakceptować. Zestawienie badań szczegółowych powyższych zagadnień można znaleźć w opracowaniu [7].

LITERATURA

[1] Bilski E., Budzianowski L., Muehleisen T., Wietrzych J., Żabnieński W.: Opis systemu operacyjnego sieciowego procesora czołowego do m.c. ODRA 1305. Wersja heterogeniczna. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Raport SPR 4, 1984, Wrocław 1984

[2] Budzianowski L., Muehleisen T., Wietrzych J.: Protokoły wyższych warstw sieci do obsługi terminalów w MSK, Przegląd Telekomunikacyjny, nr 11, 1983

[3] Bieleninik E., Hudyma E., Kosmulska-Bochenek E., Lewoc J., Stanisz A.: Podsieć Komunikacyjna Międzyuczelnianej Sieci Komputerowej, Informatyka nr 11—12, 1985

[4] Huzar Z., Kwiatkowski J.: Protokół transferu i manipulacji zadaniami dla sieci MSK, Centrum Obliczeniowe Politechniki Wrocławskiej. Raport SPR 16/84, Wrocław 1984

[5] ISO/DIS 7498 — Data Processing — Open System Interconnection Basic Reference Model, as October 1981

[6] ISO/DIS 8073. Transport Protocol Specification — Draft Proposal as of March 1983

[7] Janczewski K.: Badania eksperymentalne i pomiary pilotowej MSK. Centrum Obliczeniowe Politechniki Wrocławskiej, Raport SPR 44, 1984, Wrocław 1984

[8] Kaszuba R., Kwiatkowski J., Skoczylas Ł.: Usługi transferu zbiorów i zadań w KO ODRA 1305. Centrum Obliczeniowe Politechniki Wrocławskiej, Raport SPR 26, 1984, Wrocław 1984

[9] Kaszuba R.: Protokół transferu zbiorów dla MSK. Centrum Obliczeniowe Politechniki Wrocławskiej. Raport SPR 16, 1983, Wrocław 1983

[10] Wietrzych J.: Usługi i protokół wirtualnego terminala interakcyjnego w MSK. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Raport SPR 3, 1983, Wrocław 1983

[11] YELLOW BOOK, Data communication networks services and facilities, terminal equipment and interfaces, Recommendation X.25, Vol. VIII — Fascicle VIII.2 CCITT, Geneva 1981.

ZBYSZKO KRÓLIKOWSKI
Instytut Automatyki
Politechnika Poznańska

Generowanie planów wykonania transakcji w systemie rozproszonych baz danych (2)

Przegląd metod

W pierwszej części artykułu omówiono podstawowe pojęcia i założenia przyjmowane w badaniach nad problematyką optymalizacji planów wykonywania transakcji w systemach rozproszonych baz danych (SRBD). Obecnie zostanie dokonany przegląd dotychczas opracowanych algorytmów generowania tych planów.

Problem opracowania efektywnego algorytmu generowania planów wykonywania transakcji jest jednym z najtrudniejszych i najbardziej złożonych problemów, stojących przed projektantem systemu zarządzania rozproszoną bazą danych (SZRBD). Złożoność tego problemu wynika przede wszystkim z dwóch faktów:

- **możliwości równoległego wykonywania transakcji w SRBD**
- **konieczności transmisji danych między stanowiskami komputerowymi SRBD.**

Problem generowania planów wykonywania transakcji należy do klasy problemów silnie NP-zupełnych (por. m.in. [1, 23, 29]). Nawet w wypadku przyjęcia silnych założeń upraszczających, takich jak założenie o zaniedbywalności kosztów przetwarzania danych lub założenie o jednorodnym rozkładzie wartości atrybutów relacji, algorytmy generowania planów wykonywania transakcji są algorytmami heurystycznymi.

KLASYFIKACJA ALGORYTMÓW

Znane w literaturze algorytmy generowania planów wykonywania transakcji w SRBD są trudno porównywalne, ze względu na różnorodność przyjmowanych założeń dotyczących modelu rozproszonej bazy danych (RBD) i sieci komunikacyjnych, jak również ze względu na założenia o informacji dostępnej podczas generowania planu, a dotyczącej stanu RBD i aktualnego stanu wykonywanej transakcji. Czynnik drugi, tj. zakres i dokładność dostępnej informacji o stanie bazy danych, ma istotny wpływ na efektywność algorytmów generowania planów wykonywania transakcji w SRBD.

Zdecydowana większość algorytmów generowania planów wykonywania transakcji w SRBD, należy do grupy **algorytmów statycznych** [1, 4, 5, 7—10, 12—15, 18—21, 23—26, 29, 30, 35, 38], które generują plany wykonywania transakcji podczas ich kompilacji. Statyczna metoda generowania planu wykonywania transakcji wymaga informacji statystycznych o zawartości bazy danych. Informacje te mogą nie być precyzyjne, a więc mogą powodować błędy w oszacowaniu sprawności generowanych planów. Algorytmy statyczne cechują się małymi narzutami czasowymi związanymi z ich wykonywaniem.

W algorytmach dynamicznych poszczególne kroki planu wykonywania transakcji generuje się podczas wykonywania transakcji. Umożliwia to optymalizację powstającego planu, przy wykorzystaniu precyzyjnej informacji o aktualnym stanie bazy danych i stanie wykonywanej transakcji. Okazuje się jednak, iż koszt dynamicznego zbierania tych informacji jest tak duży, że przewyższa potencjalny zysk wynikający ze zwiększenia precyzji uzyskiwanej informacji [27] (po każdym kroku transakcji jej wykonanie musi być przerywane, w celu zebrania informacji potrzebnych do określenia optymalnego kroku następnego). Dotychczas opracowano tylko kilka algorytmów dynamicznych [3, 31, 32, 34].

Algorytmy generowania planów wykonywania transakcji można również sklasyfikować ze względu na klasy transakcji, których dotyczą (por. cz. 1 artykułu). Wyróżnić więc można **algorytmy dla transakcji ogólnych** [1, 4, 5, 10, 13, 21, 29, 30, 35], **drzewowych** [8, 16, 17, 38], **cyklicznych** [24] i **łańcuchowych** [9]. Wiele opracowanych algorytmów dotyczy SRBD, wykorzystujących sieci komputerowe o specyficznej architekturze, na przykład: **gwiazdowych** [7, 25] lub **magistralowych** [20, 22, 31, 32, 34].

Zdecydowana większość proponowanych w literaturze metod ogranicza obszar poszukiwania planów wykonywania transakcji, przez przyjęcie założeń upraszczających. Uzasadnienie takiego podejścia podano w [5], na przykładzie stosunkowo prostego przypadku transakcji wymagającej wykonania operacji połączenia trzech relacji na dwóch atrybutach połączeniowych. Założono, że transakcja będzie realizowana z wykorzystaniem techniki operacji półpołączenia. Bez przyjmowania dodatkowych założeń upraszczających, dla znalezienia optymalnego planu wykonywania tej transakcji należałoby rozpatrzyć około 1,3 miliarda unikalnych sekwencji operacji półpołączenia. Działanie takiego algorytmu byłoby oczywiście nieefektywne.

W konsekwencji omówionego — w pierwszej części artykułu — założenia o dominacji kosztów transmisji w ogólnych kosztach wykonywania transakcji w SRBD, zdecydowana większość opracowanych dotychczas algorytmów generowania planów wykonywania transakcji z SRBD koncentruje się wyłącznie na redukcji woluminów danych przesyłanych między stanowiskami SRBD i wykorzystuje w tym celu strategię operacji półpołączenia.

Algorytmami o fundamentalnym znaczeniu w problematyce optymalizacji planów wykonywania transakcji w SRBD są: algorytm stosowany w systemie rozproszonej bazy danych SDD-1 (produkt Computer Corporation of America [4]), algorytm Apears'a, Hevnera i Yao, nazywany dalej algorytmem AHY [1], oraz jego odmiana zaproponowana przez Perrizo — algorytm S [30]. Algorytmy te dotyczą klas transakcji ogólnych, co wydatnie zwiększa ich wartość w porównaniu z innymi znanymi z literatury algorytmami, które dotyczą szczególnych klas transakcji (por. część 1 artykułu), i które w wielu wypadkach są przystosowaniem jednego z algorytmów SDD-1 lub AHY do specyfiki szczególnej klasy transakcji.

Algorytm SDD-1

W algorytmie SDD-1 wykorzystano wyniki prac E. Wonga, dokonując modyfikacji i rozszerzenia algorytmu zaproponowanego w [35]. Użytkownicy systemu rozproszonej bazy danych SDD-1 kontaktują się z systemem za pomocą proceduralnego języka wysokiego poziomu Datalanguage.

Tabela 1. Realizacja transakcji w systemie SDD-1

1. Przekształcenie transakcji T zapisanej w Datalanguage do postaci transakcji wyrażonej w algebrze relacyjnej, przy jednoczesnym określeniu lokalizacji relacji, wyszczególnionych w Q(T) na stanowiskach komputerowych SRBD.
2. Konstrukcja programu B stanowiącego sekwencję operacji półpołączenia oraz wybór stanowiska komputerowego, na którym po wykonaniu programu P znajdują się dane o największym rozmiarze.
3. Przesłanie danych wymaganych do wykonania transakcji na stanowisko wybrane w kroku 2; wykonanie operacji końcowych na tym stanowisku komputerowym i przesłanie ostatecznego wyniku na stanowisko wynikowe.

Realizacja transakcji w systemie SDD-1 odbywa się w trzech krokach [4], przedstawionych w tabeli 1.

Kluczowym punktem tego schematu jest punkt 2, tj. konstrukcja programu P. W celu wyboru odpowiedniego wariantu programu P szacuje się koszty, efekty i zyski dla poszczególnych operacji półpołączenia, tworzących dany wariant. **Koszt** operacji półpołączenia $R_i \triangleright\!< R_j$ jest zdefiniowany jako rozmiar danych (size) przesyłanych, w celu wykonania tej operacji między stanowiskami komputerowymi ST_i i ST_j, tzn. jest to size $(R_i.X)$. **Efekt** wykonania operacji półpołączenia $R_j \triangleright\!< R_i$ jest rozumiany jako rozmiar relacji R_j po wykonaniu tej operacji. **Zysk** z wykonania operacji półpołączenia $R_j \triangleright\!< R_i$ jest rozumiany jako rozmiar danych wyeliminowanych z bazy danych, tzn.: size $(R_j)_{przed \triangleright\!<}$ — size $(R_j)_{po \triangleright\!<}$. Do oszacowania efektów operacji półpołączenia wykorzystuje się teorię współczynnika selektywności, przy czym zakłada się, że atrybuty niepołączeniowe są redukowane według aproksymacji funkcji Yao [37].

Algorytm SDD-1 jest przedstawiony w tabeli 2.

Tabela 2. Algorytm SDD-1

1. Wykonaj wszystkie operacje lokalne.
2. Dla zbioru wszystkich możliwych operacji półpołączenia dla danej transakcji T wykonaj następujące kroki:
(a) oszacuj koszty, efekty i zyski operacji półpołączenia,
(b) wybierz operację półpołączenia, która daje największy zysk,
(c) uaktualnij dane potrzebne do oszacowania kosztów, efektów i zysków operacji półpołączenia w następnej iteracji,
(d) ze zbioru badanych operacji wyeliminuj operację wybraną w punkcie (b),
(e) powtarzaj od kroku (a) aż zyski ze wszystkich operacji półpołączenia będą równe zeru lub zbiór badanych operacji półpołączenia będzie zbiorem pustym.
3. Wybierz stanowisko komputerowe, na którym znajdują się dane o największym rozmiarze i na to stanowisko prześlij dane wyszczególnione w specyfikacji relacji wynikowej; wykonaj operacje końcowe i prześlij ostateczny wynik na stanowisko wynikowe.

Złożoność obliczeniowa algorytmu SDD-1 wynosi:

$$O_{SDD-1}(m) = 2*NAP*(m-1)$$

gdzie: **NAP** jest liczbą atrybutów połączeniowych transakcji **T**, **m** — liczbą relacji wyszczególnionych w kwalifikacji transakcji **T**.

Algorytm AHY

Podstawową ideą algorytmu AHY jest dekompozycja transakcji złożonej do zbioru podtransakcji prostych, których plany wykonywania — jak wykazano w [1, 21] — można łatwo optymalizować. Algorytm AHY polega na generowaniu planów przetwarzania każdej relacji osobno, przy czym przez plan przetwarzania relacji rozumie się sekwencję operacji półpołączenia użytych do redukcji tej relacji. Plan wykonywania transakcji zawiera plany dla wszystkich relacji, które są wyszczególnione w kwalifikacji transakcji. Algorytm AHY [1] jest podany w tabeli 3.

Ze względu na obszerność i dużą złożoność, uniemożliwiającą skrótowe przedstawienie poszczególnych procedur algorytmu AHY, ograniczymy się tylko do omówienia procedury RESPONSE, dla której autorzy algorytmu AHY wykazali optymalność generowanych planów wykonywania transakcji. Do twierdzenia o optymalności algorytmu AHY (RESPONSE) i specyficznych założeń metody AHY, ustosunkowano się w [6, 28].

Procedura RESPONSE wykorzystywana w kroku 3 algorytmu AHY jest przedstawiona w tabeli 4.

Algorytm AHY wykorzystuje procedury PARALLEL i SERIAL [21], w celu minimalizacji odpowiednio czasu odpowiedzi i sumarycznego obciążenia systemu dla kandydujących planów. Procedury te przedstawiono w tabelach 5 i 6.

Złożoność obliczeniowa algorytmu AHY wynosi:

$$O_{AHY}(m) = NAP*m^2*\log_2 NAP$$

gdzie: **m** jest liczbą relacji wyszczególnionych w kwalifikacji transakcji **T**, **NAP** — liczbą atrybutów połączeniowych transakcji **T**.

Tabela 3. Algorytm AHY

1. Wykonaj wszystkie operacje lokalne.
2. Wygeneruj kandydujące plany dla poszczególnych relacji:
- wyizoluj każdy atrybut, na którym jest wykonywana operacja połączenia i rozważ każdy z nich dla zdefiniowania podtransakcji prostych z nieokreślonym stanowiskiem wynikowym,
- w celu zminimalizowania czasu odpowiedzi zastosuj procedurę PARALLEL; dla każdej podtransakcji prostej zachowaj wszystkie kandydujące plany w celu ich integracji w kroku 3,
- w celu zminimalizowania sumarycznego obciążenia systemu zastosuj procedurę SERIAL dla każdej podtransakcji prostej,
- z uzyskanych planów dla każdej podtransakcji prostej utwórz kandydujące plany dla każdego atrybutu, na którym jest wykonywane połączenie.
3. Dokonaj integracji kandydujących planów; dla każdej relacji wyszczególnionej w kwalifikacji transakcji kandydujące plany są integrowane do formy planu przetwarzania tej relacji. W zależności od kryterium optymalizacji, integracja jest wykonywana przez procedury RESPONSE oraz TOTAL lub CELLECTIVE.
4. Usuń plany nadmiarowe, tzn. te plany przetwarzania relacji, których transmisje są uwzględnione w planach dla innej relacji.

Tabela 4. Procedura RESPONSE

1. Uporządkowanie kandydujących planów.
Dla każdej relacji R_i uporządkuj według rosnących czasów kandydujące plany dla atrybutu połączeniowego d_{ij} ($j = 1,2,...,NAP_i$ jest liczbą atrybutów połączniowych relacji R_i).
2. Integracja planów.
Z kandydujących planów, uporządkowanych według rosnących czasów, skonstruuj zintegrowane plany dla relacji R_i, które zawierają równoległe transmisje wszystkich uporządkowanych kandydujących planów. Wybierz plan z minimalnym czasem odpowiedzi.

Tabela 5. Procedura PARALLEL

1. Uporządkuj relacje R_i wyszczególnione w kwalifikacji transakcji, tak aby size $(R_1) \leqslant$ size $(R_2) \leqslant ... \leqslant$ size (R_n).
2. Rozważ każdą z relacji R_i w kolejności ich rosnących rozmiarów. Dla każdej relacji R_i skonstruuj plan, który zawiera równoległe transmisje planów dla relacji R_j (gdzie $j < i$); wybierz plan z minimalnym czasem odpowiedzi.

Tabela 6. Procedura SERIAL

1. Uporządkuj relacje R_i wyszczególnione w kwalifikacji transakcji, tak aby size $(R_1) \leqslant$ size $(R_2) \leqslant ... \leqslant$ size (R_n).
2. Jeśli na stanowisku wynikowym nie ma relacji, wówczas wybierz strategię:
$R_1 \rightarrow R_2 \rightarrow ... \rightarrow R_n \rightarrow$ stanowisko wynikowe.
3. Jeśli R_r jest relacją na stanowisku wynikowym, wówczas spośród dwóch strategii:
$R_1 \rightarrow R_2 \rightarrow ... \rightarrow R_r \rightarrow ... \rightarrow R_n \rightarrow R_r$ lub
$R_1 \rightarrow R_2 \rightarrow ... \rightarrow R_{r-1} \rightarrow R_{r+1} \rightarrow ... \rightarrow R_n \rightarrow R_r$
wybierz tę, która daje minimum sumarycznego obciążenia systemu.

Algorytm S

Przedstawiony w [30] algorytm S jest odmianą algorytmu AHY. Istotną zmianą w stosunku do algorytmu AHY jest odrzucenie założenia o nieredukowalności atrybutów połączeniowych. Algorytm S do konstrukcji planu dla danej relacji wykorzystuje efekty wykonania operacji półpołączenia zarówno na atrybutach połączeniowych jak i niepołączeniowych. Szacowanie rozmiarów redukcji relacji i atrybutów po wykonaniu operacji półpołączenia jest dokonywane na podstawie rozwiniętej teorii współczynnika selektywności [30].

Z takich samych względów jak w wypadku algorytmu AHY, ograniczymy się do przedstawienia algorytmu S w wersji RESPONSE (tab. 7).

Tabela 7. Algorytm S

1. Konstrukcja kandydujących planów dla poszczególnych atrybutów połączeniowych.
(a) wykonaj sortowanie atrybutów połączeniowych według ich rosnących rozmiarów,
(b) dla wszystkich kolejnych atrybutów połączeniowych wykonaj następujące czynności:
— weź kolejny (pierwszy) atrybut,
— dla tego atrybutu skonstruuj plany, w których uwzględnij plany wybrane dla atrybutów poprzedzających go, biorąc pod uwagę równoległe i szeregowe wykonywanie operacji półpołączenia,
— wybierz plan z minimalnym czasem odpowiedzi,
— wykonaj sortowanie skonstruowanych planów według rosnących czasów odpowiedzi.
2. Dla każdej relacji R_i wyszczególnionej w kwalifikacji danej transakcji skonstruuj zintegrowane plany przetwarzania, w których uwzględnij równoległe wykonywanie skonstruowanych i uporządkowanych w kroku 1 planów kandydujących dla atrybutów.
3. Ze skonstruowanych w ten sposób planów przetwarzania dla relacji usuń operacje nadmiarowe.

Jak widać, idea algorytmu S jest identyczna z ideą algorytmu AHY. Jednak wprowadzenie do konstrukcji planów operacji półpołączenia na atrybutach niepołączeniowych oraz wielokrotne sortowanie planów według czasu odpowiedzi umożliwiło uzyskanie wyników lepszych nawet o 50% w stosunku do planów generowanych przez algorytm AHY.

Złożoność obliczeniowa algorytmu S wynosi:

$$O_s(m) = m^{2*}\left(\sum_{i=1}^{m} NAP_i + 3\right)$$

gdzie: **m** jest liczbą relacji wyszczególnionych w kwalifikacji transakcji **T**, NAP_i — jest liczbą atrybutów połączeniowych w kwalifikacji transakcji **T**.

Inne algorytmy

Oprócz omówionych metod, w literaturze znanych jest wiele innych algorytmów generowania planów wykonywania transakcji w SRBD. Algorytmy te mają jednak dużo mniejsze znaczenie dla omawianej problematyki, ponieważ albo dotyczą tylko wybranych klas transakcji, albo są odmianą algorytmów SDD-1 i AHY lub też zakładają pewne dodatkowe uproszczenia w środowisku SRBD. Tak więc algorytmy te mają znacznie mniejsze wartości aplikacyjne niż omówione wcześniej metody SDD-2, AHY i S.

Spośród algorytmów, które są odmianami metod SDD-1 i AHY, na szczególną uwagę zasługuje algorytm zaproponowany w [5, 29]. Jest on rozwiązaniem pośrednim między starą wersją algorytmu SDD-1 z 1979 roku a algorytmem G przedstawionym przez Hevnera i Yao w [21]. Istotną nowością w stosunku do tych wersji algorytmów SDD-1 i AHY jest to, że wyboru operacji półpołączenia, które mają być włączone do programu redukcyjnego, dokonuje się na podstawie dwóch warunków:

1 — koszt operacji półpołączenia jest minimalny,
2 — po wykonaniu operacji półpołączenia atrybut połączeniowy w redukowanej relacji ma najmniejszą liczność.

Dla klasy transakcji prostych znalezienie operacji półpołączenia, spełniającej jednocześnie oba kryteria, jest możliwe. Nie jest to jednak możliwe dla transakcji złożonych. Black i Luk proponują wówczas formułę pośrednią między warunkami 1 i 2, tzn. podstawą wyboru operacji półpołączenia jest wówczas minimum sumy składników występujących w warunkach 1 i 2. Dla klasy transakcji prostych, na podstawie badań symulacyjnych [5], stwierdzono przewagę tego algorytmu nad wersją algorytmu SDD-1 z 1979 roku i algorytmu G-Hevnera i Yao również z 1979 roku [21]. Dla transakcji o wysokiej złożoności algorytmu Blacka i Luka staje się jednak nieefektywny. W odniesieniu do nowych wersji algorytmów SDD-1 i AHY, badań porównawczych nie przeprowadzono.

W [11] zaproponowano algorytm D, będący odmianą algorytmu AHY. W algorytmie tym odrzucono założenie o nieredukowalności atrybutów niepołączeniowych. Algorytm ten ma już tylko znaczenie historyczne, bowiem Perrizo w [30] zaproponował jego ulepszenie i rozwinięcie.

W literaturze zaproponowano również wiele algorytmów generowania planów wykonywania wybranych klas transakcji. Algorytmy te nie są zdolne do generowania dobrych planów dla transakcji ogólnych. W [24] zaproponowano algorytm generowania planów dla klasy transakcji cyklicznych, w [9] — dla klasy transakcji łańcuchowych, a w [38] — dla klasy transakcji drzewowych. Wszystkie te algorytmy, w poszukiwaniu optymalnego planu wykonywania danego typu transakcji, stosują technikę zbliżoną do zaproponowanej w algorytmie SDD-1.

Wiele prac dotyczy optymalizacji planów wykonywania transakcji w takich SRBD, w których używane są sieci komunikacyjne o specyficznej konfiguracji albo występują dodatkowe ograniczenia.

W [10] zaproponowano algorytm generowania planów wykonywania transakcji, minimalizujący koszty transmisji i przetwarzania w funkcji lokalizacji stanowisk komputerowych i sekwencji wykonywanych operacji relacyjnych. Dana transakcja jest dekomponowana do sekwencji operacji relacyjnych. Bazując na powiązaniach między tymi operacjami oraz na ich przechodniości, generuje się dla danej transakcji wiele wstępnych równoważnych drzew transakcji. Następnie, na podstawie wyprowadzonych w [10] twierdzeń ogranicza się obszar poszukiwań rozwiązania optymalnego.

Podstawową wadą tej metody jest jej niewielomianowa złożoność obliczeniowa, co czyni tę metodę nieefektywną obliczeniowo i bardzo trudną w implementacji. W modelu kosztów wykorzystywanym w tej metodzie zakłada się — między innymi — znajomość tzw. funkcji redukcyjnych danych. Funkcje te są trudne do określenia dla poszczególnych kroków planu wykonywania transakcji, a uzyskiwanie ich wartości przez wykonywanie pomiarów symulacyjnych może być powodem nieprzewidzianych błędów i utraty optymalności. W algorytmie tym zakłada się, że na stanowiskach komputerowych SRBD znajdują się komputery tego samego typu oraz identyczne systemy zarządzania z lokalną bazą danych. Ograniczenie stosowalności tego algorytmu tylko do klasy homogenicznych SRBD, degraduje wartość tego algorytmu, wobec obowiązujących tendencji projektowania heterogenicznych SRBD.

W [15] zaprezentowano algorytm generowania planów wykonywania transakcji, stosowany w rozproszonej wersji systemu INGRES. Jest ona rozwinięciem idei dekompozycji zapytań przedstawionej w [36]. Przy wyborze planu wykonywania transakcji stosuje się kryterium minimalizacji transmisji w sieci komunikacyjnej, tzn. dokonuje się wyboru stanowisk komputerowych, na których ma być wykonane przetwarzanie, by przesyłane woluminy danych były jak najmniejsze.

W systemie R [2, 13] problem optymalizacji planów wykonywania transakcji został sprowadzony do wyboru najtańszego algorytmu wykonywania operacji połączenia. W [7, 25] zaproponowano algorytm generowania planów wykonywania transakcji w homogenicznych systemach rozproszonych baz danych typu gwiaździstego, a w [14] — dla SRBD typu CODASYL.

Najnowsze badania w omawianej problematyce dotyczą optymalizacji planów wykonywania transakcji w SRBD implementowanych w lokalnych sieciach komputerowych [22, 31, 34]. Istotna różnica (w porównaniu z przedstawio-

nymi pracami) sprowadza się do nieco innego przedstawienia funkcji kosztów transmisji danych między stanowiskami systemu rozproszonego. Funkcja ta została dostosowana do specyfiki poszczególnych typów sieci lokalnych [22]. Nadal jednak podstawą algorytmów generowania planów wykonywania transakcji jest wykorzystywanie operacji relacyjnej półpołączenia.

Na zakończenie warto wspomnieć o pewnych algorytmach, w których odrzucono część powszechnie akceptowanych założeń. W [33, 39] zaproponowano pewne algorytmy generowania planów wykonywania transakcji w SRBD przy założeniu, że istnieją redundantne kopie relacji na różnych stanowiskach komputerowych SRBD oraz przy założeniu, że relacje mogą być podzielone na mniejsze fragmenty. Prace te mają jednak charakter wstępny i jak na razie nie mają większego znaczenia praktycznego.

LITERATURA

[1] Apers P. M., Hevner A. R., Yao S. B.: Optimization algorithms for distributed queries. IEEE Trans. on Software Engineering, Vol. SE-9, No. 1, pp. 57—67, January 1983

[2] Astrahan M., Kim W., Schkolnick M.: Evaluation of the system R access path selection mechanism. Research Report RJ 2798, IBM Research Laboratory, San Jose, 1980

[3] Baldissera C., Bracchi G., Ceri S.: A query processing strategy for distributed data bases. Proc. EURO-IFIP 79, pp. 667—678, 1979

[4] Bernstein P. A. et al.: Query processing in a system for distributed data bases (SDD-1). ACM Trans. on Database Systems, Vol. 6, No. 4, 1981, pp. 602—625

[5] Black P. A., Luk W. S.: A new heuristic for generating semi-joins programs for distributed query processing. Proc. Conf. IEEE COMPSAC, pp. 581—588, 1981

[6] Cellary W., Królikowski Z., Morzy T.: Others comments on „Optimization algorithms for distributed queries". IEEE Trans. on Software Engineering, 1985 (przygotowano do druku)

[7] Chen A. L. P., Li V. O. K.: Optimizing Star queries in distributed database system. Proc. 10th Intern. Conf. on VLDB, pp. 344—347, 1984

[8] Chen A. L. P. Li V. O. K.: Improvement algorithms for semi-join query processing programs in distributed database systems. IEEE Trans. on Computers, Vol. C-33, No. 11, 1984

[9] Chiu Dah-Ming, Bernstein P. A., Ho Yu-Chi: Optimizing chain queries in distributed database systems. SIAM J. Comput., Vol. 13, No. 1, pp. 116—134, 1984

[10] Chu W. W., Hurley P.: Optimal query processing for distributed database systems. IEEE Trans. on Computers, Vol. C-31, No. 9, pp. 835—850, 1982

[11] Cheung T. Y.: A method for equijoin queries in distributed relational databases. IEEE Trans. Comput., Vol. C-31, 1982

[12] Chung Ch. W., Irani K. G.: A semi-join strategy for distributed query optimization. Proc. 4th Intern. Conf. on Distributed Computing, pp. 368—377, San Francisco, 1984

[13] Daniels D.: Query compilation in a distributed database system. IBM Research Report RJ 8423, IBM Research Laboratories, San Jose, 1982

[14] Dayal U., Goodman N.: Query optimization for Codasyl Database Systems, pp. 138—150, Proc. ACM-SIGMOD Conf. on Management of Data, Orlando, 1982

[15] Epstein R., Stonebraker M. R., Wong E.: Distributed query processing in a relational data base system, pp. 169—180, Proc. ACM-SIGMOD Intern. Conf. on Management of Data, Austin, 1978

[16] Goodman N., Shmueli O.: The tree property is fundamental for query processing, pp. 40—48, Proc. ACM-SIGACT-SIGMOD Conf. on Principles of Database Systems, Los Angeles, March, 1982

[17] Goodman N., Shmueli O.: Transforming cyclic schemas into trees, pp. 49—54, Proc. ACM-SIGACT-SIGMOD Conf. on Principles of Database Systems, Los Angeles, 1982

[18] Goodman N., Shmueli O.: The tree projection theorem and relational query processing. Journal of Computer and System Sciences, Vol. 28, No. 1, pp. 60—79, 1984

[19] Goldhirsch D., Yedwab L.: Processing read only queries — over views with generalization, pp. 344—347, Proc. 10th Intern. Conf. on VLDB, 1984

[20] Gouda M., Dayal U.: Optimal semi-join schedules for query processing in local distributed database systems, pp. 164—175, Proc. ACM-SIGMOD Intern. Conf. on Management of Data, Ann Arbor, New York, 1981

[21] Hevner A. R., Yao S. B.: Query processing in distributed database systems. IEEE Trans, on Software Engineering, Vol. SE-5, No. 5, pp. 177—187, 1979

[22] Hevner A. R., Wu Q., Yao S. B.: Query optimization on local area networks. ACM Trans. on Office Information, Vol. 3, No. 1, 1985

[23] Huang T. K.: A cost evaluation model for processing distributed query. Proc. 18th Hawaii International Conf. on System Sciences, 1985

[24] Kambayashi Y. et al.: Query processing for distributed database using generalized semi-joins, pp. 151—160, Proc. ACM-SIGMOD Intern. Conf. on Management of Data, Orlando, 1982

[25] Kerschberg L., Ting P. D., Yao S. B.: Query optimization in star computer networks. ACM Trans. on Database Systems, Vol. 7, No. 4, pp. 678—711, 1982

[26] Krishnamurthy R., Morgan S. P.: Distributed query optimization — on engineering approach. Intern. Conf. on Data Engineering, Los Angeles, 1984

[27] Królikowski Z.: Some Remarks on Query Processing in Distributed Data Base Management Systems. 8th International Seminar on DBMS, Piestany, Czechoslovakia, 1985

[28] Lakhani G. D., Wang J. S.: Comments on „Optimization Algorithms for Distributed Queries", IEEE Trans. on Software Engineering, Vol. SE-10, No. 4, 1984

[29] Luk W. S., Luk L.: Optimizing semi-join programe for distributed query processing, pp. 298—316, Proc. Conf. ICOD-2, Cambridge, 1983

[30] Perrizo W.: A method for processing distributed database queries. IEEE Trans on Software Engineering, Vol. SE-10, Nr. 4, pp. 466—470, 1984

[31] Sacco G. M.: Distributed query evaluation in local area networks. Intern. Conf. on Data Engineering, Los Angeles, 1984

[32] Toan N. G.: Distributed query management for a local network database system. Proc. 2nd Intern. Conf. on Distributed Computing Systems. Paris, 1981

[33] Vard Y. L., Vrbsky V.: Distributed query processing allowing for redundant data, pp. 389—396, Proc. 4th Intern. Conf. on Distributed Computing, 1984

[34] Wah B. W., Went Y. N.: The File-assinnment and query processing problems in local multiaccess networks. Intern. Conf. on Data Engineering, Los Angeles, 1984

[35] Wong E.: Retieving dispersed data from SDD-1 — a system for distributed data bases, pp. 217—235, Proc. 2nd Berkeley Conf. on Ditributed Data Management and Computer Networks, 1977

[36] Wong E., Youssefi K.: Decomposition — a strategy for query processing. ACM Trans. on Database Systems, Vol. 1, No. 3, pp. 223—241, 1976

[37] Yao S. B.: Approximating block accesses in database organizations. CACM, Vol. 20, No. 4, pp. 260—261, 1977

[38] Yu C. T., Lam K., Ozsoyoglu M.: Distributed query optimization for tree queries. Technical Report, Dept. of Information Engineering, University of Illinois at Chicago Circle, pp. 1—55, 1980

[39] Yu C. T., Chang C. C.: On the design of a query processing strategy in a distributed database environment, pp. 30—39, Proc. SIGMOD 83 Conf., New York, 1983.

JANUSZ STOKŁOSA
Politechnika Poznańska

Z dziejów sterowania

Sterowanie jest oddziaływaniem na układ z zamiarem osiągnięcia pożądanego jego zachowania, określonego celem, który chce się uzyskać. Zmiany zachowania układu są osiągane przez oddziaływanie na niego urządzeniem sterującym. Jeśli rolę urządzenia sterującego pełni maszyna [1]), to mamy do czynienia ze sterowaniem automatycznym.

Początki sterowania automatycznego sięgają później starożytności i są związane z imieniem Herona z Aleksandrii (I w.n.e.), autora „Automatów" (albo „Teatru automatycznego"). Jednakże systematyczny rozwój urządzeń sterujących został zapoczątkowany w średniowieczu — epoce, która stworzyła nie tylko wspaniałe kościoły romańskie i katedry gotyckie, ale w której wynaleziono lub odkryto na nowo zegar mechaniczny, wiatrak, młyn wodny, proch strzelniczy, soczewkę, busolę i inne urządzenia stanowiące cenne składniki kultury materialnej.

Kołowe zegary mechaniczne, które pojawiły się w XIII wieku, są uważane za jeden z najdonioślejszych wynalazków technicznych średniowiecza. Wykonywano je przeważnie jako zegary wieżowe, katedralne lub pałacowe. W konstrukcjach tych wykorzystywano fakt, że pod wpływem przywiązanego do linki ciężarka następuje ruch wałka, na którym jest nawinięta linka. Wał obracałby się nie ruchem jednostajnym, lecz jednostajnie przyspieszonym, gdyby zegar nie był wyposażony w kolebnik — specjalny regulator służący do utrzymywania równomiernego obrotu koła zapadkowego [17]. Dokładność takich zegarów nie była jednak zbyt duża. Pierwszym krokiem w konstrukcji dokładnych zegarów było włączenie do ich mechanizmu wahadła w charakterze regulatora chodu. Uczynił to Ch. Huygens (1629—1695), konstruując zegar wahadłowy, w którym zastosował mechanizm zliczający wahania. Huygens przedstawił swój wynalazek w 1658 roku w dziele „Horologium" (Zegar).

Jednakże warto zaznaczyć, że przeszło 160 lat przed Huygensem, Leonardo da Vinci projektował mechanizmy z systemem kół zębatych i wahadłem regulującym działanie urządzenia. O zastosowaniu mechanizmu wahadłowego do regulacji pracy pomp informują J. Besson (ok. 1500—1569) i A. Ramelli (ok. 1530—1590) [8]. W 1675 roku Huygens wynalazł balans ze sprężyną, który w zegarach przenośnych odgrywa tę samą rolę co wahadło. W 1685 roku w lipskich „Acta Eruditorum", najpoważniejszym wówczas czasopiśmie naukowym, opublikowano pracę pt. „Novum genus perpendiculi pro horologiis rotatis portatilibus" (Nowy rodzaj wahadła dla obrotowych zegarków przenośnych) autorstwa Adama A. Kochańskiego (1631—1700), polskiego jezuity, matematyka i mechanika, bibliotekarza Jana III Sobieskiego. W pracy tej Kochański opisuje zegar magnetyczny swego pomysłu, wynaleziony już w 1659 roku, w którym balans oscyluje w polu magnetycznym. Wzmiankę o nim Kochański zamieścił w napisanym przez siebie rozdziale (dotyczącym mechanizmów zegarowych) dzieła Kaspra Schotta pt. „Technika curiosa" (Osobliwości techniki), opublikowanym w 1664 roku w Norymberdze. Uważa się, że był to „pierwszy w historii zegarmistrzowstwa wykład tej sztuki" [6].

W XVIII wieku zaczęto udoskonalać krosna tkackie. B. Bouchon w 1725 roku zrealizował układ, który kilkadziesiąt lat później całkowicie przekształcił wyrób tkanin wzorzystych. Zadaniem układu było rozdzielenie, za pomocą specjalnych igieł sterowanych taśmą dziurkowaną — nici osnowy materiału tkanego na krośnie. Falcon (ok. 1705—1765) zmienił w 1728 roku układ sterowania wynaleziony przez Bouchona, zastępując taśmę dziurkowaną kartami prostokątnymi i zwiększającymi jednocześnie liczbę igieł. W 1744 roku J. J. de Vaucanson (1709—1782) wynalazł warsztat tkacki sterowany za pomocą walca mechanicznego. J.-M. Jacquard (1752—1834) znając wynalazki Bouchona i Falcona, a także Vaucansona, udoskonalił je [1], konstruując w 1805 roku maszynę sterowaną kartami perforowanymi, która dzisiaj nosi jego imię.

Żywiołowy rozwój przemysłu w XIX wieku uczynił maszynę Jacquarda szeroko znaną. Nic więc dziwnego, że Ch. Babbage (1792—1871) w swojej „maszynie analitycznej" wykorzystał do sterowania karty perforowane. Dla tej maszyny Ada Byron opracowała koncepcję takiego ich zastosowania, które dziś można nazwać podprogramem. Pomysł polegał na „cofaniu karty" w dowolnej fazie wykonywania programu [5].

W czasie, gdy Huygens sprawował prezesurę Akademii Nauk w Paryżu, zetknął się z nim D. Papin (1647—1714). Pracując pod opieką Huygensa, wynalazł w 1679 roku kocioł (uważany za poprzednika współczesnych autoklawów), z którym po raz pierwszy współpracował — w charakterze regulatora ciśnienia — zawór bezpieczeństwa. Nie był to jednak czas, aby wynalazki tego rodzaju rewolucjonizowały technikę. Tak było również kilkadziesiąt lat później, gdy w Rosji Iwan I. Połzunow (1728 lub 1729—1766) zbudował w latach 1765—66 siłownię parową z regulatorem poziomu wody. Dopiero wynalazek Jamesa Watta (1736—1819) pojawił się w odpowiednim momencie i w stosownym miejscu. W 1784 roku Watt uzyskał patent na silnik parowy, w którym zastosował odśrodkowy regulator prędkości obrotowej. Regulator wynaleziony przez Huygensa, adaptowany później do potrzeb młynów wietrznych i regulacji obrotów kół wodnych, został zmodernizowany i użyty przez Watta do regulacji prędkości obrotowej maszyn parowych [2]. W dobie rewolucji przemysłowej następuje intensywny rozwój przemysłu. Pojawiają się nowe rozwiązania urządzeń i układów sterowania [10]. W 1868 roku, w Royal Society, londyńskiej akademii nauk, J. Clerk Maxwell (1831—1879) wygłosił referat pt. „On governors" („O regulatorach") [2]), w którym przedstawia wyniki badań stabilności układu maszyna parowa — regulator Watta („Regulator jest częścią maszyny, za pomocą której utrzymywana jest prawie jednostajna prędkość maszyny, pomimo zmian mocy napędowej lub oporu" [9]).

Pięć lat później, w 1873 roku, inżynier francuski Leon Farcot (1823—1909) skonstruował układ regulacji, w którym zastosował serwomotor hydrauliczny ze sztywnym (tj. proporcjonalnym) sprzężeniem zwrotnym. Była to nowa jakość, ponieważ dotąd nie budowano regulatorów korzystających z energii zasilającej; energia potrzebna do uruchomienia elementu wykonawczego pochodziła z czujnika pomiarowego. W 1884 roku pojawił się regulator o zmiennej strukturze: gdy uchyb regulacji był duży, za pomocą przekaźnika włączało się sztywne sprzężenie zwrotne (uzyskano przez to działanie proporcjonalne), gdy był mały — sprzężenie podatne (otrzymywano dodatkowo działanie całkujące). Uwieńczeniem tego okresu rozwoju układów regulacji była nagroda Nobla z fizyki przyznana w 1912 roku szwedzkiemu inżynierowi Nilsowi G. Dalénowi (1869—1937), m.in. za skonstruowanie w 1907 roku układu automatycznej regulacji częstotliwości pulsacji i wielkości płomienia palnika acetylenowego boi świetlnej [3, 10]. Nowy etap rozwoju regulacji automatycznej, czyli sterowania ze sprzężeniem zwrotnym, nastąpił wtedy, gdy zaczęto stosować w regulatorach elementy elektroniczne i wykorzystywać

[1]) Maszyna jest tu rozumiana tak, jak słowo machina rozumieli Grecy, a więc jako dzieło inżynierskie.

[2]) Angielskie słowo governor pochodzi od łacińskiego gubernus (wywodzącego się z greckiego kybernetes), który znaczy sternik.

osiągnięcia matematyków w dziedzinie badania stabilności układów dynamicznych i badania procesów optymalnych.

W latach czterdziestych, Norbert Wiener (1894—1964) wraz ze współpracownikami stworzył nową dziedzinę nauki, obejmującą sterowanie w maszynach i zwierzętach, której w 1947 roku nadano nazwę cybernetyka. Tak m.in. uzasadniał wprowadzenie tego terminu [15]: *„Zdecydowaliśmy się nadać całej dziedzinie teorii sterowania i komunikacji w maszynach i zwierzętach nazwę cybernetyki, którą utworzyliśmy z greckiego kybernetes, czyli sternik. Wybierając ją chcieliśmy podkreślić, że pierwszą z ważnych opublikowanych prac o mechanizmach ze sprzężeniem zwrotnym był artykuł Clerka Maxwella z 1868 roku o regulatorach, które autor określił nazwą governors, pochodzącą właśnie od greckiego słowa kybernetes".*

Na połowę XIX wieku przypadają prace G. Boole'a (1815—1864) nad formalizacją logiki. Pierwszy wykład algebry logiki zawarł on w dziele z 1854 roku pt. „An Investigation of the Laws of Thought". W drugiej połowie XIX wieku system ten stanowił podstawę prac grupy logików, którzy go ulepszyli i uzupełnili [4]. Lecz dopiero przed drugą wojną światową wskazano na możliwości zastosowań technicznych: w 1938 roku C. E. Shannon zauważył [14], że przekaźnikowe sieci przełączające można opisywać za pomocą symbolicznej notacji algebry Boole'a. Podejście takie jest z powodzeniem realizowane także dzisiaj, m.in. w wielu mikroprocesorowych sterownikach programowalnych [7], zapewne ze względu na tradycję i przyzwyczajenia inżynierskie związane z projektowaniem algorytmów sterowania napędami, maszynami i urządzeniami.

Pierwszy układ mikroprocesorowy wyprodukowano na początku lat siedemdziesiątych, a pierwsze komputery w latach czterdziestych. W 1944 roku Amerykanin H. Aiken zbudował przy pomocy firmy IBM komputer przekaźnikowy o nazwie Mark I, w którym kolejnością wykonywania operacji sterował program wydziurkowany na taśmie papierowej. Technikę przekaźnikową wykorzystał również K. Zuse w konstrukcji swej wcześniejszej nieco maszyny Z3. Sterowanie było tutaj zrealizowane także za pomocą taśmy dziurkowanej, na której umieszczono ciąg 8-bitowych rozkazów [7].

Pierwszy komputer elektroniczny COLOSSUS został wykonany w 1943 roku pod kierunkiem M. H. A. Newmana i A. Turinga (1912—1954) w Wielkiej Brytanii [13]. Zapamiętywał on program wyłącznie układowo. W Stanach Zjednoczonych zespół z Uniwersytetu Pensylwanii, pracujący pod kierunkiem J. W. Mauchley'a (1907—1980) i J. P. Eckerta, zbudował maszynę elektroniczną o nazwie ENIAC (pierwsze wykorzystanie w końcu 1945 roku). Sterowanie obliczeniami zrealizowano w niej za pomocą tablic programowych i przełączników ustawianych ręcznie. Tę uciążliwą czynność wyeliminowano w następnych rozwiązaniach zarówno brytyjskich, jak i amerykańskich. Na 1945 rok w Wielkiej Brytanii przypada skonstruowanie prototypu komputera z zapamiętywanym programem (A. Turing, F. C. Williams — 1909—1977; M. H. A. Newman, T. Kilburn) [13]. Nieco później w USA, A. W. Burks, A. Goldstin i J. von Neumann (1903—1957), w zaprojektowanej przez siebie maszynie EDSAC, umieścili program także w pamięci [12].

Teoretyczne podstawy budowy komputerów opracował w latach 1945—1947 John von Neumann. Wyodrębnił on funkcje spełniane przez komputer w postaci bloków: operacyjnego, pamięci, wejścia i wyjścia oraz sterowania. Koncepcje te przedstawione w pracy pt. „Preliminary discussion of the logical design of an electronic computing instrument", napisanej wspólnie przez von Neumanna, Goldstina i Burksa, stanowiły punkt wyjścia przy konstruowaniu, m.in. takich maszyn jak MANIAC 1, ORACLE, SILLIAC, IBM-701 [12].

W połowie lat sześćdziesiątych IBM zastosował technikę mikroprogramowania w większości modeli Systemu 360 [11]. W dzisiejszym rozumieniu mikroprogram jest ciągiem operacji logicznych (mikrorozkazów) tworzących jeden rozkaz. Wykonanie rozkazu interpretuje się jako wykonanie programu, składającego się z wielu mikrorozkazów.

Koncepcję sterowania mikroprogramowego, jako alternatywę w stosunku do układowego sterowania sekwencyjnego, wprowadził w 1951 roku M. Wilkes. W artykule [16] pisał: *„...rozważmy sterowanie w ścisłym tego słowa znaczeniu, to jest część maszyny, która dostarcza impulsów do obsługi bramek współpracujących z rejestrami arytmetycznymi i sterowania. Projektant tej części maszyny często postępuje ad hoc, rysując schematy blokowe aż do momentu spostrzeżenia uporządkowania, które odpowiada jego wymaganiom i wydaje się być sensowne ekonomicznie. Chciałbym zaproponować sposób, w którym sterowanie może być uczynione systematycznym, a przeto mniej złożonym.*

Każde działanie wywołane kodem rozkazu maszyny obejmuje ciąg kroków, które mogą wywoływać przeniesienia z pamięci do rejestrów sterowania lub arytmetycznych, lub vice versa, i przeniesienia z jednego rejestru do innego. Każdy z tych kroków dokonuje się przez impulsowanie pewnych przewodów połączonych z rejestrami sterowania i arytmetycznymi; będę to nazywał mikrooperacją. Tak więc każda rzeczywista operacja maszynowa składa się z ciągu lub mikroprogramu, mikrooperacji".

Nowym etapem zastosowania komputerów było ich wykorzystanie do pracy w czasie rzeczywistym. Na początku lat pięćdziesiątych skonstruowano w Massachusetts Institute of Technology maszynę SAGE, której zadaniem była interpretacja danych uzyskiwanych ze stacji radarowych i wyznaczenie trajektorii rakiet, mających niszczyć cele znajdujące się w locie [12]. Na wczesne lata sześćdziesiąte przypada początek zastosowania komputerów do sterowania procesami produkcyjnymi. Wykorzystuje się je w układach sterowania bezpośredniego, nadrzędnego i w układach hierarchicznych. A dzieje się to ponad czterdzieści wieków po tym, jak Utnapisztî zbudował przed wielkim potopem korab, który wyposażył w ster: *„Przybiłem obręcze i wytyczywszy uczyniłem w nim sześć pokładów, podzieliłem jak dom korab na siedem pięter, wzwyż go przegrodziłem na dziewięć części, od wody kołkami go uszczelniłem, rudło dlań wybrałem i włożyłem na korab, co było potrzeba"* („Gilgamesz", tablica 11. Tłum. R. Stiller).

LITERATURA

[1] Alphandéry M. F.: Dictionnaire des invanteurs français. Editions Segers, Paris, 1962

[2] Bateman H.: The control of an elastic fluid. Bulletin of the AMS, Vol. 51, pp. 601—646, 1945. Także: pp. 19—64, Bellman R., Kalaba R., eds. Selected Papers on Mathematical Trends in Control Theory. Dover Publications, New York, 1964

[3] Bonin W. von, Bagge E., Herrlinger R.: Laureaci nagrody Nobla. Iskry, Warszawa, 1969

[4] Bourbaki N.: Elementy historii matematyki. PWN, Warszawa, 1980

[5] Bytniewski M.: Ada Byron — romantyczna prekursorka programowania maszyn cyfrowych. „Informatyka", nr 9, str. 36—37, 1977

[6] Dianni J., Kochański: Polski słownik biograficzny, t. XIII, str. 205—208, Zakład Narodowy im. Ossolińskich, Wrocław, 1967—1968

[7] Lechner K., Staufer K.: Flexible Automation. VTR-Verlag, Wien, 1983

[8] Lilley S.: Ludzie, maszyny i historia. PWN, Warszawa, 1963

[9] Maxwell J. C.: On governors. Proceedings of the Royal Society of London, Vol. 16, pp. 270—283, 1868 (Także: pp. 4—17, Bellman R., Kalaba R., eds., Selected Papers on Mathematical Trends in Control Theory. Dover Publications, New York, 1964)

[10] Niederliński A.: Teoria i technika regulacji. Archiwum Automatyki i Telemechaniki, t. XVIII, z. 2, str. 135—152, 1973

[11] Rauscher T. G., Adams P. M.: Microprogramming — a tutorial and survey of recent developments. IEEE Transactions on Computers, Vol. C-29, No. 1, pp. 2—20, 1980

[12] Rose J.: Anatomia i fizjologia automatyzacji. Wiedza Powszechna, Warszawa, 1973

[13] Rudnicki P.: Czy rzeczywiście Anglicy byli pierwsi? „Informatyka", nr 6, str. 22—23, 1977

[14] Shannon C. E.: A symbolic analysis of relay and switching circuits. Transactions of the American Institution of Electrical Engineers, Vol. 57, No. 1, pp. 1—11, 1938

[15] Wiener N.: Cybernetyka, czyli sterowanie i komunikacja w zwierzęciu i maszynie. PWN, Warszawa, 1971

[16] Wilkes M. V.: The best way to design an automatic calculating machine. Report of the Manchester University Computer Inaugural Conference. Manchester, England, pp. 16—18, July 1951, Także: pp. 266—270, Swartzlander E. E., ed., Computer Design Development — Principal Paper. Hayden Book, Rochelle Park (NJ), 1976

[17] Zajdler L.: Dzieje zegara. Wiedza Powszechna, Warszawa, 1977.

VASILII ZAKHAROV
Uniwersytet Genewski
Genewa
Szwajcaria

Komputery osobiste oraz indywidualne stanowiska robocze

Od wprowadzenia w roku 1981 komputerów osobistych IBM PC i przeznaczonego dla nich systemu operacyjnego PC-DOS 1.1, ich rozwój — pod względem właściwości i liczby sprzedanych egzemplarzy — przebiegał niemal nadzwyczajnie. Jednakże koncepcja komputera osobistego istnieje znacznie dłużej i obecne realizacje zawodzą oczekiwania pierwszych projektantów. W rzeczywistości przez te pięć lat wykształciły się dwa rodzaje terminali użytkowych, różniących się właściwościami. Pierwszy z nich, komputer osobisty, jest w zasadzie urządzeniem jednoużytkowym i jednozadaniowym; drugi, zwany indywidualnym stanowiskiem roboczym (ang. personal workstation, PWS), ma większe możliwości i jest znacznie bliższy pierwotnym zamierzeniom projektowym.

Celem tego artykułu jest porównanie właściwości komputerów osobistych i stanowisk roboczych oraz dokonanie prognozy dotyczącej ich rozwoju w ciągu najbliższych kilku lat. Istnieją już oznaki świadczące o tym, że obecna różnica między komputerami osobistymi a stanowiskami roboczymi będzie zanikać, a przyszłe stanowiska robocze raczej nie będą stanowić autonomicznych urządzeń, lecz będą częścią rozproszonego systemu komputerowego, przeznaczonego do wszelkich zastosowań. Tak więc nie eliminując potrzeby istnienia dużych zasobów komputerowych indywidualne stanowiska robocze nawet wzmacniają taką potrzebę.

Rozważanym przykładem komputera osobistego jest IBM PC. Nie znaczy to, że nie istnieją świetne komputery osobiste wytwarzane przez innych producentów, lecz komputery IBM PC i kompatybilne zdecydowanie zdominowały rynek mikrokomputerowy.

ROZWÓJ KOMPUTERÓW OSOBISTYCH

Pierwszy komputer IBM PC oparto na mikroprocesorze Intel 8088 o częstotliwości zegara 4,8 MHz. W konfiguracji znajdowała się pamięć RAM o pojemności 640 KB, dwa napędy jednostronnych dysków elastycznych i monitor monochromatyczny o rozdzielczości 720 × 350 pixeli (elementów obrazu, ang. picture elements, pixels). Jednoużytkowy i jednozadaniowy system operacyjny PC-DOS umożliwiał sterowanie operacjami wejścia-wyjścia przez standardowe wywołania podstawowego systemu we-wy BIOS (ang. basic input-output system). Wprowadzono nową magistralę systemową, nie adaptując żadnego z istniejących rozwiązań (co było dość dziwną decyzją), przy czym liczbę gniazd na dodatkowe płytki (rozszerzenie konfiguracji) ograniczono do pięciu.

Wkrótce stało się oczywiste, że komputer powinien mieć większą szybkość i większą liczbę gniazd, umożliwiających rozbudowę podstawowej konfiguracji, oraz — że należy go wyposażyć w większą pamięć masową, w większą liczbę płyt sprzęgających urządzenia wejścia-wyjścia i zapewnić pewne dodatkowe możliwości, np. graficzne. Ponadto istniejący zasilacz nie pozwalał na dalszą rozbudowę konfiguracji. Ponieważ nowe programy aplikacyjne wymagały nie tylko większej pamięci operacyjnej, lecz także dyskowej, w praktyce znikły dyskietki jednostronne i w niektórych zastosowaniach nieodzowne stało się użycie dysków stałych. Oprócz wymienionych wad sprzętu istotne były także pewne ograniczenia systemu operacyjnego.

Producent odpowiedział na zwiększenie wymagań dwojako. Po pierwsze, wprowadzono nowy model IBM PC/XT, oparty również na mikroprocesorze 8088, zawierający 8 gniazd przeznaczonych do rozbudowy i nowy zasilacz o większej mocy, a także wbudowano dysk stały jako wyposażenie standardowe. Ulepszono również grafikę. Po drugie, napisano od nowa system operacyjny, wzbogacając go o wiele nowych cech, m.in. o nowe wywołania systemowe BIOS. Podstawowe ulepszenie polegało na zwiększeniu szybkości działania, głównie dzięki znacznie krótszemu czasowi dostępu do dysku.

Pod koniec roku 1984 wprowadzono model IBM PC/AT, oparty na 16-bitowym mikroprocesorze 80286, o częstotliwości zegarowej 6 MHz. Nowy model był nie tylko szybszy, lecz oprócz dysków stałych zawierał dodatkowo opcjonalną stację dysków elastycznych o pojemności 1,2 MB. Do standardowego wyposażenia należała także pamięć RAM o pojemności 512 KB, zainstalowana na płycie systemowej, i koprocesor arytmetyczny 80287. Wprowadzono również nową wersję systemu operacyjnego PC-DOS 3.0, a później 3.1. Po raz pierwszy zaoferowano także wielozadaniowy system operacyjny, Xenix firmy Microsoft, łącznie z kompilatorem języka C. Xenix jest pochodną systemu operacyjnego Unix [1]) (System III), opracowanego w ATT (American Telegraph and Telephone) Bell Laboratories, i zapewnia przenośność plików do systemu PC-DOS 3.0. Nieco wcześniej pojawiły się monitory kolorowe o rozdzielczości 640 × 480 pixeli.

Mniej więcej w czasie wprowadzania modelu PC/AT, na rynku pojawiła się duża liczba komputerów kompatybilnych z IBM PC/XT, z których najciekawsze były oparte na szybszym mikroprocesorze 8086. Ponadto w firmie Microsoft napisano od nowa system operacyjny DOS, ulepszając go i wzbogacając na wiele sposobów, tworząc faktycznie wersję dla mikroprocesora 8086. Nowa wersja MS-DOS 2.0 stała się bardzo użytecznym narzędziem. Przykładem konkurencyjnego rozwiązania dla modelu PC/XT, ze względu na stosunek ceny do możliwości, jest komputer osobisty M24 firmy Olivetti.

PROBLEM KOMPATYBILNOŚCI

Ocenia się, że w 1985 roku sprzedano ponad ćwierć miliona egzemplarzy modelu PC/XT. Obecnie istnieje już około pięćdziesięciu różnych producentów kompatybilnych z nim komputerów osobistych. W niektórych z tych komputerów stosuje się procesor 80286 o częstotliwości zegarowej 8 MHz, monitory o zwiększonej rozdzielczości, większe pamięci operacyjne i dodatkowe koprocesory. Nie wszystkie jednak z tych komputerów są identyczne z którymś z modeli IBM PC/1, PC/XT lub PC/AT. Nawet wewnątrz rodziny modeli IBM istnieje problem kompatybilności.

W ciągu ostatnich kilku lat powstała niezliczona ilość oprogramowania aplikacyjnego dla komputerów osobistych,

[1]) Por. cykl artykułów pt. Czym jest Unix, Mikroklan, nr 1—3, 1986 (przyp. red.).

przeznaczonego głównie dla systemów jednoużytkowych do zastosowań w zarządzaniu. Bardzo dobrze znane stały się takie pakiety, jak Framework, Lotus 1-2-3, SAMNA, dBASE-III i in., a oprócz tego dostosowano do użytkowania na komputerach osobistych tak dobre systemy zarządzania relacyjnymi bazami danych, jak ORACLE lub INGRES. Spowodowało to jednak powstanie dodatkowych problemów kompatybilności.

Wiele zastosowań opracowanych w początkowym okresie rozwoju systemu operacyjnego DOS wymagało dużej pojemności pamięci i w praktyce zużywało całą dostępną pamięć. Ponadto, w celu zwiększenia szybkości, w wielu programach omijano system BIOS lub bezpośrednio korzystano z jego podprogramów. Często w programach aplikacyjnych dokonywano przeprogramowania kanału DMA, czasomierza lub portu szeregowego, bez sprawdzania, czy nie używają ich inne programy lub system operacyjny! Takie postępowanie wynika oczywiście z założenia, że cały system jest jednoużytkowy i jednozadaniowy, a jego skutkiem jest totalny chaos. Ostatecznie znaczy to, że oprogramowania przeznaczonego dla jednego zastosowania nie można użyć w innym. Często zdarza się także, że program aplikacyjny, działający na jednym modelu PC z określoną wersją systemu DOS lub BIOS, nie działa na zestawie złożonym z innych wersji tych samych elementów.

Problem kompatybilności występuje szczególnie często przy ochronie oprogramowania, gdyż w tej dziedzinie stosuje się przeprogramowanie sprzętu, głównie sterowników dyskowych. Jest oczywiste, że nie można tego robić bez (przynajmniej) poinformowania systemu operacyjnego.

STAN OBECNY — MODEL IBM RT/PC

Obecną sytuację można określić jako punkt zwrotny.

Z jednej strony, większość oprogramowania aplikacyjnego (jak również praktycznie systemy operacyjne komputerów osobistych) jest przeznaczona do zastosowań jednoużytkowych i jednozadaniowych na pojedynczych komputerach. Wiele programów wciąż traktuje urządzenia peryferyjne dołączone do komputera jako swoją prywatną własność, a nie jako współdzielone zasoby. Przykładowo, wydruki są wysyłane najczęściej wprost na drukarkę, bez pośrednictwa dysku i zapewnienia tzw. spoolingu, jak w nowoczesnych systemach operacyjnych.

Jednocześnie rozszerzyły się znacznie możliwości komputerów osobistych i powstało wiele udogodnień dostępnych dla użytkowników. Oprócz koprocesora arytmetycznego, wprowadzono — na przykład — płytę sterownika graficznego z dodatkowym procesorem i pamięcią obrazu oraz elementem graficznym. Sterownik może obsługiwać monitor kolorowy o rozdzielczości 640 × 480 pixeli. Sprzężenie programowe obejmuje system GKS (ang. Graphical Kernel System) i sprzężenie VDI (ang. Virtual Device Interface).

Z drugiej strony wprowadzono istotne ulepszenia w celu osiągnięcia wielozadaniowości i wieloużytkowości. Taką próbą było wprowadzenie systemu operacyjnego Xenix do zastosowań wieloużytkowych. Jednakże, w systemie Xenix nie działają programy aplikacyjne utworzone pod kontrolą systemu PC-DOS, a ponadto nie istnieje metoda współdzielenia danych między nimi. Inny kierunek rozwoju polega na dobudowaniu wielozadaniowego egzekutora na wierzchu systemu DOS. Takim programem jest TopView, umożliwiający współbieżne wykonywanie kilku programów aplikacyjnych i współdzielenie przez nie zasobów. Najważniejszą cechą programu TopView jest współbieżne wyświetlanie wyników w zbiorze okien i wybór opcji za pomocą ikon. Jednakże, wszystko co wykonuje TopView, polega w najlepszym razie na rozszerzeniu systemu DOS w celu zapewnienia prostego przełączania zadań, zarządzania pamięcią i sterowania wejściem-wyjściem ekranowym. Inne istotne funkcje, jak sprzężenie z siecią i zarządzanie operacjami dyskowymi, pozostawiono systemowi DOS. Takie rozwiązanie nie oznacza oczywiście, że jest to prawdziwie wielozadaniowy system operacyjny, który powinien nie tylko wykonywać wszystkie wymienione funkcje, lecz być od początku zaprojektowany do tego celu. Ponieważ TopView stanowi nadbudowę pierwotnego systemu operacyjnego, wiele bardzo pożądanych funkcji, jak np. sprzężenie z siecią, jest z nim niezgodnych.

Podejmuje się próby opracowania prawdziwie wielozadaniowego systemu operacyjnego, jak np. Concurrent DOS firmy Digital Research, lecz podlegają one innym ograniczeniom. Taki system jak Unix również nie rozwiązuje problemu, częściowo dlatego, że istnieje w wielu różnych wersjach o niepowtarzalnych rozszerzeniach, ale także dlatego, że nie można pod jego kontrolą wykonywać programów działających w systemie PC-DOS. Jednakże oba wymienione systemy operacyjne mogą stanowić początek drogi do rozwiązania problemu wieloużytkowego i wielozadaniowego wykorzystania komputerów osobistych.

Spośród wielu proponowanych rozwiązań problemu wieloużytkowości wyróżnia się dwa główne kierunki. W pierwszym z nich użycie komputera osobistego polega na zastosowaniu go, np. modelu PC/AT, do obsługi obliczeń i obsługi plików dla niewielkiej liczby odległych terminali (lub ich emulatorów). Jest to jednak rozwiązanie bardzo ograniczone, przeznaczone głównie do współdzielenia pamięci dyskowej i posiadania wspólnej puli plików w środowisku lokalnym. Znacznie większe możliwości zapewnia alternatywne rozwiązanie polegające na sprzężeniu komputera osobistego z lokalną siecią komputerową.

Obecnie istnieje co najmniej 40 różnych, wzajemnie niezgodnych wyrobów przemysłowych, przeznaczonych do łączenia komputerów osobistych w sieci lokalne. Najważniejsze z nich są te, które zapewniają sprzężenie sprzętowe (tj. jedną lub więcej kart instalowanych w gniazdach płyty systemowej) ze standardową siecią lokalną, wraz z odpowiednimi programami obsługi (ang. software drivers). Takim urządzeniem jest, na przykład, produkowana przez firmę 3COM, płyta sprzęgająca z siecią Cheapernet (zawierająca także nadajnik-odbiornik, ang. transceiver) umożliwiająca użycie komputera osobistego jako stacji obsługi plików (ang. file server) lub jako niezależnego, specjalizowanego procesora. Istniejące oprogramowanie umożliwia przesyłanie plików w oparciu o protokoły TCP-IP (Terminal Communication Protocol — Internetworing Protocols) i zarządzanie dyskami wirtualnymi na poziomie woluminów. Istnieją również opcje umożliwiające spooling i pocztę elektroniczną.

Rozwiązanie proponowane przez firmę 3COM jest bardzo efektywne. Niestety, nie rozwiązanych pozostaje jeszcze wiele innych problemów, wynikających z faktu, że podstawowym systemem operacyjnym wciąż jest DOS, a dla wielu programów aplikacyjnych niekorzystne jest to, że zostały napisane dla systemu jednoużytkowego.

Ostatnio zaanonsowany model IBM RT/PC jest jeszcze zbyt nowy, aby można poddać uczciwej ocenie jego możliwości i sprzężenie użytkowe (ang. performance and user interface). Podstawowa różnica między modelem RT a pozostałymi komputerami osobistymi polega na tym, że jego procesor ma architekturę RISC (ang. Reduced Instruction Set Computer), a podstawowy system operacyjny AIX jest oparty na Unixie, a nie na systemie PC-DOS.

Koncepcja architektury RISC powstała w firmie IBM około roku 1975 i została najpierw zrealizowana w minikomputerze 801. W modelu RT/PC użyto 32-bitowego mikroprocesora produkcji IBM, którego lista rozkazów zawiera tylko połowę 200—300 rozkazów konwencjonalnego procesora. Ze względu na bardzo krótki czas cyklu (170 ns) można osiągnąć szybkość działania równą około 4 Mips (ang. megainstructions per second, megainstrukcji na sekundę). Jednak programy będą miały większą liczbę rozkazów, dłuższe czasy ładowania i będą wymagały częstszego stronicowania, co powoduje, że dokonywanie porównań na tym etapie jest utrudnione. Ważną właściwością tego modelu jest pamięć wirtualna zrealizowana przy użyciu oddzielnego mikroukładu zarządzania pamięcią, o 40-bitowym słowie adresowym.

W modelu RT/PC zastosowano magistralę systemową PC/AT, dzięki czemu można używać urządzeń peryferyjnych i płyt tego starszego modelu. Jednakże, należy napisać dla nich nowe programy obsługi, ponieważ nie może być już użyty stary system BIOS. Do rozbudowy służy 8 dodatkowych gniazd na płycie głównej, dysk stały może mieć pojemność 40 lub 70 MB, a pamięć operacyjną można rozszerzyć do 4 MB. W skład podstawowej konfiguracji wchodzi również tzw. akcelerator zmiennoprzecinkowy (ang. floating-point accelerator).

Oprócz systemu operacyjnego AIX można używać także systemu PC-DOS po dołączeniu dodatkowej karty z procesorem 80286. PC-DOS i AIX mogą działać współbieżnie, współdzieląc pamięć dyskową, ekran i pamięć operacyjną, lecz nie są w żadnym sensie zintegrowane. Przesyłanie plików między obydwoma systemami odbywa się za pomocą programów usługowych.

Doskonałym ulepszeniem jest zarządca okien (ang. window manager) oparty na karcie graficznej z mapą bitową (ang. bit-mapped graphics), wyposażonej w dodatkową pamięć i specjalizowany procesor. Niestety, nie jest on zintegrowany z systemem PC-DOS. W każdym oknie może być wykonywany tylko jeden program aplikacyjny, działający pod kontrolą sytemu PC-DOS. W pierwszej implementacji, jak się wydaje, tylko jedno okno może być wyświetlone w danej chwili na ekranie, co oczywiście jest znaczną wadą.

Jednak największą wadą modelu RT/PC wydaje się brak jakiegokolwiek sprzętu i oprogramowania sieciowego. W chwili obecnej połączenie z lokalną siecią komputerową może być wykonane tylko przez sprzężenie z systemem PC-DOS, co oczywiście nie jest zintegrowane z systemem AIX. Ponadto, jasne jest, że sprzężenia użytkowe obu systemów znacznie się różnią.

STANOWISKA ROBOCZE

Wydaje się, że pierwsze stanowiska robocze powstały z połączenia minikomputerów i interakcyjnej grafiki. Używano ich w wielu zastosowaniach jednoużytkowych, od gier do projektowania wspomaganego komputerowo (CAD). Takie połączenia istnieją już od początku lat sześćdziesiątych. Opracowanie takiego systemu jak Sketchpad odegrało również istotną rolę we wprowadzaniu grafiki do indywidualnej współpracy z systemami komputerowymi. Jednakże główne prace, które przyczyniły się do powstania stanowisk roboczych, prowadzono w Xerox Palo Alto Research Center, w Kalifornii, gdzie opracowano m.in. komputery Alto i STAR, a także lokalne sieci komputerowe w znanej dzisiaj postaci. W Xerox PARC powstały niemal wszystkie koncepcje nowoczesnego sprzężenia użytkowego stanowisk roboczych, jak nakładane okna, technika menu, ikony i sterowanie kursora myszką (choć dokładnie mówiąc, samą mysz opracowano gdzie indziej w roku 1964). Istotną rolę w tych pracach odegrał język Smalltalk, powstały również w Xerox PARC.

Czym jest stanowisko robocze?

Nie można podać ustalonej definicji stanowiska roboczego, ponieważ wraz z rozwojem technologii i zmniejszaniem się kosztów sprzętu zmieniają się wymagania użytkowników. Jednakże istnieje pewna granica jakości, od której począwszy można sklasyfikować komputer jako stanowisko robocze; istnieją także określone cechy architektury, które musi mieć stanowisko robocze.

Stanowisko robocze jest w zasadzie jednoużytkową, wielozadaniową, interakcyjną stacją o wszechstronnych możliwościach graficznych, z klawiaturą i myszką (lub drążkiem itp.), o sprzężeniu programowym opartym nie tylko na przyjaznym języku poleceń, lecz także na technice okienkowania. Rodzaje menu i sposoby wyboru opcji są szczegółami nieistotnymi, gdyż istnieją różne metody interakcji.

Choć stanowisko robocze jest terminalem jednoużytkowym, nie wszyscy użytkownicy są jednakowi i nie wszystkie ich wymagania obliczeniowe mogą być spełnione przez pojedynczą stację. Dlatego stanowisko robocze musi mieć dwie podstawowe właściwości architektury. Po pierwsze, całe stanowisko musi mieć modularną budowę, aby można dostosować jego konfigurację do potrzeb (lub możliwości finansowych) użytkownika. Po drugie, stanowisko robocze musi mieć możliwość dołączenia do lokalnej sieci komputerowej. Choć nie jest to nieodzowne wymaganie, można dodać, że budowa modularna powinna być oparta na standardowej magistrali (takiej jak VME, Multibus lub Versabus), a sieć lokalna służąca do łączenia stanowisk roboczych powinna być zgodna z normami ISO.

Jeśli chodzi o system operacyjny, to powinien on być zaprojektowany całościowo, w sposób integrujący wszystkie cechy stanowiska, jak wielozadaniowość, okienkowanie, sprzężenie użytkowe, połączenie z siecią i zarządzanie plikami, bez uzupełniania o te właściwości jakiegoś istniejącego systemu operacyjnego.

Właściwości stanowisk roboczych

Poniżej przedstawiono listę najważniejszych atrybutów stanowisk roboczych:

1) modularna architektura oparta na standardowej magistrali, umożliwiająca łatwą rozbudowę

2) w pełni 32-bitowy procesor o szybkości operacji co najmniej 1 Mips, a w przypadku architektury RISC — o równoważnej szybkości 1,2—2 Mips; dostępność operacji zmiennoprzecinkowych np. przy użyciu koprocesora

3) dostępna dla użytkownika pamięć, o pojemności przynajmniej 1 MB; duży obszar adresowy pamięci wirtualnej

4) użytkowa przestrzeń dyskowa o pojemności co najmniej 20 MB, dostępna lokalnie lub współdzielona w systemie obsługi plików

5) monitor o rozdzielczości przynajmniej 1024 × 800 pixeli z mapą bitową w oddzielnej pamięci; do odwzorowania mapy na obraz i wykonywania operacji graficznych (rastrowych) powinien istnieć specjalny sprzęt

6) zarządzanie oknami i redagowanie wymianą informacji między nimi jako standardowa funkcja systemu operacyjnego; każdy program systemu wielozadaniowego powinien umożliwiać uaktualnianie i odbiór informacji we własnym oknie, a kilka nakładanych okien — współbieżne wyświetlanie

7) bezpośrednie połączenie ze standardową lokalną siecią komputerową przy użyciu standardowych protokołów; system operacyjny powinien zapewniać dostęp do plików i urządzeń przez sieć

8) możliwość wykonywania programów aplikacyjnych zgodnych z systemem PC-DOS, w jednym lub kilku oknach współbieżnie z innymi programami użytkowymi.

Do powyższej listy można dodać jeszcze wiele innych, równie ważnych cech. Bardzo pożądana jest, na przykład, możliwość programowania ukierunkowanego obiektowo — do szybkiego operowania obiektami graficznymi i wyświetlania. Ta metoda jest bardziej użyteczna niż wywołania standardowego pakietu graficznego, takiego jak GKS, który zaprojektowano dla obrazów wektorowych i dlatego działa znacznie wolniej w środowiskach z mapą bitową. Tym niemniej, osiem wyliczonych atrybutów dość dobrze charakteryzuje właściwości, którymi powinny dysponować stanowiska robocze w najbliższych kilku latach. Jeżeli należałoby dodać jakąkolwiek inną cechę, to powinna nią być cena minimalnej konfiguracji — nie przekraczająca 5000 dolarów (pod koniec roku 1986).

Na powyższej liście nie uwzględniono celowo jednego elementu, mianowicie — systemu operacyjnego. Żaden z istniejących systemów operacyjnych nie ma z pewnością wszystkich wymaganych właściwości ani Unix, ani PC-DOS lub MS-DOS. Nie wydaje się jednak, że daleka jest droga do wyłonienia się systemu operacyjnego, stanowiącego syntezę wszystkich najlepszych właściwości i będącego jednocześnie wygodnym i wydajnym.

Istnieje jeszcze jedno wymaganie, nie umieszczone na powyższej liście, którego spełnienie jest konieczne do skutecznego działania stanowisk roboczych. Należy, mianowicie, zapewnić dostęp do szybkiej i niezbyt kosztownej stacji obsługi plików (ang. file server), dostosowując jej oprogramowanie i system operacyjny stanowiska roboczego do operowania globalną przestrzenią identyfikatorów plików i urządzeń. Urządzenie obsługi plików powinno być niewątpliwie dołączone do tej samej lokalnej sieci komputerowej, co stanowisko robocze. Można również dodać, że w tej sieci lokalnej powinny istnieć odpowiednie bramy (ang. gateways) do standardowych, rozległych sieci komputerowych (ang. wide area networks, WAN), np. w oparciu o protokół X.25 (co już jest faktem).

Czy komputer osobisty jest stanowiskiem roboczym?

Porównując rozważania zawarte w dwóch poprzednich punktach można łatwo dojść do wniosku, że żaden z modeli komputerów osobistych poniżej PC/AT, nie spełnia wymagań nałożonych na stanowisko robocze. Z drugiej strony wydaje się, że bardzo bliski spełnienia tych założeń jest model RT/PC i może je spełnić całkowicie w drodze dalszej ewolucji.

Podstawową wadą tego modelu jest obecny brak sprzężenia z siecią komputerową według zaleceń ISO. Wydaje się też, że grafika i operacje ekranowe nie są w pełni zintegrowane z systemem operacyjnym. Jednakże w chwili pisania tego artykułu (początek roku 1986) jest jeszcze za wcześnie, aby dokonać pełnej oceny modelu RT/PC. Oczywiste jest tylko rozczarowanie spowodowane brakiem nakładających się okien.

PRZYKŁADY STANOWISK ROBOCZYCH

Wiele urządzeń produkowanych przemysłowo, opisywanych jako stanowiska robocze, ma rzeczywiście szereg cech

wymienionych w tym artykule. Bezpośrednim skutkiem prac prowadzonych w Xerox PARC było powstanie komputerów STAR (firmy Xerox) i Lisa (firmy Apple Computer). Niestety, Lisę wkrótce wycofano z produkcji, a STAR jest jedynie biurowym stanowiskiem roboczym, przeznaczonym do pracy w dość ograniczonym środowisku sieciowym. Komputer PERQ firmy ICL, jedno z pierwszych stanowisk roboczych, miał z pewnością najlepsze obrazowanie, ale nie rozwijał się tak szybko jak inne stanowiska. Był oparty na 16-bitowym mikroprogramowanym procesorze segmentowym.

Wydaje się, że właściwości najbardziej zbliżone do wyżej przedstawionych wymagań (wyłączając ceny) mają dwa stanowiska robocze: Apollo i SUN.

Stanowiska robocze Apollo

Apollo jest w rzeczywistości rodziną różnych, lecz wzajemnie zgodnych stanowisk roboczych. W najbardziej ograniczonej wersji występuje procesor Motorola 68010 z pamięcią o pojemności 1,5 MB. W systemie może istnieć 15 współbieżnie wykonywanych procesów, a każdy proces może użytkować 16 MB pamięci wirtualnej. Pamięć dyskowa może być dostępna lokalnie (od 34 MB) lub zdalnie przez dołączoną do sieci stację obsługi plików.

Podstawowy monitor jest czarno-biały z mapą bitową 1024 × 1024 pixele, z których 1024 * 800 służy do odświeżania przy wyświetlaniu, a pozostałe tworzą szybką pamięć notatnikową. Pamięć wyświetlania jest wspomagana przez sprzętowy przesuwnik (ang. bit mover). Wprowadzanie informacji odbywa się przez klawiaturę, mysz lub zwykły port synchroniczny bądź asynchroniczny. Istnieje również sprzęg magistrali Multibus, zawierający pięć gniazd do dołączania urządzeń zewnętrznych.

Bardzo istotnym elementem systemu Apollo jest sieć o nazwie Domain, mająca strukturę pierścienia znamieniowego (ang. token ring, token = znamię) o szybkości transmisji 12 Mb/s. Wszystkie stanowiska robocze i stacje obsługi plików mają sprzężenie z siecią Domain, a stację standardową można skonfigurować jako drukarkę lub bramę do innych sieci. Pliki są przesyłane między stacją obsługi a węzłami obliczeniowymi metodą stronicowania na żądanie (ang. demand paging). System operacyjny Aegis jest specyficzny dla tych stanowisk i bardzo efektywny. Umożliwia interaktywne i współbieżne wykonywanie procesów, nakładane wyświetlanie okien i komunikację między procesorami. Struktura skorowidzów, plików i procesów jest rozciągnięta na całą sieć. Język poleceń powłoki (ang. shell command language) pozwala na skoordynowane wykonywanie wielu programów, nawet jeśli rezydują w różnych węzłach sieci.

Ostatnio ogłoszono dostępność sprzężeń do systemu operacyjnego Unix w wersjach System V i BSD4.2 (ang. Berkeley Software Distribution) mogących działać jednocześnie w każdym węźle. Jest to dość unikalne rozwiązanie i trudno jest jeszcze ocenić jego wartość. W tym samym czasie rozszerzono standardowe sprzężenie stacji Domain do sieci Ethernet o moduł trasowania międzysieciowego (ang. internet router) i protokół TCP/IP.

Nie ma wątpliwości, że indywidualne stanowiska robocze Apollo są doskonałe, szczególnie do zastosowań w procesach komputerowo wspomaganego projektowania i wytwarzania (CAD-CAM), dla których istnieje znaczna ilość dobrego oprogramowania. Jedynie pod dwoma względami stanowiska Apollo nie spełniają wymienionych w tym artykule wymagań. Po pierwsze, minimalna konfiguracja stanowiska roboczego jest jeszcze ciągle zbyt kosztowna. Po drugie, każde rozszerzenie stanowiska lub połączenie komunikacyjne (z siecią lokalną lub rozległą) musi być wykonane przez tę stację, z którą stanowisko jest połączone w sieci Domain, nie będącą systemem otwartym. Tak więc, jedno z wymagań nie jest spełnione w sposób jawny, a połączenie z siecią OSI jest niełatwe i kosztowne. Niekorzystny jest również brak systemu operacyjnego PC-DOS, lecz nie stanowi to tak istotnej wady, tym bardziej że wkrótce oczekuje się wprowadzenia tego systemu.

Stanowiska robocze SUN

Oryginalne stanowisko robocze SUN powstało około 1980 roku na Uniwersytecie Stanforda i było przeznaczone do połączenia z siecią Ethernet. Od początku uważano je za stanowisko Unixa ze sprzężeniem do sieci Ethernet, stanowiącym integralną część sprzętu i systemu operacyjnego. Model, który wszedł do szerszego użytku, SUN-2, oparty na mikroprocesorze MC 68010 o częstotliwości zegarowej 9,8 MHz, produkowano w dwóch wersjach, z magistralą wewnętrzną Multibus lub VME. Ostatnio SUN-2 zastąpiono nową rodziną, SUN-3, opartą na 32-bitowym procesorze MC 68020 o częstotliwości zegarowej 15—17 MHz, wyłącznie w wersji VME (z wyjątkiem najmniejszego modelu).

SUN-3 jest komputerem modularnym o wszystkich jednostkach dołączonych do magistrali systemowej VME. Podstawową jednostką jest płyta CPU, zawierająca procesor, pamięć operacyjną, układ zarządzania pamięcią, sprzężenie sieciowe i standardowe wejście-wyjście. Płyta CPU ma również bardzo szybką magistralę wewnętrzną, do której można dołączyć dodatkową pamięć i tzw. akcelerator zmiennoprzecinkowy. Do innych płyt dołączanych do magistrali VME należą: procesor graficzny, bufor graficzny, bufor grafiki kolorowej oraz standardowy sprzęt SCSI (ang. standard computer system interface) do dysków i innych urządzeń peryferyjnych.

SUN-3 ma pamięć wirtualną o maksymalnej pojemności 256 MB, stronicowaną na żądanie w pamięci pomocniczej. Pamięć rzeczywista może mieć maksymalną pojemność 16 MB. Zależnie od modelu można używać dysków lokalnych (71-megabajtowych) lub odległych stacji obsługi plików, tzn. istnieją „bezdyskowe" stanowiska robocze, operujące plikami zdalnie przez sieć.

SUN-3 ma grafikę rastrową z mapą bitową. Pamięć odświeżana jest dwuwejściowa, aby była dostępna z procesora. Dostępne są pewne operacje rastrowe, realizowane przez specjalizowane mikroprocesory, umożliwiające bardzo szybkie przesuwanie i maskowanie bitowe. Standardowy monitor ma rozdzielczość 1152× 900 pixeli. Opcjonalny monitor kolorowy, 8 bitów na pixel, wymaga dodatkowej pamięci buforowej. Zainstalowany na oddzielnej płycie procesor graficzny służy do wykonywania transformacji geometrycznych, obcinania (ang. clipping), skalowania i operowania wektorami oraz wielokątami. Ponadto wykonuje wyświetlanie i obcinanie okien. Możliwości tego stanowiska roboczego są olbrzymie. Wystarczy powiedzieć, że przy użyciu operacji rastrowych możliwa jest aktualizacja obrazu z szybkością 46 megapixeli/s, a rysowanie wektorów — z szybkością 2 megapixele/s.

Sprzężenie z siecią Ethernet jest szczególnie efektywne, gdyż transmisja odbywa się bezpośrednio do pamięci. Ma to istotne znaczenie ze względu na pracę węzłów bezdyskowych. Systemem operacyjnym jest Unix w wersji BSD4.2 z udoskonaleniami do operowania oknami i do grafiki. Konieczne też były jego modyfikacje w celu zapewnienia obsługi węzłów bezdyskowych, np. do ładowania wstępnego (ang. booting) przez sieć.

Obecnie planuje się dwuetapowe przejście do ujednoliconego Unixa. Pierwszy krok będzie polegał na zachowaniu funkcjonalności wersji BSD4.2 (szczególnie do obsługi sieci) i utrzymania wersji System V na poziomie aplikacyjnym. Drugi krok ma polegać na uzgodnieniu ostatecznej postaci nowego Unixa z firmą ATT i jego wprowadzeniu do eksploatacji. Gdy do tego dojdzie i cena systemu zbliży się do granicy podanej w tym artykule, wtedy nie będzie żadnych wątpliwości, że kolejny model stanowiska SUN będzie spełniał podaną specyfikację (być może, z wyjątkiem wymagania dotyczącego systemu PC-DOS, ale wtedy może ono już nie być aktualne!).

Wydaje się, że istnieje kilka wyrobów przemysłowych, mających podaną w tym artykule charakterystykę indywidualnych stanowisk roboczych. Co więcej, obecne tendencje w technologii i w programowaniu zdają się jasno wskazywać, że istniejące jeszcze wady zostaną szybko usunięte.

Komputery osobiste nie są indywidualnymi stanowiskami roboczymi. Jednakże różnice między nimi są bardzo małe, szczególnie po pojawieniu się modelu IBM RT/PC. Można oczekiwać, że różnice te zanikną całkowicie w ciągu najbliższych dwóch lat, lecz nadal otwarta pozostanie kwestia wyboru systemu operacyjnego. Choć przyszłość systemu PC-DOS jest niejasna, na pewno nie będzie go można zaniechać jeszcze przez pewien czas.

Podczas licznych dyskusji w czasie przygotowywania tego artykułu, wiele cennych uwag z których skorzystał Autor, przedstawił Ian Willers. Treść artykułu stanowi tekst referatu przedstawionego na konferencji CAMAC'86, w Warszawie, 23 kwietnia 1986 r.

WALDEMAR DWORAKOWSKI
Instytut Maszyn Matematycznych
Warszawa

Komputery osobiste IBM PC

Po kilku latach rozwoju techniki mikrokomputerowej, w końcu lat siedemdziesiątych wytworzyła się sytuacja znacznego zróżnicowania w zakresie sprzętu mikrokomputerowego. Brak standardu sprzętowego ograniczył rozwój oprogramowania i prowadził do kompatybilności między komputerami jedynie na niezbyt ściśle zdefiniowanym poziomie systemu operacyjnego. Istnienie kilku różnych standardów zapisu informacji na nośnikach magnetycznych utrudniało przenośność zarówno danych, jak i programów. Brak powszechnie uznawanych standardów w zakresie oprogramowania graficznego utrudniał rozwój zaawansowanego oprogramowania wykorzystującego grafikę jako środek do komunikacji człowiek-maszyna.

W tym okresie najpopularniejszym mikrokomputerowym systemem operacyjnym był CP/M firmy Digital Research, który zapewniał łatwą przenośność, zarówno systemu jak i programów użytkowych, oraz możliwość obsługi także użytkowników nie znających technik komputerowych. Standardem w dziedzinie oprogramowania użytkowego stały się programy WordStar, dBase II, Supercalc, Microsoft Basic. Brak standardu w dziedzinie oprogramowania graficznego oraz niewielkie moce obliczeniowe typowych w tamtym czasie mikrokomputerów utrudniały powstawanie bardziej wyrafinowanych programów użytkowych.

Pojawiające się już wówczas mikrokomputery z 16-bitowym procesorem były niestety rzadkie i nie podporządkowane uznanym standardom. Oczekiwano pojawienia się nowego komputera, który dzięki masowej produkcji mógłby stać się standardem zarówno w zakresie sprzętu jak i oprogramowania.

Mikrokomputerem, który można uznać za taki standard, jest komputer osobisty IBM PC. Jego modułowa konstrukcja umożliwia tworzenie różnych konfiguracji w zależności od potrzeb użytkownika — zarówno obecnych jak i przyszłych. Podstawowa konfiguracja bez napędów dyskowych nie znalazła nabywców, jednak wersja z jednym lub dwoma napędami dyskowymi i z rozszerzoną pamięcią operacyjną przyjęła się powszechnie. Stworzone później moduły znacznie rozszerzyły zakres stosowania tego mikrokomputera.

Jego systemem operacyjnym jest PC-DOS, będący mutacją systemu MS-DOS firmy Microsoft. System ten pod względem użytkowym przypomina CP/M, jednak umożliwia pełne wykorzystanie walorów sprzętu i wydaje się być wygodniejszym w obsłudze, szczególnie dla ludzi nie znających dobrze systemów mikrokomputerowych.

Konstrukcja systemów IBM PC/1 i IBM PC/XT

W artykule skoncentrowano się na bardziej popularnych modelach PC/1 i XT oraz na komputerach kompatybilnych z nimi. Poniższy opis nie pretenduje do przedstawienia wszystkich informacji o tych komputerach, jednak powinien wystarczyć do poznania ogólnych zarysów ich konstrukcji. Zainteresowanych szczegółami można odesłać do bogatej literatury na temat tych komputerów, a w szczególności do dość dobrze opracowanej dokumentacji technicznej firmy IBM.

Konstrukcja systemu IBM-PC jest oparta o płytę główną (ang. motherboard) zawierającą gniazda złącz krawędziowych dla płyt dodatkowych (od 5 do 9). Płyta główna zawiera procesor, koprocesor, pamięć RAM, pamięć ROM, układ czasowy, układ DMA, układ współpracy z klawiaturą, układ dekodowania. Niektórzy wytwórcy na płycie głównej umieszczają także inne układy, np. układ współpracy z dyskiem elastycznym, układ sterownika znakowo-graficznego itp.

Dodatkowe płytki, wkładane w pakiet główny, zawierają sterowniki pozostałych urządzeń lub rozszerzenie pamięci. W związku z ograniczeniami mechanicznymi obudowy, niektóre płytki muszą mieć mniejszą długość. W obudowie znajduje się też zasilacz impulsowy o mocy około 130—150 W z wentylatorem oraz jeden lub dwa napędy dysków elastycznych i (lub) napęd dysku stałego typu Winchester.

Procesor i koprocesor numeryczny

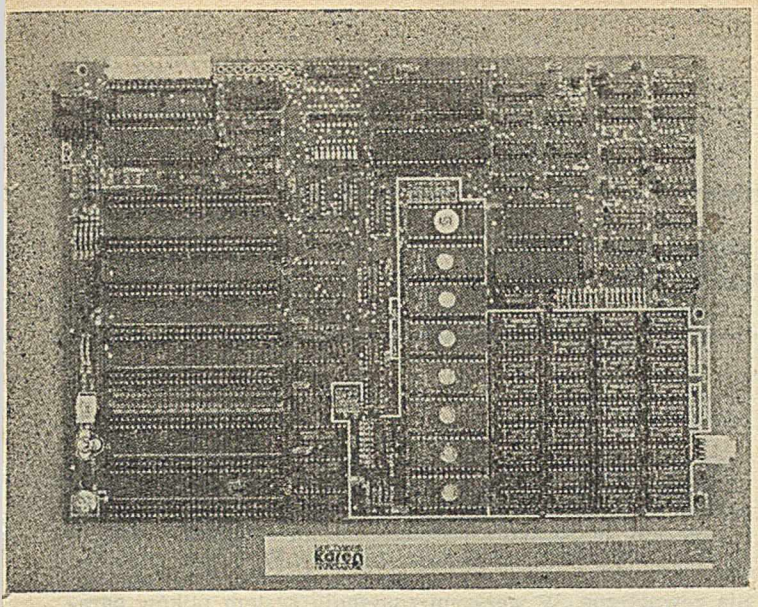
Wewnętrzna konstrukcja komputerów osobistych serii IBM PC/1 i IBM PC/XT opiera się na procesorze 8088, będącym odmianą procesora 8086. Komputer IBM PC/AT, mający lepsze parametry, wykorzystuje procesor 80286, będący rozszerzoną wersją procesora 8086. Procesor 8088 jest zgodny z procesorem 8086 zarówno pod względem architektury jak i listy rozkazów, jednak ma jedynie 8-bitową szynę zewnętrzną, co ogranicza jego moc obliczeniową. Wejściowa kolejka rozkazowa w tym procesorze jest krótsza niż w procesorze 8086 i wystarcza jedynie na przechowywanie jednego długiego rozkazu z parametrami. Konstruktorzy systemu IBM PC zastosowali zegar systemowy o szybkości 4,77 MHz, co spowodowało dalsze ograniczenie mocy obliczeniowej tego mikrokomputera. Istnieją jednak wersje komputera, które w pełni wykorzystują szybkość procesora (np. TURBO-PC) lub posiadają lokalną 16-bitową szynę danych (np. COMPAQ PC).

W oryginalnym IBM PC, jak i w większości jego mutacji, częstotliwość podstawowa zegara 14,31818 MHz jest wykorzystywana do sterowania pakietem sterownika monitora graficznego. Istnieje możliwość regulacji tej częstotliwości w niewielkich granicach w celu uzyskania dokładnego działania zegara systemowego lub w celu uzyskania optymalnego obrazu kolorowego na klasycznych odbiornikach TV. Przez podział tej częstotliwości uzyskuje się sygnał zegarowy procesora i koprocesora.

Koprocesor numeryczny jest opcjonalnym rozszerzeniem możliwości funkcjonalnych tego komputera i znacznie przyspiesza wykonywanie programów operujących liczbami zmiennoprzecinkowymi. Programy muszą być pisane z myślą o wykorzystaniu możliwości koprocesora, a każdy program użytkowy pisany w języku Fortran lub Pascal przez podanie odpowiedniej opcji podczas kompilacji lub przez użycie odpowiedniej biblioteki procedur może używać koprocesora. Zysk wynikający ze stosowania koprocesora zależy od rodzaju programu. W konstrukcji komputera IBM PC użyto koprocesora 8087, zaś w komputerze IBM PC/AT — koprocesora 80287, dopasowanego do procesora 80286. Koprocesor numeryczny może działać współbieżnie z procesorem głównym, a synchronizacja działania odbywa się przez dodatkową linię BUSY/TEST.

Pamięci RAM i ROM

Pamięć RAM zawarta na pakiecie głównym może mieć różną pojemność. Przykładowo, na płytce IBM PC/1 standardowo instaluje się 64 KB pamięci RAM, na płytce IBM PC/XT do 256 KB, zaś na nowszych płytkach (np. tzw.



Megaboard) do 1 MB RAM, z czego tylko 640 KB jest dostępnych bezpośrednio, pozostała część może być wykorzystywana jako tzw. RAM-dysk do symulacji dodatkowego napędu dyskowego w wydzielonym fragmencie pamięci RAM.

Do sprawnego funkcjonowania systemu IBM PC i typowych programów konieczne jest zainstalowanie co najmniej 256 KB pamięci RAM. Wartość maksymalna pojemności pamięci RAM jest określona konstrukcją komputera i wynosi 640 KB. Poprawność działania pamięci RAM jest kontrolowana przy użyciu dodatkowego bitu parzystości na 1 bajt (8 bitów) słowa pamięci. W niektórych komputerach tej klasy pamięć jest wyposażona w układy wykrywania i korekcji błędów.

Pamięć ROM, złożona ze zwykłych elementów EPROM, przechowuje programy obsługi urządzeń standardowych, program służący do ładowania systemu operacyjnego do pamięci RAM, testy podstawowe sprzętu, oraz w niektórych wersjach (np. w oryginalnym IBM PC) interpreter języka Basic, który jest przystosowany do współpracy z taśmą magnetofonową. Niektóre programy wymagają istnienia pamięci ROM z Basikiem, choć jest to dość rzadkie. Istnieją wersje interpretera języka Basic dostosowane do pracy bez instalowania tego interpretera w pamięci RAM, a jedynie w oparciu o dysk.

Poprawność danych zawartych w pamięci ROM nie jest kontrolowana sprzętowo, jednak podczas startu systemu jest obliczana suma kontrolna pamięci ROM. Niestety, w przypadku niezainstalowania pamięci test sumy kontrolnej może nie wykazać błędu. Pojemność pamięci ROM na płytce głównej wynosi od 8 do 40 KB.

Sterownik DMA

Układ sterownika DMA 8237A-5 zastosowany w tym komputerze spełnia wiele funkcji: zapewnia odświeżanie pamięci RAM przez cykliczny odczyt kolejnych bajtów pamięci, umożliwia pracę układu dysku elastycznego oraz dysku stałego Winchester, umożliwia prowadzenie szybkich transmisji przez interface szeregowy lub równoległy interface pomiarowy. Na pakiecie głównym jest używany jedynie kanał układu DMA, pozostałe kanały są wykorzystywane przez sterowniki zawarte na płytkach dodatkowych (np. kanał 2 jest użyty przez układ sterownika dysku elastycznego). Dzięki użyciu układów trójstanowych na liniach żądania transmisji DMA, możliwe jest (przez odpowiednie oprogramowanie) wykorzystywanie tego samego kanału DMA do wielu celów. Układ sterujący transmisją DMA umożliwia równoczesną pracę 4 kanałów.

Ponieważ układ 8237A umożliwia adresowanie pamięci jedynie w zakresie 64 K słów, w IBM PC układ DMA rozbudowano o rejestr (zbiór rejestrów) rozszerzający zakres adresowania do 1 MB. Uproszczona konstrukcja tego układu utrudnia prowadzenie transmisji przez granice stron po 64 KB, jednak jest to rozwiązywane na poziomie systemu operacyjnego. W niektórych komputerach zamiast układu 8237A stosuje się wbudowane w procesor 80188 dwa kanały DMA, jednak zmniejsza to kompatybilność sprzętu.

Układ czasowy i układ równoległy

Działania czasowe w komputerze IBM PC są wykonywane w oparciu o układ 8253-5. Na wejścia zegarowe wszystkich kanałów podany jest sygnał o częstotliwości 1,1993180 MHz. W kanale 0 jest wytwarzane przerwanie okresowe o częstotliwości ok. 18,2 Hz, które służy do pomiaru czasu oraz ułatwia obsługę dysku elastycznego, drukarki i portów szeregowych. Może być także wykorzystywane przez programy użytkowe. Kanał 1 służy do wytwarzania okresowego sygnału odświeżania. Kanał 2 może wytwarzać sygnał akustyczny. Jest on bramkowany sygnałem z portu wyjściowego. Sygnał z kanału 2, oprócz bramkowania w układzie 8253-5, jest też bramkowany i wzmacniany na zewnątrz tego układu i podawany na głośnik.

Układ równoległego we-wy 8255A-5 spełnia następujące funkcje: odczytuje dane przyjęte przez rejestr z klawiatury, wytwarza sygnały sterujące klawiaturą, umożliwia odczytanie 2×8 kluczy, wytwarza sygnały do bramkowania kanału 2 układu 8253-5, do bramkowania sygnału akustycznego, do sterowania silnikiem magnetofonu i do bramkowania układu wykrywania błędów w pamięci RAM. Część wytwórców nie zachowuje pełnej kompatybilności na poziomie sprzętowym, zapewniając jednak kompatybilność na poziomie programu obsługi wejścia-wyjścia BIOS. Niektóre firmy (np. Olivetti) jako układ współpracy z klawiaturą, zamiast układu z rejestrem przesuwnym, stosują mikrokomputer jednoukładowy. Niektóre typy mikrokomputerów mają jedynie jeden zestaw ośmiu mikroprzełączników. W większości mikrokomputerów pomijane są układy współpracy z magnetofonem.

Układ przerwań

Układ przerwań został zaprojektowany w oparciu o element typu I8259A-5. Układ przerwań może przyjmować sygnały przerwań z 8 źródeł. Zgłoszenie przerwania jest dokonywane poprzez zmianę stanu na linii wejściowej ze stanu NIE do stanu TAK. Po wykryciu zgłoszenia wewnętrzny układ priorytetowy wyznacza jeszcze nie obsługiwane przerwanie o najwyższym priorytecie i zgłasza przerwanie do procesora. Procesor (o ile jest w stanie przyjmować przerwania) odczytuje z układu sterownika przerwań numer przerwania, następnie posługując się nim jak wskaźnikiem do tablicy przerwań odczytuje adres i wykona skok ze śladem do procedury obsługi przerwania. W wyniku działania tej procedury musi nastąpić usunięcie przyczyny przerwania. Na końcu procedury obsługi przerwania musi być wysłana do kontrolera przerwań komenda informująca go o zakończeniu obsługi danego przerwania i musi nastąpić powrót do przerwanego programu. W komputerach typu IBM PC/XT przerwaniom sprzętowym odpowiadają przerwania programowe INT 08H do INT 0FH.

* * *

Artykuł ten jest bardzo ogólnym wprowadzeniem w sprzęt komputerów osobistych, jednak wystarcza do poznania konstrukcji komputera IBM PC. Dokładne informacje o poszczególnych elementach składowych tego komputera można znaleźć w dość precyzyjnej dokumentacji firmowej lub w publikacjach specjalistycznych. W następnej części artykułu zostaną przedstawione płyty dodatkowe komputera IBM PC.

ROMAN GRABOWICZ
JANUSZ ZALEWSKI
Warszawa

System operacyjny PC-DOS (I)

Coraz szersze rozpowszechnienie w Polsce komputerów IBM PC i nowszych odmian powoduje, że znacznie wzrasta zainteresowanie ich użytkowaniem. Nie istnieje jednak w języku polskim dobra i łatwo dostępna dokumentacja tych komputerów ani ich oprogramowania. Celem tego artykułu jest zaznajomienie Czytelnika z podstawowymi pojęciami przeznaczonego dla tych komputerów systemu operacyjnego PC-DOS w takim zakresie, aby ułatwić samodzielne używanie podręczników firmowych. Ze względu na ograniczoną objętość tekstu, niniejsze omówienie jest skrótowe i z konieczności nie obejmuje wszystkich zagadnień.

System operacyjny PC-DOS jest oparty na bardzo podobnym systemie firmy Microsoft zwanym MS-DOS, który powstawał w drodze ewolucji od 1980 roku z systemów 86-DOS i wcześniejszego QDOS. Celem autora systemu QDOS było opracowanie systemu operacyjnego dla mikrokomputerów 16-bitowych, pozbawionego ograniczeń systemu CP/M, przy zachowaniu z nim jak największej zgodności. Choć PC-DOS jest własnością firmy IBM, utrzymuje się, że istnieje jego pełna zgodność z poprzednikiem (MS-DOS) na poziomie oprogramowania aplikacyjnego do wywołań funkcji włącznie. Poniższy opis oparto na wersji 2.00, aczkolwiek są już rozpowszechniane nowsze wersje.

Pełny opis systemu operacyjnego PC-DOS powinien objąć sześć dokumentów:

1) BIOS, tj. podstawowy system wejścia-wyjścia (ang. Basic Input-Output System), zapisany w pamięci ROM, a więc wbudowany w komputer i w zasadzie niezależny od systemu operacyjnego

2) program samoładujący (ang. bootstrap loader), zapisany w pierwszym sektorze dysku systemowego, służący do zainicjowania procesu ładowania dwóch plików zawierających główną część systemu operacyjnego

3) plik dyskowy IBMBIO.COM, będący wymiennym rozszerzeniem modułu BIOS, umożliwiającym dołączenie programów obsługi urządzeń (ang. device drivers), nie przewidzianych w zestawie podstawowym; stanowi sprzężenie z jądrem systemu operacyjnego

4) plik dyskowy IBMDOS.COM, stanowiący jądro systemu operacyjnego, przede wszystkim zarządzający plikami oraz buforami dyskowymi i zawierający kod poszczególnych funkcji systemowych [1])

5) plik dyskowy COMMAND.COM, stanowiący interpreter poleceń rezydentnych, tj. wbudowanych do systemu operacyjnego (ang. internal commands), będący jednocześnie programem ładującym do pamięci operacyjnej polecenia nierezydentne

6) samodzielne programy usługowe, uzupełniające system operacyjny, stanowiące tzw. polecenia nierezydentne.

Do wstępnego zapoznania się z systemem PC-DOS i do jego praktycznego użytkowania wystarczy znajomość poleceń operatorskich, dlatego w poniższym opisie nie poruszono zagadnień objętych punktami 1—4. Aby przyswoić zasady użytkowania systemu operacyjnego, należy najpierw zapoznać się z organizacją systemu plików, gdyż nie jest ona konwencjonalna, a w każdym razie nie spotykana dotąd w komputerach rozpowszechnionych w Polsce.

[1]) Funkcje systemowe, których kod zawiera plik IBMDOS.COM realizują stosunkowo złożone operacje, w których wykorzystuje się elementarne funkcje zawarte w pliku IBMBIO.COM bądź w module BIOS.

SYSTEM PLIKÓW

Niżej opisana postać systemu plików występuje dopiero od wersji PC-DOS 2.00 i zawiera kilka koncepcji przeniesionych z systemu Unix.

Plik jest tu zorganizowanym zbiorem informacji dostępnych w określonym fizycznie miejscu. W takim rozumieniu system plików obejmuje oprócz dysków inne urządzenia zewnętrzne. W węższym znaczeniu system plików ogranicza się do plików dyskowych. Pliki dyskowe mogą znajdować się na czterech stacjach dyskowych oznaczonych literami:

A: — pierwsza stacja dyskietek
B: — druga stacja dyskietek
C: — pierwszy dysk stały
D: — drugi dysk stały

Identyfikatory plików

Identyfikator pliku składa się z dwóch części: nazwy właściwej pliku (ang. filename) oraz rozszerzenia nazwy pliku. Nazwa składa się maksymalnie z 8 znaków. Po nazwie może wystąpić kropka "." oraz rozszerzenie nazwy, które może mieć maksymalnie 3 znaki długości. W nazwie i jej rozszerzeniu mogą wystąpić następujące znaki: litery, cyfry, $, , #, &, !, %, ', ', (,), —, {, }, —'

Przykładowymi nazwami plików są:

MAGAZYN.MAG
INCOME%5
NAZ-WA-.0

Tabela 1. Nazwy zastrzeżone dla urządzeń

Zastrzeżona nazwa	Urządzenie
CON	Klawiatura lub ekran
AUX lub COM1	Pierwszy port asynchronicznego układu transmisji (ang. Asynchronous Communication Adapter port)
COM2	Drugi port asynchronicznego układu transmisji
LPT1 lub PRN	Pierwsza drukarka (port równoległy) tylko jako urządzenie wyjściowe
LPT2	Druga drukarka (port równoległy)
LPT3	Trzecia drukarka (port równoległy)
NUL	Zaślepka — nieistniejące fizycznie urządzenie

Tabela 2. Inne zastrzeżone nazwy plików

Nazwa	Znaczenie
% PIPE	Pliki tymczasowe na dane w potokach
$	Dowolny plik rozpoczynający się od $, używany przez konsolidator w trybie odpowiedzi automatycznych
FILEnnnn.CHK	Stracone fragmenty pliku (ang. cluster) odzyskane poleceniem CHKDSK
FILEnnnn.REC	Identyfikatory plików odtworzone poleceniem RECOVER
VM.TMP	Plik utworzony przez konsolidator, gdy brak miejsca w pamięci operacyjnej na moduł ładowania

Pewne nazwy mają specjalne znaczenie dla systemu DOS. Część z nich oznacza standardowe urządzenia zewnętrzne (tab. 1). Z tego powodu nie mogą one być używane jako nazwy plików, choć w niektórych poleceniach można używać urządzeń zewnętrznych w roli plików. Inne zarezerwowane nazwy plików przedstawiono w tabeli 2, a zarezerwowane rozszerzenia nazw — w tabeli 3.

Tabela 3. Zarezerwowane rozszerzenia nazwy

Rozszerzenie	Znaczenie
.$$$	Plik roboczy (utworzony np. przez edytor EDLIN)
.ASM	Plik źródłowy asemblera
.BAK	Kopia pliku źródłowego używanego przez edytor EDLIN
.BAS	Plik z programem źródłowym w Basicu
.BAT	Plik wsadowy
.BIN	Plik binarny
.COM	Polecenie systemu PC-DOS
.EXE	Plik wykonywalny, tworzony przez konsolidator
.HEX	Plik znakowy zakodowany szestnastkowo; można go przekształcić na .COM
.LIB	Plik biblioteczny
.LST	Listing z asemblera
.MAP	Mapa pamięci z asemblera
.OBJ	Wynikowy kod z asemblera
.REF	Mapa odwołań
.TMP	Plik tymczasowy konsolidatora
.CHK	Patrz tabela 2
.REC	Patrz tabela 2

Do identyfikowania plików można używać tzw. zmiennych identyfikatorów (ang. wildcards):

? — który zastępuje pojedynczy znak

* — który zastępuje ciąg znaków.

Przykładowo, identyfikator

TEST?.COM

oznacza każdy z następujących plików:

TEST1.COM

TEST2.COM

TEST7.COM

TESTA.COM

a być może jeszcze inne, jeżeli występują.
Podobnie identyfikator

TEST.*

oznacza wszystkie z następujących plików

TEST.COM

TEST.EXE

TEST.BAT

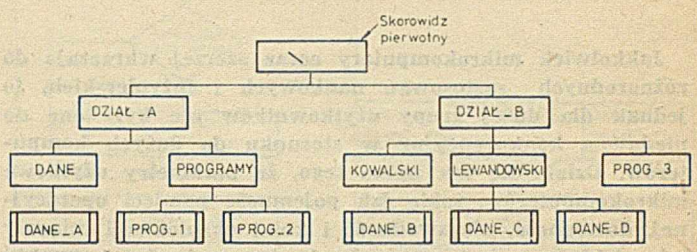
itd., tzn. wszystkie pliki o nazwie TEST i dowolnym rozszerzeniu. Zmiennych identyfikatorów należy używać dość ostrożnie, gdyż można omyłkowo skasować wszystkie pliki na dysku używając odpowiedniego polecenia z argumentem:
.

Hierarchiczna struktura systemu plików

Wprowadzenie mikrokomputerów z dyskami stałymi spowodowało, że współczesny użytkownik ma do czynienia z dziesiątkami lub setkami plików na jednym nośniku. Utrudnia to poważnie sprawne gospodarowanie dyskiem zarówno samemu użytkownikowi jak też systemowi operacyjnemu.

System PC-DOS ma tę zaletę, że umożliwia tworzenie hierarchicznej (drzewiastej) struktury systemu plików. Skorowidze w tej strukturze zawierają pliki bądź inne skorowidze. Każdy skorowidz niższego poziomu może zawierać mieszaninę plików oraz innych skorowidzów.

Na każdym dysku znajduje się jeden skorowidz nadrzędny zwany skorowidzem pierwotnym lub systemowym (ang. root). Tworzony jest on automatycznie podczas formatowania dysku. W wypadku dyskietek jest on zarazem jedynym skorowidzem, w którym liczba pozycji (plików lub skorowidzów) jest ograniczona do 64 (dla dyskietek z zapisem jednostronnym) lub 112 (dla dyskietek z zapisem dwustronnym). Wszystkie skorowidze oprócz skorowidza systemowego są także plikami, lecz ich zawartość stanowi opis plików i innych skorowidzów. Przykładową strukturę plików na dysku dla dwóch działów instytucji przedstawiono na rysunku.



Przykładowa struktura plików dla dwóch działów instytucji

Pliki o tych samych nazwach mogą występować w różnych skorowidzach. Aby określić, w którym skorowidzu znajduje się określony plik, należy jednoznacznie określić tzw. ścieżkę dostępu do pliku, czyli listę skorowidzów, począwszy od skorowidza systemowego, przez którą należy przejść, aby dotrzeć do danego pliku. Nazwy kolejnych skorowidzów łączy się znakiem ukośnika (ang. backslash) [2]. Przykładowo, dla pliku DANE_B ścieżką dostępu jest ciąg:

/DZIAL_B/LEWANDOWSKI/DANE_B

Początkowy znak informuje, że ścieżka dostępu rozpoczyna się od skorowidza systemowego danego dysku. Pominięcie tego znaku oznacza, że ścieżka dostępu rozpoczyna się w tzw. skorowidzu bieżącym, przyjmowanym domyślnie. Po załadowaniu sytemu operacyjnego, jako skorowidz bieżący przyjmowany jest na ogół skorowidz systemowy.

W każdym skorowidzu, z wyjątkiem skorowidza systemowego, zawsze znajdują się dwie pozycje symbolizowane kropkami:
. oznacza skorowidz bieżący
.. oznacza skorowidz bezpośrednio nadrzędny.
Obu symboli można używać tak jak nazw innych skorowidzów.

Pełna specyfikacja pliku obejmuje:
— symbol dysku (opcjonalnie)
— ścieżkę dostępu (opcjonalnie)
— nazwę pliku (obligatoryjnie)
— rozszerzenie nazwy pliku (opcjonalnie)
na przykład:

A:/DZIAL_A/KOWALSKI/PROGRAM1.BAS

* * *

System PC DOS zawiera grupę poleceń operowania skorowidzami:

CHDIR — zmiana skorowidza

MKDIR — utworzenie skorowidza

PATH — specyfikacja ścieżki dostępu

RMDIR — usunięcie skorowidza

TREE — wyświetlenie istniejącej struktury skorowidzów.

Bardziej szczegółowy opis poleceń do operowania skorowidzami i plikami zwykłymi zostanie przedstawiony w drugiej części artykułu.

---

[2]) 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.

MICHAŁ KLEIBER
MICHAŁ SARAN
Warszawa

Doświadczenia z użytkowania Fortranu na mikrokomputerze IBM PC

Jakkolwiek mikrokomputery coraz szerzej wkraczają do różnorodnych zastosowań naukowych i inżynierskich, to jednak dla dużej grupy użytkowników nie były one do niedawna konkurencyjne w stosunku do dużych komputerów. Działo się tak mimo tego, że parametry użytkowe mikrokomputerów, takie jak pojemność pamięci operacyjnej, dostępność, niezawodność i koszt eksploatacji osiągały lub niekiedy nawet przewyższały (przynajmniej z punktu widzenia pojedynczego użytkownika) te same parametry dużych komputerów (ang. mainframe). Niewątpliwie zasadniczą przyczyną początkowego braku zainteresowania naukowców i inżynierów mikrokomputerami był brak dla tego sprzętu efektywnego kompilatora języka Fortran, który mimo swych znanych wad jest, i chyba jeszcze długo będzie, podstawowym narzędziem do obliczeń numerycznych w wielu dyscyplinach naukowo-technicznych. Tworzone od dawna kompilatory Fortranu dla mikrokomputerów miały niestety pewne zasadnicze niedogodności, w znacznym stopniu utrudniające stosowanie tego języka.

Sytuacja zmieniła się radykalnie w momencie pojawienia się kompilatora Fortranu firmy Microsoft — MS-FORTRAN wersja 3.20 [1, 2]. Wyeliminowano w nim szereg podstawowych wad wersji poprzednich tego kompilatora. Jest on obecnie w pełni porównywalny z kompilatorami dla komputerów dużych, ma nawet pewne rozszerzenia i ulepszenia. Dla użytkowników posiadających zaawansowane oprogramowanie w FORTRANIE jest on dodatkowo atrakcyjny ze względu na możliwość łatwego przenoszenia tego oprogramowania na IBM PC.

Artykuł niniejszy jest próbą pierwszej oceny przydatności kompilatora MS-FORTRAN 3.20 w zastosowaniach naukowo-technicznych. Autorzy podsumowują w nim doświadczenia zebrane w ciągu kilkunastomiesięcznej, intensywnej pracy przy tworzeniu oprogramowania fortranowskiego na różnych mikrokomputerach, zgodnych z IBM PC i IBM PC/XT (Compaq, IBM, Olivetti, Redstone).

MOŻLIWOŚCI JĘZYKA MS-FORTRAN

MS-FORTRAN (w skrócie MS-F) jest rozszerzeniem podzbioru Fortranu ANSI, a ponadto ma szereg cech pełnego standardu ANSI, znanego jako FORTRAN 77. Ma też pewne drobne rozszerzenia w stosunku do tej wersji. Ponadto jest wyposażony w metajęzyk, będący językiem sterującym dla kompilatora. Przy użyciu jego dyrektyw można, na przykład, sterować tworzeniem listingu kompilacji lub poziomem śledzenia błędów. Generalnie można stwierdzić, że możliwości kompilatora MS-F 3.20 są bardzo duże.

Z punktu widzenia użytkownika programującego w Fortranie na Odrze lub R-32 można uwypuklić kilka różnic. Pierwsza różnica dotyczy grupy instrukcji we-wy. W MS-F dostępny jest swobodny format wejściowy w następującej, wygodnej postaci:

READ (nr., *) lista we

Kolejne liczby ze zbioru danych są podstawiane pod zmienne listy, przy czym możliwa jest konwersja liczb typu integer na zmienne typu real.

Oprócz podstawowych instrukcji we-wy (READ, WRITE) istnieją też instrukcje operowania plikami zewnętrznymi, takie jak OPEN, CLOSE itd. Dozwolone jest tworzenie plików nazwanych i roboczych, zredagowanych i niezredagowanych, sekwencyjnych i bezpośredniego dostępu. Bardzo cenna jest możliwość przypisywania typu urządzenia we-wy dopiero w trakcie wykonywania programu, co pozwala na zmiany tych urządzeń bez powtórnej kompilacji i konsolidacji.

W MS-F wyróżnia się 6 podstawowych typów zmiennych:
— integer (16 i 32 bity)
— single precision real (32 bity)
— double precision real (64 bity)
— logical (16 i 32 bity)
— complex (64 lub 128 bity)
— znakowy CHARACTER (od 1 do 127 znaków).

Wyróżnione są więc zmienne typu znakowego, na których można wykonywać wszystkie operacje. Ułatwia to znacznie przechowywanie tekstu, np. zamiast następujących sekwencji:

```
      REAL TE(3)
      DATA TE/8HWYNIKI ,8HDLA KROK,5HU NR.)
      WRITE (6,100) (TE(I),I=1,3)
100   FORMAT (2A8,A5)
```

w MS-F należy napisać:

```
      CHARACTER TE*21
      DATA TE /"WYNIKI DLA KROKU NR."/
      WRITE (6,100) TE
100   FORMAT (A21)
```

lub jeszcze prościej:

```
WRITE (6,"(A)") "WYNIKI DLA KROKU NR."
```

Z drugiej jednak strony należy pamiętać o zmiennych typu znakowego, przy przenoszeniu programów z innych komputerów, bowiem — na przykład — próba podstawienia tekstu pod zmienną typu REAL, jak podano na początku ostatniego przykładu, nie będzie akceptowana, jeśli nie użyje się odpowiedniej dyrektywy.

Jeśli chodzi o instrukcje sterujące, to użytkownik takich komputerów jak Odra zauważy tu pewne rozszerzenia w postaci instrukcji ELSE, ELSE IF, END IF, przy użyciu których można konstruować blok o przykładowej postaci:

Dr MICHAŁ KLEIBER jest profesorem w IPPT PAN, specjalistą w dziedzinie zastosowań metod numerycznych w nieliniowej mechanice ciał odkształcalnych. Wykładał przez dłuższy czas na Uniwersytecie w Stuttgarcie (RFN) oraz Uniwersytecie w Berkeley (USA). Jest autorem książki „Metoda elementów skończonych w nieliniowej mechanice kontinuum", PWN, 1985 oraz współautorem książki „Nieliniowa mechanika konstrukcji". PWN, 1982.

Dr MICHAŁ SARAN jest adiunktem w Laboratorium Informatyki Instytutu Dróg i Mostów Politechniki Warszawskiej. Prowadzi zajęcia z zakresu programowania komputerów i zastosowania metod komputerowych w mechanice konstrukcji. Jest autorem systemów numerycznej analizy konstrukcji, opublikował wiele prac badawczych z tego zakresu.

```
IF(C.EQ."A")THEN
  CALL SUB1
ELSE
  CALL SUB2
ENDIF
```

Podsumowując można stwierdzić, że MS-F ma bogate możliwości, jest bardzo zbliżony do Fortranu dla dużych komputerów, a przy przenoszeniu oprogramowania ewentualne kłopoty mogą dotyczyć operacji na znakach i innej postaci formatu swobodnego. Zmienia się też sposób definiowania plików zewnętrznych (jeśli są używane). Żaden z tych problemów nie jest trudny do rozwiązania, szczególnie gdy ma się do dyspozycji bardzo dobre edytory tekstu dostępne na IBM PC.

WYMAGANIA SPRZĘTOWE

Minimalne wymagania MS-F podane przez producenta to 140 KB wolnej pamięci RAM i jedna stacja dyskietek. Praktycznie przy tej konfiguracji praca jest niewygodna i jako rozsądne minimum można polecić dwie stacje dwustronnych dyskietek oraz 256 KB pamięci RAM. Przy opracowywaniu bardziej zaawansowanych, dużych programów (rzędu tysięcy linii), znacznie ułatwia i przyspiesza pracę posiadanie twardego dysku lub dużej pamięci, umożliwiającej założenie dysku RAM, a najlepiej — obu tych urządzeń jednocześnie. Dla orientacji można podać wielkości poszczególnych programów, biorących udział w procesie tworzenia programu binarnego: system operacyjny DOS 2.0 — 40 KB, FOR1.EXE — 120 KB, PAS2.EXE — 98 KB, LINKER — 40 KB, FORTRAN.LIB — 86 KB, MATH.LIB — 36 KB (w wypadku posiadania koprocesora, tę ostatnią bibliotekę należy zastąpić biblioteką 8087.LIB zajmującą 19 KB).

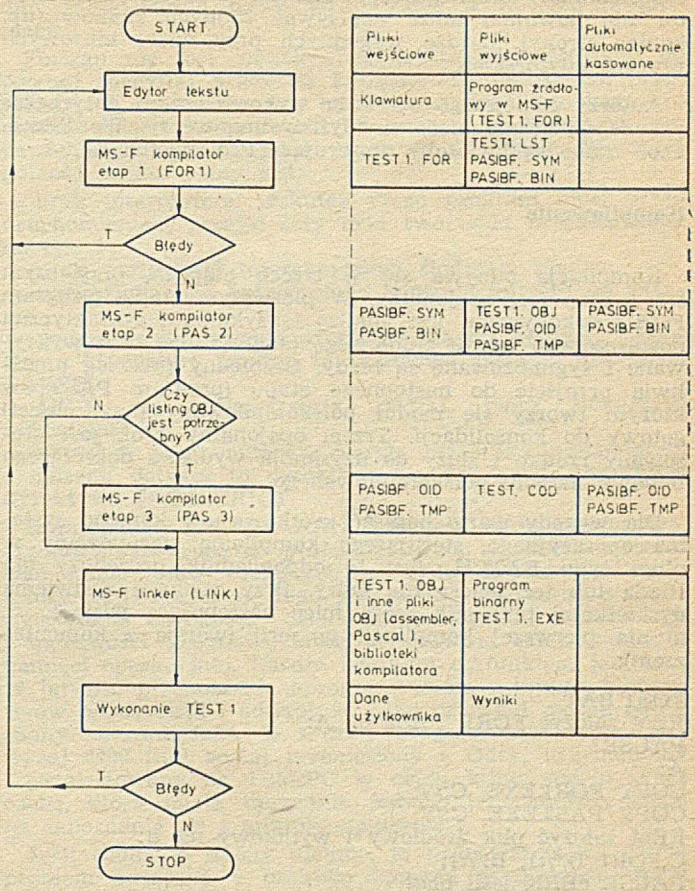
Oprócz tych programów musi być jeszcze miejsce na edytor, moduły źródłowe, program binarny i ewentualnie pliki robocze tworzone przez kompilator i linker (konsolidator). Jest oczywiste, że nie wszystkie one muszą znajdować się w pamięci jednocześnie.

TWORZENIE PROGRAMU

Proces tworzenia i testowania programu wykonywalnego (binarnego) przedstawiono schematycznie na rys. 1. Program źródłowy, tworzony z wykorzystaniem edytora tekstu, jest kompilowany i konsolidowany na program wykonywalny, który może być uruchomiony z różnymi zestawami danych.

W wypadku posiadania dużego programu źródłowego najkorzystniej jest podzielić go na segmenty, które powinny być tworzone, poprawiane i kompilowane oddzielnie. Zasada ta jest dobrą praktyką doświadczonych programistów także na dużych komputerach, a na komputerach osobistych jej zalety uwypuklają się szczególnie silnie, tym bardziej że cały system jest przystosowany do tego trybu pracy, a wielkość programu wynikowego oczywiście nie zmienia się. Konkretnym ograniczeniem jest tylko wielkość segmentu binarnego, tworzonego przez kompilator, która nie może przekraczać 64 KB. Jest to jednak równoważne średnio około 1000 linii programu źródłowego, a więc w świetle powyższych uwag nie stanowi istotnego ograniczenia.

Drugi problem wiąże się z odpowiedzią na pytanie, gdzie najlepiej umieszczać program redagujący moduły źródłowe, kompilator, moduły półskompilowane, linker i program binarny w trakcie procesu tworzenia programu, pamiętając dodatkowo o miejscu na pliki robocze, tworzone przez kompilator i linker. Odpowiedź nie może być jednoznaczna, zależy bowiem od wielu czynników takich, jak dostępna konfiguracja (liczba stacji dyskietek, posiadanie twardego dysku, możliwość założenia dysku RAM), wielkość programu źródłowego lub faza tworzenia programu. Pewne wnioski można wyciągnąć na podstawie danych liczbowych podanych niżej. Generalnie można powiedzieć, że należy optymalizować czas pracy przez minimalizację czasu przebiegu programów oraz liczby zmian dyskietek. Pamiętając, że najkrótszy czas dostępu do informacji ma dysk RAM, następnie zaś dysk twardy, należy świadomie zaplanować miejsca poszczególnych plików, starając się umieszczać na dysku RAM wszystkie pliki robocze. W wypadku posiadania wyłącznie stacji dyskietek, jedna z nich powinna być przeznaczona na moduły źródłowe, półskompilowane i program wynikowy, a druga — na edytor, kompilator i linker. W tym ostatnim wypadku nie uniknie się jednak przy dużych programach żmudnego przekładania dyskietek.



Proces tworzenia programu w systemie MS-FORTRAN

Poniżej zamieszczono przykład ilustrujący to zagadnienie dla zestawu: 2 stacje dyskietek i dysk RAM zawierający komplet instrukcji oraz odpowiedzi systemu pojawiających się kolejno na ekranie. Przed uruchomieniem założono dysk RAM (C:), na dyskietce A: umieszczono FOR1, PAS2, LINK i biblioteki, a na dyskietce B: program źródłowy.

```
A> COPY FOR1.EXE,C:  return
A> COPY PAS2.EXE,C:  return
A> C: return
C> FOR1 return
source file [.FOR] B:TEST1 return
object file [TEST.OBJ] B: return
source listing [NUL.LST] return
object listing [NUL.COD] return
  pass one    no errors detacted
C> PAS2  return
code area size=.....
cons area size=.....
date area size=.....
  pass two   no errors detected
C> A:LINK  return
object modules [.OBJ] B:TEST1 return
run file [TEST1.EXE] B: return
list map [NUL.MAP] return
libraries [.LIB] A: return
C> B: TEST1
```

Redagowanie tekstu

Na komputerach osobistych dostępnych jest obecnie wiele edytorów tekstu, od najprostszych edytorów wierszowych aż po bardzo efektywne i wygodne, wyspecjalizowane programy edytorskie. Pozwalają one w bardzo prosty sposób wykonywać skomplikowane operacje z przetworzonym tekstem. Umożliwiają przesuwanie tekstu na ekranie w górę i w dół z wybraną szybkością, wpisywanie, kasowanie i zamianę znaków w dowolnym miejscu lub w całym pliku, wycinanie, przenoszenie bądź wstawienie określonej liczby linii przy użyciu kursora, bez konieczności podawania numerów linii. Nie wszystkie procesory tekstowe (ang. word procesors) nadają się do redagowania w Fortranie.

Konieczna jest możliwość redagowania plików o niezdefiniowanej długości oraz możliwość szybkiego przeszukiwania i zamiany zadanego ciągu znaków w całym pliku. Edytor nie powinien także dopisywać żadnych znaków (np. koniec strony), bardzo wygodnych przy wielu biurowych pracach edytorskich.

Autorzy niniejszego artykułu wykorzystywali dotychczas prawie wyłącznie nieco zmodyfikowaną wersję WordStara, choć znane są im osoby preferujące Professional Editor.

Kompilowanie

Kompilacja odbywa się w trzech etapach, przy czym trzeci etap jest opcjonalny. W pierwszym etapie (program FOR1) dokonywana jest analiza leksykalna i syntaktyczna dostarczonego tekstu źródłowego (ang. source) oraz wykrywane i sygnalizowane są błędy. Bezbłędny przebieg umożliwia przejście do następnego etapu (program PAS2), w którym tworzy się moduł półskompilowany (ang. object) gotowy do konsolidacji. Trzeci opcjonalny krok jest stosowany rzadko i służy do uzyskania wydruku dotyczącego budowy modułu półskompilowanego.

Dla wygody warto napisać krótki zestaw komend systemu operacyjnego, sterującego kompilacją, zgrupowany w pliku typu BATCH. Jest to odpowiednik procedury dla Riada lub też macra dla Odry. Przy zestawie z dwiema dyskietkami taki plik może mieć następującą postać:
a) dla pierwszej kompilacji w serii (wersja z komentarzem):

```
FOST.BAT
REM założyć FOR1, PAS2 na A:
PAUSE
A:
COPY FOR1.EXE, C:/V
COPY PAS2.EXE, C:/V
REM założyć plik źródłowy i wynikowy na B:
C:FOR1 B:%1, B:%1;
PAUSE ęBRK jeśli błąd
C:PAS2 C
```

b) dla następnych kompilacji (wersja bez komentarza, szybsza, choć mniej elegancka):

```
FO.BAT
C:
FOR1 B:%1, B:%1,
PAS2 C
```

Na klawiaturze pisze się wtedy tylko parametr będący nazwą segmentu, co przyspiesza pracę i zwiększa jej wygodę (np. FO.sub1).

Mając tak opracowane narzędzie można przystąpić do kompilacji poszczególnych segmentów. Na początku należy kompilować przy użyciu dyrektywy $DEBUG, która ustawia wysoki poziom śledzenia błędów. Są wtedy wykrywane np. błędy dzielenia przez zero i sygnalizowane łącznie z informacją, w której linii programu źródłowego występują. Oczywiście, obszar zajęty przez program jest wtedy większy (o ok. 40—50%), a czas wykonywania dłuższy, więc po osiągnięciu poprawnych przebiegów testowych należałoby powtórnie skompilować i skonsolidować program bez tej opcji.

Czas kompilacji zależy od użytego zestawu komputerowego. Eksperymentalnie kompilowano program złożony z szeregu procedur o długości 439 aktywnych linii (bez komentarzy) dla trzech różnych rodzajów pamięci zewnętrznej. Należy podkreślić, że jednorazowa kompilacja tak długiego tekstu jest nieekonomiczna i została wykonana tylko w celu uzyskania wyraźnych różnic czasowych. Wyniki zamieszczono w tabeli 1. Obliczenia wykonano dla systemu operacyjnego DOS 2.0 i kompilatora MS-F 3.20.

Z tab. 1 wynika, że dysk RAM przyspiesza kompilację prawie dwukrotnie w stosunku do dysków elastycznych. Trzeba jednak stwierdzić, że dysk twardy jest wygodniejszy w użyciu przy dużej liczbie modułów do kompilacji, bowiem nie zachodzi wtedy konieczność częstej wymiany dyskietek. Należy pamiętać jednak, że ponieważ obecnie rzadkością jest posiadanie urządzenia do wykonywania kopii (składowania) zawartości twardego dysku, zdarzają się wypadki nieumyślnego skasowania jego zawartości (najczęściej przez nieumiejętne użycie komendy FORMAT), bez możliwości jej odtworzenia.

Tabela 1. Czas komplikacji programu testowego dla trzech rodzajów pamięci zewnętrznej

Rodzaj dysku	Elastyczny [min:sek]	Twardy [min:sek]	RAM [min:sek]
Ładowanie FOR1	0:13	0:05	0:02
Przebieg etapu 1	1:25	0:52	0:44
Ładowanie PAS2	0:08	0:03	0:01
Przebieg etapu 2	4:55	3:31	3:05
Pełna kompilacja	6:41	4:31	3:52

Tworzenie programu binarnego (konsolidacja)

Jak wynika z rys. 1 danymi dla linkera są poszczególne moduły półskompilowane, które mogą być pisane w różnych językach (np. Fortran, Pascal, Assembler, ale nie Basic) oraz biblioteka kompilatora. Praktycznie nie ma ograniczenia liczby konsolidowanych modułów. Jedynym ograniczeniem jest dostępna pamięć operacyjna. Dla większej liczby modułów, wygodnie jest je zapisać w pliku, aby nie wyliczać dużej liczby nazw modułów przy każdej konsolidacji. Czasy konsolidacji programu tekstowego są zamieszczone w tabeli 2.

Tabela 2. Czasy konsolidacji programu testowego

Rodzaj dysku	Elastyczny [min:sek]	Twardy [min:sek]	RAM [min:sek]
Ładowanie	0:05	0:02	0:01
Przebieg	3:18	1:42	1:38
Pełna konsolidacja	3:23	1:44	1:39

Jedną z podstawowych zalet kompilatora MS-F 3.20 jest możliwość użycia koprocesora zmiennoprzecinkowego 8087. Aby ocenić wartość tego urządzenia w obliczeniach naukowo-technicznych, wystarczy podać, że przyspiesza ono czas wykonania programu kilkadziesiąt razy. Ponadto, co ma też ogromne znaczenie, operuje on słowami o długości 80 bitów, a więc stosowanie podwójnej precyzji nie zwiększa czasu wykonania programu.

W związku z możliwością użycia koprocesora 8087 istnieje kilka sposobów kompilowania i konsolidacji, dających w efekcie programy o różnych wielkościach i różnych czasach wykonania. Wiąże się to z użyciem różnych bibliotek kompilatora o nazwach MATH.LIB i 8087.LIB, oraz dyrektywy $NO/FLOATCALLS.

Tabela 3. Długość i czas wykonywania programu testowego dla różnych opcji kompilatora i linkera

	Długość [byte]	Czas (dysk RAM) [min:sek]
Kompilacja z $NOFLOATCALLS	30355	3:38
Konsolidacja z 8087. LIB	91168	1:17
Konsolidacja z MATH. LIB	100836	1:31
Kompilacja z $FLOATCALLS	31761	3:52
Konsolidacja z 8087. LIB	94304	1:20
Konsolidacja z MATH. LIB	103424	1:39

Jeśli układ 8087 jest dostępny, to użycie biblioteki 8087.LIB powoduje utworzenie najkrótszego i najszybszego programu binarnego, jednak programy tak konsolidowane mogą być wykonywane tylko na komputerach z koprocesorem. W wypadku braku koprocesora, należy konsolidować program z biblioteką MATH.LIB. Takie programy będą używać koprocesora, jeśli są wykonywane na komputerze z tym układem, jednak czas wykonywania będzie dłuższy niż po konsolidacji z 8087.LIB. Tabela 3 obrazuje otrzymane długości programów testowych i czasy ich wykonywania dla omówionych opcji kompilatora i linkera.

Interesujące jest porównanie czasów wykonania programu testowego, działającego na macierzach, dość charakterystycznych w obliczeniach naukowo-technicznych. Tekst programu jest następujący:

```
      DIMENSION A(3),B(1,1,1,1)
      DATA A,B/4*2.3/
      READ (5,*)L
      DO 10 I=1,L
10    A(I/I+2)=A(I/I+1)/A(1)*(2)+B(1,1,1,1)
      WRITE (6,2)
2     FORMAT ('KONIEC')
      STOP
      END
```

Program ten uruchomiono na różnych komputerach osobistych z podwójną precyzją, bez opcji $DEBUG, oraz na Odrze 1305 z opcją TRACE 1 i pojedynczą precyzją. Uzyskane czasy przebiegu przedstawiono w tab. 4.

Tabela 4. Czasy wykonania programu działającego na macierzach [minisek]

Długość pętli — L	Compaq PC bez koprocesora	Olivetti M24 bez koprocesora	Compaq PC z koprocesorem	Odra 1305
10000	0:33	0:15	0:05	0:05
100000	5:18	2:22	0:47	0:45

ZALETY I WADY MS-F

Do najważniejszych zalet omawianej wersji Fortranu należą następujące cechy:

- Dobra zgodność ze standardem, co umożliwia przenoszenie prawie bez modyfikacji oprogramowania z dużych komputerów i odwrotnie.
- Kompilator MS-F tworzy bardzo szybki program binarny z operacjami zmiennoprzecinkowymi, podstawowymi dla zastosowań naukowo-technicznych. Możliwość użycia koprocesora 8087 eliminuje jedną z ostatnich barier na drodze do tworzenia złożonego oprogramowania. Jak wykazano, czas wykonania programu jest porównywalny z czasem zajętości jednostki centralnej na Odrze, tak więc biorąc pod uwagę tryb pracy na komputerze osobistym użytkownik otrzymuje wyniki szybciej.
- Umożliwienie przez MS-F 3.20 deklarowania dużych tablic, przekraczających 64 KB (co nie było możliwe w poprzednich wersjach tego kompilatora), a tym samym udostępnienie użytkownikowi bardzo dużego obszaru pamięci operacyjnej mogącego dojść do 600 KB.
- Kompilator MS-F tworzy program wynikowy o małej zajętości pamięci, w stosunku do długości programu źródłowego, co pozwala uruchomić stosunkowo duże programy nawet na zestawach o pamięci 256 KB.
- Fortran jest językiem bardzo popularnym i mimo swoich wad ma pewną przewagę w zastosowaniach nad Pascalem lub językiem C. Wynika to nie tylko z przywiązania programistów do tego języka, ale także z olbrzymich zasobów istniejących i gruntownie przetestowanych procedur, które można łatwo wykorzystać w nowych programach.

Fortran nie jest jednak zbyt dobrze przystosowany do pewnej specyfiki pracy na komputerach osobistych, przejawiającej się możliwością bardzo bezpośredniego kontaktu użytkownika z komputerem. W związku z tym brak mu pewnych możliwości, jakie zapewnia np. Basic. W szczególności niekorzystny jest:

— brak interpretera, wskutek czego program musi przed uruchomieniem przejść cały cykl tworzenia przedstawiony na rys. 1
— brak możliwości interakcyjnego śledzenia błędów
— niemożliwa jest bezpośrednia kontrola kursora na ekranie
— podczas wykonywania programu nie ma kontroli sygnałów z klawiatury (można to częściowo zastąpić instrukcją PAUSE)
— MS-F nie wykorzystuje grafiki, pióra świetlnego i dźwięku
— używana przez autorów wersja ma kilka niezgodności z opisem; dotyczy to dyrektyw, nie działa np. poprawnie dyrektywa $NOTSTRICT.

Kompilator MS-FORTRAN 3.20 jest oprogramowaniem otwierającym drogę do profesjonalnych, złożonych zastosowań komputerów osobistych w wielu dziedzinach nauki i inżynierii. Przyczyną tego jest przede wszystkim znaczne zwiększenie szybkości działania programu wynikowego i możliwości tworzenia tablic w zakresie całej dostępnej pamięci operacyjnej. Bardzo istotnym czynnikiem jest także łatwość przenoszenia oprogramowania z innych komputerów na osobiste i odwrotnie. Jako przykład można podać znany autorom fakt, że złożony program zawierający powyżej 6000 linii został przeniesiony z Odry, uruchomiony i przetestowany na IBM/PC w ciągu kilku godzin przez osobę, która miała uprzednio jedynie trzy godziny czasu na zapoznanie się z mikrokomputerem.

Jeśli chodzi o strony ujemne, to najpoważniejszy chyba problem związany z brakiem komend graficznych może być rozwiązany dzięki możliwości konsolidacji procedur fortranowskich z procedurami w Pascalu lub C. Można też wykorzystać możliwość zapisu w Fortranie danych, dotyczących grafiki na pliki dyskowe, które następnie mogą być traktowane jako pliki wejściowe do wielu rozbudowanych samodzielnych pakietów graficznych dostępnych na komputerach osobistych.

LITERATURA

[1] Microsoft Fortran Compiler for the MS-DOS Operating System. User's Guide. Microsoft Corporation, Bellevue (WA), 1984
[2] Microsoft Fortran. Reference Manual. Microsoft Corporation, Bellevue (WA), 1984.

MACIEJ A. MARKOWSKI
Warszawa

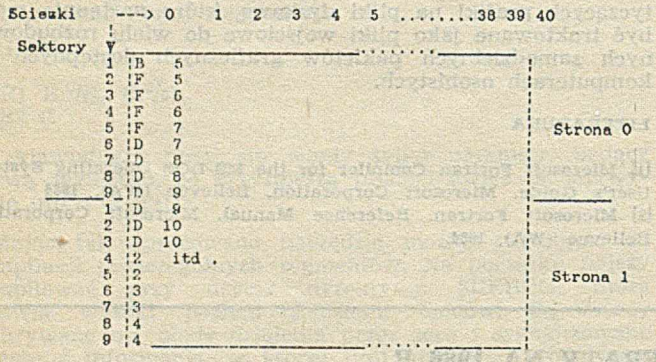
Logiczna organizacja dysku w systemie operacyjnym PC-DOS

W artykule przedstawiono wiadomości użyteczne przy odzyskiwaniu przypadkowo skasowanych lub uszkodzonych plików zapisanych na dyskietkach i dyskach w systemie operacyjnym PC-DOS lub MS-DOS.

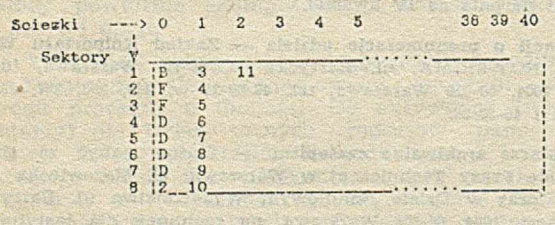
Większość informacji zaczerpnięto z książki [2], będącej czymś w rodzaju technicznego opisu teorii (ang. technical theory textbook) dla programów NORTON UTILITY. Zostały one sprawdzone przy opracowywaniu programu służącego do przepisywania plików z dyskietki zapisanej w systemie DOS na dyskietkę systemu CP/M-86. Było to jedno z zadań laboratorium do przedmiotu obieralnego „Oprogramowanie mikrokomputerów". Program ten napisany w systemie CP/M nie mógł korzystać ze standardowych funkcji odczytywania plików, skąd wynikła potrzeba poznania organizacji dysku w systemie PC-DOS.

LOGICZNA ORGANIZACJA DYSKU

Na rysunkach 1 i 2 przedstawiono położenie obszarów zajętych przez system operacyjny i obszaru danych dla dwu różnych rodzajów dyskietek w systemie DOS.



Rys. 1. Dwustronna dyskietka z dziewięcioma sektorami na ścieżce



Rys. 2. Jednostronna dyskietka z ośmioma sektorami na ścieżce

Litery na rysunkach oznaczają odpowiednio:
B — sektor, w którym zapisany jest program ładujący (ang. boot-strap loader)
F — sektor zajmowany przez tablicę alokacji plików FAT (ang. file allocation table)
D — sektor zajmowany przez skorowidz (ang. directory).
Cyfry oznaczają numer jednostki alokacji na obszarze danych, przy czym dla dyskietki dwustronnej pojedyncza jednostka zajmuje dwa sektory. Ta część dysku jest wykorzystywana do zapisu plików użytkownika.

Terminem **niepodzielna jednostka alokacji** proponuję określić najmniejszą jednostkę, jaką można odczytać z dysku podczas jednorazowego pobierania informacji. Dosłowne tłumaczenie angielskiego słowa cluster (kiść, pęk, grono, kępka, gromada) raczej nie oddaje sensu w tym wypadku. Każda jednostka alokacji ma przypisany numer, który określa jej fizyczne położenie na dysku.

Pierwszym etapem odczytywania plików z dysku jest znalezienie nazwy pliku w skorowidzu, skąd jest pobierany numer pierwszej jednostki alokacji pliku. Określa on adres (w tablicy alokacji) numeru następnej jednostki w łańcuchu, a po zdekodowaniu wyznacza fizyczne położenie informacji przeznaczonych do odczytania. Po odczytaniu pierwszej jednostki, pobiera się wartość zapisaną pod podanym wcześniej adresem. Wartość ta może być numerem jednostki lub znacznikiem końca pliku. W pierwszym wypadku postępuje się analogicznie jak z pierwszą jednostką łańcucha alokacji, w drugim — kończy się odczytywanie.

SKOROWIDZE

Każdy plik, który zapisany na dysku ma swoje pole w skorowidzu. Zawiera ono:

nazwę pliku	— bajty 0—7
rozszerzenie (typ)	— bajty 8—10
atrybut	— bajt 11
obszar zarezerwowany	— bajty 12—21
datę powstania pliku	— bajty 22 i 23
czas powstania pliku	— bajty 24 i 25
numer pierwszej jednostki alokacji w łańcuchu	— bajty 26 i 27
wielkość pliku	— bajty 28—31.

Jedno pole zajmuje 32 bajty, więc w jednym sektorze znajduje się dokładnie 16 pól, czyli np. na dyskietce dwustronnej z dziewięcioma sektorami na ścieżkę, może być zapisanych co najwyżej 112 plików (bez użycia podskorowidzów).

Nazwa pliku (ang. file name) jest zapisana w kodzie ASCII dużymi literami i zajmuje osiem bajtów. W wypadku, gdy długość nazwy jest mniejsza niż osiem liter, wektor ten jest dopełniany spacjami (32 dziesiętnie w kodzie ASCII). Pierwszy bajt tego wektora pełni jeszcze dodatkową funkcję — kasowanie pliku poleceniem DEL lub ERASE powoduje wpisanie na ten bajt wartości 5EH. Dzięki temu łatwo jest odtworzyć przypadkowo skasowany plik, który mieści się w jednej jednostce alokacji. W wypadku dłuższych plików należy przeszukać dyskietkę, aby znaleźć dalszą część pliku, gdyż łańcuch alokacji znajdujący się w FAT jest podczas kasowania usuwany.

Rozszerzenie (typ) pliku zajmuje trzy następne bajty i tak samo jak nazwa jest zapisane w kodzie ASCII. Jeżeli długość typu jest mniejsza, to dopisywane są spacje.

Atrybut pliku zajmuje jednobajtowe pole o adresie 11 względem początku pola w skorowidzu. Atrybut może przebierać następujące wartości:
01 — plik tylko do odczytu (ang. read only)
02 — plik ukryty (ang. hidden), nie wykazywany podczas standardowych przeszukiwań dysku
04 — plik systemowy, funkcjonalnie pełni rolę podobną do atrybutu 02

08 — informuje, że bajty od 0 do 11 zawierające znaki ASCII oznaczają nazwę (etykietę) całej dyskietki (ang. volumne label)
10 — plik podskorowidza (ang. subdirectory), jego zawartością jest skorowidz o identycznej budowie jak aktualnie przedstawiany
20 — plik archiwalny.
Atrybut może być równy sumie liczb dla jednoczesnego nadania plikowi kilku wymienionych właściwości.

Czas i datę powstania lub uaktualnienia pliku stanowią cztery bajty, zapisane w formacie liczb całkowitych bez znaku. Na dwóch pierwszych jest zakodowany dokładny czas ostatniej operacji na pliku. Kodowanie jest określone następującym wzorem:

CZAS = GODZINY * 2048 + MINUTY * 32 + SEKUNDY / 2

Następne dwa bajty informują o dacie ostatniego otwarcia pliku i są kodowane za pomocą wzoru:
DATA = (ROK — 1980) * 512 + MIESIĄC * 32 + DZIEŃ
Mimo że według wzoru można kodować daty do 2107 roku, to programy systemu operacyjnego DOS za najpóźniejszą poprawną datę przyjmują rok 2099.

Rozmiar pliku stanowią dwie liczby całkowite bez znaku, określające jego długość w bajtach. Dokładność, z jaką ta liczba jest podawana, zależy od rozmiaru bloku, jaki jest zapisywany na dysk. Niektóre programy przetwarzające pliki tekstowe formują bloki zawierające 128 bajtów.

TABLICA ALOKACJI PLIKÓW

W sektorach od drugiego do trzeciego (dla dyskietki jednostronnej z ośmioma sektorami na ścieżce) lub do piątego (dla dyskietki dwustronnej z dziewięcioma sektorami na ścieżce) znajdują się dwie identyczne tablice określające za pomocą struktur łańcuchowych położenie całego pliku na dysku. Każda jednostka alokacji ma 12-bitowe (półtorabajtowe) pole w tablicy.

Budowa

Pierwszy bajt tablicy określa rodzaj dyskietki i może przybierać następujące wartości:
FFH — dla dyskietki dwustronnej z ośmioma sektorami na jednej ścieżce
FEH — dla dyskietki jednostronnej z ośmioma sektorami na jednej ścieżce
FDH — dla dyskietki dwustronnej z dziewięcioma sektorami na jednej ścieżce
FCH — dla dyskietki jednostronnej z dziewięcioma sektorami na jednej ścieżce
F8H — dysk stały (ang. fixed disk).

Drugi i trzeci bajt mają zawsze wartość FFH.

Następne bajty służą do odwzorowania obszaru zajmowanego na dyskietce przez dane, przy czym:
— wartości od 3 do EFFH służą do wskazywania następnej jednostki alokacji
— wartości od FF0H do FF7H są używane do wskazania jednostek zarezerwowanych; liczba FF7H wskazuje na mechaniczne uszkodzenie nośnika w obrębie jednostki — nie może ona znajdować się w łańcuchu alokacji (wielkość ta jest wstawiana podczas formatowania dysku)
— wartości od FF8H do FFFH informują, że poprzednio czytana jednostka była ostatnią w łańcuchu alokacji dla danego pliku
— wartość 000 występuje, gdy jednostka jest dostępna i nie używana.

Dekodowanie

W celu zaoszczędzenia miejsca na dyskietce tablica alokacji jest zakodowana. Informacja o adresie (względem początku tablicy) następnej jednostki jest pobierana następująco:
— numer poprzednio czytanej jednostki mnoży się przez 1,5; jest to adres słowa, w którym znajduje się zakodowany numer następnej jednostki
— pobiera się to słowo (np. instrukcją MOV), tak aby zmienić kolejność bajtów w słowie
— jeżeli numer poprzednio pobranej jednostki jest nieparzysty, to z wartości pobranego słowa należy uzyskać wartość modulo 4096
— w przeciwnym wypadku pobraną wartość należy podzielić całkowicie przez 16.
Otrzymany wynik jest numerem następnej jednostki do odczytania lub informacją, że poprzednio odczytana jednostka była ostatnią w łańcuchu.

Uzyskiwanie informacji o położeniu sektorów z numeru jednostki alokacji

Lokalizacja jednostki jest zdefiniowana za pomocą następujących współrzędnych: strona dyskietki, ścieżka, sektory. Algorytmy dekodujące dla różnych rodzajów dyskietki są następujące:

— dyskietka jednostronna, 8 sektorów na ścieżkę

```
TEMP = JEDNOSTKA + 5
ŚCIEŻKA = TEMP DIV 8
STRONA = 0
SEKTOR = TEMP MOD 8
```

— dyskietka dwustronna, 8 sektorów na ścieżkę

```
TEMP = JEDNOSTKA * 2 + 6
ŚCIEŻKA = TEMP DIV 16
TEMP = TEMP MOD 16
STRONA = TEMP DIV 8
SEKTOR = (TEMP MOD 8) + 1
```

— dyskietka dwustronna, 9 sektorów na ścieżkę

```
TEMP = JEDNOSTKA * 2 + 8
ŚCIEŻKA = TEMP DIV 18
TEMP = TEMP MOD 18
STRONA = TEMP DIV 9
SEKTOR = (TEMP MOD 9) + 1
```

Zmienna JEDNOSTKA oznacza numer jednostki alokacji.

* * *

Informacje zawarte w tym artykule powinny być pomocne przy korzystaniu z programów NORTON UTILITY, ale mogą też służyć do pisania programów zabezpieczających dyskietki lub dyski przed kopiowaniem lub przed czytaniem przez niepowołane osoby. Wszelkie uwagi dotyczące treści artykułu można kierować do autora (tel. 44-34-04 — Warszawa).

LITERATURA

[1] IBM Personal Computer XT — Technical Reference. First Edition. May, 1983
[2] Norton P.: Inside the IBM PC — Access to Advanced Features and Programming. Robert J. Brady Company, 1983.

Ada dla IBM PC

Kompilator Ady dla komputerów IBM PC, działający pod kontrolą systemu operacyjnego PC-DOS, opracowała firma Artek. Pełny system, obejmujący edytor ekranowy, interpreter, program uruchomieniowy, konsolidator, programy biblioteczne, disasembler A-kodu i dokumentację, kosztuje 895 dolarów. Jak twierdzi firma, implementacja jest zgodna z normą Ady, lecz nie zawiera wielozadaniowości.

Do głównych zaimplementowanych konstrukcji należą podprogramy i pakiety rodzajowe, agregaty tablicowe i rekordowe, przeciążanie operatorów, tablice dynamiczne i wyjątki. W pierwszym przebiegu kompilatora następuje translacja programu źródłowego na wykonywalny pseudokod. W drugim przebiegu A-kod jest tłumaczony na język maszynowy mikroprocesora Intel 8086.

Cały system wymaga pamięci operacyjnej o pojemności 384 KB. Do opracowywania dużych programów zaleca się użycie dysku stałego. Adres producenta: Artek Corp., 100 Seaview Drive, Secaucus, New Jersey 07094, USA.

(JZ)

WALDEMAR DWORAKOWSKI
Instytut Maszyn Matematycznych
Warszawa

Techniki testowania układów a analiza sygnatur

Klasyczne metody testowania są nieefektywne w zastosowaniu do współczesnych układów mikroprocesorowych. Dotychczasowe metody testowania automatycznego opierały się na metodzie pobudzeń i odpowiedzi. Pobudzenia i odpowiedzi obliczano metodami symulacji lub algorytmicznymi. Liczba wektorów pobudzeń była ograniczona możliwościami aparatury testującej i czasem testowania. Doboru wektorów dokonywano biorąc pod uwagę ich możliwości detekcji uszkodzeń w badanym układzie. Do ustalenia takiego minimalnego zestawu wektorów testowych wykorzystywano dokładnę modele układu i przez metody symulacyjne, poddając układ uszkodzeniom (symulowanym według zakładanego modelu uszkodzeń) badano przydatność wektorów testowych do wykrywania uszkodzeń. Niestety, takie postępowanie jest opłacalne ekonomicznie jedynie w produkcji wieloseryjnej. Przy produkcji małoseryjnej nie opłaca się tworzenie stanowisk testowych, generowanie komputerowych modeli układów oraz dokonywanie doboru wektorów testowych.

Należy pamiętać, że przy stosowaniu nowoczesnych elementów o wysokim stopniu scalenia pojawiają się dodatkowe problemy, nieistniejące dla klasycznych układów logicznych. Do tych problemów można zaliczyć: stopień komplikacji układów, niepełne zdefiniowanie badanego obiektu, trudna detekcja uszkodzeń oraz ograniczenia ekonomiczne.

NOWE PROBLEMY TESTOWANIA

Stopień komplikacji układu wpływa na czas ustalania optymalnych wektorów testowych, przy czym zależność ta jest silnie nieliniowa. Skutkiem tego jest długi czas generowania testów albo tworzenie testów nieoptymalnych. Ponadto, liczba testów wzrasta w sposób wykładniczy wraz ze wzrostem komplikacji układów. Wpływa to na czas generowania testów jak i testowania. Zmusza również do stosowania bardziej skomplikowanych urządzeń testujących. Co więcej, pojawienie się układów zawierających kilka milionów elementów o nieznanej strukturze wewnętrznej musi prowadzić do innej strategii testowania. Przykładem takiej strategii jest rodzina iAPX 432, która zawiera wbudowane układy do samotestowania metodą porównywania z wzorcem. Niestety, takie rozwiązania nie są dostosowane do klasycznych testerów automatycznych.

Znaczna komplikacja układów cyfrowych stosowanych we współczesnej technice prowadzi do trudności w zdefiniowaniu struktury wewnętrznej układu scalonego przez producenta sprzętu elektronicznego. Producenci układów scalonych nie są zainteresowani rozpowszechnianiem informacji o wewnętrznej konstrukcji swoich produktów. Ponadto, konstrukcja wewnętrzna układów jest różna u różnych producentów i zmienia się w czasie uzyskiwania doświadczeń przez poszczególnych wytwórców. Prowadzi to oczywiście do trudności z utworzeniem modelu układu przy zachowaniu odpowiedniej dokładności. Powstałe w ten sposób testy mogą pomijać pewne specyficzne uszkodzenia układów scalonych lub odrzucać układy działające poprawnie, lecz pochodzące od innego wytwórcy.

Innym problemem jest wierność pomiarów i ryzyko przez nie wprowadzane w wypadku układów VLSI. Układy te pracują na granicy swoich możliwości prądowych i termicznych. Dodatkowe obciążenia wprowadzane przez dobrze zaprojektowane układy pomiarowe są zazwyczaj do pominięcia, jednak układy wymuszania stanów logicznych (służące do wprowadzania wektorów testowych) powodują zazwyczaj przepływ dużych prądów przez stosunkowo długi czas. Wprowadza to oczywiście ryzyko przy prowadzeniu testowania tych złożonych i kosztownych układów. Dodatkowym problemem jest konstruowanie układów sond pomiarowych tak, aby nie wprowadzały zmian w układzie badanym (np. przez pojemności rozproszone) i zapewniały niezakłócony pomiar przy możliwie niewielkim koszcie w całym cyklu produkcyjno-eksploatacyjnym.

Detekcja uszkodzeń w klasycznych układach cyfrowych przy użyciu automatycznego testowania jest możliwa dzięki odpowiedniemu doborowi wektorów testowych oraz dzięki odpowiedniej obróbce informacji uzyskiwanej z testowania. Lokalizacja uszkodzonych układów opiera się na prostym modelu uszkodzeń jak zwarcie z masą lub przebicie napięciem na poziomie zasilania. Wprowadzanie nowych typów układów i nowych konfiguracji oraz względy technologiczne spowodowały pojawienie się dodatkowych rodzajów uszkodzeń jak zwarcie dwóch szyn i błędy w dostępie do wspólnych szyn. Wprowadzenie układów LSI i VLSI oraz układów dynamicznych wprowadziło dalsze zamieszanie. W tych wypadkach detekcja niektórych rodzajów błędów w warunkach produkcyjnych może być trudna i mogą się one ujawniać dopiero w gotowym urządzeniu u użytkownika. Wtedy często stosuje się wymianę pakietów i ich naprawę u producenta (przy użyciu testerów), co może prowadzić do ciągłej cyrkulacji niesprawnych pakietów.

Dodatkowym utrudnieniem jest pojawienie się uszkodzeń przemijających (takich, które uniemożliwiają działanie układu tylko przez pewien czas), uszkodzeń lawinowych (polegających na tym, że uszkodzenie jednego z elementów pociąga za sobą uszkodzenia kolejnych elementów) i innych, ściśle związanych z technologią wykonania układów, jak np. tzw. zjawisko zatrzaskiwania w układach CMOS lub obciążenie szyn przez wyłączone bramki trójstanowe. Do dnia dzisiejszego nie pojawiły się jeszcze elementy LSI, ułatwiające klasyczne metody testowania, co mogłoby znacznie pomóc w testowaniu tych złożonych układów.

Korzystne natomiast jest to, że we współczesnej technice cyfrowej stosowane są układy programowalne, co umożliwia wbudowanie testów do gotowego produktu. Ma to znaczenie nie tylko dla serwisu, lecz także dla uruchamiania układu dla producenta, umożliwia zbadanie sprawności układu w warunkach zbliżonych do rzeczywistej pracy. Niestety, ten sposób testowania nie gwarantuje wykrycia wszystkich uszkodzeń. Pozwala jednak na wprowadzenie do układu wektorów testowych przez odpowiedni program testowy. Układ testujący synchroniczny z wewnętrznym pobudzeniem musi jedynie ustalić, przez wyznaczone punkty testowe, czy przebiegi w tych punktach są prawidłowe i następnie na podstawie charakteru rozbieżności zlokalizować uszkodzenia. Do rozróżnienia poszczególnych przebiegów można wykorzystać zamiast pełnej informacji o przebiegu, także informacje skompresowane w celu przyspieszenia procesu testowania i zmniejszenia ilości danych w testerze. Jest to ogólna zasada wszystkich metod testowania zbliżonych do analizy sygnatur.

METODA ANALIZY SYGNATUR

Analiza sygnatur została opracowana w firmie Hewlett-Packard w połowie lat siedemdziesiątych. Jej poprzedniczką była metoda liczenia zboczy, która jednak nie zdobyła uznania. Natomiast metoda analizy sygnatur jest powszechnie stosowana jako nowoczesna metoda serwisowa, stanowi także podstawę innych nowych metod testowania układów.

O ile w klasycznych metodach testowania i projektowania ideą nadrzędną było zapewnienie testowalności i obserwowalności przy założeniu podawania wektorów testowych przez układy zewnętrzne, które też oceniały zachowanie układu, to w technice analizy sygnatur założono, że źródłem testu będzie sam układ. W związku z tym nie jest istotne optymalizowanie testu, lecz synchroniczne działanie testera z układem badanym.

Nowością jest zastosowanie w tej technice dzielenia wielomianów do kompresji informacji o mierzonym przebiegu. Dzięki temu udaje się uzyskać prawdopodobieństwo omyłkowego niewykrycia błędnego przebiegu równe $1/2^{16}$ (dla 16-bitowego rejestru przesuwnego). Tak małe i niezależne od długości ciągu wejściowego prawdopodobieństwo popełnienia błędu skłania do stosowania ciągów wejściowych o dużych długościach, ograniczonych jedynie przez czas pomiaru i względy praktyczne.

Przyrząd do testowania metodą analizy sygnatur opracowany przez firmę Hewlett-Packard [7] był w pierwszej wersji dostosowany głównie do zastosowań serwisowych i umożliwiał pomiary po doprowadzeniu czterech sygnałów z układu testowanego: sygnału zegarowego CLOCK, sygnałów synchronizacyjnych START i STOP oraz sygnału badanego (przez sondę logiczną). Trzy pierwsze sygnały są dostarczane z układu badanego do sondy pomiarowej (zrealizowanej w oparciu o układy ECL) za pomocą krótkich kabli zakończonych chwytakami, a następnie po wzmocnieniu do przyrządu. Sygnał badany jest wzmacniany w sondzie logicznej i przetwarzany na dwa sygnały informujące o poziomie napięcia na wejściu (poziom wysoki, poziom niski, poziom nieokreślony), a następnie przekazywany do przyrządu pomiarowego, gdzie wchodzi na wejście układu wstępnej kompresji informacji. W przyrządzie HP5004A [18], układem kompresji wstępnej jest przerzutnik JK, na wejście którego wchodzą sygnały „poziom wysoki" i „poziom niski". W przyrządach innych firm spotyka się różne rozwiązania wstępnej kompresji informacji.

Po wstępnej kompresji dane wejściowe, przez układ bramkujący i układ obliczania sumy modulo 2, dostają się do układu 16-bitowego rejestru przesuwnego. Układ obliczania sumy modulo 2 służy do realizacji obliczenia wielomianu charakterystycznego $X^{16} + X^9 + X^7 + X^4 + 1$.

Układ bramkujący, przez próbkowanie wejść START i STOP, wykrywa aktywne zbocza na tych wejściach i na ich podstawie wytwarza sygnał okna pomiarowego, służący do sterowania pracą całego przyrządu. Użytkownik, przez zestaw przełączników przyrządu, może ustalić jako aktywne jedno z dwóch zboczy sygnału zegarowego, jak i sygnałów START i STOP oraz sygnału QUAL, który w nowszych modelach analizatora może być używany jako sygnał warunkujący pracę układu bramkującego.

Wynik pomiaru po zakończeniu sygnału okna pomiarowego jest przekazywany z rejestru przesuwnego do pamięci i porównywany z poprzednim pomiarem. Do przedstawienia wyniku służy czterocyfrowy wyświetlacz siedmiosegmentowy, w którym zamiast standardowego zestawu cyfr szesnastkowych (0 do 9, A do F) zastosowano zestaw zmodyfikowany, zawierający następujące symbole: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, C, F, H, P, U. Producent wybrał taki zestaw znaków ze względów ergonomicznych oraz przypuszczalnie ze względów reklamowych. Oprócz aktualnego wyniku pomiaru wyświetla się także aktualny stan okna pomiarowego i wynik porównania aktualnej i poprzedniej sygnatury (stabilność sygnatury). Niestabilność sygnatury może sygnalizować błędne działanie układu lub źle zaprojektowany test.

Nowsze modele analizatora sygnatur zostały wyposażone w możliwość współpracy z komputerem przez sprzęg HP-IB lub HP-IL. Ponadto możliwości pomiarowe przyrządów zostały rozszerzone o zdolność współpracy z dowolnymi układami logicznymi (przez regulowanie napięcia odniesienia), pomiary napięcia, różnicy napięć, oporności, częstotliwości, zliczanie impulsów i inne (np. w HP 5005B [19]). Analizatory są też wbudowywane do nowoczesnych testerów automatycznych (np. HP 3060A lub GUEST firmy Hewlett-Packard [22] lub nowy tester opracowywany w IMM). Należy jednak pamiętać, że analiza sygnatur została opracowana jako technika pomiarowa z myślą o serwisie urządzeń, do czego najlepiej nadają się przyrządy (takie jak HP 5006A [20]), umożliwiające pomiar sygnatur złożonych, będących sumą wielu sygnatur indywidualnych, co umożliwia przyspieszenie pomiarów. Warto więc zwrócić także uwagę na polski serwisowy analizator sygnatur PAS 80.

Niestety analiza sygnatur ma też wady. Podstawową wadą tej metody jest jej nietypowość i dodatkowe wymagania dotyczące konstrukcji układów przeznaczonych do uruchamiania przy jej użyciu oraz istnienie dodatkowej precyzyjnej dokumentacji serwisowej. Kosztowne może też być wyposażenie serwisu w przyrządy do analizy sygnatur.

Dodatkowe wymagania dotyczące projektowania można ująć w następujących punktach:

1. Układ przeznaczony do badania metodami analizy sygnatur musi być skonstruowany z myślą o testowaniu.

2. Musi istnieć możliwość wyłączenia wszystkich globalnych pętli sprzężenia zwrotnego (warunek rozróżnialności uszkodzeń).

3. Wszystkie pomiary dotyczące jednego pakietu powinny być możliwe do wykonania przy użyciu minimalnej liczby przełączeń. Płytka badana powinna mieć wyróżnione i opisane punkty z sygnałami START, STOP, CLOCK. Punkty testowe powinny być możliwie nieliczne i łatwo dostępne.

4. Płytka powinna mieć wbudowany element wytwarzający pobudzenia, który powinien być synchroniczny z zegarem i powinien wytwarzać sygnały START i STOP (oraz niekoniecznie sygnał QUAL), wszystkie synchroniczne z zegarem i pobudzeniem, a umożliwiające poprawny pomiar. Zwykle elementem takim jest np. mikroprocesor pracujący w trybie „free run".

5. Wszystkie linie o charakterze asynchronicznym w stosunku do testu lub zegara powinny podczas testowania być odłączone od układu.

6. Linie trójstanowe badane metodą analizy sygnatur nie powinny przyjmować stanu dużej impedancji w momencie aktywnego zbocza sygnału zegarowego, ponieważ w pewnych specyficznych przypadkach może to prowadzić do niestabilności sygnatur.

7. Testowanie powinno być zrealizowane w sposób strukturalny, tzn. należy je rozpoczynać od możliwie niewielkiego jądra, a następnie testować kolejne warstwy systemu. Do celów testowania niższych warstw nie można używać elementów związanych z warstwami wyższymi. Poszczególne testy powinny być wybierane przy możliwie niewielkim zaangażowaniu układów sprzętowych.

8. Należy zwrócić szczególną uwagę na przygotowanie odpowiedniej dokumentacji serwisowej. Na schematach muszą być naniesione sygnatury wraz z układami pomiarowymi. We wszystkich punktach układu powinny być dołączone tablice sygnatur, a także drzewa testowania i słowniki uszkodzeń. Korzystne jest wyposażenie przyrządu w nakładki z sygnaturami i drogami testowania układu.

Zestawienia wymagań dla konstruktorów, jak i materiały służące pomocą w projektowaniu są publikowane zarówno przez firmę Hewlett-Packard w postaci not aplikacyjnych [8—15], jak też w pismach fachowych [1], [7], [22] i w książkach. Pojawiają się też prace w języku polskim poświęcone tej tematyce. Metoda analizy sygnatur jest ciągle rozwijana, o czym świadczy stosunkowo duża liczba publikacji, jak i częste pojawianie się tego tematu na konferencjach naukowych. Obecnymi kierunkami rozwoju tej metody są: równoległa i wielowartościowa analiza sygnatur oraz jej modyfikacje w celu umożliwienia łatwej obserwowalności, kontroli i łatwego diagnozowania układów w warunkach automatycznego testowania, jak i w celu samotestowania.

LITERATURA

[1] Ashkenas D. J., Degabriele R. F.: Powerful diagnostic philosophy reduced down time. Hewlett-Packard Journal, marzec 1982

[2] Bennetts R. G.: Design of testable logic circuits. Addison-Wesley, Reading (M.A), 1984

[3] Breuer M. A.: Automatyczne projektowanie maszyn cyfrowych. PWN, Warszawa 1966

[4] Breuer M. A.: Diagnosis and reliable design of digital systems. Pitman, 1977

[5] Dworakowski W.: Komputerowo wspomagane projektowanie, symulacja i uruchamianie układów cyfrowych z uwzględnieniem analizy sygnatur. Praca dyplomowa. Instytut Telekomunikacji, Politechnika Warszawska, 1984

[6] Friedman A. D., Menon P. R.: Wykrywanie uszkodzeń w układach cyfrowych. PWN, Warszawa, 1974

[7] Hewlett-Packard: Zestaw artykułów poświęconych analizie sygnatur. Hewlett-Packard Journal, maj 1977

[8] Hewlett-Packard: A designers guide to signature analysis. Application note 222

[9] Hewlett-Packard: A managers guide to signature analysis. Application note 222

[10] Hewlett-Packard: Implementing signature analysis for production testing with the HP 3060A board test system. Application note 222-1
[11] Hewlett-Packard: Application articles on signature analysis. Application note 222-2, maj 1979
[12] Hewlett-Packard: Guidelines for signature analpsis — understanding the signature measurements. Application note 222-4
[13] Hewlett-Packard: Troubleshooting with composite signatures. Application note 222-6
[14] Hewlett-Packard: A signature analysis case study of a Z80 based personal computer. Application note 222-10
[15] Hewlett-Packard: A signature analysis case study of a 6800 based display terminal. Application note 222-11
[16] Hewlett-Packard: Microprocesor Exerciser. Model HP 5001C
[17] Hewlett-Packard: Microprocesor Exerciser. Model HP 5001D
[18] Hewlett-Packard: HP 5004A/5005A technical data
[19] Hewlett-Packard: HP 5005B technical data
[20] Hewlett-Packard: HP 5006A technical data
[21] Hewlett-Packard: Financial justification of circuit test systems
[22] Holland E. R., Robertson J. L.: GUEST — A signature analysis based test system for ECL logic. Hewlett-Packard Journal, marzec 1982
[23] Ignaczak A., Dworakowski W.: Zastosowanie analizy sygnatur do komputerowej generacji dokumentacji serwisowej układów cyfrowych. Zeszyt 111, Instytut Telekomunikacji, Politechnika Warszawska, 1982

120 lat „Przeglądu Technicznego"

„Przegląd Techniczny", to jedno z najstarszych czasopism polskich, które zostało założone jako „pismo miesięczne poświęcone przemysłowi krajowemu i praktycznym zastosowaniom inżynierii". Powstało z inicjatywy Pawła Kaczyńskiego i wydawane było przez oficynę księgarską Gebethnera i Wolffa. „Przegląd Techniczny" zapoczątkował rozwój polskiego czasopiśmiennictwa technicznego, odegrał inspiratorską i organizatorską rolę w zrzeszaniu się inżynierów we wspólnych organizacjach: początkowo Stowarzyszeniu Techników (1898 r.), potem Związku Polskich Zrzeszeń Technicznych (1922 r.), Naczelnej Organizacji Inżynierskiej (1935 r.) i wreszcie Naczelnej Organizacji Technicznej (1946 r.).

W okresie po Powstaniu Styczniowym czasopismo skierowało uwagę inteligencji ku sprawom umacniania podstaw materialnych społeczeństwa polskiego, pozbawionego niepodległości. „Przegląd Techniczny" podejmował wiele inicjatyw, m.in.: unowocześniania Warszawy, tworzenie systemów komunikacyjnych oraz organizację i rozwój zawodowych szkół technicznych.

Po roku 1918 „Przegląd Techniczny" zmienił nieco tematykę, ponieważ problemy ściśle branżowe poszczególnych grup inżynierskich podjęły powstałe wówczas różne czasopisma techniczne. Na łamach „Przeglądu Technicznego" ukazywały się publikacje dotyczące ogólnych zagadnień przemysłowych i techniczno-ekonomicznych.

Po II Wojnie Światowej „Przegląd Techniczny", którego wydawanie wznowiono 1 kwietnia 1945 r. w Łodzi, powrócił z początkiem 1949 r. do Warszawy, jako główny organ prasowy Naczelnej Organizacji Technicznej.

Czasopismo towarzyszyło wysiłkom twórczym i organizatorskim polskich inżynierów w czasie odbudowy i uprzemysłowienia kraju.

W latach 1949—1986 „Przegląd Techniczny" zwiększył kilkakrotnie nakład, zmieniał format, szatę graficzną i charakter publikacji. Stał się czasopismem bardziej uniwersalnym. Był i pozostaje wyrazicielem poglądów zorganizowanej w ramach Federacji NOT społeczności inżynieryjno-technicznej.

W „Przeglądzie Technicznym" publikowane są artykuły przedstawiające:

- politykę techniczną państwa oraz opinie środowiska na ten temat,
- polskie i światowe osiągnięcia nauki i techniki,
- rolę cywilizacji technicznej oraz jej skutki społeczne i ekonomiczne,
- zadania i status inżyniera i technika,
- związki techniki z gospodarką.

W 120-lecie PT redakcja podjęła wiele ciekawych przedsięwzięć: wydanie w roku jubileuszowym specjalnego numeru czasopisma, zorganizowanie wspólnie z dyrekcją Międzynarodowych Targów Poznańskich i Zakładami ELWRO (w kwietniu 1987 r.) międzynarodowej wystawy systemów informatycznych (INFOSYSTEM), we Wrocławiu oraz liczne spotkania z czytelnikami.

Redaktor Naczelny „Przeglądu Technicznego"
mgr inż. BRONISŁAW HYNOWSKI

Konstruowanie lokalnych sieci komputerowych według specyfikacji Ethernet

Firmy Digital, Intel i Xerox (w skrócie: DIX) opublikowały standard sieci lokalnej Ethernet, który specyfikuje warstwę fizyczną i warstwę łącza danych tej sieci. Przypuszcza się, że będzie to jeden z dwóch typów lokalnych sieci komputerowych najszerzej używanych w zastosowaniach przemysłowych i biurowych. Opublikowanie tej specyfikacji jest handlową próbą pobudzenia sprzedaży i wzrostu liczby implementacji w zakresie elementów sieciowych kompatybilnych z Ethernetem. Poniżej opisano próbę implementacji sieci lokalnej typu Ethernet na Wydziale Informatyki Uniwersytetu Strathclyde. Sieć ta nosi nazwę **Strathnet.** W implementacji tej warstwa fizyczna i warstwa łącza są wbudowane w mikroprocesorowe jednostki dostępu do sieci (ang. network access units), za pomocą których dołącza się do sieci komputery, terminale i inne urządzenia. W artykule przedyskutowano wykonalność i celowość konstruowania sieci według specyfikacji Ethernet i zarysowano podejście przyjęte przy implementacji sieci Strathnet.

WPROWADZENIE

Idea lokalnej sieci komputerowej (ang. local area network) ma już około 10 lat. Trudno byłoby jednak ściśle określić moment, kiedy się pojawiła lub kiedy po raz pierwszy użyto terminu „lokalna sieć komputerowa" w jego dzisiejszym znaczeniu. Podobnie jak to często dzieje się z kluczowymi słowami lub zwrotami w informatyce, tak i to pojęcie nie ma ścisłej definicji. Potrzeba było trochę czasu, aby dzięki publikacjom i językowi mówionemu stało się ono znane społeczności informatycznej. Obecnie istnieje już obszerna i stale wzrastająca bibliografia na temat sieci lokalnych. Przyjmuje się w niej zgrubną definicję lub klasyfikację tych sieci według posiadanych właściwości. Charakterystykę sieci lokalnych można podsumować następująco:

— sieć lokalna obejmuje terytorialnie mały obszar, najczęściej w promieniu 0,5 km,
— łączy ona różnorodne urządzenia, jak: komputery, terminale lub inne urządzenia peryferyjne, za pomocą taniego systemu komunikacyjnego,
— szybkości przesyłania danych są rzędu 1—10 Mb/s.

Sieci lokalne stosowane są już w wielu różnych dziedzinach, np.: w ośrodkach wytwarzających oprogramowanie (m.in. w ośrodkach uniwersyteckich), w systemach biurowych z przetwarzaniem tekstów i pocztą elektroniczną, w systemach zbierania danych i zarządzenia w fabrykach i szpitalach. Miejsce sieci lokalnych w obrębie „widma", na którego jednym krańcu znajdują się systemy wieloprocesorowe, a na drugim sieci komputerowe o dużym zasięgu terytorialnym zilustrowano na rys. 1.

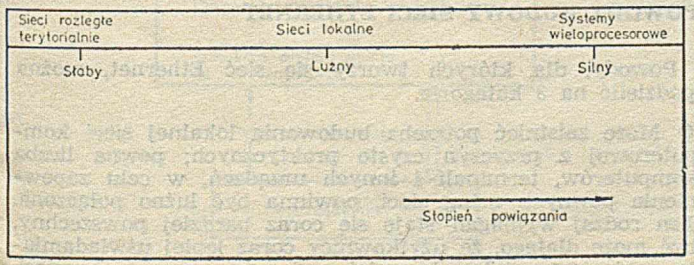
Sieci lokalne można porównywać z systemami innych rodzajów biorąc pod uwagę różne kryteria, m.in. odległość na jaką przesyłane są komunikaty lub szybkość przesyłania danych. W niniejszym artykule autorzy porównują ich „stopień powiązania". To niezbyt jasne, ale intuicyjnie użyteczne, pojęcie obejmuje zakres współpracy współdziałających ze sobą elementów systemu, opisuje ich odległość i wzajemne połączenia. Jedną z najistotniejszych cech sieci lokalnych, wyróżniającą je wśród innych systemów, jest mały koszt komunikacji. Inaczej mówiąc, sieci lokalne cechują się względnie małym kosztem instalowania i eksploatacji ośrodka transmisji oraz dołączania do niego urządzeń, przy umiarkowanie dużej szybkości transmisji.

Dwoma dominującymi architekturami sieci lokalnych są: system rozgłaszania (ang. broadcasting system) i pętla (ang. loop). Obie występują w licznych odmianach. Najbardziej chyba znanym przykładem systemu z rozgłaszaniem jest Ethernet, opracowany przez Xerox Corporation w Stanach Zjednoczonych, zaś najbardziej znanym przykładem pętli — przynajmniej w Wielkiej Brytanii — jest sieć Cambridge Ring. Oba te systemy były przedmiotem licznych badań i jest bardzo prawdopodobne, że będą miały duże znaczenie handlowe.

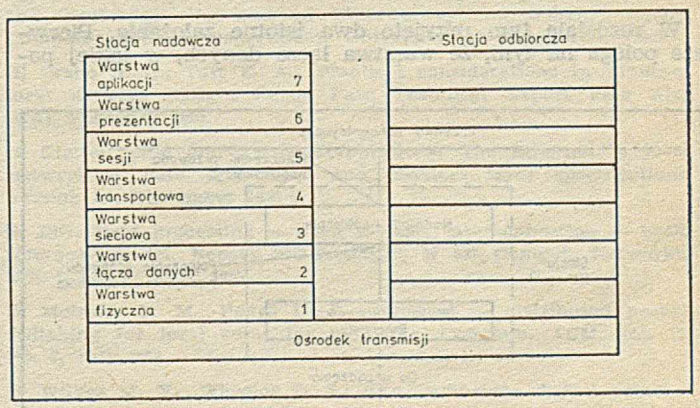
MODEL ISO I WARSTWY SIECI

Aby w pełni zrozumieć specyfikację Etherneta należy znać podział sieci na warstwy. Ponieważ pełny tytuł publikacji DIX [3] brzmi: „Ethernet: lokalna sieć komputerowa — specyfikacja warstwy łącza danych i warstwy fizycznej", należy wyjaśnić, co rozumie się przez „warstwę łącza danych i warstwę fizyczną". W dużym stopniu odpowiedzialna za opracowanie zbioru zaleceń [4], mówiących jak powinna być ustrukturalizowana architektura sieci, jest Międzynarodowa Organizacja Normalizacyjna (ISO). Mimo że zalecenia te były opracowywane dla rozległych sieci komputerowych (ang. wide area networks), które (włączając w to publiczne sieci z komutacją pakietów) do wzajemnej komunikacji wymagają określonego zbioru standardów, to są one użyteczne również przy projektowaniu sieci lokalnych.

Koncepcje, na których oparto zalecenia ISO polegają na tym, że obsługa sieci może być ustrukturalizowana jako hierarchiczny zbiór warstw, z których każda realizuje określony zbiór funkcji oraz ma dobrze określone sprzężenie z warstwą położoną powyżej. Natura sprzężenia każdej warstwy z warstwą położoną poniżej niej jest niewidoczna. Tak więc istnieje możliwość sprzętowych systemów komunikacyjnych, z różnymi protokołami komunikacyjnymi, które zapewniają identyczne narzędzia na jednym z poziomów wyższych. Podstawowy model ISO architektury sieci zawiera 7 warstw, przedstawionych na rys. 2, spośród których 3 dolne mają wszystkie cechy specyficzne dla systemu komunikacji sieci.



Rys. 1. Relacja między sieciami lokalnymi a innymi systemami sieciowymi



Rys. 2. Model architektury sieci (ISO)

Warstwa transportowa i warstwy wyższe modelu ISO zapewniają narzędzia niezależne od sieci. Tak więc przykładowo, przez wspólne sprzężenie warstwy transportowej można wymieniać informacje między dwiema zupełnie różnymi sieciami. Warstwa czwarta i wyższe, które nie są omawiane w tym artykule, obejmują protokoły: przesyłania plików, obsługi terminali i manipulacji pracami. Warstwy te, przy wysyłaniu i odbiorze danych, opierają się na warstwach niższych, zapewniających kierowanie danych pod właściwym adresem w sieci i wykrywanie błędów transmisji.

Trzy dolne warstwy spełniają następujące funkcje:
— warstwa fizyczna odpowiada za transmisję pojedynczych bitów przez fizyczne połączenie między stacją nadawczą i odbiorczą,
— warstwa łącza danych grupuje bity w ramki (ang. frames), tak aby nadajnik i odbiornik rozumiały znaczenie pojedynczych bitów; warstwa ta odpowiada także za wykrywanie i sygnalizację błędów transmisji oraz za retransmisję ramek,
— warstwa sieciowa jest odpowiedzialna za nawiązanie połączenia między nadawcą i odbiorcą oraz za multipleksowanie wielu dialogów na jednym łączu.

Dla wygody terminologicznej autorzy definiują „element sieci" jako implementację trzech dolnych warstw, które przez sprzężenie z warstwą sieciową umożliwiają dołączenie urządzeń do sieci. Pełną kompatybilność elementów danej sieci można osiągnąć tylko przez implementację identycznych sprzężeń na poziomach wyższych i niższych dla wszystkich wymagań elementu sieci.

SPECYFIKACJA DIX

Wersja 1.0 specyfikacji sieci Ethernet definiuje jedynie warstwę fizyczną i warstwę łącza danych. Warstwy te mają ściśle odpowiadać dwom najniższym warstwom modelu ISO. Wersja ta zawiera 7 rozdziałów, wstęp i 5 dodatków.

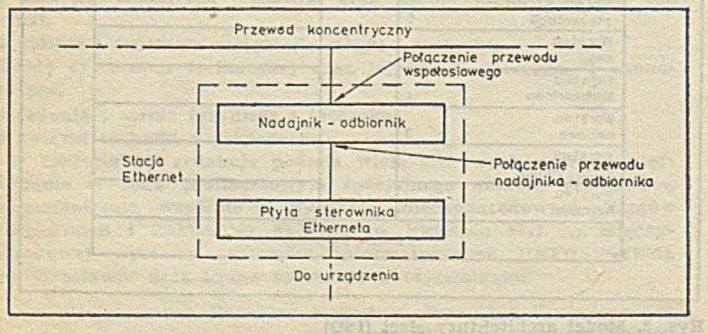
Główne rozdziały są zatytułowane następująco:
— cele specyfikacji,
— model funkcjonalny architektury Etherneta,
— sprzężenia międzywarstwowe,
— specyfikacja warstwy łącza danych,
— specyfikacja warstwy fizycznej.

We wprowadzeniu nakreślono ogólną charakterystykę sieci Ethernet i jej otoczenia, podkreślając potrzebę kompatybilności na różnych poziomach architektury sieci. Spośród literatury, na którą powołuje się specyfikacja DIX, jedynie odwołania do modelu ISO są niezbędne do jej czytania i zrozumienia.

W rozdziale 3 specyfikacji wyszczególniono cele projektu Ethernet, a także to, czego do celów nie zaliczono, jak np. zapewnienie poufności lub całkowita kontrola błędów. Oba te zadania przesunięto do warstw położonych powyżej warstwy łącza danych. Rozdział 3 stanowi kontekst, w którym zrozumiałe powinny być specyfikacje z późniejszych rozdziałów.

W rozdziale 4 opisano ideę podziału architektury sieci na warstwy oraz przedstawiono model funkcjonalny sieci z podziałem na warstwy łącza danych i fizyczną, zalecane przez ISO. Jest to przeglądowy opis obu warstw, ilustrujący jak zasady zaleceń ISO są implementowane w systemie Ethernet. Warstwy wyższe są określone jako „warstwa klienta" i nie są dalej omawiane.

W rozdziale tym przyjęto dwa istotne założenia. Pierwsze polega na tym, że warstwa łącza danych, w takiej postaci jak została opisana, ma zastosowanie do klasy sieci lokalnych o dostępie wspólnym z wykrywaniem kolizji (ang. carrier-sense multiple-access with colision-detection, CSMA-CD). Drugim założeniem jest fakt, że specyfikację podporządkowano przede wszystkich architekturze, która uwypukla logiczny podział systemu Ethernet. W implementacji systemu występują jednak istotne sprzężenia nie odpowiadające międzywarstwowym powiązaniom w modelu architektury. W szczególności, w warstwie fizycznej istnieje połączenie z przewodem koncentrycznym i połączenie z przewodem nadajnika-odbiornika (ang. transceiver), jak przedstawiono na rys. 3.



Rys. 3. Fizyczne połączenia sieci Ethernet

Nadajnik-odbiornik stanowi niewielką część stacji w sieci Ethernet, umieszczoną blisko przewodu koncentrycznego, której rola polega na dokonywaniu konwersji sygnałów między tym przewodem a płytą sterownika, mogącą znajdować się w pewnej odległości od przewodu. Zalecana jest zgodność implementacji ze specyfikacją połączenia przewodu nadajnika-odbiornika, ale nie jest to konieczne. Należy jednak zauważyć, że pełna kompatybilność połączeń z przewodem koncentrycznym ma zasadnicze znaczenie dla komunikacji w sieci Ethernet. Dzieje się tak mimo tego, że warstwa łącza danych w modelu architektury miała być w zamierzeniach niezależna od leżącej pod nią warstwy fizycznej.

W rozdziale 5 zdefiniowano w notacji języka Pascal sprzężenia między warstwą klienta a warstwą łącza danych oraz między warstwą łącza danych a warstwą fizyczną, posługując się funkcjami, procedurami i zmiennymi. Zasadniczo, warstwa łącząca danych zapewnia warstwie klienta dwie funkcje („nadaj ramkę" i „odbierz ramkę"), a warstwa fizyczna warstwie łącza danych jedną funkcję („odbierz bit"), dwie procedury („nadaj bit" i „czekaj") oraz trzy zmienne typu Boolean („wykrycie kolizji", „wykrycie nośnej" i „nadawanie"). Użycie języka Pascal nie oznacza, że powyższe funkcje, procedury i zmienne mogą lub powinny być zrealizowane programowo.

W rozdziałach 6 i 7 szczegółowo zdefiniowano warstwę łącza danych i warstwę fizyczną. O ile poprzednie rozdziały zawierają tylko opisy sprzężeń, to w tych dwóch ostatnich rozdziałach dokumentu przedstawiono dokumentację projektową dla obu warstw, która według zamierzeń autorów ma być podstawowym dokumentem w przypadku implementacji Etherneta. Rozdział 6 zawiera model proceduralny warstwy łącza danych, opisany przy użyciu języka Pascal. Warstwa fizyczna nie nadaje się do opisu proceduralnego, wobec czego rozdział 7 jest mieszaniną tekstu, rysunków i tabel. Realizacja warstwy fizycznej zależy od ośrodka transmisji, dlatego też większa część specyfikacji tej warstwy dotyczy elektrycznych szczegółów przewodu koncentrycznego oraz jego połączenia. Rozdział 7 ma podtytuł „System pasma podstawowego oparty na przewodzie koncentrycznym" (ang. „Baseband coaxial system"), co oznacza, że dokument może być zastąpiony przyszłą wersją dla innego ośrodka transmisji.

Wydaje się, że więcej kłopotów w utworzeniu sieci w pełni kompatybilnej z Ethernetem, będzie sprawiać implementacja szczegółów wyspecyfikowanych dla warstwy fizycznej niż dla warstwy łącza danych, a zwłaszcza zachowanie pełnej kompatybilności z przewodem współosiowym. Z drugiej strony, system typu Ethernet mógłby być łatwo zbudowany przez zastosowanie zasad sieci lokalnej o dostępie wspólnym z wykrywaniem kolizji (CSMA-CD), zarealizowanie warstwy fizycznej według zredukowanej specyfikacji (szczególnie ze zmniejszoną szybkością przesyłania danych, w porównaniu do 10 Mb/s w specyfikacji DIX) oraz warstwy łącza danych, bardzo podobnej do wyspecyfikowanej w dokumentach sieci Ethernet.

POWODY BUDOWY SIECI ETHERNET

Powody, dla których tworzy się sieć Ethernet, można podzielić na 3 kategorie.

● Może zaistnieć potrzeba budowania lokalnej sieci komputerowej z przyczyn czysto praktycznych; pewna liczba komputerów, terminali i innych urządzeń, w celu zapewnienia pewnych usług sieci, powinna być luźno połączona. Ten rodzaj wymagań staje się coraz bardziej powszechny, być może dlatego, że użytkownicy coraz lepiej uświadamiają sobie potencjalne korzyści związane z pracą w sieci. Użytkownicy mogą też zdecydować się na własną implementację sieci Ethernet.

● Pod względem handlowym systemy Ethernet są perspektywiczne. Producenci komputerów, terminali i innych urządzeń wyposażają swoje produkty w sprzęgi Etherneta, tak więc są to elementy sieci w pełni kompatybilne z Ethernetem. Niektóre firmy budują kompletne systemy lub zestawy elementów, które mogą być kupowane i używane przez organizacje, nie posiadające środków ani chęci do samodzielnego budowania całych systemów.

● Pomimo tego, że produkty handlowe są już dostępne, nadal widoczna jest potrzeba kontynuowania badań podstawowych w różnorodnych dziedzinach związanych z teorią i praktyką sieci lokalnych. Z pewnością dużo jest do zrobienia w zakresie architektury sprzętowej, protokołów programowych czy oceny efektywności. Istnieje więc potrzeba budowy systemów Ethernet lub przynajmniej systemów tego typu, aby dać podstawy narzędziowe do tych badań.

Jedynie w przypadku, gdy poszczególne elementy sieci Ethernet (z wyjątkiem jeszcze niewyspecyfikowanej warstwy sieciowej) są oferowane na rynku indywidualnie, istnieje potrzeba zapewnienia ich pełnej kompatybilności z Ethernetem. Elementy te muszą być łatwo dołączalne do standardowego systemu Ethernet. Produkowane i sprzedawane systemy typu Ethernet, które używają warstwy fizycznej i ośrodka transmisji, różniących się od specyfikacji w dokumencie DIX, mogą stać się w pełni kompatybilne z systemami Etherneta na poziomie warstwy łącza lub na poziomie warstw wyższych (po opracowaniu ich specyfikacji).

Na skonstruowanie autentycznego systemu Ethernet wymagane jest otrzymanie (za jednorazową opłatą 1000 dolarów) licencji od Xerox Corporation. Wiele firm, zwłaszcza w USA, podjęło już tę ofertę. Digital, Intel i Xerox chcą zachęcić innych producentów do budowania systemów w pełni kompatybilnych z Ethernetem, tak aby specyfikacja Etherneta stała się przemysłowym standardem.

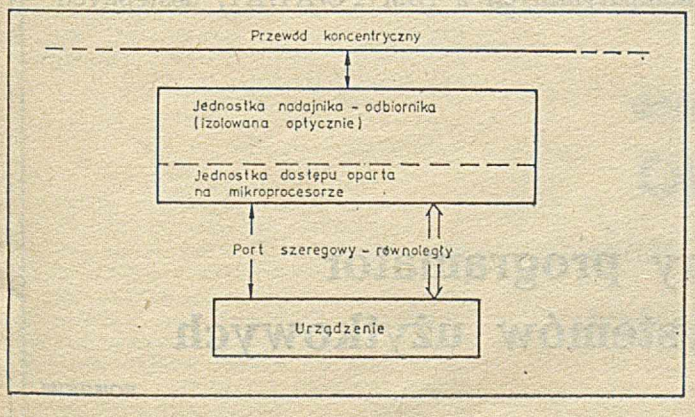
W większości przypadków sieć typu Ethernet jest adekwatna do wymogów użytkowników chcących zbudować własną, usługową sieć lokalną. Przykładowo autorzy tego artykułu budują sieć typu Ethernet na Wydziale Informatyki Uniwersytetu Strathclyde.

Autorzy będą używali tej sieci zwanej Strathnet głównie do badań porównawczych architektury i efektywności Etherneta oraz systemów o topologii pierścienia. Oprócz sieci Strathnet instalowana jest sieć Cambridge Ring [6], aby umożliwić pełniejsze eksperymentowanie z sieciami. Prace te są kontynuacją wcześniejszych badań symulacyjnych nad porównaniem sieci obu typów [1]. Sieć Strathnet będzie ostatecznie użytkowana jako sieć usługowa na Wydziale.

SIEĆ STRATHNET

Podstawą systemu jest oparta na mikroprocesorze jednostka dostępu do sieci (ang. network access unit), która w chwili obecnej zawiera układ sprzętowy i oprogramowanie odpowiadające warstwie fizycznej i warstwie łącza danych modelu ISO, a wkrótce będzie także zawierała oprogramowanie warstwy sieciowej. Jednostka dostępu jest więc w swoich zasadach działania zbliżona do pary: nadajnik-odbiornik Etherneta i płyta sterownika.

Ośrodek transmisji w sieci Strathnet stanowi przewód koncentryczny o oporności 50 ohm, podobny do zaleconego



Rys. 4. Fizyczne połączenia jednostki dostępu sieci Strathnet

dla Etherneta. Każda jednostka dostępu jest dołączona jednym końcem do przewodu koncentryrznego, a drugim do urządzenia (rys. 4). Jednostka dostępu sieci Strathnet oraz płyta sterownika i nadajnik-odbiornik Etherneta (rys. 3.) są funkcjonalnie podobne. Wyróżnia się dwa rodzaje jednostki dostępu. Pierwszy jest całkowicie niezależny, z własnym zasilaniem w energię elektryczną. Ma on dwa typy sprzężenia z urządzeniami: sprzęg szeregowy V.24 (z różnymi szybkościami transmisji) i sprzęg równoległy 8-bitowy (odpowiadający standardowemu portowi wejścia-wyjścia mikrokomputerów). Drugi rodzaj jednostki dostępu jest projektowany tak, aby działał jako zewnętrzna płyta dołączona do magistrali urządzenia i był z niego zasilany. Różnorodne odmiany drugiego typu są potrzebne do sprzęgania różnych (ale w ograniczonej liczbie) magistrali komputerów. Wszystkie typy jednostek dostępu mają dokładnie takie samo sprzężenie z ośrodkiem transmisji.

Cele stawiane przed projektem sieci Strathnet są trojakie: elastyczność, niski koszt dołączania urządzeń do sieci i umiarkowanie duża szybkość transmisji (rzędu 1 Mb/s) Osiągnięto to przez zastosowanie jednostki dostępu, która tworzy programowalną jednostkę sprzęgającą sieci, opartą na standardowych układach LSI, m.in. na rozwiniętym sterowniku łącza danych (ang. advanced data link controller, ADLC) Motorola 6854. Układ ADLC realizuje większość funkcji warstwy łącza danych, pozostawiając do zaimplementowania programowego jedynie warstwę sieciową. Jednostka dostępu umożliwia sprzęganie urządzeń przez sprzęg szeregowy V.24 i równoległy 8-bitowy adapter (ang. 8-bit parallel peripheral interface adapter, PIA) używany zwykle w mikrokomputerach.

W obecnej wersji projektu jednostki dostępu można osiągać na przewodzie koncentrycznym szybkości transmisji danych do 4 Mb/s. Autorzy są także usatysfakcjonowani innymi właściwościami ich implementacji sieci typu Ethernet. Nie osiąga ona jednak poziomu efektywności Etherneta.

* * *

Specyfikacja DIX przedstawia wartość tylko w wypadku autentycznego zapotrzebowania na elementy kompatybilne z Ethernetem. Taka sytuacja istnieje dla wielu firm, które chciałyby wykorzystać spodziewaną popularność systemów Ethernet. Firmy te, aby tworzyć systemy według specyfikacji DIX, muszą jednak nabyć licencję od Xerox Corporation za jednorazową opłatą 1000 dolarów. Cena za oryginalny patent Etherneta jest dość niska, lecz Digital, Intel i Xerox mają nadzieję uzyskać znacznie większe korzyści z będącej rezultatem tej polityki popularności Etherneta.

W wypadku jednak, gdy planuje się systemy typu Ethernet, zarówno do pełnienia funkcji usługowych, jak i odgrywania roli narzędzi badawczych, to użyteczność specyfikacji Etherneta jest ograniczona. Bardziej wartościowym źródłem informacji są wówczas oryginalne artykuły [2, 5] zawierające praktyczne uwagi na temat eksperymentalnego projektu Etherneta.

Oprac. i tłum.: ROMAN GRABOWICZ
na podst. Computer Communications,

LITERATURA

[1] Blair G. S., Hutchinson D., Shepherd W. D.: A comparison of the Cambridge ring and an Ethernet system. Internal report, Computer Science Department. University of Strathclyde, 1980

[2] Crane R. C., Taft E. A.: Practical considerations in Ethernet local network design. Xerox Parç Technical Report Palo Alto (CA), February 1980

[3] Digital, Intel and Xerox Corporations: The Ethernet: a local network — data link layer and physical layer specifications. Version 1.0, September 1980

[4] ISO: Data processing — open systems interconnection — basic reference model. Report ISO/TC97/SC16 N 537 Comput. Networks, Vol. 5, No. 2, April 1981

[5] Metcalfe R. M., Boggs D. R.: Ethernet — distributed packet switching for local computer networks. Commun. ACM Vol. 19, No. 7, July 1976

[6] Wilkes M. V., Wheeler D. I.: The Cambridge digital communication ring. Proc. Local Area Communications Network Symp., USA (May 1979)

COMPUTER STUDIO KAJKOWSCY

PROFESJONALNE OPROGRAMOWANIE MIKROKOMPUTERÓW

ul. Balladyny 3B, 81-524 Gdynia, tel.: 29-00-18, 24-01-50

Już od czterech lat działamy na rynku oprogramowaniu mikrokoputerów w Polsce, tworząc systemy znajdujące szerokie zastosowanie w zarządzaniu, finansach, księgowości i gospodarce materiałowej — narzędzia wspomagające pracę programistów, dyrektorów, księgowych i sekretarek.

Nasze programy sprawdziły się już w ponad 500 przedsiębiorstwach i instytucjach.

Dzięki wysiłkowi najlepszych fachowców jesteśmy w stanie co miesiąc przedstawić Wam nowy produkt o najwyższym standarcie światowym. Zaletą naszego oprogramowania jest możliwość wzajemnej współpracy poszczególnych systemów i tworzenia z nich pakietów o specjalnej, dostosowanej do potrzeb klienta, strukturze.

Jednocześnie firma nasza nieodpłatnie zapewnia serwis gwarancyjny i wymianę zakupionych już programów na najnowsze, udoskonalone ich wersje. Wszystkie nasze systemy są łatwe w użyciu, opracowane w języku polskim i zaopatrzone — dla wygody klienta — w wyczerpującą dokumentację.

Dla zainteresowanych oferujemy również pomoc we wdrożeniu naszych pakietów i szkolenie w zakresie ich obsługi.

Współczesny świat wymaga szybkiej i dokładnej informacji, a my chcemy pomóc w jej uzyskaniu — dlatego jesteśmy pewni, że w Twoim przedsiębiorstwie znajdą zastosowanie oferowane przez nas systemy.

	Nazwa programu	8-B	16-B
	Oprogramowanie użytkowe		
1	**BANK-DANYCH CSK** — System zarządzania bazą danych	+	+
2	**TABPLAN CSK** — Komputerowy arkusz kalkulacyjny	+	+
3	**TEKST CSK** — Pakiet redagowania tekstów	+	+
4	**TRANSCOM CSK** — Program komunikacji z ODRĄ	+	+
5	**TRANSCOM/M CSK** — Program komunikacji między mikrokomputerami	+	+
6	**BANK-GSK** — Graficzny system komunikacji z bazą danych	—	+
7	**BGRAF CSK** — Pakiet grafiki prezentacyjnej	—	+
8	**FK CSK** — System finansowo-księgowy	+	+
9	**EM CSK** — System ewidencji materiałowej	+	+
10	**PL_TEKST** — Pakiet redagowania tekstów (polskie znaki)	—	+
11	**CSK SOFT#1** — Oprogramowanie pomocnicze — BANK	—	+
12	**BANK-TP CSK** — Komunikacja między BANK CSK i TABPLAN	+	+
13	**OZ CSK** — Odtwarzanie zbiorów danych BANK CSK	+	+
	Oprogramowanie systemowe		
14	**SOMIK** — Rozbudowa systemu operacyjnego CP/M 2.0		
15	**W/SYS16 CSK** — System operacyjny wielozadaniowy — wielokonsolowy		
16	**GKS CSK** — Pakiet procedur graficznych wg normy GKS		

+ tak; — nie

Oprogramowanie CSK działa na wszystkich mikrokomputerach 8-bitowych z systemem operacyjnym CP/M oraz na komputerach 16-bitowych kompatybilnych z IBM PC/XT/AT, dostępnych w Polsce.

NOWOŚĆ!

INTEGRATOR — uniwersalny programator automatycznego łączenia systemów użytkowych

EO/81/K/86

Podprogramy obsługi przerwań dla IBM PC

Jednym z najważniejszych pojęć w technice komputerowej jest przerwanie sprzętowe. Przerwania zwiększają ogólną wydajność systemu komputerowego, ponieważ w ten sposób urządzenia zewnętrzne wymagają od procesora podjęcia obsługi tylko wtedy, gdy zachodzi potrzeba. Gdyby system nie miał przerwań, procesor musiałby systematycznie sprawdzać, czy któreś z urządzeń systemu nie wymaga obsługi.

Niestety, zarówno sprzętowe jak i programowe aspekty systemu przerwań są słabo udokumentowane i na ogół zarezerwowane dla funkcji systemu operacyjnego, takich jak zarządzanie urządzeniami zewnętrznymi. Programiści, chcący używać systemu przerwań w programach użytkowych, są często zmuszeni do uciekania się do skomplikowanych, trudnych do napisania i zrozumienia podprogramów w języku asemblera.

Celem artykułu jest podanie opisu — jak należy używać systemu przerwań na mikrokomputerze IBM PC oraz zademonstrowanie — jak można napisać podprogram obsługi przerwań (ang. interrupt service routine, ISR) w języku wysokiego poziomu (w tym wypadku w języku Turbo Pascal firmy Borland International). Czytelnicy zainteresowani tym tematem powinni również zajrzeć do artykułu „Interrupts and the IBM PC", Parts 1—2, Chris Dunford, PC Technical Journal, November-December 1983, str. 173, January 1984, str. 144. W oparciu o poniższe wskazówki, napisanie programu ISR powinno być możliwe także w innych językach.

Uwagi ogólne

Urządzenia zewnętrzne mogą zażądać od procesora podjęcia obsługi z wielu różnych powodów. Przykładowo, mogą wymagać podania sygnałów sterujących, zgłaszać gotowość do przesyłania danych, a także przekazać procesorowi meldunek o zakończeniu swojego zadania. Jakikolwiek byłby powód przerwania, jest ono zdarzeniem, które sprawia, iż procesor zawiesza wykonywanie bieżącego programu, aby podjąć wymaganą akcję.

Linia przerwania jest końcówką (ang. pin) układu mikroprocesora, której uaktywnienie powoduje, że procesor składuje bieżący stan swojego licznika rozkazów (zwykle procesor realizuje to przez umieszczenie jego zawartości na stosie) i przekazuje sterowanie pod ustalony adres. Ten ustalony adres jest początkiem programu ISR. ISR realizuje swoje zadania i przekazuje sterowanie z powrotem do miejsca, w którym wystąpiło przerwanie, w ten sposób, aby przerwany program mógł kontynuować działanie tak, jakby nic się nie zdarzyło. Program ISR musi zapamiętać zawartość każdego rejestru, zanim go zmodyfikuje, oraz musi przywrócić jego stan, zanim sterowanie wróci do przerwanego programu. W przeciwnym wypadku informacje, które przerwany program przechowywał w tych rejestrach, mogą ulec zniekształceniu.

Jeżeli procesor ma tylko jedną linię przerwań, a dołączonych urządzeń jest wiele, to pierwszą rzeczą, którą musi wykonać ISR jest sprawdzenie, które urządzenie spowodowało przerwanie. Większość procesorów jest jednak wyposażona w wiele linii przerwań, a każda linia jest związana z innym początkowym adresem obsługi. W takim wypadku każde urządzenie jest obsługiwane przez swój własny program ISR. Często — między innymi w IBM PC — adresy początku ISR są przechowywane w spójnym fragmencie pamięci, zwanym tablicą wektorów przerwań. Zwykle nazywa się to systemem przerwań wektorowych (ang. vectored interrupt system).

Niezależnie od tego, czy system ma jedno, czy wiele źródeł przerwań, zarówno część z nich jak i wszystkie naraz mogą być ignorowane (maskowane) w chwilach, gdy procesor realizuje pewien krytyczny kod i nie może być absorbowany przerwaniem. Poza tym, gdy występuje wiele linii przerwań, często mają one nadane priorytety tak, aby system mógł decydować, która powinna być obsłużona jako pierwsza, jeśli co najmniej dwa urządzenia równocześnie żądają obsługi.

System przerwań w IBM PC

Mikroprocesor 8088 mikrokomputera IBM PC posiada w swoim repertuarze specjalną instrukcję INT. Dzięki tablicy wektorów przerwań pozwala ona programom użytkowym synchronicznie przechodzić do podprogramów w sposób bardzo podobny do przerwań sprzętowych. System operacyjny korzysta z tych przerwań programowych, aby wirtualnie wykonać wszystkie operacje wejścia-wyjścia. Należy więc mieć świadomość faktu, że większość przerwań w IBM PC właściwie odwołuje się do instrukcji INT.

Dla prawdziwych przerwań sprzętowych w architekturze IBM PC istnieje osiem linii oznaczonych przez IRQ0, ..., IRQ7. Standardowy, oferowany do sprzedaży, model IBM PC używa trzech z tych linii. Linia IRQ0 jest zarezerwowana dla zegara systemu, IRQ1 dla klawiatury, zaś IRQ6 do obsługi sterownika stacji dysków elastycznych NEC PD765.

Pozostałych linii system nie używa. Mogą one natomiast być używane przez umieszczone w gniazdach (ang. peripheral slots) płyty głównej tzw. płyty (karty) dodatkowe. Linie przerwań IBM PC są uporządkowane według priorytetu; IRQ0 ma priorytet najwyższy, a IRQ7 — najniższy. Linie, każda z osobna lub wszystkie jednocześnie, mogą być zamaskowane. Do realizacji operacji przekazywania sterowania i maskowania oraz decydowania o priorytetach w IBM PC używa się programowalnego sterownika przerwań (ang. programmable interrupt controller, PIC) Intel 8259. Procesor 8088 może obsługiwać tylko jedno przerwanie w danej chwili, dlatego też układ 8259 ocenia każde żądanie i określa, czy powinien przekazać to przerwanie do procesora. Podprogram przerwania może być przerwany, jeżeli układ 8259 rozpozna żądanie o wyższym priorytecie. W standardowej konfiguracji IBM PC zegar może przerwać obsługę dysku, ale nie odwrotnie.

Załóżmy, że układ 8259 odbiera żądanie przerwania, linia nie jest zamaskowana i nie jest aktualnie obsługiwane żadne żądanie o wyższym priorytecie. Sterownik 8256 sygnalizuje procesorowi, że odebrał poprawne żądanie przerwania. Procesor kończy realizację aktualnie wykonywanej instrukcji, a następnie wysyła do układu 8259 sygnał potwierdzenia, iż jest gotów obsłużyć przerwanie. Z kolei sterownik dodaje do numeru żądania liczbę 8 i otrzymuje 8-bitowy numer przerwania. Przykładowo, jeśli obsługi przerwania żąda dysk, to numer rodzaju przerwania wynosi 14 (8 + + IRQ6). Ten numer przerwania jest umieszczany na szynie danych systemu. Procesor odczytuje numer z szyny danych i używa go jako indeksu dostępu do przechowywanej w pamięci tablicy wektorów przerwań.

Każda pozycja w tablicy wektorów przerwań składa się z czterech bajtów. Dwa pierwsze określają adres segmentu ISR, a dwa następne zawierają adres wyrównania (ang. offset) względem początku segmentu. Dla IRQ0, IRQ1 i IRQ6 adresy te wskazują początki podprogramów w systemie BIOS (ang. basic input-output system) obsługujących odpowiednie urządzenia systemu. Adresy kierują wektory pozostałych przerwań do ślepego (bezczynnego) podprogramu (ang. dummy routine) w systemie BIOS, który przekazuje sterowanie z powrotem do wykonywanego programu. Aby dodać do tablicy nowy podprogram ISR należy po prostu w miejsce adresu odpowiedniego ślepego podprogramu wpisać adres nowego podprogramu obsługi przerwania.

Funkcjonowanie ISR

Pierwszą czynnością, którą wykonuje podprogram ISR, jest zapamiętanie stanu rejestrów. Rejestry te są modyfikowane w czasie obsługi przerwania, lecz po jej zakończeniu należy przywrócić ich pierwotny stan. Najłatwiej zrealizować to przez umieszczenie ich zawartości na stosie. Jeżeli podprogram ISR jest napisany w języku asemblera, to można zachować tylko te rejestry, których rzeczywiście się

używa. Jeżeli jednak pisze się ISR w języku wysokiego poziomu, np. w Pascalu, to niemożliwe jest określenie z góry, których rejestrów będzie używał kompilator. Dlatego w podprogramie ISR należy zachować wszystkie rejestry. Turbo Pascal posiada instrukcję inline, która pozwala wstawić kod maszynowy bezpośrednio do programu. Można użyć tej instrukcji na początku ISR do wygenerowania instrukcji PUSH i ponownie na końcu do wygenerowania instrukcji POP.

Czy można przystąpić do pisania właściwego kodu, który będzie obsługiwał przerwania, jeśli wiadomo już jak zachować stan rejestrów? Otóż, jeszcze nie można. Należy bowiem pamiętać, że ISR jest wywoływany asynchronicznie. Nie powoduje to kłopotów, jeżeli przerwany kod jest fragmentem programu użytkownika. Jednak większość języków wysokiego poziomu (włączając w to Turbo Pascal) do wykonania operacji wejścia-wyjścia używa systemu BIOS. Załóżmy, że program jest przerywany w takim momencie. Sterowanie zostanie przekazane do ISR użytkownika, ale rejestr segmentu danych DS będzie zawierał segment danych BIOS. Jeżeli kod ISR będzie usiłował uzyskać dostęp do którejś ze zmiennych Pascala, to rejestr DS będzie niewłaściwy i pozycje w pamięci, do których ISR się odwołuje będą położone prawdopodobnie gdzieś w systemie BIOS, a nie w programie pascalowym. Nie trzeba dodawać, że spowoduje to, iż program zacznie zachowywać się w sposób trudny do przewidzenia, szczególnie jeśli podprogram ISR będzie usiłował zmodyfikować te zmienne.

Jak pokonać tę trudność? Po umieszczeniu rejestrów na stosie należy pzywrócić stan rejestru segmentu danych programu w Pascalu. Zadanie to można zrealizować na kilka sposobów; poniżej opisano jedną z metod.

Jak zapewnia firma Borland, słowa od 5 do 15 w segmencie kodu zawierają kod niewykonywalny i program nie zostanie uszkodzony, jeśli w tym obszarze będą przechowywane dane. Dlatego w programie głównym, w tym obszarze tworzy się zmienną bezwzględną, w której pierwsze wykonywane instrukcje, używając funkcji Turbo Pascala Dseg, zachowują adres z DS.

Gdy zawartość rejestru segmentu danych DS zostanie zachowana pod znanym adresem bezwzględnym, podprogram ISR może przywracać jego stan za każdym razem, gdy jest wywoływany. Pierwszą czynnością, którą wykonuje ISR jest kopiowanie rejestru segmentu kodu CS do rejestru DS; pozwala to na wczytanie zawartości zmiennej bezwzględnej do rejestru AX. Następnie, aby zakończyć operację, przesyła się AX do rejestru DS.

```
Program Test(input,output);

var
  dsave         :integer absolute Cseg:$0006;

Procedure Interrupt_Service_Routine;

  begin
    inline($FB/                 { STI   umożliwia dalsze przerwania     }
           $1E/                 { PUSH DS                               }
           $50/                 { PUSH AX                               }
           $53/                 { PUSH BX                               }
           $51/                 { PUSH CX                               }
           $52/                 { PUSH DX                               }
           $57/                 { PUSH DI                               }
           $56/                 { PUSH SI                               }
           $06/                 { PUSH ES                               }

    inline($8C/$C8/             { MOV AX,CS przywraca stan segmentu danych }
           $8E/$D8/             { MOV DS,AX                             }
           $A1/dsave/           { MOV AX,dsave                          }
           $8E/$D8);            { MOV DS,AX                             }
                .
                .
        { Kod programu w Pascalu do obslugi przerwar                    }
          .     .
                .
    port[$0020] := $20;         { niespecyficzne EOI do 8259 PIC        }

    inline($07/                 { POP ES                                }
           $5E/                 { POP SI                                }
           $5F/                 { POP DI                                }
           $5A/                 { POP DX                                }
           $59/                 { POP CX                                }
           $5B/                 { POP BX                                }
           $58/                 { POP AX                                }
           $1F/                 { POP DS                                }
           $CF);                { IRET    powrót z przerwania           }
  end

begin
  dsave := Dseg;
end.
```

Wydruk 1. Zasadnicza struktura podprogramu obsługi przerwań ISR, napisanego w Turbo Pascalu. Kolejność i liczba rejestrów różni się nieco od tego, co zaleca się w podręcznikach Turbo Pascala, ale zdaniem Autora jest to niezbędne, aby zapewnić właściwe funkcjonowanie programu

Bezpośrednio przed usunięciem zawartości rejestrów ze stosu i powrotem z podprogramu musi on zasygnalizować układowi 8259, że już skończył obsługę urządzenia. Sterownik 8259 ma 8-bitowy rejestr, w którym zapisuje informacje o działających podprogramach ISR. Odpowiedni dla podprogramu obsługi (np. dysku) bit należy wyzerować, gdy podprogram zakończy działanie. W IBM PC realizuje się to przez wysłanie wartości 20 (szesnastkowo) do portu sterującego układu 8259. Nazywa się to niespecyficznym końcem przerwania (ang. nonspecific end of interrupt, EOI). Gdy sterownik 8259 otrzymuje sygnał niespecyficznego końca przerwania, zeruje bit odpowiadający przerwaniu o najwyższym priorytecie wśród przerwań będących w toku realizacji. Podprogramem o najwyższym priorytecie będzie zawsze ten podprogram, który wysłał rozkaz EOI.

W IBM PC porty 8259 mają adresy 20 i 21 (szesnastkowo). Dostęp do portów wejścia-wyjścia można mieć bezpośrednio z programu w języku wysokiego poziomu, używając standardowo zdefiniowanej w Turbo Pascalu tablicy portów. Zasadniczy szkielet podprogramu ISR przedstawiono na wydruku 1.

Inicjalizacja systemu w celu obsługi przerwań

Gdy wiadomo już, jak konstruuje się podprogram ISR, należy przygotować system do obsługi przerwań. Po pierwsze, trzeba zasygnalizować układowi 8259, że odpowiednia linia IRQ ma nie być maskowana. Sterownik 8259 PIC ma 8-bitowy rejestr, zwany rejestrem maski przerwań (ang. interrupt-mask register, IMR). Każdy bit od 0 do 7 odpowiada linii żądania przerwania. Jeśli bit odpowiadający linii jest ustawiony na 1, to linia ta jest zamaskowana i kolejno następujące przerwania nie będą przekazywane do procesora. Podczas początkowego ładowania systemu, MS-DOS inicjalizuje sterownik 8259 ustawiając wszystkie bity rejestru na 1, z wyjątkiem bitów odpowiadającym liniom IRQ0, IRQ1, IRQ6. Aby uaktywnić dodatkową linię IRQ należy wyzerować odpowiedni bit. Na wydruku 2 przedstawiono procedurę, która to realizuje.

Następnym krokiem jest modyfikacja tablicy wektorów przerwań tak, aby odpowiednia pozycja w tablicy wskazywała generowany podprogram ISR. Adres ISR można znaleźć używając funkcji Turbo Pascala Cseg i Ofs. Należy jednak pamiętać o tym, że kompilator dodaje kilka własnych instrukcji na początku każdej procedury. W konsekwencji, pierwszą instrukcją skompilowanego podprogramu ISR nie będzie wstawiona instrukcja Pascala inline — instrukcja STI. Właściwy początek podprogramu ISR znajduje się siedem bajtów wcześniej niż pierwsza instrukcja kodu źródłowego. Ponieważ instrukcje dodane przez kompilator używają rejestrów, zanim można przechować ich zawartość, należy dodać do adresu wyrównania (względ-

```
Procedure Set_IVT(entry : integer);

  var
     offset, segment, first_word, second_word : integer;

  begin
     offset := Ofs(Interrupt_Service_Routine) + 7;
     segment                  := Cseg;
     first_word               := (entry + 8) * 4;
     second_word              := first_word + 2;
     memW[$0000:first_word]   := offset;
     memW[$0000:second_word]  := segment;
  end;
```

Wydruk 2. Procedura Turbo Pascala, która umożliwia przerwanie. Bajtowa zmienna IRQ wylicza linię przerwania, która ma być „odmaskowana"

```
Procedure Enable_IRQx(IRQ : byte);

  var
     imr, mask : byte;

  begin
     mask       := not (1 shl IRQ );
     imr        := port[$21];          { Weź IMR OD 8259          }
     imr        := imr and mask;       { wyzeruj bit maski        }
     port[$21]  := imr;                { i powróć do kontrolera   }
  end;
```

Wydruk 3. Procedura Turbo Pascala służąca do ustawienia wektora przerwań. Zmienna entry wybiera pozycję w tablicy, która ma być ustawiona; pozycja O odpowiada linii przerwań 0 itd.

dem początku segmentu) zwróconego przez funkcję Ofs liczbę 7. Dzięki temu instrukcje zawarte w pierwszych siedmiu bajtach nie będą wykonywane. Tablica wektorów przerwań jest następnie modyfikowana za pomocą standardowo zdefiniowanej w implementacji Turbo Pascala tablicy memW. Na wydruku 3 przedstawiono procedurę, która to realizuje.

Przykładowe implementacje

W celu zademonstrowania użycia systemu przerwań w IBM PC przy wykorzystaniu Turbo Pascala opiszemy program odczytu danych z taśmy kasetowej, transmitujący 16-bitowe słowa z częstotliwości 2000 Hz i zapisujący je na dysku. Zakłada się, że nie ma możliwości uzgadniania transmisji (ang. handshaking), ponieważ czytnik taśmy po prostu umieszcza dane na liniach wejściowych i bramkuje (ang. strobe) urządzenie odbierające co 500 mikrosekund.

Aby odczytać dane, należy wybrać programowalny port równoległy Intel 8255. Program powinien odczytać dwie 8-bitowe porcje z portów A i B układu 8255 i uformować 16-bitowe słowo danych. Układ 8255 należy zaprogramować tak, aby generował przerwanie, gdy zatrzaśnie (ang. latch) dane, i przełączał linię IRQ7.

Program powinien jednocześnie zbierać dane i zapisywać je na dysku. Ponieważ pojemność dyskietki jest ograniczona do 360 KB, w chwili jej zapełnienia program może kończyć swoje działanie. Po to, by równocześnie zbierać dane i wysyłać je na dysk, należy użyć dwóch dużych buforów o pojemności po 12 800 słów każdy. W programie powinny znaleźć się dwa podprogramy obsługi przerwań, do wypełniania buforów. Tablicę wektorów przerwań należy ustawić początkowo w ten sposób, aby wskazywała podprogram ISR1. Gdy pierwszy bufor będzie pełny, tablica wektorów przerwań powinna być zmieniona tak, aby wskazywała ISR2. W czasie, gdy drugi bufor będzie wypełniany, dane z pierwszego powinny być wysyłane na dysk.

Czytelnikowi pozostawiamy wdzięczne zadanie samodzielnego napisania tego programu.

* * *

Jak wykazano, cały podprogram obsługi przerwań, z wyjątkiem kilku wstawianych w tekst programu instrukcji maszynowych, może być napisany w Pascalu. Przykładowe wydruki mogą służyć programistom, chcącym opracować swoje moduły obsługi przerwań, za szkielet ich własnych programów.

Oprac. ROMAN GRABOWICZ
na podstawie BYTE, No 11, 1985

European Workshop on Industrial Computer Systems (EWICS)
Technical Committee 7 (TC 7)

EWICS TC7 (TC on Reliability, Safety and Security) grupuje europejskich ekspertów, zajmujących się zagadnieniami niezawodności, bezpieczeństwa i poufności systemów komputerowych w zastosowaniach przemysłowych. W szczególności, prace Komitetu obejmują zastosowania systemów komputerowych w energetyce jądrowej, przemyśle chemicznym, transporcie, telekomunikacji, sterowaniu ruchem lotniczym itp. Członkowie Komitetu rekrutują się ze środowisk przemysłowych (zarówno użytkowników jak i producentów systemów komputerowych) oraz ze środowisk akademickich z 12 krajów Europy.

Prace Komitetu są wspierane przez Wydział Technologii Przetwarzania Informacji i Telekomunikacji Europejskiej Wspólnoty Gospodarczej za pośrednictwem Brytyjskiego Towarzystwa Niezawodności i Bezpieczeństwa (United Kingdom Safety and Reliability Society). Przewiduje się opracowanie (do 1987 r.) zaleceń i przewodników przeznaczonych dla wytwórców i użytkowników systemów komputerowych czasu rzeczywistego, w których niezawodność i bezpieczeństwo stanowią krytyczne elementy w ocenie jakości systemu.

Prace EWICS TC7 są prowadzone w czterech grupach tematycznych:

- utrzymanie integralności w ramach pełnego cyklu istnienia systemu,
- metody oceny systemu w celu wykazania, że osiągnął on wymagany poziom niezawodności i bezpieczeństwa,
- miary jakości oraz techniki polepszania jakości oprogramowania dla systemów krytycznych ze względu na niezawodność i bezpieczeństwo,
- metody konstruowania systemów zwiększonej niezawodności i bezpieczeństwa (zarówno w zakresie sprzętu jak i oprogramowania).

Zakłada się, że finalne opracowania EWICS TC7 będą przedstawione w formie nadającej się do bezpośredniego wykorzystania w środowiskach przemysłowych. Treść tych opracowań uwzględnia istniejące normy ISO i IEC.

Dalsze informacje dotyczące prac EWICS TC7 można otrzymać od członka Komitetu — dr. inż. Janusza Górskiego:

Instytut Informatyki
Politechnika Gdańska
Majakowskiego 11—12
80-952 Gdańsk

Informacje dotyczące United Kingdom Safety and Reliability Society można otrzymać pisząc pod adresem:

B. K. Daniels
P O Box 25
Cambridge Arcade
Lord Street
Southport PR8 1AS
Wielka Brytania

(JG)

Kto jest kim w IFIP

Ashley Goldsworthy
(Australia)

Ashley Goldsworthy zaczął współpracę z IFIP w 1974 roku. Będąc wtedy prezesem Australijskiego Towarzystwa Komputerowego brał udział w Zgromadzeniu Ogólnym IFIP. Jako przedstawiciel Australii złożył ofertę zorganizowania Kongresu w 1980 roku, w Australii. Ostatecznie Kongres był wspólnym przedsięwzięciem Japonii i Australii, i odbył się w dwóch częściach: pierwszy tydzień w Tokio, drugi w Melbourne. Goldsworthy był w latach 1974—1980 przewodniczącym Australijskiego Komitetu Organizacyjnego Kongresu IFIP. Od 1974 roku uczestniczył we wszystkich posiedzeniach Rady i Zgromadzeniach Ogólnych, najpierw jako przewodniczący Komitetu Organizacyjnego Kongresu, a później — od 1980 roku — jako delegat Australii. W tym roku wybrano go członkiem zarządu, a w 1983 wiceprezesem IFIP. Od 1981 roku jest przewodniczącym Komitetu Planowania.

A. Goldsworthy ukończył studia ekonomiczno-administracyjne na Uniwersytecie stanu Queensland, a na Uniwersytecie Griffitha otrzymał tytuł magistra nauk przyrodniczych. Z informatyką zetknął się na początku lat sześćdziesiątych, pracując w Telecom Australia, a później w Urzędzie Statystycznym i przez kilka lat w towarzystwie ubezpieczeniowym.

Obecnie jest dyrektorem wykonawczym jednej z największych australijskich spółek budowlanych (towarzystw oszczędnościowo-pożyczkowych). Głównym obszarem jego zainteresowań w dziedzinie przetwarzania informacji są społeczne i ekonomiczne skutki komputeryzacji i wprowadzania nowoczesnych metod zarządzania; opublikował ponad 70 artykułów na te tematy.

A. Goldsworthy był pierwszym Australijczykiem uhonorowanym przez królową tytułem Oficera Orderu Imperium Brytyjskiego za zasługi w dziedzinie komputeryzacji. W 1981 roku wybrano go w Australii „Człowiekiem Roku w Branży Komputerowej".

Jest jednym z siedmiu członków rzeczywistych (najwyższych rangą) Australijskiego Towarzystwa Komputerowego, którego był prezesem przez dwie kadencje (1974—1976 i 1982—1983), co nie miało precedensu. Nieprzerwanie od 1968 roku jest członkiem Rady Towarzystwa. Od 1979 roku, a więc od chwili powstania, jest przewodniczącym Australijskiej Krajowej Rady Technologii Informacyjnej, a także Gubernatorem Międzynarodowej Rady do spraw Komunikacji Komputerowej (ang. International Council for Computer Communications, ICCC), jednego z członków stowarzyszonych IFIP. A. Goldsworthy uczestniczy w pracach różnych komitetów uniwersyteckich i narodowych komitetów doradczych, takich jak: Biblioteka Australijska czy Komisja Reformy Prawa Australijskiego.

Pan Goldsworthy i jego żona Shirley, która jest nauczycielką tańca, mieszkają w Brisbane w stanie Queensland. Mają czworo dorosłych dzieci i ostatnio zostali szczęśliwymi dziadkami swojej pierwszej wnuczki.

(MK)

Nareszcie ИНФОРМАТИКА = INFORMATYKA

Po blisko ćwierćwieczu rozbieżności w interpretacji nazwy информатика w fachowej literaturze radzieckiej w ZSRR nastąpiło rozstrzygnięcie problemu. Świadczy o tym poniższy przekład fragmentu artykułu akademika A. A. Dorodnicyna, na temat przedmiotu i zadań informatyki [1]).

(Red.)

W Akademii Nauk ZSRR powołano nowy wydział — informatyki, techniki obliczeniowej i automatyzacji. Co to jest — informatyka?

Termin ten wprowadzili 15 lat temu naukowcy francuscy. Oznacza on *naukę o przekształcaniu informacji.* Pod pojęciem informacji rozumie się dowolny zbiór sygnałów, oddziaływań czy też wiadomości — które dowolnie obrany obiekt czy też system pobiera z zewnątrz (informacja wejściowa), przekazuje do otoczenia (informacja wyjściowa) lub zachowuje w sobie (informacja wewnętrzna). Przekształcanie tego rodzaju informacji opiera się — oczywiście — na technice obliczeniowej. Tak więc strukturę informatyki wyznaczają jej trzy nierozdzielne składniki: maszyny liczące, urządzenia programujące oraz środki algorytmiczne.

Informatyka, podobnie jak matematyka, jest służebna w stosunku do innych nauk. Nie bada ona żadnego konkretnego obiektu materialnego ani zachodzącego w przyrodzie procesu, lecz uzbraja swymi metodami inne, specjalistyczne gałęzie wiedzy. Początkowo było to szczególnie wyraźnie widać w fizyce jądrowej czy też w mechanice konstrukcji lotniczych. Potem metody informatyczne zaczęto stosować w ekonomice przedsiębiorstw i zarządzaniu — a wreszcie informatyka zawładnęła sterowaniem procesów produkcyjnych oraz automatyzacją projektowania.

W obecnej dobie charakterystyczne jest także przenikanie informatyki do tzw. nauk opisowych jak: biologia, medycyna czy też socjologia, które jeszcze nie tak dawno uważano za wręcz niedostępne dla precyzyjnych metod ilościowych (matematycznych). Do tych nauk informatyka wniosła dwa nowe podejścia: **modelowanie matematyczne** oraz **rozpoznawanie obrazów.** W oparciu o nie stało się możliwe przewidywanie rozwoju tych zjawisk, jakimi zajmują się nauki opisowe, które dzięki temu otworzyły sobie furtkę do wiedzy ścisłej.

Nietrudno sobie wyobrazić, jak ogromne znaczenie będzie to miało dla ludzkości. Wszak *matematyzacja* wiedzy umożliwia z założenia ścisłe opisywanie obiektów rzeczywistych, takich jak: domy, mosty, parowozy, radioodbiornik. Natomiast w biologii, medycynie, agrotechnice, hodowli, socjologii — nowe obiekty nadal bada się metodami prób i błędów. Dlatego też wdrożenie metod informatycznych do nauk opisowych będzie jednym z ważniejszych zadań doby jutrzejszej.

A. A. DORODNICYN

(Tłumaczył A.B.E.)

[1]) Anatolij Aleksiejewicz Dorodnicyn: Informatika — priedmiet i zadaczi. Wiestnik AN SSSR, No. 2, 1985 (cyt. za: Nauka i Żizń, No. 6, s. 37, 1985).

Informatyka doświadczalna

Kiedy do mojego nauczyciela programowania zwracał się któryś ze studentów, kwestionując działanie translatora, ten przybliżał do oczu tabulogram (aktualny — bez niego nie rozpoczynał rozmowy), zastygał na chwilę w milczeniu, po czym... kierował delikwenta na pogłębione studia gramatyki Algolu, języka asemblera lub zalecał wnikliwą lekturę wybranej partii dokumentacji przedmiotu.

Obrazek taki był regułą. By go dopełnić, dodam, że od czasu do czasu konstatowaliśmy przy wywoływaniu translatora, że zmienił się jego numer — tak bezszelestnie i szybko usuwane zeń były błędy, niedostrzegalne dla zwykłego użytkownika. W rzadszych odstępach czasu wiadomość o nowej wersji translatora czy systemu operacyjnego ogłaszano publicznie, a w ślad za nią użytkownik dostawał do ręki zwięzły — lecz o dziwo — składną polszczyzną napisany biuletyn.

Działo się to w czasach, gdy nowa specjalność wybijała się z matematyki na informatykę, i nie wyartykułowano jeszcze dobitnie takich pojęć, jak: metodologia programowania, cykl istnienia oprogramowania, serwis i pielęgnacja systemów.

Dziś, kiedy uczeń poddany informatycznemu eksperymentowi pyta: „Czy liczba w Logo jest słowem?" — nie odeślę go do gramatyki, bo jej nie ma (nie podołałaby!), odpowiem za to obiecująco, cytując źródłowy podręcznik: „Liczby w Logo są również słowami, serdeńko". Pół biedy, jeśli to uczniowi wystarczy, gdy jednak chłopak staje się zbyt dociekliwy, robi się niebezpiecznie (znających Logo zachęcam do oceny pozaartystycznych walorów załączonej jednoaktówki).

Dziś, kiedy propaguję informatykę w gronie zainteresowanych urzędników (to co, że na Spectrum; w kwestii, którą poruszam, winno to być bez znaczenia), rumienię się od środka — by nie było widać, gdy mile w ojczystym języku gwarzący elektroniczny formularz, rzuca co pewien czas sygnałem — obelga-zagadka: NONSENSE IN BASIC — zwykle wtedy, gdy Basicowi ducha winny użytkownik popełnia błąd w którymś z poleceń dla owegoż programu.

Dziś, kiedy kontemplując, po zawezwaniu pomocy, menu mikrokomputerowej bazy danych, chcę wydać któreś z uwidocznionych na ekranie poleceń, daremnie gniotę klawiaturę. Po krótkim mozole zostaję przez program wyćwiczony: albo widzisz jadłospis, albo zamawiasz danie, leniwcze — chciałoby ci się jednego i drugiego na raz? Tak nie potraktowano mnie w żadnej restauracji.

Dziś, kiedy obserwuję, jak znajomy matematyk, penetrując rewiry dostępnego mu oprogramowania, mężnie i ze spokojem stawia czoła kolejnym — jak mówi — pluskwom, nie dziwię się już, ni smucę.

Dziś, gdy wysyłam — czy to prywatnie, czy oficjalnie — do firmy, która upowszechnia software'owe kawałki, całkiem uprzejmą informację-reklamację, nie otrzymuję nawet słowa odpowiedzi.

Dziś, kiedy wesół jak skowronek wyprowadzam pod koniec sesji katalog dyskietki, widok nowych wpisów o niedopuszczalnych nazwach (których nigdy nie dokonałem, choćby dlatego, że są niedopuszczalne) paraliżuje mnie tylko na moment, bo już po chwili dziarsko przekopiowuję do nowej koperty to, co jeszcze da się uratować z okaleczonego „systemowo" nośnika.

Dziś, kiedy widzę w TV „wygodną pamięć kasetową" typu NNNNX (za jedyne mmmmm złotych) albo takąż jej reklamę w gazecie, trzęsę się niczym dźwięk z nierówno przesuwanej taśmy, a gdy zerkam na licznik owej pamięci, zamontowany do góry nogami, śmieję się już nie gorzej niż wtedy, gdy — przez właściwość sprzęgów, niewydolność zasilaczy, a może i z innego powodu — jak Spectrum ma dyskietki, to nie drukarkę (i na odwrót), albo jak ma pierwsze lub drugie, to traci wzmocnienie dźwięku (bo kabelek nie sięga). A gdy z przykrótkiego gniazda udałej przystawki wypadnie zasilanie w najważniejszym momencie (a czy może być inny?), śmieje się nawet telewizor.

Dziś, gdy oglądam w fachowym wydawnictwie straszydełka „32-, 16- i 8-bitowe" nie drgnie mi nawet z dawna nawykła powieka, a kiedy dowiaduję się, że: „skoncentrujemy się na wersji (podkr. moje) BASIC-80 Rev 5.21 CP/M Version Copyright 1977— —1981 by Microsoft created: 28 Jul — 81 32824 Bytes free", lub że: „Zwykle przedstawia się go (żółwia Logo — przyp. aut.) na ekranie w formie małego trójkąta równobocznego, tak że zarówno jego położenie, jak i kierunek są dobrze widoczne" — myślę tylko: czy to żółw, czy to kaczka (i czyja)?

Weselę się zatem bez przerwy, nawet mnie od tego już trochę kłuje w dołku i doświadczam, doświadczam, doświadczam... Doświadczam sprawności sprzętu, zależności między dokumentacją a rzeczywistym programem, związków rzeczywistości z moimi wyobrażeniami, własnego odchodzenia od celu, z którym zasiadłem przy komputerze i trwałości zdrowia, którym się (jeszcze?) cieszę.

I tak sobie myślę, eksperymentując: co to będzie, gdy wraz ze mną (który, niczym stary wirus grypy, jestem już na te doświadczenia uodporniony) doświadczać tak zaczną informatyki uczniowie, nauczyciele, rzesze chętnych i życzliwych niezawodowców?

Gdy Kajowi wpadł w oko odłamek z rozbitego zwierciadła złego czarownika, nie najlepiej mu to posłużyło w ocenie otaczającego go świata i ludzi. Co będzie, gdy „informatyka doświadczalna" utknie złym pierwszym wspomnieniem w pamięci tych, ku którym się teraz skłania?

Informatyka, a informatyka powszechna osobliwie, ma naturalne powody, by stać się szkołą działań zaplanowanych, skutecznie realizowanych i osiąganych zgodnie z początkowymi (korygowanymi, gdy trzeba) założeniami — szkołą klarownych całości i wypieszczonych detali. Zatem, dość informatyki doświadczalnej! Niech wzrasta (wszędzie) informatyka odpowiedzialna! Tylko — czy to się jeszcze opłaca?

ZDZISŁAW PŁOSKI

Zabrodzki J.: Stan obecny i perspektywy rozwoju układów LSI i VLSI

INFORMATYKA 1986, nr 7—8, s. 1

Charakterystyka aktualnego stanu oraz perspektywy rozwoju projektowania i technologii wytwarzania układów LSI i VLSI.

Pierzchała E.: Interakcyjne środowisko języka Lisp (1)

INFORMATYKA 1986, nr 7—8, s. 5

Pierwsza część charakterystyki języka Lisp, zawierająca omówienie tych jego cech, które wpływają na jego szczególną przydatność do budowy środowiska interakcyjnego.

Owczarczyk J., Stolarski M., Woźniak E.: Architektura współbieżnych systemów przetwarzania obrazów

INFORMATYKA 1986, nr 7—8, s. 8

Charakterystyka rozwiązań oraz światowych tendencji rozwoju architektury współbieżnych systemów przetwarzania obrazów.

Iszkowski W.: Architektura systemów n-mikroprocesorowych

INFORMATYKA 1986, nr 7—8, s. 12

Charakterystyka architektury systemów wieloprocesorowych zbudowanych w oparciu o mikroprocesory. Omówiono klasyfikację systemów cyfrowych oraz różne rozwiązania systemów wieloprocesorowych.

Budzianowski L., Kwiatkowski J., Wietrzych J.: Komputer ODRA 1305 w Międzyuczelnianej Sieci Komputerowej

INFORMATYKA 1986, nr 7—8, s. 16

Omówienie sposobu dołączenia do podsieci MSK komputera ODRA 1305 i związanych z nim terminali. Podano podstawowe rozwiązania oraz uzyskane doświadczenia eksploatacyjne.

Królikowski Z.: Generowanie planów wykonywania transakcji w systemach rozproszonych baz danych (2)

INFORMATYKA 1986, nr 7—8, s. 20

Druga część charakterystyki metody generowania planów wykonywania transakcji w systemach rozproszonych baz danych. Podano klasyfikację oraz różne algorytmy generowania planów.

Stokłosa J.: Z dziejów sterowania

INFORMATYKA 1986, nr 7—8, s. 24

Krótki przegląd głównych etapów rozwoju sterowania od starożytności do czasów współczesnych.

Zakharov V.: Komputery osobiste i indywidualne stanowiska robocze

INFORMATYKA 1986, nr 7—8, s. 26

Porównanie właściwości komputerów osobistych i komputerowych indywidualnych stanowisk roboczych. Przedstawiono ewolucję komputerów osobistych na przykładach sprzętu firmy IBM, charakterystykę obecnych rozwiązań indywidualnych stanowisk roboczych oraz prognozę ich rozwoju w ciągu najbliższych lat.

Dworakowski W.: Komputery osobiste IBM PC

INFORMATYKA 1986, nr 7—8, s. 30

Charakterystyka konstrukcji modeli 1 oraz XT komputera osobistego IBM PC. Omówiono procesor i koprocesor numeryczny, pamięci RAM i ROM, sterownik DMA oraz układy: czasowy i równoległy.

Grabowicz R., Zalewski J.: System operacyjny PC-DOS (1)

INFORMATYKA 1986, nr 7—8, s. 32

Pierwsza część charakterystyki systemu operacyjnego PC-DOS dla mikrokomputerów rodziny IBM PC. Omówiono jego genezę oraz organizację systemu plików.

Kleiber M., Saran M.: Doświadczenia z użytkowania Fortranu na mikrokomputerze IBM PC

INFORMATYKA 1986, nr 7—8, s. 34

Charakterystyka możliwości oraz ocena przydatności kompilatora MS-FORTRAN 3.20 firmy Microsoft w zastosowaniach naukowo-technicznych. Ocena została opracowana na podstawie doświadczeń zebranych przez autorów podczas tworzenia oprogramowania użytkowego na różnych mikrokomputerach kompatybilnych z IBM PC oraz IBM PC/XT.

Markowski M. A.: Logiczna organizacja dysku w systemie operacyjnym PC-DOS

INFORMATYKA 1986, nr 7—8, s. 38

Szczegółowa charakterystyka logicznej organizacji dysku w systemie operacyjnym PC-DOS. Omówiono organizację i zasady działania skorowidzów oraz tablicy alokacji plików z uwzględnieniem możliwości odzyskiwania skasowanych lub uszkodzonych plików zapisanych na dyskietkach lub dyskach.

Dworakowski W.: Techniki testowania układów a analiza sygnatur

INFORMATYKA 1986, nr 7—8, s. 40

Charakterystyka problemów związanych z testowaniem układów o duzym stopniu scalenia. Omówiono opracowaną przez firmę Hewlett-Packard metodę analizy sygnatur.

Забродски Я.: Настоящее состояние и перспективы развития схем LSI и VLSI

INFORMATYKA 1986, № 7—8, с. 1

Характеристика настоящего состояния и перспектив развития проектирования и технологии производства систем LSI и VLSI.

Пешхала Э.: Интерактивная среда языка Лисп (1)

INFORMATYKA 1986, № 7—8, с. 5

Первая часть характеристики языка Лисп, содержащая описание тех его свойств, которые влияют на его особую пригодность для построения интерактивной среды.

Овчарчик Я., Столярски М., Возняк Э.: Архитектура согласованных систем обработки изображений

INFORMATYKA 1986, № 7—8, с. 8

Характеристика решений и мировых тенденций развития архитектуры согласованных систем обработки изображений.

Ишковски В.: Архитектуры n-микропроцессорных систем

INFORMATYKA 1986, № 7—8, с. 12

Характеристика архитектуры многопроцессорных систем, построенных на базе микропроцессоров. Описание классификации цифровых систем и различные решения многопроцессорных систем.

Будзяновски Л., Квятковский Я., Ветрых Е.: ЭВМ ОДРА 1305 в Межвузовской вычислительной сети

INFORMATYKA 1986, № 7—8, с. 16

Описание способа включения в подсеть МВС ЭВМ ОДРА 1305 и связанных с ней терминалов. Приведены основные решения и достигнутый эксплуатационный опыт.

Круликовски З.: Разработка планов выполнения сделок в рассеянных системах баз данных (2)

INFORMATYKA 1986, № 7—8, с. 20

Вторая часть характеристики метода разработки планов выполнения сделок в системах рассеянных баз данных. Классификация и разные алгоритмы разработки планов.

Стоклоса Я.: Из истории управления

INFORMATYKA 1986, № 7—8, с. 24

Краткий обзор основных этапов развития управления с древних до современных времен.

Захаров В.: Персональные компьютеры и индивидуальные рабочие посты

INFORMATYKA 1986, № 7—8, с. 26

Сравнение свойств персональных компьютеров и компьютерных индивидуальных рабочих постов. Эволюция персональных компьютеров на примерах оборудования фирмы ИБМ, характеристика существующих конструктивных решений индивидуальных рабочих постов и прогнозы их развития на ближайшие годы.

Двораковски В.: Персональные компьютеры ИБМ PC

INFORM TY A 1986, № 7—8, с. 30

Характеристика конструкций модели 1 и XT персонального компьютера ИБМ PC. Описание процессора и числового сопроцессора, постоянной памяти и памяти со свободной выборкой данных, контроллера ДМА, а также временной и параллельной схемы.

Грабович Р., Залевский Я.: Операционная система PC-ДОС (1)

INFORMATYKA 1986, № 7—8, с. 32

Первая часть характеристики операционной системы PC-ДОС для микро-ЭВМ семейства ИБМ PC. Описание ее создания и организации системы файлов.

Клейбер М., Саран М.: Опыт использования языка ФОРТРАН на микро-ЭВМ ИБМ PC

INFORMATYKA 1986, № 7—8, с. 34

Характеристика возможности и оценка пригодности компилятора МС-ФОРТРАН 3.20 фирмы Микрософт в научно-технических расчетах. Оценка разработана на основе опыта, накопленного авторами в ходе разработки прикладного программного обеспечения на разных микро-ЭВМ, совместимых с ИБМ PC и ИБМ PC/XT.

Марковски М.А.: Логическая организация диска в операционной системе PC-ДОС

INFORMATYKA 1986, № 7—8, с. 38

Подробная характеристика логической организации диска в операционной системе PC-ДОС. Описание организации и принципов действия индексов, а также таблицы расположения файлов с учетом возможностей восстановления ликвидированных или поврежденных файлов, записанных на малых дисках или дисках.

Двораковски В.: Техники отладки микросхем и анализ сигнатур

INFORMATYKA 1986, № 7—8, с. 40

Характеристика проблем, связанных с отладкой микросхем с большой степенью интеграции. Описание разработанного фирмой Hewlett-Peackard метода анализа сигнатур.

Zabrodzki J.: State of art and development perspectives of LSI and VLSI circuits

INFORMATYKA 1986, No. 7—8, p. 1

Characteristics of the state of art and development perspectives of LSI and VLSI designing and manufacturing technology.

guage (1)
Pierzchała E.: Interactive environment of the Lisp Lan-
INFORMATYKA 1986, No. 7—8, p. 5

First part of Lisp characteristics, which include discussion of their features influencing its special ability for interactive environment building.

Owczarczyk J., Stolarski M., Woźniak E.: Architecture of concurrent image processing systems

INFORMATYKA 1986, No. 7—8, p. 8

Characteristic of solutions and world trends in concurrent image processing systems architecture.

Iszkowski W.: Architecture of n-microprocessor systems

INFORMATYKA 1986, No. 7—8, p. 12

Characteristics of architecture of multiprocessor systems, which are based on microprocessors. Digital systems classification and different multiprocessor system solutions are discussed.

Budzianowski L., Kwiatkowski J., Wietrzych J.: ODRA 1305 computer in MSK Interuniversity Computer Network

INFORMATYKA 1986, No. 7—8, p. 16

Method of ODRA 1305 computer and terminals connection with MSK network. Basic solutions and operating experience are presented.

Królikowski Z.: Generation of transaction plans in distributed data base systems (2)

INFORMATYKA 1986, No. 7—8, p. 20

Second part of characteristics of the optimalization method for transaction plans generation in distributed data base systems. Classification and different plans generation algorithms are presented.

Stokłosa J.: A short history of control

INFORMATYKA 1986, No. 7—8, p. 24

Short overview of main stages in development of control from ancient ages to contemporary times.

Zakharov V.: Personal computers and personal workstations

INFORMATYKA 1986, No. 7—8, p. 26

Comparison of personal computers and personal workstations features. Evolution of personal computers on example of IBM hardware, characteristics of actual personal workstation solutions and forecast of their development in next years are presented.

Dworakowski W.: IBM personal computers

INFORMATYKA 1986, No. 7—8, p. 30

Characteristics of IBM PC models 1 and XT. Processor and numeric coprocessor, RAM and ROM storages, DMA controller, timer and parallel I/O circuits are discussed.

Grabowicz R., Zalewski J.: PC-DOS operating system (1)

INFORMATYKA 1986, No. 7—8, p. 32

First part of characterises of the PC-DOS operating system for IBM PC family. System's genesis and file system organization are discussed.

Kleiber M., Saran M.: Experience in Fortran use on IBM PC

INFORMATYKA 1986, No. 7—8, p. 34

Feasibility characteristics and evoluation of Microsoft's MS-FORTRAN 3.20 compiler in scientific-technical applications. The evaluation is based on authors experience in application software building on different IBM PC and IBM PC/XT compatible microcomputers.

Markowski M. A.: Logical disc organization in the PC-DOS operating system

INFORMATYKA 1986, No. 7—8, p. 38

Detailed characteristics of logical disc organization in the PC-DOS operating system. Organization and working principles of directory and file allocation table with recovery possibility of cancelled or disturbed files on flexible or hard disc, are discussed.

Dworakowski W.: Circuit testing technics and signature analysis

INFORMATYKA 1986, No. 7—8, p. 40

Characteristics of problems connected with testing of large scale integration circuits. Signature analysis, elaborated by Hewlett-Packard, is discussed.

Zabrodzki J.: Jetziger Stand und Entwicklungsvoraussichten der LSI und VLSI Schaltungen

INFORMATYKA 1986, Nr. 7—8, S. 1

Eine Charakteristik von jetzigen Stand und Entwicklungsvoraussichten in der Projektierung und Herstellungstechnologie der LSI und VLSI Schaltungen.

Pierzchała E.: Interaktive Umwelt der Lisp-Sprache (1)

INFORMATYKA 1986, Nr. 7—8, S. 5

Erster Teil einer Charakteristik der Lisp-Sprache mit Besprechung dieser Spracheeigenschaften, die auf eine besondere Eignung für das Bau einer interaktiven Umwelt beeinflüssen.

Owczarczyk J., Stolarski M., Woźniak E.: Architektur der simultanen Bildverarbeitungssysteme

INFORMATYKA 1986, Nr. 7—8, S. 8

Eine Charakteristik von Lösungen und Welttrends in der Architektur der simultanen Bildverarbeitungssysteme.

Iszkowski W.: Architekturen der n-Mikroprozessorsysteme

INFORMATYKA 1986, Nr. 7—8, S. 12

Eine Charakteristik der Architektur von den auf Mikroprozessorbasia gebauten Multiprozessorsystemen. Es wurden Klassifikation der Digitalsysteme und verschiedene Lösungen von Multiprozessorsystemen besprochen.

Budzianowski L., Kwiatkowski J., Wietrzych J.: ODRA 1305-Computer im MSK-Hochschulrechnernetz

INFORMATYKA 1986, Nr. 7—8, S. 16

Eine Besprechung des Anschlusses von ODRA 1305-Computer mit Terminalen zum MSK-Hochschulrechnernetz. Es wurden Grundlösungen und gesammelten Betriebserfahrungen angegeben.

Królikowski Z.: Generierung der Transaktionausführungspläne in verteilten Datenbanksystemen (2)

INFORMATYKA 1986, Nr. 7—8, S. 20

Zweiter Teil einer Charakteristik von Optimierungsmethode der Transaktionspläneherstellung in verteilten Datenbanksystemen. Es wurden Klassifizierung und verschiedene Algorithmen von Plangenerierung angegeben.

Stokłosa J.: Aus des Geschichte der Steuerung

INFORMATYKA 1986, Nr. 7—8, S. 24

Ein kurzer Übersicht von Hauptetappen in der Entwicklung der Steuerung von Altertum bis heutiger Zeit.

Zakharov V.: Personal Computer und Arbeitsplatzcomputer

INFORMATYKA 1986, Nr. 7—8, S. 26

Ein Eigenschaftenvergleich der Personal Computer und Arbeitsplatzcomputer. Es wurden Evolution der Personal Computer am Beispiel der IBM-Hardware, eine Charakteristik der heutigen Lösungen von Arbeitsplatzcomputer, sowie Prognose ihrer Weiterentwicklung im Laufe der nächsten Jahre, besprochen.

Dworakowski W.: IBM PC

INFORMATYKA 1986, Nr. 7—8, S. 30

Eine Charakteristik der Hardware von Modellen 1 und XT des IBM PC's. Es wurden Prozessor und numerischer Koprozessor, RAM- und ROM-Speicher, DMA-Steuerung, sowie Zeit- und Parallelschaltung, besprochen.

Grabowicz R., Zalewski J.: PC-DOS-Betriebssystem (1)

INFORMATYKA 1986, Nr. 7—8, S. 32

Erster Teil einer Charakteristik von PC-DOS-Betriebssystem für IBM PC-Mikrorechnerfamilie. Es wurde sein Ursprung, sowie Organisation des Dateisystems, besprochen.

Kleiber M., Saran M.: Erfahrungen mit Fortran auf IBM PC

INFORMATYKA 1986, Nr. 7—8, S. 34

Eine Charakteristik von Möglichkeiten und eine Beurteilung von Nutzbarkeit des Microsoft's M-S FORTRAN 3.20-Kompilators in wissenschaftlich-technischen Anwendungen. Die Beurteilung wurde auf Grund der von Autoren gesammelten Erfahrungen während Erarbeitung der Anwendungssoftware auf verschiedenen IBM PC und IBM PC/XT kompatiblen Mikrorechnern, formuliert.

Markowski M. A.: Logische Plattenorganisation im PC--DOS-Betriebssystem

INFORMATYKA 1986, Nr. 7—8, S. 38

Eine detaillierte Charakteristik der logischen Plattenorganisation im PC-DOS-Betriebssystem. Es wurden Organisation und Wirkungsgrundlage von Registern und Dateizuteilungstafel mit Berücksichtigung der Wiedergewinnung der gelöschten oder beschädigten Dateien auf Disketten oder Platten, besprochen.

Dworakowski W.: Schaltungentestungstechniken und die Signaturenanalyse

INFORMATYKA 1986, Nr. 7—8, S. 40

Eine Charakteristik von Problemen, die mit Testung von hochintegrierten Schaltunpen verbunden sind. Es wurde die von der Firma Hewlett-Packard erarbeitete Signaturenanalyse besprochen.

Nowoczesny dalekopis elektroniczny przystosowany
do pracy w kodzie TTA 2 lub ITA według CCITT.
Wykonanie RO, KSR, ASR, ESR.
Mozaikowy. Pisze literami łacińskimi i cyrylicą.
Odmienny kształt pisma
dla tekstów przyjmowanych i nadawanych.
Niski poziom hałasu. Komfortowa obsługa.
Dostarcza:
PZO KOVO,
Praha 7, ul. Jankovcova 2, CSRS

EO/600/K/86