5 1989

informatyka

Rozpraszanie obliczeń
Mikrokomputery Mera 600
Fortran 88

WYDAWNICTWO NOT SIGMA

Cena 300 zł

informatyka

5 1989

Nr 5
Miesięcznik Rok XXIV
Maj 1989

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

Materiałów nie zamówionych redakcja nie zwraca

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

RSW „PRASA-KSIĄŻKA-RUCH"
PRASOWE ZAKŁADY GRAFICZNE
ul. Dworcowa 13, 85-950 BYDGOSZCZ
Zam. 848/89.E-9
Obj. 4,0 ark. druk. Nakład 8950 egz.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 300 zł
Prenumerata roczna 3600 zł

WYDAWNICTWO
NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

W NAJBLIŻSZYCH NUMERACH:

- **Zdzisław Szczerbiński omawia najważniejsze zagadnienia architektury i oprogramowania superkomputerów Cray X-MP.**
- **Waldemar Koczkodaj przedstawia zalety stosowania uniwersalnych pakietów programowych na przykładzie pakietu arkusza kalkulacyjnego Lotus 1-2-3.**
- **Wacław Iszkowski kontynuuje cykl artykułów na temat przyspieszania obliczeń przez ich rozpraszanie.**
- **Jan Bielecki charakteryzuje grafikę w Turbo Pascalu 4.0.**
- **Maciej M. Sysło dzieli się refleksjami na temat edukacji informatycznej w Japonii.**
- **Jacek Skrzymowski opisując nową normę Fortranu omawia podprogramy, moduły i instrukcje.**

WARUNKI PRENUMERATY

Prenumeratorzy zbiorowi jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat wyłącznie na blankiecie „wpłata-zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia). Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: Państwowy Bank Kredytowy III O Warszawa nr 370015-7490-139-11.

Prenumerata ulgowa przysługuje włącznie osobom fizycznym członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po jednym egzemplarzu każdego czasopisma. Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumeratę ze zleceniem wysyłki za granicę zamawia się tak, jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach: do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny; do 28 lutego na II, III i IV kwartał oraz II półrocze; do 31 maj na III i IV kwartał oraz II półrocze; do 31 sierpnia na IV kwartał. Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21, wewn. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 26-80-16), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena prenumeraty

miesięczna		kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
300 zł	60 zł	900 zł	180 zł	1800 zł	360 zł	3600 zł	720 zł

WACŁAW ISZKOWSKI
Instytut Informatyki
Politechnika Warszawska

Pierwsza wersja artykułu została przedstawiona podczas Szkoły Jesiennej PTI „Współczesne kierunki rozwoju informatyki", Mrągowo, 5–9 grudnia 1988 r.

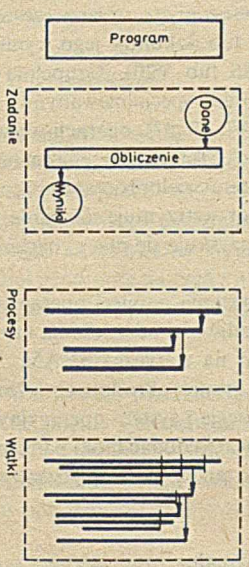
Przyspieszanie obliczeń przez ich rozpraszanie (1)

Rozpraszanie sprzętowe

Podstawowym celem informatyki było i jest budowanie coraz szybszych komputerów. Jeżeli staje się to niemożliwe lub zbyt kosztowne, to wtedy podejmuje się prace nad poszukiwaniem innych metod przyspieszania obliczeń. Rozwiązania te mogą z natury dotyczyć tylko oprogramowania, w którego projekcie jest możliwe lepsze lub gorsze wykorzystanie cech funkcjonalnych sprzętu oraz poszukanie nowego algorytmu dającego inną – mniejszą złożoność obliczeniową dla tego samego problemu. Zagadnienia te mają szczególne znaczenie w przypadku konieczności znacznego przyspieszenia wykonania pojedynczego programu, zamiast zwiększania przepustowości całego systemu cyfrowego.

Opierając się na doświadczeniach z życia codziennego poszukuje się metod podziału problemu obliczeniowego na mniejsze podproblemy, które następnie mogą być wykonywane współbieżnie. Możliwość rozproszenia obliczenia problemu musi uwzględniać cechy wykorzystywanego sprzętu (jego praktyczną równoległość działania) oraz właściwości oprogramowania systemowego. Równocześnie konieczne jest dostarczenie odpowiedniego narzędzia – języka i translatora – za pomocą którego zapis i podział problemu będzie stosunkowo łatwy. To proste z pozoru rozwiązanie w praktycznych zastosowaniach staje się bardzo złożone i co gorsze nie przynosi często oczekiwanych efektów – w myśl przysłowia *„Gdzie kucharek sześć tam nie ma co jeść"*.

Przedmiotem tego artykułu jest przegląd podstawowych rozwiązań sprzętowych oraz programowych, których celem jest szybsze, w porównaniu z typowym obliczeniem sekwencyjnym, obliczenie danego problemu. Przegląd rozpoczyna się od najtańszych, dostępnych nawet u nas w kraju, rozwiązań wykorzystujących standardowy sprzęt mikrokomputerowy oraz odpowiednie oprogramowanie. Do tej klasy zaliczam typowy system jednoprocesorowy, ewentualnie uzupełniony o koprocesory. Kontynuując poszukiwanie metod sprzętowego rozpraszania obliczeń zajmę się strukturami sprzętu działającego z szybkościami rzędu setek milionów operacji zmiennoprzecinkowych na sekundę, którego najpopularniejszym reprezentantem, znanym nam niestety ze słyszenia, jest superkomputer Cray.



Rys. 1. Program → obliczenie → zadanie → procesy → wątki

Druga część przeglądu jest poświęcona metodom rozpraszania obliczeń, opartym głównie na programowaniu równoległym. Oczywiście, dla rzeczywistego wykorzystania tego podziału obliczenia na współbieżnie wykonywane procesy, konieczne jest również istnienie odpowiedniego sprzętu *n*-procesorowego. W tej części zajmę się głównie ograniczeniami, jakie niesie to rozwiązanie, podając równocześnie przewidywane tendencje rozwojowe.

Tabela 1. Podstawowe pojęcia

Obliczeniem (ang. *computation*) jest wykonywanie kodu pewnego programu dla określonego zestawu danych w celu otrzymania wyników (rys. 1).
Zadaniem (ang. *task*) jest całość pracy potrzebnej do realizacji obliczenia.
Procesem (ang. *process*) jest wykonywanie pojedynczego zadania, przy czym do jego wykonania może być powołany jeden lub więcej procesów oraz użyty jeden lub więcej procesorów.
Wątkiem (ang. *thread*) jest wykonywanie sekwencji instrukcji w ramach określonego procesu, przy czym wątków takich w procesie może być określonych więcej niż jeden.
Dane współdzielone (ang. *shared data*) są danymi dostępnymi dla wielu procesów.
Zamek (ang. *lock*) jest operacją do synchronizacji współdziałania wątków na podstawie danych lokalnych procesu.
Semafor (ang. *semaphore*) jest obiektem zlokalizowanym w obszarze danych współdzielonych, będącym zmienną operacji synchronizacji współdziałania procesów.
Wiadomość (ang. *message*) jest daną operacji przesyłania informacji między procesami.
Równoległość (ang. *parallelism*) jest odnoszona do obiektów statycznych – algorytmów, programów, maszyn, zawierających elementy, które mogą być używane równocześnie.
Współbieżność (ang. *concurrency*), jest odnoszona do obiektów dynamicznych – obliczeń, procesów, wątków, wykonywanych równocześnie.

W podsumowaniu, porządkuję pojęcia związane z konstruowaniem programów obliczeń przewidzianych przez programującego do rozpraszania w sposób świadomy, uświadamiany lub nieświadomy. Artykuł jest napisany na poziomie podstawowym i łączy w sobie prezentację architektury sprzętu oraz oprogramowania. Podstawowe pojęcia przedstawiono w tabeli 1 i na rys. 1.

ŚRODOWISKO OBLICZENIA

Pojęcie przyspieszania oznacza tu poszukiwanie metody szybszego wykonania danego obliczenia po wyczerpaniu podstawowych metod optymalizacji czasowej programu oraz po dostrojeniu systemu cyfrowego – czyli sprzętu i oprogramowania systemowego.

Dla porządku wymienię z pewnością znane Czytelnikom od dawna (a może nie zawsze sobie uświadamiane) – metody optymalizacji złożoności czasowej programu.

1. Zacznę od stwierdzenia banalnego – z systemu usuwamy wszystkie inne programy, aby nawet ich przypadkowe uruchomienie nie wstrzymywało aktywowania naszego gotowego programu. Interesuje nas bowiem w danym wypadku przyspieszenie tylko jednego obliczenia, a nie zwiększenie przepustowości systemu dla wykonywania wsadu programów czy sesji interakcyjnych.
2. Do zapisu tekstu programu wybiera się taki język programowania i taki kompilator, aby przy maksymalnej wygodzie programisty możliwe było uzyskanie optymalnego czasowo kodu wynikowego. Zakłada się ewentualną konieczność napisania pewnych, wielokrotnie wykorzystywanych i szybkich procedur w języku asemblera danego komputera.
3. Wykorzystuje się maksymalnie pamięć operacyjną, modyfikując algorytm w myśl zasady odwrotnej proporcjonalności czasu wykonywania do wielkości obszaru danych wewnętrznych.

Teraz można dokonać wstępnej analizy możliwości systemu cyfrowego w uzyskaniu przyspieszenia obliczeń. Jedną z miar jest szybkość systemu określana liczbą rozkazów wykonywanych na sekundę, *MIPS* (ang. *Million Instructions Per Second*)

$$MIPS = \frac{liczba\_rozkazów}{liczba\_sekund} \cdot 10^{-6}$$

Miara ta może być stosowana jedynie jako orientacyjna, gdyż dla różnych architektur różne są moce poszczególnych rozkazów. Trudno jest na przykład porównywać szybkość komputera klasy RISC, o małej liczbie bardzo prostych rozkazów, z komputerem VAX mającym rozkazy operacji na kolejkach i grupowego przesyłania danych w bloku pamięci. Również w wypadku poszukiwania metod przyspieszania obliczeń, interesują nas głównie obliczenia na wartościach rzeczywistych, a te mogą być wykonywane przez koprocesory zmiennoprzecinkowe bądź realizowane innymi metodami. Tym bardziej, trudno jest dobrać jednolitą mieszankę listy rozkazów umożliwiającą porównanie różnych architektur.

Dlatego też dla obliczeń na wartościach rzeczywistych stosuje się miarę określającą liczbę operacji zmiennoprzecinkowych wykonywanych w jednostce czasu, *MFLOPS* (an. *Million Floating Point Operations Per Second*)

$$MFLOPS = \frac{liczba\_operacji\_zmiennoprzecinkowych}{liczba\_sekund} \cdot 10^{-6}$$

Również i z tą miarą są kłopoty. Wyraża ona bowiem moc obliczeniową danego komputera, ale podobnie jak maksymalna prędkość na liczniku samochodu, stwierdza jedynie, że jest możliwe uzyskanie takiej szybkości. Oczywiście, jest to możliwe na płaskiej, prostej drodze, przy braku wiatru, na dobrej benzynie, po wyregulowaniu silnika i z bardzo sprawnym kierowcą oraz przy braku innych użytkowników tej drogi. Podobne zastrzeżenia należy podać dla danego komputera. Pamiętając o tym można już przeanalizować samodzielnie tabelę 2, obrazującą szybkości różnych klas komputerów.

Tabela 2. Szybkości niektórych komputerów

Komputer	Częstotliwość lub cykl zegara	MIPS	MFLOPS
PC/XT	4,77 MHz	0,1	
PC/AT	8 MHz	0,5	
PS/2(80)	20 MHz	3,0	
Sun 3/50	15 MHz	1,5	
Sun 3/200	25 MHz	4,0	
T800 (transputer)	30 MHz	15,0	2,25
Cray-1	12 ns		160
Cray-2	5 ns		500

Poprzednią miarę można przedstawić w nieco innej postaci:

$$MIPS = \frac{liczba\_rozkazów}{liczba\_cykli} \cdot \frac{liczba\_cykli}{liczba\_sekund} \cdot 10^{-6}$$

Pierwszy iloczyn jest zależny od architektury sprzętu, a drugi od zastosowanej technologii. Przyspieszenie obliczeń można więc uzyskać przez zastosowanie innej, lepszej architektury lub odkrycie nowej technologii. Obecnie wykorzystywane są trzy różne technologie – MOS, ECL i GaAs. Najpopularniejszą i najtańszą jest technologia MOS, w której są realizowane wszystkie popularne układy sprzętu mikro- i minikomputerowego. Ograniczenia szybkości przełączania technologii MOS są mniejsze w technologii ECL, która jest wykorzystywana do realizowania układów dla superkomputerów. Znaczne zainteresowanie budzi technologia GaAs, ale jeszcze kilka lat upłynie zanim stanie się ona dostatecznie tania.

Po wyczerpaniu możliwości wykorzystania danej technologii, pozostaje tylko rozproszenie tego obliczenia, czyli podzielenie koniecznych do wykonania operacji między wiele współbieżnie działających jednostek wykonawczych. Rozpraszanie to może się odbywać na kilku poziomach struktury systemu cyfrowego:
– przez zakładkowanie lub równoczesne wykonywanie cykli kolejnych rozkazów,
– przez zwiększenie przepustowości jednostki przetwarzającej wskutek podziału jednostkowej operacji obliczenia na fazy wykonywane w trybie potokowym,
– przez zwielokrotnienie liczby jednakowych lub specjalizowanych jednostek przetwarzających,
– przez zwielokrotnienie liczby procesorów,
– przez zwielokrotnienie liczby komputerów.

Pierwsza grupa metod warunkujących rozpraszanie jest zależna od rozwiązań sprzętowych. Analizą tych rozwiązań praktycznych zajmę się w dalszej części artykułu. W drugiej grupie nacisk musi być położony na zapis rozpraszanego obliczenia w programie, przy stosunkowo prostym złożeniu n-procesorowego czy n-komputerowego sprzętu do jego współbieżnego wykonania. Zagadnienia te rozważono w drugiej części artykułu.

DOSTRAJANIE SYSTEMU CYFROWEGO

Zanim będziemy poszukiwać nowego sprzętu i oprogramowania umożliwiającego radykalne przyspieszenie obliczenia, spróbujmy lepiej wykorzystać te środki, które są obecnie dostępne. W tych rozważaniach jako przykład niech posłuży najpopularniejszy obecnie w kraju sprzęt, jakim są mikrokomputery PC i PS/2. Dodatkowym argumentem za zajęciem się tym sprzętem jest fakt, że według mojej oceny większość używanych w kraju konfiguracji mikrokomputerowych nie jest w pełni wykorzystana.

Dostrojenie systemu mikrokomputerowego (a według tych zasad, również i każdego innego) polega między innymi na wykonaniu następujących działań.

1. Przyspiesza się procesor przez zmianę częstotliwości zegara (np. z 4,77 MHz na 8 MHz) lub dokonuje jego wymiany na inny, szybszy (np. Intel 8088 na 8086 lub V20). Uzupełnia się też architekturę dodatkowymi koprocesorami specjalizowanymi – do obliczeń zmiennoprzecinkowych (np. 80287, 80387), operacji we-wy oraz grafiki (jeżeli program potrzebuje takich operacji). Trzeba zaznaczyć, że w ten sposób nie uzyskuje się komputera wieloprocesorowego, gdyż wykonywanie operacji przez koprocesor wstrzymuje działania procesora głównego; przyspieszenie obliczeń uzyskuje się przez znacznie szybsze wykonanie danej operacji.
2. Efektywność wykorzystania pamięci operacyjnej można polepszyć przez zmniejszenie częstotliwości odświeżania lub zmianę typu pamięci z dynamicznych DRAM na statyczne SRAM. Warto też zwiększyć pojemność pamięci do granic określonych przez daną architekturę układu zarządzania pamięcią i system operacyjny. Nawet dla systemu operacyjnego MS-DOS warto dodać bloki pamięci o adresach powyżej 1 MB, wykorzystując je przez moduły rozszerzające lub jako dysk wirtualny.

3. Celowe jest też maksymalne zmniejszenie czasu dostępu do informacji zapisanych w pamięci dyskowej. Warto więc dobrać kartę

sterownika oraz zbadać najoptymalniejszy sposób sformatowania dysku polegający głównie na dobraniu właściwej liczby przeplotów. Następnie, już z poziomu systemu operacyjnego należy dobrać odpowiednią dla danego typu obliczeń liczbę buforów oraz wielkość wewnętrznie pamiętanej tabeli adresów fizycznych dla ostatnio używanych bloków informacji (polecenie Fastopen w MS-DOS 3.3). Warto też zainstalować pamięć podręczną (ang. *cache*) oraz dyski wirtualne.

4. Uzupełnia się też sprzęt dodatkowym specjalizowanym sprzętem:
– kartami pamięci masowych RAM o krótkim czasie dostępu,
– procesorami sterowania wyświetlaniem w trybie tekstowym oraz graficznym i autonomicznymi operacjami przetwarzania graficznego,
– autonomicznymi kartami buforowania (np. 256 KB) i wyprowadzania wydruków na drukarkę,
– autonomicznymi procesorami z wbudowanymi programami, np. translatorem języka Ada,
– procesorami wektorowymi, macierzowymi czy nawet automatu komórkowego, jak np. CAM-6 o siatce 256 × 256 4-bitowych procesorów (za 1500 dolarów!).
Podane przykłady dotyczą komputerów IBM PC czy PS/2. Ulepszenia te nie są zbyt kosztowne, ale wymagają sporych umiejętności w dostrojeniu systemu do optymalnego funkcjonowania dla określonego typu obliczeń.

ROZPRASZANIE PROSTE

Po dostrojeniu systemu cyfrowego można zacząć poszukiwanie metod dalszego przyspieszenia obliczeń. Zacznę od rozwiązań najprostszych, możliwych do natychmiastowego zastosowania (pewnie znanych większości Czytelników). Analiza prawie każdego programu pod kątem możliwości przyspieszenia jego wykonania szybko prowadzi do wniosku, że największe straty powstają w momencie oczekiwania procedury na wprowadzenie kolejnych wartości oraz procedury wyprowadzania wyników. Oczywiście, zupełnie nas nie interesuje fakt, że nieco lepszy system operacyjny realizuje operacje we-wy asynchronicznie – wykorzystując wolny procesor na rzecz innego programu. Można jednak na tej podstawie podać najprostsze, wręcz banalne, metody przyspieszania obliczania:

1. Do wprowadzania danych i wyprowadzania wyników wykorzystuje się najszybsze urządzenie, czyli pamięć dyskową, a niekiedy dysk wirtualny zlokalizowany w pamięci operacyjnej. Zapełnienie plików danymi, czy przesyłanie ich na dysk wirtualny, wykonuje się innym programem. Podobnie realizuje się drukowanie wyników.

2. Program dzieli się na mniejsze fragmenty – programy realizujące kolejne wyodrębnione kroki algorytmu. Następnie wykorzystując systemowe mechanizmy przekazywania wartości między programami, powołuje się przetwarzanie potokowe:

progr_A || *progr_B* || ... || *progr_Z*

Zysk z przyspieszenia obliczeń notuje się dopiero wtedy, gdy to samo obliczenie wykonuje się dla kilku lub kilkuset zbiorów danych. Trzeba też pamiętać o dodatkowych opóźnieniach spowodowanych koniecznością przekazania do następnego programu wyników częściowych. Należy zwrócić uwagę, że cały czas wykorzystuje się tylko jeden procesor. Taki podział programu na części pozwala jedynie zrównoważyć obciążenie zasobów systemu cyfrowego przy obliczaniu tylko jednego problemu.

3. Poszukuje się możliwości przerzucenia części obliczeń na inny procesor. Wybrane fragmenty obliczeń muszą być łatwo wyróżnialne – możliwe do zsynchronizowania – logicznie lub nawet sprzętowo. Wprowadzenie bowiem dodatkowych mechanizmów synchronizacji na pewno zwiększy czas obliczeń (do tego problemu powrócę w następnej części artykułu).

Spróbuję teraz poszukać praktycznej odpowiedzi, na jakim sprzęcie i z jakim oprogramowaniem można zrealizować podane postulaty. Pkt. 1 można zrealizować na każdym komputerze osobistym wykorzystując system operacyjny MS-DOS. Przełączenie standardowego we-wy programu na współpracę z plikiem jest łatwe. Pliki te mogą być umieszczone na dysku wirtualnym – co pozwala między innymi wykorzystać pamięć ponad 1 MB (której inaczej system MS-DOS nie widzi). Można też wykorzystać pracujący w tle program drukowania PRINT. Istnieją też bardziej wyrafinowane programy przechwytywania wydruków do plików pomocniczych, wyprowadzanych następnie w miarę szybkości działania drukarki. Oczywiście, zarówno program PRINT, jak i te inne, obciążają procesor.

Do realizacji pkt. 2 konieczne jest tu wykorzystanie odpowiedniego systemu operacyjnego o cechach wieloprocesowości (lub wielozadaniowości – tutaj jest to synonim). Spośród dostępnych systemów najżyczliwiej są oceniane systemy klasy Unix-Xenix, QNX, Microport Unix V, IPIX itp. Niestety dla naszych celów trzeba je zdecydowanie odrzucić, gdyż Unix – zresztą bardzo elegancki, chociaż dość trudny w opanowaniu – zużywa sporo czasu procesora i obszaru pamięci operacyjnej na kontrolę środowiska oraz wykonywanych w trybie wieloprogramowym, wielożytkowym i wielodostępnym programów. Wada ta dotyczy nie tylko mikrokomputerów – również dla maszyny VAX 11/780 chęć szybkiego wykonania złożonego czasowo programu powoduje zastąpienie systemu Unix systemem VMS lub nawet poczciwym RSX-11M.

Drugim kandydatem jest jednożytkowy, wieloaplikacyjny, wielozadaniowy, wieloprocesowy i wielowątkowy system OS/2 firmy Microsoft Corp., opracowany specjalnie dla komputerów rodziny PS/2 (od modelu 50 wzwyż oraz PC/AT). System ten nie jest jeszcze rozpowszechniony, różne też krążą o nim opinie, ale moim zdaniem z czasem zostanie określony odpowiedni dla niego obszar zastosowań. W każdym razie, do naszych celów nadaje się, gdyż dostarcza odpowiednich mechanizmów współbieżnego wykonania procesów z wątkami oraz ich wzajemnej synchronizacji czy wymiany informacji w postaci potoku lub kolejek wiadomości. Pozytywną cechą tego systemu jest też możliwość wykorzystania całej pamięci operacyjnej (do 16 MB).

Zgodnie z postulatem pkt. 3 należy znaleźć inny procesor do wykonania przynajmniej niektórych obliczeń. Możliwe są tu następujące rozwiązania:

1. Wykorzystuje się sieć lokalną, zrealizowaną sprzętowo, przenosząc wprowadzanie danych i wyprowadzanie wyników, łącznie z ich drukowaniem, na inny komputer działający w sieci. W tym rozwiązaniu komunikacja z drugim komputerem odbywa się z wykorzystaniem drugiego procesora we-wy, przy odciążeniu procesora głównego. W ten sposób uzyskuje się praktycznie całą moc procesora tylko na same właściwe obliczenia. Inne zastosowania sieci lokalnych nie są tu istotne.

2. Wykorzystuje się zainstalowane w systemie koprocesory, przy czym kod wynikowy programu musi zostać dopasowany do nowej konfiguracji maszyny. Z reguły odbywa się to przez wymianę generowanego kodu po podaniu odpowiedniej opcji translatorowi języka. Programista ma tu niewiele do roboty.

Prezentując rozwiązania umożliwiające, przez dostrojenie systemu i rozproszenie obliczeń, uzyskanie ich przyspieszenia, nie chciałbym w żadnym wypadku pozostawić złudzeń, że możliwe jest pokonanie problemu mocy obliczeniowej opierając się nawet na PS/2 Model 80. Te proste metody mogą dać jedynie skutek chwilowy – często nie do pogardzenia – ale rozwiązania połowiczne niewiele dają. Konieczne jest więc zastosowanie szybszego komputera.

ULEPSZONE MASZYNY SISD

Po wyczerpaniu prostych metod przyspieszenia obliczeń musimy rozejrzeć się za innymi rozwiązaniami o rozbudowanej architekturze wykorzystującej naturalną równoległość działania pojedynczych układów czy całych bloków funkcjonalnych. Choć stosowano tu szereg różnych rozwiązań, koncepcja maksymalnego ulepszenia klasycznej

architektury von Neumanna, najpełniej została zrealizowana w komputerach serii Cray. Maszyny te obok systemów Cyber i CDC 205 są praktycznie jedynymi superkomputerami dostępnymi handlowo.

W projekcie architektury podobnych superkomputerów starano się pokonać wszelkie bariery ograniczające szybkość obliczeń. W tym celu [1, 2, 3, 4, 5, 7, 16]:
- zwielokrotniono dostęp do pamięci,
- zbuforowano z wcześniejszym rozpoznawaniem sekwencje rozkazów do wykonania,
- radykalnie zwiększono liczbę rejestrów skalarnych,
- dołączono bufory umożliwiające czasowe przechowywanie zawartości wektorów,
- zwielokrotniono liczbę specjalizowanych, równocześnie działających jednostek wykonawczych, realizujących operacje w trybie potokowym,
- wprowadzono łańcuchowanie operacji wektorowych polegające na rozpoczęciu obliczania następnej operacji natychmiast po uzyskaniu wyników z poprzedniej dla jednego elementu wektora (oczywiście, jeżeli ma to sens).

W rezultacie, dla tego typu maszyn uzyskano szybkości działania rzędu kilkuset MFLOPS (tab. 3).

Tabela 3. Przykładowe szybkości komputerów serii CRAY; 116M - 1 procesor i 16M słów 64-bitowych, 42G – 4 procesory i 2G słów 64-bitowych

Typ	Czas cyklu	MFLOPS
Cray-1	12,5 ns	160
Cray-X-MP 116M	9,5 ns	210
22M		420
48M		840
648M		13440 (w budowie)
Cray-2 12G	4,1 ns	500
42G	5 ns	2000
Cray-3	1 ns	1000

Warto zauważyć, że po dopracowaniu danego modelu maszyny, podwojenie tej mocy następowało przez dodanie drugiego procesora. Potem rozpoczynano opracowanie następnego modelu w innej nowszej technologii, już z założenia znacznie szybszego – uzyskując z jednego procesora moc większą niż uprzednio z dwóch. Niestety, w klasycznej architekturze komputerów, ze względu na technologię trudno będzie przekroczyć granicę czasu cyklu rozkazowego 1 ns.

Do podstawowych ograniczeń należy zaliczyć problemy związane z odprowadzaniem ciepła oraz czasem opóźnienia w przesyłaniu informacji. Proszę sobie wyobrazić konieczność rozwiązania problemu utrzymywania w temperaturze pokojowej przedmiotu o objętości herbatnika, stale podgrzewanego grzałką o mocy 1 KW. Z tego też względu, każda karta w superkomputerze Cray jest osobno chłodzona freonem, a sam komputer ma kształt walca, dla zmniejszenia długości połączeń wiążących poszczególne karty. Następne wersje, hiperkomputery, chyba powinny być kulami zawieszonymi pod sufitem.

OPROGRAMOWANIE SUPERKOMPUTERÓW

Najważniejszym językiem programowania superkomputerów jest Fortran 77, w niewielkim stopniu Pascal, C i Occam. Popularność Fortranu wynika z przyzwyczajeń kręgów naukowo-technicznych. Już w latach sześćdziesiątych projektowano i implementowano translatory tego języka, zawierające elementy wyszukiwania możliwości zrównoleglania obliczeń na podstawie zależności między zmiennymi wykorzystywanymi w kolejnych instrukcjach. Analiza ta była szczególnie efektywna dla instrukcji cyklu (pętli). Metody te, wykorzystujące między innymi grafy przepływu danych, były stale doskonalone. Wadą rozwiązań z tamtych lat jest przede wszystkim mała efektywność działania takiego translatora.

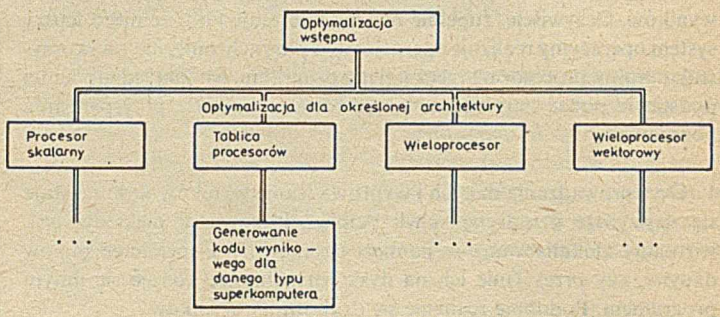
Okazuje się, że najefektywniejszą metodą zrównoleglania obliczeń jest przekształcanie instrukcji cyklu (również zagnieżdżonych). Oprócz klasycznej instrukcji DO języka Fortran:

```
      DO etyk I =...
      ...
etyk  CONTINUE
```

wyróżniono podobnie skonstruowane instrukcje zawierające dodatkowo informacje o możliwym sposobie wykonywania instrukcji w nich zawartych:
DOSPREAD – cykl rozprzestrzeniający kolejne iteracje obliczeń na inne istniejące i wolne procesory ze statycznym (w trakcie kompilacji) lub dynamicznym (w trakcie wykonywania programu) ich przydziałem do obliczeń.
DOALL – cykl nie zawierający żadnych zależności iteracyjnych, tzn. wszystkie kroki cyklu mogą być wykonane współbieżnie,
DOACROSS – cykl zawierający zależności, ale z możliwością jego wykonania w trybie wielopotokowym.

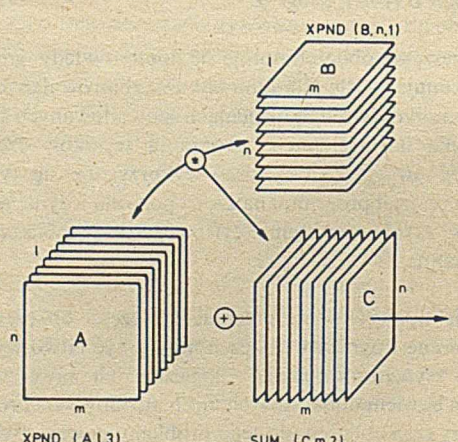
Obecnie krystalizują się dwie drogi postępowania. Pierwsza z nich polega na opracowywaniu półautomatycznych narzędzi wspomagających programistę w określaniu możliwości zrównoleglenia obliczeń. Narzędzia te – jak np. PTOOL [1] – konwersacyjnie analizują kolejne fragmenty programu wskazując zależności lub ich brak między zmiennymi występującymi w bloku instrukcji DO.

Druga metoda polega na implementacji preprocesora języka Fortran, którego zadaniem jest wykrywanie w programie możliwości zrównoleglenia obliczeń, a następnie zastąpienia znalezionego fragmentu kodu sekwencją wywołań podprogramów realizujących to obliczenie równolegle, łącznie z koniecznymi synchronizacjami. Preprocesory te, chociaż dotyczą tego samego języka, różnią się w zależności od rodzaju maszyny docelowej, na której ma działać tak zmodyfikowany program. Przykładem takiego zestawu preprocesorów są produkty firmy Kuck and Associates (rys. 2).



Rys. 2. Struktura translatora Parafrase firmy Kuck and Associates Inc.

Zaawansowane są już prace nad normalizacją rozszerzeń Fortranu 77, które mają być zawarte w Fortranie 8x (por. str. 18 tego numeru).



Rys. 3. Zasada mnożenia dwóch macierzy

dokończenie na s. 13

HERBERT A. SIMON
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, stan Pensylwania,
USA

Procesy przeszukiwania i wnioskowania w rozwiązywaniu problemów (3)

W poprzednich częściach artykułu faktycznie przedefiniowałem problem postawiony przez McCarthy'ego, a nie rozwiązałem go. Trudność, na którą zwraca uwagę McCarthy, jest poważna (natura tej trudności nie została uwidoczniona w rozważaniach poświęconych „wnioskowaniu zdroworozsądkowemu"). Problem polega na określeniu sposobu przejścia od nieformalnego do formalnego opisu zadania, niezależnie od tego, czy opis formalny będzie sporządzony w rachunku predykatów pierwszego rzędu, czy też za pomocą reprezentacji przestrzeni stanów.

TWORZENIE REPREZENTACJI

Problemy mogą być formułowane w języku naturalnym (np. problemy umieszczone na końcu rozdziału w podręczniku fizyki) lub mogą być określane przez bodźce przychodzące bezpośrednio ze świata rzeczywistego. McCarthy czyni pożyteczne rozróżnienie między rzeczywistymi problemami a zagadkami. W zagadkach występuje niejawnie gwarancja, że nie istnieją żadne inne możliwości działania oprócz tych, które znamy (nie ma żadnych mostów w górze rzeki). Pomimo tego w rozwiązywaniu zagadek niezbędna jest czasem wiedza empiryczna, jak na przykład prawa zachowania. Przy rozwiązywaniu rzeczywistych problemów nie mamy pewności, czy dysponujemy wszystkimi niezbędnymi informacjami. Z tego względu, zdobywając więcej informacji o świecie rzeczywistym (w trakcie rozwiązywania problemu lub w wyniku celowego namysłu przed przystąpieniem do jego rozwiązania), możemy doprowadzić do drastycznej zmiany sformułowania problemu, czyli faktycznie postawić nowy (inny) problem.

Niezależnie od tego, czy mamy do czynienia z zagadką, czy z rzeczywistym problemem, w pewnej chwili musimy wykorzystać dostępną informację w celu stworzenia formalnej reprezentacji problemu, którą następnie będziemy posługiwać się przeszukując przestrzeń rozwiązań lub wykonując wnioskowanie. Dopiero po utworzeniu takiej reprezentacji pojawia się zagadnienie logicznego wnioskowania – do tej chwili bowiem nie dysponujemy formalnymi obiektami, na których system logiki mógłby wykonać jakieś operacje.

W podejściu opartym na wnioskowaniu skonstruowanie reprezentacji wymaga „po prostu" określenia zbioru asercji mówiących o tym, co jest w świecie rzeczywistym prawdą, a co nie – zbioru na tyle kompletnego, aby mógł posłużyć jako aksjomatyczna baza maszyny wnioskującej. W podejściu opartym na przeszukiwaniu przestrzeni stanów należy określić obiekty i relacje definiujące przestrzeń stanów oraz operatory przejść między stanami. Jeśli system wnioskowania jest rozszerzony o makrooperatory (uzupełnienia proceduralne i semantyczne), to operatory te muszą być określone dokładnie w taki sam sposób.

Co można powiedzieć o konstruowaniu reprezentacji dla procesów przeszukiwania? Obecnie istnieje kilka programów sztucznej inteligencji, które rzucają nieco światła na to zagadnienie. Omówię dwa z nich – program UNDERSTAND zbudowany przez Hayesa i przeze mnie [1] oraz program ISAAC zbudowany przez Gordona Novaka [3]. Podstawową cechą tych programów – istotną z naszego punktu widzenia – jest to, że dokonują one transformacji problemu postawionego w języku naturalnym na formalną reprezentację przestrzeni stanów.

Program UNDERSTAND

Program UNDERSTAND jest przeznaczony do rozwiązywania problemów podobnych do zagadek. Inaczej mówiąc nie ma on wiedzy empirycznej z żadnej dziedziny świata rzeczywistego i zawiódłby, gdyby taka wiedza była potrzebna do poprawnego sformułowania problemu. Aczkolwiek program nie dysponuje wiedzą empiryczną, to jednak jest w stanie robić różnego rodzaju założenia empiryczne w trakcie generowania reprezentacji problemów. Operatory budowane przez program zawierają niejawne zasady zachowania, a sam program tworzy reprezentacje tylko takich obiektów, właściwości i akcji, które można w sensowny sposób wywnioskować ze sformułowania problemu.

Na podstawie opisu danego problemu, sporządzonego w języku naturalnym, program UNDERSTAND próbuje określić **obiekty** i klasy obiektów, **relacje** zachodzące między obiektami oraz **operatory**, które można zastosować w danym kontekście (biorąc przy tym pod uwagę warunki poprawności stosowania operatorów). Program usiłuje również zidentyfikować charakterystyczne cechy sytuacji początkowej i sytuacji docelowej procesu rozwiązywania. Następnie dokonuje wyboru reprezentacji obiektów i relacji między nimi, jak również buduje strukturę danych, opisującą sytuację początkową. Wreszcie konstruuje programy, które działają na reprezentację w sposób określony przez operatory przejścia (poprawnie z punktu widzenia sformułowania problemu). Jeśli to się powiedzie, to program UNDERSTAND generuje na koniec strukturę danych i zbiór operatorów, które można byłoby przekazać do jakiegoś systemu rozwiązującego problemy w sposób podobny do systemu GPS i które dostarczyłyby do tego systemu informacje niezbędne do podjęcia próby znalezienia rozwiązania[1)].

Na przykład po otrzymaniu zagadki o misjonarzach i ludożercach, opisanej w języku naturalnym, program UNDERSTAND zadziałałby mniej więcej w następujący sposób. W sformułowaniu problemu rozpoznałby cztery rodzaje obiektów: brzegi rzeki, misjonarzy, ludożerców i łodzie. Następnie ustaliłby liczby poszczególnych obiektów. Zauważyłby również, że pewne zdania w sformułowaniu problemu mogą posłużyć do określenia operatorów przejścia i warunków ich stosowalności.

Korzystając z tych informacji, program UNDERSTAND przekształciłby reprezentację języka naturalnego na reprezentację przestrzeni stanów z odpowiednim operatorem określającym poprawne przejścia. Reprezentacja sytuacji początkowej mogłaby składać się ze struktury danych dla każdego brzegu rzeki. Lista misjonarzy, ludożerców i łodzi mogłaby być wartością atrybutu tej struktury. Operator przejścia usuwałby podzbiór tych elementów z listy jednego brzegu rzeki i umieszczał je na liście drugiego brzegu, sprawdzając przy tym wszystkie warunki określające poprawność przejścia (tzn. sprawdzając obecność łodzi na brzegu, z którego są przewożeni misjonarze i ludożercy, oraz sprawdzając różnicę między liczbą misjonarzy a liczbą ludożerców po wykonaniu każdego przejścia)[2)].

Aby zbudować reprezentację problemu, program UNDERSTAND nie musi rozumieć, co to są misjonarze, ludożercy, łodzie i brzegi rzeki.

[1)] Zrealizowana wersja programu UNDERSTAND nie zawiera mechanizmów definiowania rozbieżności ani mechanizmów konstruowania macierzy zależności między rozbieżnościami i operatorami. Informacje te mogłyby jednak zostać wygenerowane przez system GPS, wyposażony w mechanizmy uczenia się, na podstawie danych dostarczonych przez program UNDERSTAND.

[2)] Powyższy opis działania programu UNDERSTAND należałoby podać warunkowo, gdyż zarówno dokładna reprezentacja, którą skonstruowałby program, jak i postać odpowiednich operatorów zależałyby od języka użytego do opisu problemu. Przykłady innych możliwości działania programu znajdują się w [1].

Spostrzeżenia natury syntaktycznej wystarczą na ogół do zidentyfikowania ich jako obiekty oraz do powiązania misjonarzy, ludożerców i łodzi z brzegiem rzeki. Program potrzebuje nieco więcej wiedzy do poprawnego skonstruowania operatora przejścia. Można by dostarczyć mu, na przykład, taką definicję zwrotu *„przewieźć łodzią"*: *„przewieźć łodzią X przez Y oznacza przemieścić X, razem z obiektami związanymi z X, z jednego brzegu Y, gdzie X się znajduje, na drugi brzeg Y"*. Informacja ta mogłaby być zawarta w opisie problemu bądź też program UNDERSTAND mógłby zostać wyposażony w tę informację wcześniej.

Nie możemy jednak powiedzieć, że program UNDERSTAND nie wie nic o świecie. Działa on na podstawie pewnej minimalnej wiedzy i posługuje się – gdzie tylko jest to możliwe – wskazaniami syntaktycznymi, tak aby zmniejszyć swoje uzależnienie od wiedzy o świecie rzeczywistym. Buduje reprezentację, która nie jest bardziej złożona niż wymaga tego opis problemu i nie tworzy elementów reprezentacji samorzutnie. Ze względu na sposób konstruowania reprezentacji i operatorów, program robi kilka założeń zachowawczych (np. że operator przejścia zachowuje obiekty).

Znaczna część wiedzy programu UNDERSTAND jest zawarta w zbiorze podstawowych operatorów, takich jak *„przejdź"*, *„zamień"* i *„skopiuj"*. Każdy z tych operatorów jest oparty na różnych założeniach zachowawczych. Inny rodzaj wiedzy jest potrzebny do skonstruowania testów poprawności. Do tego celu służą takie predykaty jak *„mniej"* i *„więcej"*, które można stosować do różnych zbiorów obiektu. Wiedza ta jest jednak do tego stopnia ograniczona, że program UNDERSTAND może rozwiązywać jedynie zagadki, które celowo abstrahują od informacji o świecie rzeczywistym. Jednak nawet w dziedzinie zagadek nie istnieją żadne gwarancje, że program UNDERSTAND utworzy poprawną reprezentację (poprawną w zrozumieniu osoby, która tę zagadkę sformułowała). Nie może być takich gwarancji, ponieważ poprawność jest raczej sprawą doświadczenia, a nie logiki.

Program ISAAC

Program Novaka ISAAC [3] demonstruje, w jaki sposób moglibyśmy przejść od dziedziny zagadek do dziedziny problemów uwzględniających sytuacje ze świata rzeczywistego, ograniczając się jednak przy tym do problemów sformułowanych w języku naturalnym.

Program ISAAC jest przeznaczony do rozwiązywania problemów, które można znaleźć w podręcznikach fizyki, a zwłaszcza w podręcznikach statyki. Ogólna strategia działań programu jest podobna do strategii programu UNDERSTAND – z jedną wszakże istotną różnicą. Program ISAAC w znacznie większym stopniu jest oparty na dostarczonej mu wcześniej wiedzy empirycznej z dziedziny rozwiązywanych problemów. W odpowiedniej pamięci program gromadzi schematy definiujące obiekty (zapamiętane jako struktury list właściwości). Opisują one różne rodzaje obiektów występujących w problemach statyki, takich jak dźwignie, masy i osie. Program wie również, że „człowieka" można traktować (w problemach statyki!) jako masę lub jako oś.

Gdy program ISAAC rozpozna w sformułowaniu problemu jakieś określenie (np. dźwignia), to odwzorowuje nazwę tego określenia na odpowiedni schemat obiektu w pamięci. Posługując się informacją zawartą w sformułowaniu problemu, program generuje odpowiednią kopię schematu. Można więc, na przykład, wygenerować reprezentację dźwigni o określonych wymiarach, punkcie zamocowania itd. Program ISAAC składa wygenerowane kopie schematów w reprezentację całego problemu. Ta ogólna reprezentacja (lub schemat problemu) może być następnie użyta przez pozostałe moduły programu ISAAC, które – uwzględniając prawa mechaniki – generują odpowiednie równania i rozwiązują je.

Jest rzeczą oczywistą, że jeśli program ISAAC miałby rozwiązywać problemy z wielu różnych dziedzin, to musiałby dysponować olbrzymią pamięcią wyposażoną w odpowiednie schematy obiektów. Ekspert dysponuje dziesiątkami a nawet setkami tysięcy schematów z określonej dziedziny. Przykładowo, słownik języka naturalnego absolwenta uczelni szacuje się na 50 000 słów. Naszym aktualnym celem nie jest jednak określenie wielkości wiedzy, którą powinien mieć system, aby mógł wykonywać wnioskowanie zdroworozsądkowe. Jesteśmy zainteresowani raczej sposobem, w jaki można by tę wiedzę zorganizować. Doświadczenie związane z programem ISAAC dowodzi, że opis wiedzy za pomocą zdań logiki (w zwykle przyjętym sensie i w ramach systemu standardowej logiki) nie jest jedynym i wcale nie musi być najlepszym sposobem reprezentacji.

PROCESY WNIOSKOWANIA ROBOTÓW

Programy UNDERSTAND i ISAAC rozwiązują problemy sformułowane w języku naturalnym, nie rozwiązują natomiast takich problemów ze świata rzeczywistego, w których nowe informacje napływają w trakcie znajdowania rozwiązania. Aby lepiej zrozumieć ten nowy rodzaj problemów, musimy przeanalizować inne zarzuty wysunięte przez mędrka z cytatu McCarthy'ego. Powróćmy więc do przerwanego opowiadania McCarthy'ego [2, s. 30]:

„Widzicie już, że nasz mędrek jest wyjątkowo pomysłowy. Straciwszy więc nadzieję, że zaakceptuje problem sformułowany w duchu typowym dla rozwiązywania zagadek, sami podacie mu w końcu rozwiązanie. Ku waszemu utrapieniu mędrek zaatakuje teraz rozwiązanie na tej podstawie, że łódź mogła przeciekać lub że nie była wyposażona w wiosła..."

Tym razem z nowymi zarzutami mędrka nie jest już tak łatwo rozprawić się jak z poprzednimi. Teraz nie idzie już tylko o to, że nie możemy znaleźć najlepszego rozwiązania, bo nie wiemy nic o istnieniu mostu. Wskutek naszej niewiedzy rozwiązanie może wcale nie być rozwiązaniem, ponieważ być może nie uwzględniliśmy w naszej reprezentacji problemu wszystkich warunków związanych z operatorem. Nieznajomość warunków koniecznych do znalezienia rozwiązania jest znacznie poważniejszym zarzutem niż nieznajomość dodatkowych ścieżek możliwych rozwiązań.

Właściwa odpowiedź na ten zarzut brzmi: *„po prostu jak w świecie rzeczywistym"*. Prawdopodobnie każdego dnia ktoś wpędza się w sytuację bez wyjścia, ktoś tworzy, a następnie próbuje zrealizować jakiś plan, który jest niewykonalny. W żaden sposób „logika" lub wnioskowanie nie mogą nam zagwarantować, że zbiór naszych przesłanek jest zupełny, lub że niektóre z nich nie są empirycznie fałszywe.

Formalne reprezentacje przestrzeni stanów, którymi posługujemy się w procesach przeszukiwania i wnioskowania, są w najlepszym wypadku uproszczonymi i przybliżonymi modelami świata rzeczywistego. Wykonujemy operacje na tych reprezentacjach, jak gdyby one właśnie były światem rzeczywistym; czasem braki i niedokładności reprezentacji są na tyle nieistotne, że możemy tworzyć plany, które dadzą się sensownie zrealizować. Gdybyśmy zechcieli określić regułę, według której budujemy i stosujemy takie modele, to brzmiałaby ona tak: *„Elementy, które pominęliśmy w modelu, nie mają istotnego znaczenia, a elementy, które uwzględniliśmy są reprezentowane z taką dokładnością, jaka jest niezbędna"*. Przekonanie się, że model ma te właściwości dla poszczególnych zadań decyzyjnych jest trudnym przedsięwzięciem, lecz jest to przede wszystkim przedsięwzięcie empiryczne, a nie ćwiczenie w logice.

Spostrzeżenie, że modele reprezentują świat rzeczywisty w sposób bardzo niedoskonały i niepełny, nie podważa oczywiście ich przydatności. W znacznym stopniu sytuację łagodzi fakt, że plany są zazwyczaj realizowane z uwzględnieniem cyklicznego sprzężenia zwrotnego ze światem rzeczywistym. Gdy w trakcie wykonywania planu okaże się, że faktyczne wyniki nie pokrywają się z oczekiwanymi, to model sytuacji i rozwiązanie mogą być skorygowane, tak aby zapewnić zgodność z rzeczywistością. Niedostatki modelu planowania nie zawsze trzeba zwalczać przez zwiększanie jego realistyczności, a tym samym jego złożoności. W wielu sytuacjach znacznie korzystniejsze może być rozwiązywanie problemów na podstawie istotnie uproszczonego modelu i posługiwanie się sprzężeniem zwrotnym, zapewniającym aproksymację drugiego rzędu.

Zależność między reprezentacją przestrzeni stanów a światem zewnętrznym nie różni się specjalnie od zależności między abstrakcyjną przestrzenią planowania a oryginalną przestrzenią problemu, którą poddajemy procesowi abstrahowania. Rozwiązania problemu są generowane w bardziej abstrakcyjnej przestrzeni. Następnie należy sprawdzić, czy te rozwiązania spełniają dodatkowe ograniczenia świata zewnętrznego lub przestrzeni rozwiązań. We wczesnym programie Gelerntera i Rochestera, dowodzącym twierdzeń z zakresu geometrii, semantyczna przestrzeń diagramów spełniała tę samą funkcję, umożli-

wiając szybkie i tanie (obliczeniowo) stawianie hipotez, które następnie poddawano testowi poprawności.

Robotykę odróżnia więc od innych dziedzin sztucznej inteligencji (jeśli pominąć analogię dotyczącą planowania) przede wszystkim to, że robot jest umieszczony w pewnym rzeczywistym środowisku, w którym może się orientować i na które może oddziaływać. Ten fakt ma dwie konsekwencje. Po pierwsze, niezbędne jest posługiwanie się miarodajną oceną skuteczności działania robota: skuteczność robota ocenia się na podstawie jego zachowania w rzeczywistym środowisku, a nie na podstawie zachowania w wyimaginowanej przestrzeni problemów. Po drugie, robot może dostosować złożoność i inne właściwości swej wewnętrznej przestrzeni problemów do własnych możliwości obliczeniowych, wykorzystując sprzężenie zwrotne w celu eliminowania rozbieżności między oczekiwaniami a rzeczywistością [4, rozdz. 5.3].

Z tego właśnie powodu robotyka wydaje się być najbardziej obiecującą dziedziną sztucznej inteligencji, w której można badać procesy rozwiązywania typowych problemów życia codziennego. Specyficzną cechą w dziedzinie robotyki jest to, że badacz ani na chwilę nie może zapomnieć, że reprezentacja to nie rzeczywistość i że tylko ciągłe dostosowanie reprezentacji do rzeczywistości może doprowadzić do wygenerowania akcji, które będą w miarę efektywne w świecie zewnętrznym. Zapewnienie dostatecznego poziomu dokładności i złożoności tworzonych modeli, jak również zapewnienie wystarczająco czułych mechanizmów sprzężenia zwrotnego w celu odpowiedniego dopasowania modeli, nie jest z pewnością sprawą logiki – jest natomiast sprawą dostępu do dobrych, empirycznie poprawnych teorii naukowych, opisujących działanie świata zewnętrznego. Najważniejszym elementem ograniczonej racjonalności człowieka (p. poprzednia część artykułu) jest wiedza empiryczna.

ZAKOŃCZENIE

Celem tego artykułu było przeanalizowanie zalet i wad traktowania rozwiązywania problemów jako procesu przeszukiwania przestrzeni stanów i jako procesu wnioskowania. John McCarthy podkreślił, że oceniając procedury rozwiązywania problemów powinniśmy brać pod uwagę nie tylko skuteczność rozwiązywania problemów z użyciem gotowych już reprezentacji, lecz także dokładność reprezentowania problemów, gdy zostały one sformułowane pierwotnie w języku naturalnym lub gdy nie są prostymi zadaniami laboratoryjnymi i dotyczą rzeczywistych sytuacji świata zewnętrznego. W artykule omówiłem rolę wnioskowania zarówno w rozwiązywaniu problemów, jak i w reprezentacji problemów.

Te rozważania unaoczniły, że – w swej wyjściowej postaci – informacje o problemie (uzyskane środkami werbalnymi lub w wyniku bezpośredniej konfrontacji ze światem zewnętrznym) są zazwyczaj bardzo niekompletne i muszą być uzupełnione informacjami dostarczonymi przez osobę rozwiązującą. Większość tych informacji nie wynika z zasad logiki, lecz stanowi empiryczne założenia o dziedzinie problemu. Niezależnie od postaci reprezentacji problemu, wiedza ta musi w jakiś sposób zostać udostępniona osobie rozwiązującej i nie może być dostarczona w sformułowaniu problemu. W systemach rozwiązujących problemy przez przeszukiwanie przestrzeni stanów (takich jak UNDERSTAND i ISAAC), wiedza ta jest przechowywana w schematach i wykorzystuje się ją w trakcie analizy informacji zawartych w sformułowaniu problemu.

Tworząc reprezentację problemu należy przezwyciężyć kilka podstawowych i niemal powszechnie występujących trudności. Mamy więc trudności z interpretacją i przetłumaczeniem (na opis wewnętrzny) opisu problemu sporządzonego w języku naturalnym lub uzyskanego bezpośrednio ze świata zewnętrznego. Mamy trudności z określeniem alternatywnych akcji, o których nie było wzmianki w sformułowaniu problemu. Mamy trudności z dostarczeniem dodatkowych informacji o obiektach występujących w reprezentacji problemu. Mamy wreszcie trudności z określeniem wszystkich warunków ubocznych, które muszą być spełnione, jeśli te akcje mają być realizowalne.

W literaturze pojawiły się liczne propozycje wprowadzenia niestandardowych logik, które mogłyby dostarczyć dodatkowych informacji wymaganych przez system rozwiązywania problemów. Wadą tych propozycji jest to, że prawie wszystkie niezbędne informacje są informacjami empirycznymi, zmieniającymi się od jednej sytuacji obliczeniowej do drugiej. Nie są to informacje o dopuszczalnych formach wnioskowania logicznego. Z tego powodu poszukiwanie standardowych, niezależnych dziedzinowo reguł, które mogłyby zastąpić specyficzne i zmienne założenia empiryczne, wydaje się zadaniem bezproduktywnym.

Alternatywne podejście polega na zgromadzeniu w pamięci programu schematów, które zawierają niezbędne informacje empiryczne i które mogą być uaktywniane przez odpowiednie wskazania zawarte w sformułowaniu problemu bądź przez odpowiednie sygnały odebrane ze świata zewnętrznego. Takie podejście zastosowano w systemach rozwiązywania problemów UNDERSTAND i ISAAC, opartych na reprezentacji przestrzeni stanów, oraz w schematach systemu SCRIPT.

W systemach tych brakuje jednak istotnego mechanizmu, który zapewniłby poprawne reakcje na sytuacje docierające ze świata rzeczywistego. Chodzi o mechanizm sprzężenia zwrotnego, który otrzymując informacje ze świata zewnętrznego, umożliwia ciągłe dopasowywanie procesów obliczeniowych do procesów zachodzących w rzeczywistości. Dopiero gdy zaczniemy budować systemy o takich możliwościach, będziemy mogli mówić o „zdrowym rozsądku". Być może nie jest to tylko etymologiczny przypadek, że drugi człon powyższego idiomu odnosi się wyraźnie do sprzężenia ze światem rzeczywistym[3)].

Podkreślmy na zakończenie, że wnioskowanie (gromadzenie informacji przez wnioskowanie) jest tylko jednym z kilku różnych procesów myślowych, przydatnych w rozwiązywaniu problemów, a logika formalna – tylko jednym z narzędzi wnioskowania. Moim zadaniem było przekonać Czytelnika o potrzebie stosowania w sztucznej inteligencji innych dostępnych sposobów przetwarzania informacji, jak również wykazać, że nie należy stawiać znaku równości między myśleniem a logiką formalną.

* * *

Jestem bardzo wdzięczny Johnowi McCarthy'emu i Nilsowi Nilssonowi za ich wnikliwe uwagi do wcześniejszych wersji tego artykułu. Fakt, że ich punkt widzenia na omawianą tematykę jest zasadniczo odmienny od mojego, okazał się podwójnie pożyteczny. Czytelnik nie będzie zaskoczony, gdy się dowie, że mój artykuł nie był na tyle przekonywający, by zmienić ich poglądy, i że z tego względu mam większe niż zwykle powody do tego, by nie obciążać ich odpowiedzialnością za jakiekolwiek błędy w ostatecznej wersji artykułu i podziękować im serdecznie za zmuszenie mnie do lepszego wyjaśnienia własnego punktu widzenia. Jestem również wdzięczny wszystkim uczestnikom Seminarium Nauk Poznawczych w Uniwersytecie Carnegie-Mellon za ich cenne komentarze do ustnej prezentacji tego artykułu.

LITERATURA

[1] Hayes J. R., Simon H. A.: Understanding written problem instructions. H. A. Simon (Ed.). Models of Thought. Yale University Press, New Haven (CT), 1979

[2] McCarthy J.: Circumscription – a form of non-monotonic reasoning. Artificial Intelligence. Vol. 13, pp. 27–40, 1980

[3] Novak G. S.: Representations of knowledge in a program for solving physics problems. Proc. Fifth International Joint Conf. on Artificial Intelligence I, pp. 286–291, 1977

[4] Simon H. A.: Models of Discovery. Reidel, Dordrecht, 1977.

[3)] W języku angielskim *sense* oznacza zarówno rozsądek, sens, jak i zmysł, zdolność czucia (przyp. tłum.).

Tłumaczył i opracował
MAREK MACHURA

Bardzo prosimy Autorów przysyłających teksty artykułów do opublikowania w INFORMATYCE o podawanie danych umożliwiających bezpośredni kontakt w godzinach pracy (numer telefonu) oraz adres domowy (do korespondencji i wysłania honorarium).

JAROSŁAW DEMINET
Instytut Informatyki
Uniwersytet Warszawski

Badanie przepustowości sieci lokalnych

Badania opisane w tym artykule przeprowadzono we współpracy z Zakładem Informatyki i Automatyzacji Badań Domu Handlowego Nauki, BISTYP-em oraz Ministerstwem Pracy i Polityki Socjalnej częściowo w ramach prac nad problemem badawczym *Założenia techniczne dla eksploatacji lokalnej sieci mikrokomputerowej z centralnymi zasobami dla potrzeb biur projektów.*

WARUNKI PROWADZENIA BADAŃ

Badaniem objęto trzy sieci mikrokomputerowe: Transnet, Ethernet i Arcnet, nadzorowane przez oprogramowanie Netware 286 firmy Novell. W celu określenia szybkości przesyłania informacji przez różne rodzaje sieci oraz jej zależności od komputera użytego jako składnica plików, napisano i uruchomiono program testujący. Program ten zakłada na wskazanym dysku duży plik, zawierający od kilkuset do kilku tysięcy rekordów po 4096 bajtów (4 KB), a następnie odczytuje losowo wybrane rekordy. Duży rozmiar rekordu pozwala zminimalizować wpływ języka programowania, w którym napisano program, oraz wpływ narzutu systemu operacyjnego na przyjęcie polecenia transmisji. Obciążenie generowane przez program testujący jest porównywalne z obciążeniem powstałym przy ładowaniu dużych programów z dysku do pamięci, a zwłaszcza z wymianą nakładek dużych programów (jak kompilatory, dBase itp.). Te właśnie operacje mają przeważający wpływ na szybkość pracy komputera (zwykle większy niż np. dostęp do danych dzielonych). Do testu wybrano operację odczytu, gdyż jest ona znacznie częstszą niż zapis, a poza tym pozwala uniknąć dodatkowych narzutów związanych z zapisem przez kilka programów w tym samym pliku (w normalnych zastosowaniach tego rodzaju operacje są rzadkie, a czas ich realizacji nie jest krytyczny).

Dla każdej konfiguracji badano średni czas jednego dostępu do dysku (w rzeczywistości jest to średni czas wykonania stu takich operacji podzielony przez sto), a także jego odwrotność, tzn. średnią liczbę takich dostępów do dysku.

Ze względu na sposób odczytywania informacji z dysku przez system Netware 286 (przechowywanie kopii ostatnio odczytanych bloków dyskowych w pamięci operacyjnej) badania przeprowadzono w dwóch wariantach. W obu korzystano z pliku zawierającego 2000 rekordów (ok. 8 MB), czyli więcej, niż pamięć buforowa komputera pracującego jako składnica plików (ok. 3 MB). W wariancie *A* program sięgał do wszystkich bloków i odczyt zwykle wymagał sięgnięcia do dysku. W wariancie *B* program sięgał tylko do pierwszych 500 rekordów (ok. 2 MB), które po kilkuset pierwszych dostępach zostały załadowane do bufora. Dalsza praca programu przebiegała bez sięgania do dysku. Taki tryb pracy odpowiada częstemu dostępowi do tych samych plików, np. przy ładowaniu nakładek, pliku COMMAND.COM lub programów często używanych przez wielu użytkowników.

WYNIKI BADAŃ SIECI TRANSNET I ETHERNET

Konfiguracja testowa sieci Transnet i Ethernet obejmowała pojedynczą składnicę plików na komputerze AT pod nadzorem niededykowanej wersji systemu Netware. Jako stacje robocze pracowały dwa komputery XT Turbo, jeden dołączony przez Transnet, a jeden przez Ethernet, i jeden komputer Super AT (AT 386) – także w sieci Ethernet. Przy dostępie lokalnym, w którym składnica pracowała równocześnie jako stacja robocza, w obu konfiguracjach otrzymano następujące wyniki:
- wariant *A* – 90 ms, 11 bl/s,
- wariant *B* – 20 ms, 50 bl/s.

Dla porównania zbadano czas analogicznego dostępu do dysku w systemie DOS (bez rozróżnienia wariantów):
XT Turbo – 190 ms, 5bl/s,
AT – 65 ms, 15 bl/s,
Super AT – 57 ms, 17 bl/s,
Super AT z dyskiem RAM – 25 ms, 40 bl/s.
Jak wynika z tego, programy pracujące w stacji roboczej będącej równocześnie składnicą plików mają pozornie utrudniony dostęp do dysku (90 ms i 65 ms). Różnica ta znika, gdy programy głównie czytają informacje ze stosunkowo małego zestawu plików. Czas dostępu do bloków buforowanych w pamięci jest zbliżony do czasu dostępu do dysku wirtualnego w pamięci RAM.

Następne badania dotyczyły czasu dostępu programów pracujących w zdalnych stacjach roboczych XT Turbo. Dla stacji pracującej w sieci Transnet wyniki były następujące:
- wariant *A* – 125 ms, 8 bl/s,
- wariant *B* – 100 ms, 19 bl/s,

a dla stacji pracującej w sieci Ethernet:
- wariant *A* – 85 ms, 12 bl/s,
- wariant *B* – 45 ms, 22 bl/s,

Wynika stąd, że czas dostępu przez sieć może być krótszy od czasu dostępu do lokalnego dysku. Należy jednak oczekiwać, że wyniki mogą się pogorszyć przy sięganiu do małych rekordów, ze względu na duży narzut związany z manipulowaniem wieloma małymi komunikatami i pakietami danych.

Przy korzystaniu z sieci Transnet zaobserwowano bardzo wysoki stopień wykorzystania czasu procesora składnicy plików (od 50 do 65%). W wypadku sieci Ethernet stopień wykorzystania wynosił poniżej 2% w wariancie *B* i ok. 25% w wariancie *A*. Oznacza to, że nadzór karty Transnet jest znacznie bardziej czasochłonny i że już przy dołączeniu 2-3 stacji roboczych pojawiłoby się tu wąskie gardło. Problem ten pogłębia się przy korzystaniu z krótkich komunikatów, gdy rośnie stosunek narzutu procesora do czasu transmisji danych.

Dla stacji roboczej Super AT dołączonej przez sieć Ethernet otrzymano następujące wyniki:
- wariant *A* – 65 ms, 15 bl/s,
- wariant *B* – 25 ms, 40 bl/s.

W obu wypadkach czas dostępu skrócił się o 20 ms, co odpowiada oczekiwaniu, że efekt użycia szybszej stacji roboczej nie powinien zależeć od sposobu obsługi żądania przez składnicę plików.
Można ocenić, że w całkowitym czasie dostępu do plików czas przetwarzania w stacji roboczej wynosi ok. 25 ms na komputerze XT Turbo, a 5 ms w stacji roboczej Super AT. Pozostały czas zajmuje transmisja danych przez sieć oraz operacje w składnicy plików.

Warto zwrócić uwagę, że różnica w czasie dostępu między obydwoma wariantami jest znacznie mniejsza w wypadku dostępu przez sieć niż w wypadku dostępu z lokalnej stacji roboczej. Świadczy to o szczególnie starannym zaprojektowaniu sposobu transmisji danych z bufora składnicy plików do stacji roboczej, pracującej na tym samym komputerze.

Do zbadania zależności czasu dostępu od sprzętu pracującego jako składnica plików przeprowadzono analogiczne badania konfiguracji składającej się z komputera Super AT jako składnicy i komputera AT jako stacji roboczej dołączonej przez Transnet. Otrzymano następujące wyniki:
- wariant *A* – 110 ms, 9 bl/s,
- wariant *B* – 75 ms, 13 bl/s.

Oznacza to, że dwukrotne zwiększenie mody obliczeniowej składnicy i kilkukrotne – stacji roboczej daje tylko ok. 10% wzrost wydajności w wypadku odwołania się do dysku i 30% w wypadku korzystania z bufora w pamięci operacyjnej. Wzrost powinien być jeszcze mniejszy przy korzystaniu z sieci Ethernet, mniej obciążającej procesor. Wynika z tego, że jako składnica całkowicie wystarcza komputer AT z odpowiednio rozszerzoną pamięcią.

Kolejne badania dotyczyły czasu dostępu w wypadku równoczesnej pracy kilku stacji roboczych, wykonujących ten sam program korzystający z tego samego pliku. W wypadku równoczesnej pracy lokalnej stacji roboczej (AT składnicy) i stacji roboczej XT Turbo dołączonej przez Ethernet wyniki były następujące:

- wariant *A*

stacja lokalna – 120 ms, 8 bl/s,
stacja zdalna XT – 155 ms, 6 bl/s,
łącznie – 14 bl/s;

- wariant *B*

stacja lokalna – 25 ms, 40 bl/s,
stacja zdalna XT – 80 ms, 12 bl/s,
łącznie – 52 bl/s.

Oznacza to znaczne opóźnienie stacji zdalnej (ok. 80%) i niewielkie opóźnienie stacji lokalnej (o ok. 25%). Aby osiągnąć stabilną i szybką pracę stacji zdalnych, należy zatem zrezygnować z pracy stacji lokalnej i zainstalować dedykowaną wersję oprogramowania składnicy. W dalszych badaniach stacja lokalna nie była wykorzystywana.

Przy równoczesnej pracy dwóch zdalnych stacji XT Turbo i Super AT dołączonych przez Ethernet otrzymano następujące wyniki:

- wariant *A*

XT – 115 ms, 9 bl/s,
Super AT – 75 ms, 13 bl/s,
łącznie – 22 bl/s;

- wariant *B*

XT – 55 ms, 18 bl/s,
Super AT – 25 ms, 40 bl/s,
łącznie – 58 bl/s.

Oznacza to nieznaczne spowolnienie obu stacji w wariancie *B* (w wypadku stacji Super AT niemierzalne) i znaczące spowolnienie w wariancie *A*. Jest to zrozumiałe, gdyż w tym wariancie wąskim gardłem jest dostęp do dysku. Liczba 20–30 testowanych dostępów stanowi zapewne maksymalną przepustowość dysku. W wariancie *B* wąskim gardłem jest sama sieć i oprogramowanie składnicy. Brak widocznego spowolnienia oznacza, że wąskie gardło nie zostało osiągnięte i że maksymalna liczba testowych dostępów jest istotnie większa niż 50.

Przy równoczesnej pracy obu stacji dołączonych przez Ethernet i dodatkowo stacji XT Turbo dołączonej przez Transnet otrzymano następujące wyniki:

- wariant *A*

XT Transnet – 185 ms, 5 bl/s,
XT Ethernet – 150 ms, 7 bl/s,
Super AT – 120 ms, 8 bl/s,
łącznie – 20 bl/s;

- wariant *B*

XT Transnet – 140 ms, 7 bl/s,
XT Ethernet – 95 ms, 10 bl/s,
Super AT – 55 ms, 18 bl/s,
łącznie – 35 bl/s.

Oznacza to bardzo znaczne spowolnienie wszystkich stacji roboczych, a nawet obniżenie całkowitej wydajności składnicy, zwłaszcza w wariancie *B*. Potwierdza to poprzedni wniosek, że obsługa karty Transnet zużywa szczególnie dużo czasu procesora w składnicy plików.

WYNIKI BADAŃ SIECI ARCNET

Analogiczne badania przeprowadzono dla sieci Arcnet. Sieć obejmowała 1 komputer AT pracujący jako składnica plików oraz 1 komputer AT i 3 komputery XT pracujące jako stacje robocze. Wyniki badań przedstawiono w tabelach 1–4.

Tabela 1. Sieć Arcnet, wariant B, składnica dedykowana (brak programu w składnicy plików)

Konfiguracja	Wynik dla AT		Wynik dla XT		Suma
	ms	bl/s	ms	bl/s	bl/s
1 AT	90	11,0			11,0
1 AT, 1 XT	110	9,0	120	8,0	17,0
1 AT, 2 XT	140	7,1	140	7,1	21,3
1 AT, 3 XT	160	6,3	180	5,6	22,4

Dla wariantu *B* ze składnicą dedykowaną (tab. 1) wyniki są zbliżone do wyników otrzymanych dla sieci Ethernet, co potwierdza przypuszczenie o osiągnięciu granicy przepustowości dysku na poziomie ok. 22 bl/s. Obciążenie procesora składnicy plików wahało się w granicach od 10% do 40%.

Tabela 2. Sieć Arcnet, wariant A, składnica dedykowana

Konfiguracja	Wynik dla AT		Wynik dla XT		Suma
	ms	bl/s	ms	bl/s	bl/s
1 AT	60	16,7			16,7
1 AT, 1 XT	80	12,5	80	12,5	25,0
1 AT, 2 XT	80	12,5	120	8,3	29,2
1 AT, 3 XT	120	8,3	160	6,3	27,0

Dla wariantu *A* ze składnicą dedykowaną (tab. 2) wyniki są znacznie gorsze niż dla sieci Ethernet, co oznacza mniejszą przepustowość właściwej sieci (dłuższy czas transmisji). Wydaje się, że osiągnięto maksymalną przepustowość rzędu 25–30 bl/s. Wydajność sieci odpowiada dość dokładnie wydajności dysku, a zatem konfiguracja jest mało wrażliwa na lokalność odwołań do dysku. Obciążenie procesora składnicy wahało się w granicach od 1% do 40%. Należy zwrócić uwagę na stosunkowo małe różnice w efektywności stacji roboczej na XT i AT.

Tabela 3. Sieć Arcnet, wariant B, składnica niededykowana (działająca równocześnie jako stacja robocza)

Konfiguracja	Składnica		Wynik dla AT		Wynik dla XT		Suma
	ms	bl/s	ms	bl/s	ms	bl/s	bl/s
Sama składnica	110	9,1					9,1
1 AT	140	7,1	150	6,7			13,8
1 XT	110	9,1			170	5,9	15,0
1 AT, 1 XT	150	6,7	180	5,6	200	5,0	17,2
1 AT, 2 XT	170	5,9	220	4,5	240	4,2	18,8
1 AT, 3 XT	200	5,0	240	4,2	290	3,4	19,5

Potraktowanie składnicy jako stacji roboczej (tab. 3) powoduje obniżenie o ok. 20% całkowitej przepustowości sieci, przy czym różnica w szybkości lokalnej i zdalnej stacji roboczej jest niewielka. W stosunku do wersji dedykowanej szybkość transmisji do stacji roboczych spada o ok. 40%.

Tabela 4. Sieć Arcnet, wariant A, składnica niededykowana (działająca równocześnie jako stacja robocza)

Konfiguracja	Składnica		Wynik dla AT		Wynik dla XT		Suma
	ms	bl/s	ms	bl/s	ms	bl/s	bl/s
Sama składnica	20	50,0					50,0
1 AT	50	20,0	70	14,3			34,3
1 AT, 1 XT	60	16,7	100	10,0	110	9,1	35,8
1 AT, 2 XT	80	12,5	110	9,1	130	7,7	37,0
1 AT, 3 XT	80	12,5	140	7,1	170	5,9	37,3

Przy użyciu składnicy jako stacji roboczej w wariancie *A* (tab. 4), spowolnienie działania stacji roboczych jest mniejsze niż w wariancie *B* (ok. 20%), a sumaryczna przepustowość sieci wzrasta. Wynika to z faktu, że lokalna stacja robocza omija wąskie gardło, którym w tym wariancie jest właściwa sieć (warstwa transmisji).

NIE DAJEMY RECEPT
SPRZEDAJEMY NARZĘDZIA

ELEKTRONIKA FILM KOMPUTER
Zakład Spółdzielni Pracy UNICUM

ul. Barska 3/20, 02-315 Warszawa
tel. 23-67-57, tlx 816955

OFERUJE USŁUGI W DZIEDZINIE KOMPUTERYZACJI PRZEDSIĘBIORSTW

Podejmuje się kompleksowej obsługi kontrahentów:

- sprzedaż sprzętu mikrokomputerowego (kompletacja, dostawa, serwis gwarancyjny i pogwarancyjny)
- opracowanie oprogramowania użytkowego (wdrożenie, szkolenie personelu)

Służymy Państwu:

- doradztwem organizacyjnym
- projektowaniem, oprogramowaniem oraz wdrażaniem systemów dedykowanych dla konkretnego użytkownika
- opracowaniem unikalnych programów wraz z nadzorem autorskim

Sprzedajemy profesjonalne narzędzia dla profesjonalistów

EO 955/88

KAZIMIERZ KACZMARCZYK
Warszawa

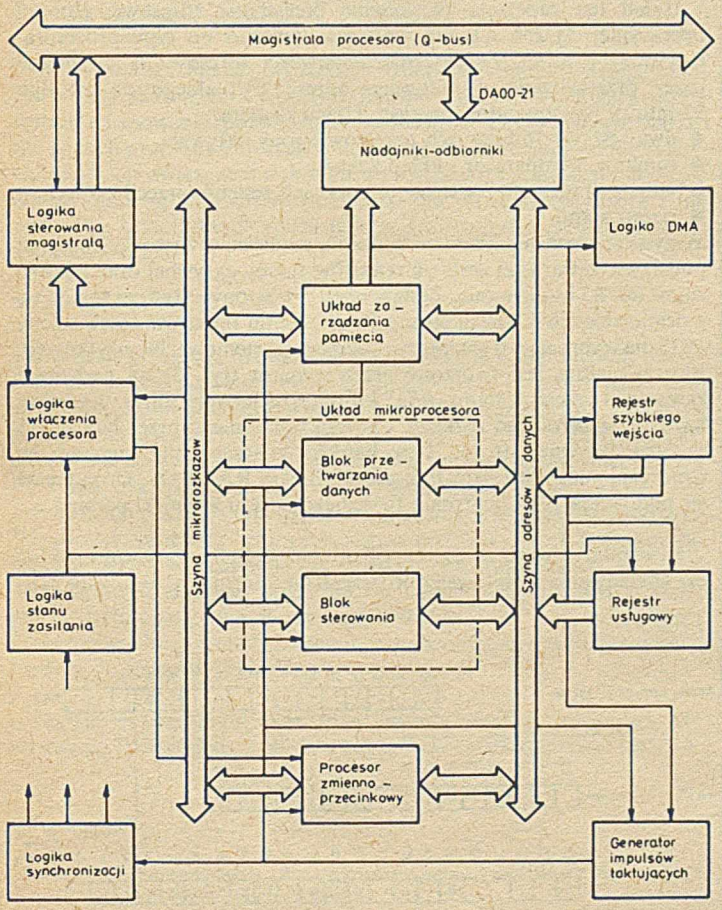
Charakterystyka modułu procesora mikrokomputerów rodziny Mera 600

W latach siedemdziesiątych przedsiębiorstwo MERASTER rozpoczęło produkcję mikrokomputerów rodziny Mera-60 opartych na procesorach oraz innych mikroukładach produkcji radzieckiej [1]. W pierwszym okresie do produkcji używano procesora M1 a następnie M2. Wzorcami tych procesorów były odpowiednio LSI-11 i LSI 11/2 firmy DEC. Kilka lat temu MERASTER rozpoczął produkcję rodziny mikrokomputerów Mera-600, opartych na procesorze M6, który jest odpowiednikiem procesora LSI-11/23 firmy DEC, mających architekturę zgodną z magistralą Q-bus [2]. Procesor M6, w porównaniu z M2, charakteryzuje się większą szybkością działania, możliwością rozbudowy pamięci operacyjnej do 4 MB i rozszerzoną listą rozkazów. Kilkakrotne zwiększenie szybkości działania uzyskano przez zwiększenie częstotliwości impulsów fazowych taktujących pracę procesora, zastosowanie procesora zmiennoprzecinkowego, mikroprogramowany cykl sterowania oraz inne rozwiązania układowe. Rozszerzona lista rozkazów zawiera dodatkowo 46 rozkazów przeznaczonych do wykonania operacji zmiennoprzecinkowych oraz rozkazy sterowania rozszerzoną pamięcią. Takie rozwiązanie pozwoliło na zapewnienie zgodności oprogramowania z minikomputerem SM-4 oraz SM-1420. Rozszerzenie adresowania pamięci operacyjnej do 256 KB lub 4096 KB w dwóch wersjach wykonania procesora osiągnięto przez wprowadzenie układu zarządzania pamięcią, przekształcającego 16-bitowe adresy wirtualne na 18- lub 22-bitowe adresy fizyczne.

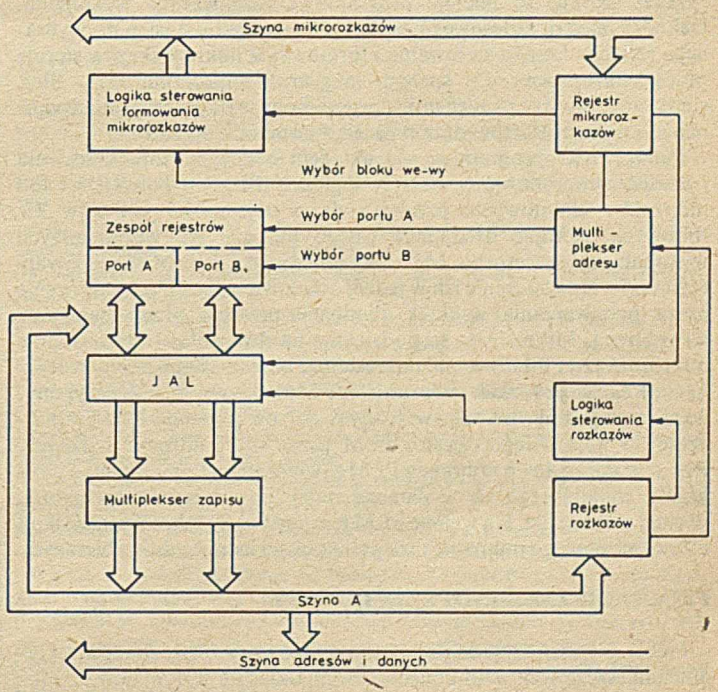
Procesor M6 jest produkowany w dwóch wariantach: MS 1601.01 oraz MS 1601.02. Procesor MS 1601.01 pracuje z szybkością ok. 500 tys. operacji na sekundę, ma możliwość rozbudowy pamięci do 256 KB, natomiast procesor MS 1601.02 charakteryzuje się szybkością do 600 tys. operacji na sekundę i możliwością rozbudowy pamięci operacyjnej do 4 MB. Obydwie wersje realizują listę 138 rozkazów (92 stałoprzecinkowych i 46 zmiennoprzecinkowych). Zbudowano je na identycznej płycie o wymiarach 252 × 143 × 12 mm. Schemat blokowy procesora M6 przedstawiono na rys. 1. Zbudowany jest on z trzech specjalizowanych układów o dużym stopniu scalenia:
- mikroprocesora,
- układu zarządzania pamięcią,
- procesora zmiennoprzecinkowego.

MIKROPROCESOR

Mikroprocesor jest zbudowany z dwóch specjalizowanych mikroukładów w jednej obudowie wykonanych w technologii *n*-MOS. Jeden z nich pełni funkcję bloku sterowania mikroprogramowego, drugi – bloku przetwarzania realizującego listę 92 rozkazów stałoprzecinkowych (rys. 2). Blok przetwarzania wykonuje wszystkie rozkazy arytmetyczne i logiczne, obsługuje wprowadzanie i wyprowadzanie 16-bitowych adresów i danych z magistrali mikroprocesora (oprócz obliczania adresu wykonywanego przez układ zarządzania pamięcią), a także wypracowuje większość sygnałów sterowania magistralą i wewnętrznymi cyklami pracy procesora. Blok ten zawiera następujące podzespoły funkcjonalne:
- jednostkę arytmetyczno-logiczną (JAL),
- zespół rejestrów,
- multiplekser zapisu,
- multiplekser adresu,
- logikę sterowania i formowania rozkazów,
- logikę rozgałęzienia wykonania rozkazów,
- rejestr mikrorozkazów i rozkazów systemowych.



Rys. 1. Struktura blokowa procesora M6



Rys. 2. Schemat bloku przetwarzania

Jednostka arytmetyczno-logiczna wykonuje wszystkie rozkazy arytmetyczne i logiczne stałoprzecinkowe. Typ i charakter przetwarzanej informacji określa 16-bitowy mikrorozkaz przychodzący z rejestru mikrorozkazów. Przetworzone dane z JAL są podawane na wejście multipleksera, który przekazuje je na dwukierunkową szynę wewnętrzną do portu *A* przechowującego rezultat operacji. Szyna dwukierunkowa umożliwia też przesłanie danych z magistrali mikroprocesora do zespołu rejestrów zawierającego porty: *A* i *B*. Port *B* jest przeznaczony tylko do odczytu danych, a port *A* do odczytu i zapisu. Zespół rejestrów zawiera także 9 rejestrów uniwersalnych i 5 roboczych rejestrów 16-bitowych, które służą do wykonywania szybkich operacji współpracy z pamięcią. Rejestry uniwersalne mogą być wykorzystane jako akumulatory, rejestry indeksowe, wskaźniki stosu, liczniki rozkazów, rejestry autoinkrementacji i autodekrementacji itp.

Procesor może pracować w jednym z dwóch trybów: nadzorczym (ang. *supervizor*) i użytkowym (ang. *user*). W celu zapewnienia szybkiego przełączania procesora z pracy w jednym trybie do pracy w drugim, zastosowano dwa rejestry R6. Wybór jednego z rejestrów R6 określają bity 14–15 rejestru stanu procesora. W trybie pracy użytkowej nie są wykonywane rozkazy *HALT*, *RESET*, i *SPL*. Znaczenie poszczególnych bitów rejestru stanu przedstawiono w tabeli 1, a zależność trybu pracy od wartości bitów 15–12 w tabeli 2.

Tabela 1. Rejestr stanu procesora

Bit	Znaczenie
15–14	Stan bieżący procesora
13–12	Stan poprzedni procesora
11–8	Nieużywane
7–5	Poziomy przerwań
4	T – zezwolenie obsługi przerwania
3	N – ujemny wynik operacji
2	Z – zerowy wynik operacji
1	V – przepełnienie arytmetyczne
0	C – przeniesienie

Tabela 2. Zależność trybu pracy procesora od wartości bitów 15–12

Tryb bieżący		Tryb poprzedni		Tryby pracy procesora	
bit 15	bit 14	bit 13	bit 12	bieżący	poprzedni
0	0	0	0	nadzorczy	nadzorczy
1	1	1	1	użytkowy	użytkowy
1	1	0	0	użytkowy	nadzorczy
0	0	1	1	nadzorczy	użytkowy
1	0	1	0	zakaz	zakaz

Rejestr rozkazów jest ładowany bieżącymi rozkazami, a jeżeli są to rozkazy skoku, to inicjuje programowe rozgałęzienie wykonania. Dekoder rejestru adresowego zapewnia bezpośredni dostęp do wybranego rejestru. Logika sterowania i formowania mikrorozkazów steruje pracą mikroprocesora i sprzęgu magistrali mikrokomputera. Blok sterowania zawiera pamięć mikroprogramów, logikę sterowania wykonaniem mikrorozkazów oraz poleceń z terminala operatora.

Pamięć mikroprogramów jest układem o dużym stopniu scalenia i zawiera programowaną matrycę logiczną (PLM) o pojemności 138 mikrosłów 25-bitowych, pamięć stałą o pojemności 414 słów 25-bitowych, sterując działaniem całego układu. We wcześniejszych wykonaniach procesorów M1 i M2 układy JAL i PLM występowały rozłącznie. Umieszczenie ich w jednej obudowie pozwoliło na skrócenie czasu przetwarzania wskutek eliminacji przesłań przez magistralę wewnętrzną. Mikrosłowa są podzielone na dwa pola: 16-bitowe pole mikrorozkazu i 9-bitowe pole określające adres następnego mikrorozkazu. Realizacja rozkazu w procesorze M6 odbywa się przez wykonywanie szeregu mikrorozkazów. Kody rozkazów systemowych są zapisywane do wejściowego rejestru PLM przez szynę adresów i danych. Pierwsze mikrosłowo pobrane z PLM zawiera adres startowy mikroprogramu znajdującego się w pamięci stałej lub adres mikroprogramu obsługi przerwania. Na wejście PLM jest podawany także 9-bitowy kod adresu następnego mikrorozkazu pobieranego jednocześnie z bieżącym.

PROCESOR ZMIENNOPRZECINKOWY

Procesor zmiennoprzecinkowy jest zbudowany z dwóch mikroukładów umieszczonych w 40-końcówkowym układzie scalonym w obudowie ceramicznej. Realizuje on następujące operacje:

– arytmetyczne i logiczne na liczbach zmiennoprzecinkowych,
– przekształcenia liczb całkowitych na liczby zmiennoprzecinkowe,
– rozkazy specjalne dla optymalizacji podprogramów matematycznych.

Ogółem wykonuje 46 rozkazów w identyczny sposób jak centralny procesor. Pobierane na bieżąco rozkazy są analizowane i jeżeli 4 bardziej znaczące bity wybranego rozkazu mają wartość 15 (ósemkowo – 17), to centralny procesor wykonuje mikroprogram obsługi procesora zmiennoprzecinkowego. Realizacja rozkazu przez procesor zmiennoprzecinkowy dokonuje się niemal równocześnie z kolejnym pobraniem rozkazu przez centralny procesor (wyjątek stanowią rozkazy ochrony i operacje kończące się przesłaniem danych do pamięci). Maksymalną korzyść czasową osiąga się dla operacji arytmetycznych, w których wykorzystuje się rejestry wewnętrzne do przechowania danych, a dane z rejestrów są pobierane przez centralny procesor. Szybkość wykonywania operacji przez procesor zmiennoprzecinkowy jest 5–10 razy większa niż przy realizacji programowej.

W czasie wystąpienia błędu procesor zmiennoprzecinkowy powiadamia centralny procesor przerwaniem wewnętrznym. Przyczynę błędu można zidentyfikować odczytując rejestr stanu procesora zmiennoprzecinkowego. Procesor ten zawiera 8 rejestrów, *A0–A7*. Rejestry *A0–A3* są rejestrami uniwersalnymi; przechowuje się w nich dane przy wszystkich obliczeniach i przesłaniach do pamięci. Rejestry *A4–A6* są przeznaczone do czasowego przechowywania danych pobranych z pamięci operacyjnej, następnie zapisywanych do rejestru wskazanego przez rozkaz. Rejestr *A7* pełni rolę rejestru stanu procesora zmiennoprzecinkowego. Każdy rejestr uważa się za 32-bitowy dla trybu pojedynczej precyzji i 64-bitowy dla podwójnej precyzji.

Osobnym blokiem procesora zmiennoprzecinkowego jest mikroprogramowalny układ sterowania. Steruje on pracą wszystkich układów logicznych przy wykonywaniu rozkazów. Składa się z pamięci mikroprogramów i logiki wypracowania sygnałów sterujących. Szybkość wykonywania operacji przez procesor zmiennoprzecinkowy jest 5–10 razy większa niż przy realizacji programowej.

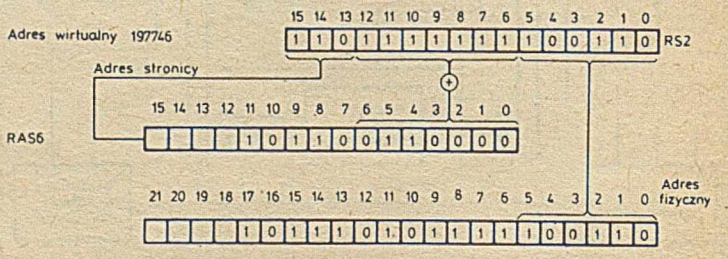
UKŁAD ZARZĄDZANIA PAMIĘCIĄ

Układ ten zapewnia zwiększenie przestrzeni adresowej pamięci operacyjnej do 256 KB lub 4096 KB (zależnie od typu procesora) i ochronę pamięci. Zwiększenie możliwości adresowania osiągnięto przez przetworzenie 16-bitowego adresu wirtualnego na 18- lub 22-bitowy fizyczny adres pamięci. Układ zawiera:

- dwa zestawy 16-bitowych rejestrów strony aktywnej,
- sumator, komparator i układ zakazu,
- sześć 64-bitowych rejestrów dla operacji zmiennoprzecinkowych,
- rejestr stanu.

Wszystkie operacje układu zarządzania pamięcią przeprowadza się na podstawie zawartości zestawu rejestrów strony aktywnej oraz rejestru stanu *R0–R3* i sterowania. Zestaw rejestrów strony aktywnej składa się z ośmiu rejestrów adresu strony RAS i ośmiu rejestrów opisu strony ROS dla wybranego trybu pracy procesora (systemowej lub użytkowej). Fizyczny adres jest tworzony przez sumator (rys. 3) na podstawie zawartości rejestru stanu *RS2*, który przechowuje adres wirtualny każdego wybieranego rozkazu, i adresu wskazanego przez bity 13–15 rejestru aktywnej strony. Trzy bardziej znaczące bity tego adresu oznaczają numer stronicy, tj. konkretnej pary RAS/ROS, którą bierze się pod uwagę przy tworzeniu fizycznego adresu komórki pamięci.

W wyniku złożenia adresu z rys. 3 otrzymuje się fizyczny adres 565746 dla adresowania 18-bitowego. Rozszerzenie adresu do 22 bitów (4 MB)



Rys. 3. Schemat tworzenia adresu fizycznego

jest uzależnione od stanu bitu 04 rejestru stanu *RS*3 układu zarządzania pamięcią. Dla jego wartości równej 1, logiczna przestrzeń adresowana wynosi 4 MB; dla wartości 0–256 KB.

POZOSTAŁE UKŁADY PROCESORA

Pozostałe układy centralnego procesora (rys. 1) realizują:
- łączność z pamięcią operacyjną i urządzeniami zewnętrznymi,
- sterowanie magistralą,
- generowanie specjalnych sygnałów sterujących i wspomagających.

Generator służy do generowania impulsów taktujących dla mikroprocesora i pozostałych elementów. Jego częstotliwość wynosi 13,3 MHz, a najkrótszy cykl pracy mikroprocesora jest równy 300 ns (nie obejmuje współpracy z magistralą). Rejestr szybkiego wejścia realizuje przesłanie do mikroprocesora informacji o stanie źródła zasilania i trybu startowego przy pierwszym lub powtórnym włączeniu mikrokomputera. Rejestr usługowy rejestruje i przekazuje do mikroprocesora dane o przyczynach przerwania i stanie stałego i zmiennego napięcia zasilania. Logika bezpośredniego dostępu do pamięci oraz sterowania magistralą realizuje wymianę danych między urządzeniami we-wy, procesorem i pamięcią operacyjną, a także formuje sygnały umożliwiające przesyłanie informacji o stanie do mikroprocesora.

Przekazywanie danych z mikroprocesora i innych układów procesora jest realizowane za pomocą dwóch wewnętrznych szyn dwukierunkowych z multipleksowaniem. Są to:
- szyna mikrorozkazów, po której w jednym cyklu przekazywane są mikrorozkazy układu zarządzania pamięcią i logiki sterowania, a w drugim blok przetworzonych danych z procesora zmiennoprzecinkowego
- szyna adresów i danych, przez którą (multipleksowaniem w czasie) przesyłane są dane, adres i informacja o stanie.

W mikrokomputerach opartych na mikroprocesorze M6 nie ma rejestru klawiszowego, za pomocą którego operator mógłby wykonywać rozkazy ręcznie, np. wybrać adres komórki, zapisać dane do pamięci itp. Wszystkie te funkcje wykonuje się przy użyciu terminala wysyłając do procesora odpowiednie polecenia. Terminalem może być monitor ekranowy z klawiaturą. Całe oprogramowanie pulpitu operatora jest zawarte w pamięci mikroprogramów, a jego wykonanie następuje z chwilą włączenia zasilania.

Procesor M6 realizuje trzy rodzaje rozkazów: bezadresowe, jednoadresowe i dwuadresowe. Typowy dla komputerów LSI-11, bogaty zestaw trybów adresowania, pozwala prowadzić bardzo efektywne przetwarzanie danych przechowywanych w dowolnej komórce pamięci lub rejestrze. Operacje związane z wykonaniem rozkazu o rejestrowanym trybie adresowania są wewnętrzne w odniesieniu do mikroprocesora i nie wymagają dodatkowych (poza pobraniem rozkazu) cykli transmisyjnych magistrali.

* * *

Na bazie procesora M6 zakłady MERASTER w Katowicach rozpoczęły produkcję systemów mikrokomputerowych o nazwie Mera 600, w różnych konfiguracjach w zależności od przeznaczenia. Typowym przykładem jest mikrokomputer Mera 385, złożony z następujących elementów podstawowych: procesora M6, pamięci operacyjnej 256 KB lub 4 MB, konsoli operatora, drukarki mozaikowej, pamięci na dyskach elastycznych 5,25 cala i na dyskach wymiennych 5-20 MB lub dyskach Winchester, systemu operacyjnego czasu rzeczywistego RT-60, kompilatorów języków programowania Basic, Fortran, Pascal, C. Jako wyposażenie dodatkowe dostarcza się: pamięć taśmową SM 5300.01, pamięć buforową 2-10 MB, interfejsy CAMAC i IEC 625, multiplekser 16-kanałowy V24, moduły transmisji synchronicznej i asynchronicznej, moduły do pracy w sieciach.

Ocenia się, że średni czas pracy bezawaryjnej dla procesora M6 wynosi do 1000 godzin, a pobór mocy jednostki centralnej ok. 1,3 kVA.

LITERATURA

[1] Grzywak A., Suchorończak Z.: Rozwój systemów mikrokomputerowych na przykładzie Mery-60 (SM-1633). Informatyka, nr 2, 1983.
[2] Grzywak A., Jakóbiec R., Tabacka D.: Mikrokomputer profesjonalny Mera 660. Biuletyn MERA, nr 7, 1986.

Rozpraszanie sprzętowe

dokończenie ze s. 4

Podsumowując ten temat warto zwrócić uwagę na dążenie do ujednolicenia postaci zapisu tekstu źródłowego programu i przeniesienie na preprocesory lub kompilatory problemu dopasowania kodu wynikowego do architektury docelowego komputera. W wielu jednak wypadkach konieczne jest uwzględnianie w projekcie algorytmu architektury komputera, na którym ma być wykonywany dany program. Co więcej, niektóre zasady z metodyki programowania sekwencyjnego powinny być pomijane, a inne wykorzystywane wręcz odwrotnie. Przykładowo, typowy algorytm mnożenia dwóch macierzy, dla wielu typów superkomputerów powinien być zrealizowany przez ekspandowanie każdej z macierzy na tablicę trójwymiarową. Dopiero wtedy można współbieżnie mnożyć wartości o identycznych indeksach z obu tablic, uzyskując trzecią, z której macierz wynikowa powstaje przez współbieżne zsumowanie wartości wzdłuż wierszy (rys. 3). Ten prosty przykład świadczy o zakresie trudności w opracowaniu algorytmów równoległych dla superkomputerów.

* * *

Jeszcze nie czas na podsumowanie tego tematu. Warto jednak już zauważyć, że na efekt przyspieszania obliczania ma wpływ wiele różnych czynników, na które z kolei niewielki wpływ ma użytkownik takiego programu. Widać też, że nawet dla komputerów osobistych konieczne jest powołanie nieco już zapomnianego stanowiska analityka systemu, którego zadaniem byłoby dostrojenie danego systemu do potrzeb użytkownika. Wart podkreślenia jest też fakt znacznej różnicy mocy superkomputera Cray w stosunku do dostępnego u nas sprzętu mikro- czy minikomputerowego. Co więcej, nie jest praktycznie możliwe nabycie doświadczeń w programowaniu równoległym dla ulepszonych maszyn typu SISD, przy użyciu jedynie prostych klasycznych mikrokomputerów. Niestety, jeżdżąc maluchem, nie nauczymy się prowadzić samochodu formuły I, chyba że będziemy nim później jeździć z szybkością 40 km/godz.

Pozostaje jeszcze jedno stwierdzenie. Zwielokrotnienie mocy superkomputera można uzyskać przez zwielokrotnienie liczby jednocześnie działających procesorów – jak na przykład 13 GFLOPS przy złożeniu 64 procesorów Cray X-MP. Czy wieloprocesor będzie rzeczywiście dobrze i efektywnie działał – odpowiedź odkładam do następnej części artykułu.

LITERATURA

Większość treści artykułu oparto na materiałach:
[1] Proc. 1986 Intern. Conf. on Parallel Processing. Penn State University, 19–22 August 1986
Pozostałe pozycje wybrano z licznej bibliografii dotyczącej tego tematu
[2] Babb II R.G. (Ed.): Programming Parallel Processors. Addison-Wesley, Reading (MA), 1988
[3] Bucher I.Y., Simmons M.L.: Performance Assessment of Supercomputers. Los Alamos National Laboratory, Report LA-UR-85-1505
[4] Fernbach S. (Ed.): Supercomputers Class VI Systems. Hardware and Software. North-Holland, Amsterdam, 1986
[5] Hockney R.W., Jesshope C.R.: Parallel Computers. Adam Hilger Ltd., Bristol, 1981
[6] Iszkowski W., Maniecki M.: Programowanie współbieżne, WNT, Warszawa, 1982
[7] Iszkowski W.: Superkomputer Cray. Informatyka, nr 5, 1985
[8] Iszkowski W.: System operacyjny OS/2. IV Szkoła Mikrokomputerowa PTI, Łódź, 1987
[9] Obermeier K.K.: Side by Side. Byte, Vol. 13, No. 12, November 1988
[10] Owczarczyk J., Stolarski M., Woźniak E.: Architektura współbieżnych systemów przetwarzania obrazów. Informatyka, nr 7–8, 1986
[11] Owczarczyk J., Stolarski M., Woźniak E.: Procesory tablicowe – współbieżne systemy przetwarzania obrazów. Informatyka, nr 2, 1987
[12] Pountain D.: Parallelizing Prolog. Byte, Vol. 13, No. 12, November 1988
[13] Rybnik J., Solak J.: Rodzina komputerów Sun-3. Informatyka, nr 3–4, 1989
[14] Randy A., Kennedy K.: Automatic Translation of Fortran Programs to Vector Form. ACM Trans. on Programming Language and Systems, Vol. 9, No. 4, October 1987
[15] Sysło M.M.: Maszyny i algorytmy równoległe. Informatyka, nr 10–12, 1988, nr 1, 1989
[16] Szczerbiński Z.: Nowa generacja superkomputerów Cray. Informatyka nr 6, 1989
[17] Shoja G.C., Gurr R.G.: Parallel Processing in Local Networks. Proc. 1986 International Conference on Parallel Processing.
[18] Wilson P.: Parallel Processing Comes to PCs. Byte, Vol. 13, No. 12, November 1988.

PIOTR FUGLEWICZ
Centrum Naukowo-Produkcyjne
Systemów Sterowania
Katowice

TADEUSZ KORNIAK
Przedsiębiorstwo Techniczno-Handlowe
Bister
Katowice

Oprogramowanie baz danych dla systemów mikrokomputerowych rodziny Mera 600

Przechowywanie przetwarzania złożonych danych dotyczących wszelkich dziedzin życia jest prawdopodobnie najważniejszą gałęzią praktycznej informatyki. Nie może więc dziwić fakt, że producenci sprzętu i oprogramowania przywiązują olbrzymią wagę do rozwoju środków programistycznych, poświęconych realizacji zadań z zakresu baz danych. Intensywny rozwój systemów baz danych w Centrum MERASTER jest dodatkowo spowodowany rosnącą konkurencją ze strony komputerów klasy PC. Jest to tym bardziej uzasadnione, że w zastosowaniu do baz danych wielostanowiskowa Mera 660 czy Mera 680 przewyższa dowolną sieć komputerów klasy IBM-PC.

Efektem pięcioletnich prac, prowadzonych zarówno w Centrum, jak i w innych jednostkach naukowo-badawczych, było opracowanie trzech systemów baz danych o rozłącznych właściwościach funkcjonalnych, ale z możliwością wymiany danych pomiędzy tymi systemami.

Artykuł prezentuje trzy różne systemy baz danych dla mikrokomputerów rodziny Mera 600; Podsystem Aktualizacji Kartotek, ReDS i mBase. Podsystem Aktualizacji Kartotek umożliwia wielodostęp do prostych baz danych dla nieprogramujących użytkowników; jest on używany głównie do gromadzenia i aktualizacji danych. System ReDS, oparty na relacyjnym modelu bazy danych, jest narzędziem do tworzenia i wykorzystywania baz danych o szczególnie złożonych powiązaniach. System relacyjnej bazy danych mBase został opracowany dla użytkowników mających jedynie podstawowe wiadomości o programowaniu. Umożliwia on wykorzystanie bazy danych w trybie interakcyjnym, a także programowym. Systemy te pokrywają zapotrzebowanie użytkowników mikrokomputerów Mera 600 na bazy danych – poczynając od zapotrzebowania na bardzo proste systemy gromadzenia danych, aż po zapotrzebowanie na złożone systemy pozwalające realizować złożone zapytania.

Podsystem Aktualizacji Kartotek

Podsystem ten, oznaczony w skrócie PAK, jest modułem programowym obsługi baz danych. Został on zaprojektowany i wykonany z myślą o małych konfiguracjach systemu Mera 600, wyposażonych w procesor typu M-2 i nie więcej niż 56 KB pamięci operacyjnej. PAK pracuje pod kontrolą systemu operacyjnego RT-60 i umożliwia pełną obsługę kartotek o indeksowo-sekwencyjnej organizacji danych. Operacje wprowadzania i aktualizacji danych mogą być wykonywane w trybie wielodostępnym.

Kartoteki obsługiwane przez PAK są dyskowymi plikami o organizacji indeksowo-sekwencyjnej. Podstawowe cechy charakteryzujące te kartoteki to:
- zmienna długość rekordu danych, nie przekraczająca 4096 bajtów,
- uporządkowanie rekordów kartoteki na podstawie pola indeksowego zawartego w rekordzie.
- umiejscowienie pola indeksowego w rekordzie danych oraz jego wielkość (nie przekraczająca 252 bajtów) definiowane przez użytkownika,
- hierarchiczna organizacja indeksów pliku kartoteki, zapewniająca krótki czas dostępu do danych,
- grupowanie rekordów danych w fizyczne porcje o ustalonej przez użytkownika wielkości (jedynym praktycznie ograniczeniem wielkości porcji jest to, aby była ona wielokrotnością 512 bajtów).

Programy składające się na Podsystem Aktualizacji Kartotek zbudowano przy wykorzystaniu biblioteki procedur MEDOS-2. Biblioteka zawiera procedury realizujące podstawowe operacje dotyczące rekordów danych zapamiętanych w plikach o organizacji indeksowo-sekwencyjnej. Procedury biblioteki pozwalają wykonywać następujące operacje:
- otwieranie i zamykanie poszczególnych plików danych,
- odczytywanie rekordów sekwencyjnie (według rosnących wartości pola indeksowego),
- zapisywanie do pliku zaktualizowanych rekordów,
- dopisywanie do pliku nowych rekordów,
- usuwanie z pliku rekordów.

Wszystkie błędy wykryte podczas operacji są przekazywane do programu, który wywołał operację, co pozwala programiście reagować na różnego rodzaju nienormalne sytuacje.

Podstawowym założeniem przy tworzeniu Podsystemu Aktualizacji Kartotek było umożliwienie użytkownikowi dostępu do danych w celu ich wprowadzania, modyfikacji, przeglądania i wyprowadzania, bez konieczności specjalizowanego szkolenia. Założenie to zostało spełnione dzięki zrealizowaniu interakcyjnego pełnoekranowego trybu współpracy użytkownika z systemem. Użytkownik steruje pracą systemu PAK przez wybór jednej spośród zestawu możliwości (menu) przedstawianych każdorazowo na ekranie monitora. Podczas wprowadzania, aktualizacji i przeglądania dane z wybranego rekordu oraz pewne informacje pomocnicze są prezentowane za pomocą odpowiedniej planszy. Informacje pomocnicze umieszczone na tejże planszy podają znaczenia poszczególnych pól rekordu danych, a także dopuszczalne w danym momencie akcje użytkownika i ich skutki. Stosowanie takiego pełnoekranowego, interakcyjnego trybu współpracy daje użytkownikom możliwość korzystania z Podsystemu Aktualizacji Kartotek już po kilkuminutowym wstępnym zapoznaniu się z nim.

Główny program systemu PAK pozwala na jednoczesną pracę czterech użytkowników. Każdy z nich po wybraniu pliku kartotek oraz po wybraniu formatu, w jakim przedstawiane będą dane z rekordu tego pliku, może wykonywać następujące operacje:

- Wprowadzanie danych

Odbywa się ono po wyświetleniu na ekranie monitora pustej planszy rekordu tj. planszy zawierającej tylko informacje pomocnicze. W trakcie wprowadzania następuje formalna kontrola zgodności wprowadzonych danych z wcześniej zdefiniowanym ograniczeniem. Po określeniu formalnie bezbłędnych wartości pól, rekord jest dopisywany do pliku w miejscu wynikającym z zawartości jego pola indeksowego.

- Modyfikacja

Pozwala na wprowadzanie zmian do zapisanego już w pliku rekordu z wykorzystaniem tej samej planszy co przy wprowadzaniu.

- Przeglądanie

Pozwala przeglądać kolejne rekordy danych w kierunku od pierwszego do ostatniego rekordu w pliku.

- Wyszukiwanie

Operacja ta wyszukuje i udostępnia dla innych operacji rekord o podanej przez użytkownika wartości pola indeksowego. Umożliwia więc sterowanie kolejnością przeglądania i modyfikacji rekordów.

- Wyprowadzanie

Pozwala na wyprowadzanie na urządzenie wyjściowe (zwykle drukarkę) wszystkich lub tylko wybranych rekordów danych. Dla operacji wyprowadzania użytkownik definiuje kryterium wyboru rekordów (w postaci wyrażenia), które musi być spełnione przez zawartość poszczególnych pól rekordu. Kryterium wyboru można zadać również w operacjach przeglądania i modyfikacji.

Inne operacje mają charakter pomocniczy i są realizowane przez niezależne programy, wymagające większej znajomości systemu. Są to takie operacje, jak:

- tworzenie pliku kartotek, połączone z definiowaniem parametrów jego struktury i sposobu indeksowania,
- reorganizacja pliku kartotek, realizująca fizyczne uporządkowanie rekordów danych zgodnie z istniejącym już uporządkowaniem logicznym, wyznaczonym przez pola indeksowe rekordów (reorganizacja pozwala również na zmianę wielkości pliku kartoteki),
- raport o stanie pliku kartotek, wyprowadza na monitor ekranowy zbiorcze informacje charakteryzujące plik, takie jak liczba zapisanych rekordów oraz informacje szczegółowe o strukturze indeksów,
- projektowanie formatu prezentacji rekordu danych, realizowane pełnoekranowo; pozwala zaprojektować nową lub zmienić istniejącą planszę dla rekordów pliku kartotek (przy projektowaniu planszy dla każdego z pól rekordu ustala się ich atrybuty określające zakres kontroli formalnej danych oraz sposób prezentacji na ekranie).

Opisany zestaw niezależnych programów pomocniczych, razem z głównym programem systemu PAK z powodzeniem realizuje zadania obsługi bazy danych, co potwierdzają dziesiątki zainstalowanych już systemów. Pozwala on, z użyciem komputera o niewielkiej konfiguracji i relatywnie małej pamięci operacyjnej, przygotować dane do późniejszego przetwarzania. Proces przygotowania danych jest tak wspomagany przez programowanie systemu, aby mógł być wykonywany przez przeciętnego użytkownika. Zestaw pomocniczych programów pozwala administratorowi systemu przygotować pracę pozostałych użytkowników.

Podsystem Aktualizacji Kartotek jest stosowany w licznych instytucjach od ok. pięciu lat. Dla wielodostępnej wersji systemu operacyjnego RT-60 – MRT-60 (odpowiednik systemu TSX PLUS) zrealizowano wariant o nazwie TSPAK. Pozwala on wykorzystywać opracowane w PAK-u systemy użytkowe również na komputerach Mera 600, dysponujących pamięcią od 256 KB wzwyż. TSPAK umożliwia jednoczesną pracę wielu użytkowników, którzy mogą wprowadzać lub aktualizować dane w tych samych lub w różnych kartotekach (maksymalnie pięć różnych kartotek). Liczba użytkowników wykorzystujących jednocześnie TSPAK jest ograniczona tylko liczbą dostępnych terminali.

System Bazy Danych ReDS

Podobnie jak PAK, system bazy danych ReDS (*Relational Data System*) został opracowany dla niewielkich konfiguracji systemu komputerowego Mera 600. System ReDS pracuje pod kontrolą systemu operacyjnego RT-60 i wymaga 56 KB pamięci operacyjnej. Implementacja jest oparta na podstawowych ideach klasycznego relacyjnego modelu bazy danych Codda oraz na pewnych nowych rozwiązaniach zaproponowanych w rozszerzonej wersji modelu relacyjnego RM/T.

System ReDS został zaprojektowany i wykonany na zlecenie CNPSS MERASTER przez zespół pod kierownictwem dr. Piotra Jasińskiego, jako możliwie proste w użytkowaniu, a jednocześnie kompletne narzędzie do tworzenia i wykorzystywania baz danych. Zdefiniowany przez użytkownika, konkretny system bazy danych wykorzystujący oprogramowanie ReDS składa się z szeregu tablic (relacji). Trzy tablice są tablicami systemowymi, a pozostałe zawierają dane użytkowe.

W dwuwymiarowej tablicy (relacja) stanowiącej podstawową formę strukturalizacji danych w systemie ReDS, wiersze reprezentują różniące się między sobą obiekty (rekordy danych), kolumny natomiast odpowiadają elementarnym (nierozdzielnym) atrybutom tych obiektów (atrybutom danych).

W celu umożliwienia definiowania struktury bazy danych z uwzględnieniem semantycznych, istotnych dla użytkownika, związków między poszczególnymi tablicami w systemie ReDS wyróżnia się trzy typy powiązań: charakterystyka, desygnacja, asocjacja.

Charakterystyka reprezentuje powiązanie typu 1:*N* między dwiema tablicami, tj. takie powiązanie, w którym jednemu wierszowi jednej z tych tablic (nadrzędnej, charakteryzowanej) odpowiada wiele wierszy drugiej tablicy (podrzędnej, charakteryzującej). Obie tablice, charakteryzowana i charakteryzująca, muszą mieć wspólny atrybut używany do stworzenia powiązania.

Desygnacja jest powiązaniem typu *N*:1 zachodzącym pomiędzy obiektami dwóch różnych typów (wierszami dwu różnych tablic). Tablica desygnująca inną tablicę musi zawierać atrybut desygnujący, tj. atrybut, którego wartości wyznaczają dokładnie jeden wiersz w tablicy desygnowanej. Tablica desygnowana jest więc swego rodzaju katalogiem.

Asocjacja jest powiązaniem typu *M:N* dwu różnych tablic. Powiązanie asocjacyjne jest zdefiniowane przez trzecią tablicę, która musi zawierać atrybuty wskazujące na wiersze w obu kojarzonych tym powiązaniem tablicach. Asocjację traktować można jako uogólnienie powiązania typu desygnacja.

Tablice systemowe definiują strukturę całej bazy danych: tablice danych, ich atrybuty i powiązania. Tablica o nazwie *TABLICE* określa wszystkie tablice użytkownika, a więc zawiera takie informacje, jak nazwy tablic, liczby atrybutów (kolumn) w poszczególnych tablicach, identyfikatory i sposób posortowania wierszy w tablicach, aktualne liczby wierszy itp. Tablica o nazwie *KOLUMNY* opisuje wszystkie atrybuty wszystkich tablic użytkownika. Zawiera ona informacje opisujące szczegółowo każdy atrybut, w tym – dopuszczalne wartości przyjmowane przez atrybut. Tablica o nazwie *ZWIĄZKI* opisuje wszystkie powiązania pomiędzy tablicami użytkownika. Dla każdego powiązania zdefiniowane są biorące w nim udział tablice oraz typ powiązania. Zawartość tablic systemowych powstaje w wyniku prowadzonego przez użytkownika procesu definiowania bazy danych. Wszelkie zmiany w opisie bazy danych, np. modyfikacje struktury tablic, są również na bieżąco odnotowane w tablicach systemowych. W dowolnym momencie użytkownik może zapoznać się z ich zawartością w sposób identyczny jak przy przetwarzaniu tablic zdefiniowanych przez siebie.

Język użytkownika systemu ReDS, *Database Inquiry and Administration Language* (DIAL), umożliwia wykonywanie następujących zadań:

- definiowanie i zakładanie bazy danych,
- aktualizowanie tablic bazy danych,
- modyfikację logicznego schematu bazy danych,
- wyszukiwanie i analizowanie informacji zawartych w bazie danych.

System ReDS wspomaga wykonywanie każdego z tych zadań. Przykładowo, przy definiowaniu nowej tablicy użytkownik ustala jej strukturę odpowiadając na pytania systemu. Do aktualizowania zawartości tablic bazy danych służy specjalny relacyjny edytor, pozwalający na dopisywanie nowych wierszy tablic.

Polecenia języka DIAL są poleceniami w języku zbliżonym do naturalnego. Dzięki temu, że system ReDS został napisany w języku wysokiego poziomu, polecenia te mogą być w prosty sposób zmienione na wyrażenia innego języka naturalnego. Zestaw poleceń realizujących wyszukiwanie i analizowanie informacji zawartych w bazie danych pozwala na proste tworzenie złożonych zapytań, wymagających selektywnego przeszukania wielu tablic dla sformułowania odpowiedzi. Wszechstronność tych poleceń jest dodatkowo zwiększana przez możliwość tworzenia w prosty sposób tymczasowych tablic zawierających informację wyselekcjonowaną z wielu tablic bazy danych.

Relacyjna baza danych mBase

System mBase przewiduje wykorzystywanie pamięci operacyjnej co najmniej 248 KB i pracuje w środowisku systemu operacyjnego RSX-60. Funkcjonalnie system mBase jest pełnym odpowiednikiem systemu dBase III firmy Ashon Tate, opracowanego dla komputerów z procesorem Intel 8086/8088. Choć system dBase III nie jest najlepszym systemem bazy danych, z powodu szerokiego rozpowszechnienia stanowi on standard światowy. Przyjęcie takiego standardu jako wzorca, dla nowo opracowanego systemu relacyjnej bazy danych dla mikrokomputera Mera 600, dało oczywiste korzyści w postaci zgodności na poziomie postaci źródłowej jego programów z tysiącami już istniejących systemów użytkowych. Co więcej, zgodność taka ułatwia przyszłe prace projektowo-programistyczne w instytucjach stosujących jednocześnie sprzęt komputerowy typu Mera 600 i IBM PC/XT/AT.

Baza danych użytkownika w systemie mBase składa się z pewnej liczby plików danych (relacji) o określonej przez użytkownika strukturze. Z każdym plikiem danych związanych może być do 7 plików indeksowych, budowanych na podstawie wartości wybranego pola

(atrybutu) pliku danych lub na podstawie wyrażenia uwzględniającego wartości kilku pól. Obsługa bazy danych odbywa się w jednym z dwu trybów pracy: interakcyjnym lub programowym. W obu trybach dostępne są praktycznie wszystkie polecenia systemu mBase. Niewielkie występujące różnice w działaniu niektórych poleceń w poszczególnych trybach dotyczą szczegółów mniejszej wagi i wynikają ze specyfiki obu trybów pracy.

Podstawowe parametry techniczne systemu bazy danych mBase są następujące:
- maksymalna liczba rekordów w pliku danych (obiektów w relacji) – 1 mld,
- maksymalna wielkość rekordu (obiektu) – 4096 bajtów,
- maksymalna liczba pól rekordu (atrybutów relacji) – 128,
- 5 typów pól (tekstowe, numeryczne, logiczne, data, informacyjne),
- maksymalna liczba jednocześnie dostępnych plików danych i (lub) indeksowych – 15.

Oprócz podstawowych typów plików, tj. plików danych i indeksowych system mBase przewiduje również takie typy plików, jak plik formatu ekranu, plik formatu raportu, plik programu.

Polecenia systemu mBase pozwalają wykonać wszelkie niezbędne operacje na plikach danych i ich rekordach (na relacjach i obiektach). Każde polecenie może być użyte w trybie interakcyjnym i wówczas jest ono natychmiast wykonywane. W trybie programowym poza poleceniami mogą być używane dodatkowe instrukcje sterujące przebiegiem programu (realizacja pętli, instrukcje warunkowe, instrukcje wywołania i powrotu z podprogramu).

Polecenia **definiujące pliki** danych pozwalają definiować nową strukturę rekordów pliku lub modyfikować strukturę istniejącego już pliku. Struktura tworzonego pliku definiowana jest interakcyjnie przez użytkownika lub pobierana z innego pliku danych. Do tej grupy poleceń należy również zaliczyć polecenia definiujące format ekranu oraz format raportów produkowanych przez system.

Polecenia **wyboru pliku** danych pozwalają wybrać plik aktualnie dostępny. Przy jednoczesnym używaniu większej liczby plików tylko jeden z nich jest aktualnie dostępny.

Polecenia **modyfikujące zawartość** plików danych pozwalają dopisywać, usuwać i modyfikować rekordy plików danych. Rekordy dopisywane mogą pochodzić z innych plików danych (niekoniecznie o identycznej strukturze) lub być określone interakcyjnie z klawiatury.

Polecenia **ustalające porządek** rekordów pliku danych pozwalają indeksować plik danych na podstawie wartości wybranych pól rekordów lub też sortować go.

Polecenia **modyfikacji zawartości** rekordów pliku pozwalają zmieniać wartości poszczególnych pól wybranego rekordu lub grupy rekordów w pliku.

Polecenia **wyboru rekordu** pozwalają przejść do najbliższego rekordu spełniającego podany warunek. Kolejność rekordu w pliku danych zmienia się lub nie, zależnie od używania (lub nieużywania) plików indeksowych.

Polecenia **ustalające powiązania** pomiędzy plikami danych pozwalają z każdym rekordem wybranego pliku danych powiązać odpowiadający mu rekord innego pliku danych.

Polecenia **wyprowadzania informacji** pozwalają wyprowadzić żądane informacje na ekran monitora, na drukarkę lub do pliku tekstowego na dysku. Wyprowadzana może być zawartość rekordów pliku danych (lub ich części), a także informacje o istniejących lub dostępnych plikach danych, o stanie przetwarzania itp. Przy wyprowadzaniu zawartości rekordów mogą być stosowane pliki formatu raportu.

Polecenia **wspomagające** ułatwiają proces tworzenia i testowania programu. Pozwalają one na pełną rejestrację wszystkich wykonanych poleceń w pliku dyskowym (przy tworzeniu programu), a także na uzyskanie śladu wykonywanego programu (przy jego testowaniu). Do grupy tej zalicza się również polecenia informujące o formacie i zasadach stosowania wszystkich poleceń systemu.

Polecenia **obliczeniowe** pozwalają uzyskać pożądaną charakterystykę przekrojową dla ustalonego pola (zwykle numerycznego) wybranej grupy rekordów pliku danych, np. wartość średnią, sumę, liczbę wystąpień.

Polecenia **konwersji formatu** mające postać funkcji pozwalają przekształcać wartość pól jednego typu na inne.

Polecenia **ustalające charakterystykę** systemu pozwalają definiować sposób reakcji systemu na określone czynności użytkownika i (lub) wykonywanego programu.

Polecenia **pomocnicze** pozwalają wykonać ogólnosystemowe operacje na plikach wchodzących w skład bazy danych (kopiowanie, przemianowanie, usunięcie). Rozszerzenie tej grupy poleceń polega na umożliwieniu wykonania dowolnego programu lub dyrektywy systemu operacyjnego bez utraty dotychczasowych wyników pracy. Po wykonaniu następuje powrót do systemu mBase bez zmiany jego stanu.

Konferencje

Systems Science X

W dniach 19–22 września br. odbędzie się we Wrocławiu Międzynarodowa Konferencja Systems Science X, poświęcona problematyce systemów automatyki i informatyki. Jest to kolejna konferencja z cyklu realizowanego wspólnie przez Coventry Polytechnic w Wielkiej Brytanii, University of Nevada w USA oraz Politechnikę Wrocławską. Ze strony polskiej jej organizatorami są: Komitet Automatyki i Robotyki PAN oraz Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej.

Problematyka konferencji:
- teoria sterowania, teoria systemów,
- identyfikacja, modelowanie i optymalizacja,
- złożone systemy sterowania, systemy wspomagające,
- podejmowanie decyzji, elastyczne systemy produkcyjne,
- zastosowanie systemotechniki i informatyki w systemach technicznych, transportowych, biomedycznych, ekonomicznych.

Informacji udziela **doc. Jerzy Świątek; Instytut Sterowania i Techniki Systemów, Politechnika Wrocławska, ul. Janiszewskiego 11/17, 50–370 Wrocław.**

JACEK SKRZYMOWSKI
Instytut Technologii Budowy Maszyn
Politechnika Wrocławska

Nowa norma Fortranu (1)

Mimo opracowania wielu nowocześniejszych języków programowania, Fortran stanowi w dalszym ciągu jedno z najważniejszych narzędzi do rozwiązywania zadań naukowo-technicznych. W ciągu wielu lat istnienia Fortranu włożono wiele wysiłku w tworzenie oprogramowania w tym języku, zatem zrozumiałe jest, że użytkownicy chcieliby wykorzystywać je także w przyszłości. Z drugiej strony, pożądane byłoby uwzględnienie w języku możliwości nowoczesnego sprzętu komputerowego. Oba te czynniki są ważnym powodem dużego zainteresowania dalszym rozwojem Fortranu.

Od chwili opublikowania normy Fortranu 77, w kwietniu 1978 r., prowadzono prace nad nową wersją tego języka. Wynikiem tych prac, w których uczestniczyli m.in. eksperci z ANSI i ISO, jest projekt nowej, radykalnie rozwiniętej normy Fortranu, nazwanej Fortran 88. Podstawowym wymaganiem wziętym pod uwagę przy opracowaniu Fortranu 88 była jego pełna zgodność z dotychczasową normą, co zostało osiągnięte w ten sposób, że Fortran 77 jest całkowicie zawarty w Fortranie 88. Fortran 88 zawiera jednak nowe rozwiązania, które sprawdziły się w nowoczesnych językach programowania.

Powszechnie znane są złe cechy Fortranu powodujące najczęstsze trudności w programowaniu, jak np.:
- operacje wymagające znajomości odwzorowania obiektów danych w fizycznej pamięci;
- sztywny format zapisu programów źródłowych;
- niedokładność numeryczna, związana z długością słowa konkretnego procesora;
- brak struktur danych.

Trudności te zostają w znacznym stopniu zmniejszone przez wprowadzenie nowych konstrukcji w Fortranie 88.

Najważniejszymi rozszerzeniami nowej normy w stosunku do Fortranu 77 są:
- swobodny format zapisu programów źródłowych;
- nowy sposób zapisu deklaracji typu;
- efektywne działania na tablicach;
- wprowadzenie tablic dynamicznych;
- zwiększone możliwości określania dokładności numerycznej;
- możliwość definiowania nowych typów (struktur) danych;
- możliwość rekurencyjnego wywoływania podprogramów;
- możliwość definiowania nowych operatorów lub przedefiniowywania standardowych (również operatora przypisania);
- wprowadzenie modułów, grupujących obiekty i podprogramy;
- nowe instrukcje sterujące.

Przewiduje się, że projekt w swej ostatecznej wersji jako nowa norma ukaże się w najbliższych miesiącach.

ALFABET, POSTAĆ PROGRAMU I DYREKTYWA TYPU

Dopuszczono stosowanie małych liter do zapisu programów, przy czym są one równoważne dużym (poza stałymi znakowymi). Dozwolone jest używanie znaku podkreślenia w nazwach (mogą one mieć długość 31 znaków). Jako nowe znaki specjalne wprowadzono: !, ", %, ;, <, >, &, [, i]. Znaki te są używane jako symbole operatorów, do nawiasowania i różnych form oddzielania i ograniczania innych jednostek leksykalnych.

Każda linia (zdanie) programu źródłowego[1] może zawierać 0...132 znaki. Wykrzyknik rozpoczyna komentarz, który sięga do końca wiersza. Średnikiem oddziela się dyrektywy i instrukcje napisane w jednym wierszu. Wystąpienie na końcu wiersza znaku & oznacza, że linia jest kontynuowana w następnym wierszu. Jeśli pierwszym różnym od spacji znakiem następnego wiersza jest &, to kontynuuje się instrukcje od następnego znaku, w przeciwnym razie – od początku wiersza. Po znaku kontynuacji może następować komentarz, z wyjątkiem wypadku, gdy ma być kontynuowana stała tekstowa.

Każdy obiekt w Fortranie 88 posiada typ i liczne inne właściwości (atrybuty). Specyfikowanie typu obiektu odbywa się za pomocą następująco zmienionej dyrektywy typu:

typ, lista atrybutów: : *lista deklaracji*

Jako typ mogą występować wszystkie dotychczasowe specyfikatory typu (*INTEGER, REAL, DOUBLE PRECISION, COMPLEX, CHARACTER i LOGICAL*). Przy użyciu słowa *TYPE* można także deklarować obiekty typu strukturalnego. Znaki „: :" można pominąć, jeśli stosuje się deklarację zgodną z Fortranem 77. Dla typów *REAL* i *COMPLEX* można dodatkowo podać dokładność dziesiętną i dopuszczalny zakres wykładnika. Dla typu *CHARACTER* zastępuje się dotychczasowy sposób określania długości **n* przez słowo *LEN:*

CHARACTER (LEN=10) : : *TEXT*

Jako atrybuty obiektu mogą wystąpić m.in. *PARAMETER* jako deklaracja stałej, *INITIAL* dla nadawania obiektom wartości początkowych, *ARRAY* jako deklaracja tablicy, *SAVE* dla zachowania wartości zmiennej lokalnej podprogramu między wywołaniami, np.:

REAL, PARAMETER :: *JEDEN* = 1.0, *TRZY* = 3.0
INTEGER, ARRAY (1:20) :: *WEKTOR*

Pierwszy wiersz deklaruje dwie stałe typu rzeczywistego, drugi tablicę jednowymiarową o elementach typu całkowitoliczbowego.

TABLICE

Operacje na tablicach zostały wprowadzone głównie z myślą o zastosowaniu w komputerach wieloprocesorowych. Operacje te można przeprowadzać zarówno na całych tablicach, jak też na wycinkach tablic. Skutkiem wprowadzenia operacji tablicowych będą krótsze i bardziej przejrzyste programy, ponieważ niepotrzebne stanie się, na przykład, stosowanie pętli do przetwarzania tablic. W szczególności, na tablicach mogą być wykonywane standardowe operacje arytmetyczne i logiczne. Możliwe są także przypisania całych tablic lub ich części, maskowane przypisania tablicowe, użycie stałych i wyrażeń tablicowych oraz definiowanie funkcji tablicowych. Wprowadzono również nowe funkcje standardowe umożliwiające tworzenie tablic i operacje na nich, łączenie tablic i dzielenie ich na części.

Tablice można tworzyć podając ciąg wartości skalarnych ujętych w nawiasy kwadratowe, np. w wyniku przypisania:

REAL, ARRAY (2,2) :: *X*
X=[3.2, 4.03, 5.6, 2.7]

poszczególne elementy tablicy *X* otrzymają wartości:

X(1,1)=3.2; *X*(2,1)=4.03; *X*(1,2)=5.6; *X*(2,2)=2.7

Każde wyrażenie tablicowe stopnia pierwszego może być użyte jako konstruktor tablicy. W pewnych wypadkach możliwy jest uproszczony zapis konstruktora, np.;

[1] Termin linia ma znaczenie geometryczne. Podobnie jak linię w znaczeniu geometrycznym można podzielić na odcinki, tak linię w znaczeniu programistycznym można podzielić na wiersze. Długość wiersza w tym samym programie nigdy zatem nie jest większa niż długość linii. Takie znaczenie terminów linia i wiersz proponujemy przyjąć w „Informatyce" (przyp. red.).

```
INTEGER I(50)
REAL R(30)
I=[2:100:2]
R=[10 [0.0, 1.0, 2.0]]
```

Kolejnym elementom tablicy *I* przypisuje się liczby parzyste z zakresu 2..100. Druga instrukcja powtarza dziesięciokrotnie przypisanie tych samych trzech wartości kolejnym elementom tablicy *R*.

Tablice, nawet różniące się typem elementów, a mające tę samą wymiarowość i te same granice indeksów, mogą być grupowane dyrektywą *RANGE*:

```
REAL, ARRAY(10,10) :: A,B
INTEGER, ARRAY(10,10) :: IX
RANGE /LISTAT/ A, B, IX
```

Postać tablic *A, B, IX* może być później zmieniona przez wykonanie instrukcji *SET RANGE*. Instrukcja ta ustala nowe granice dla wszystkich tablic zgrupowanych w danej liście:

```
N=3
SET RANGE (N:2*N, N:2*N) /LISTAT/
```

Po wykonaniu tej instrukcji wszystkie tablice z listy *LISTAT* będą miały dolną granicę indeksów 3, a górną 6. Dalsze operacje na tych tablicach będą wykorzystywać nowe wartości granic, dopóki nie zostaną one zmienione przez inną instrukcję *SET RANGE*.

Jeśli zachodzi potrzeba zmiany granic indeksów dla pojedynczych tablic, to można je wymienić w dyrektywie *RANGE* bez nazwy listy, np.:

```
DIMENSION RA(10), TA(10)
RANGE RA, TA
```

Tablice *RA* i *TA* mogą teraz występować niezależnie od siebie w różnych instrukcjach *SET RANGE*.

Wprowadzono nową konwencję zapisu, która umożliwia wybranie z każdej tablicy prostokątnego wycinka z regularnymi lukami:

$A(a_1:b_1:c_1, \ldots)$

Pierwszy parametr określa wartość początkową indeksu, drugi wartość końcową, a trzeci krok. W wypadku niepodania pewnych parametrów przyjmuje się dolną granicę indeksu dla pierwszego parametru, górną granicę indeksu dla drugiego oraz jeden dla trzeciego, np.:

```
REAL, ARRAY(5,4,3) :: D
D(3:5, 2, 1:2) = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
```

Wynikiem powyższej selekcji jest wycinek tablicy stopnia 2, o wymiarach 3 na 2, składający się z elementów:

D(3,2,1), *D*(4,2,1), *D*(5,2,1), *D*(3,2,2), *D*(4,2,2), *D*(5,2,2).

Możliwe jest nadanie wycinkowi tablicy innej nazwy. W tym celu należy zadeklarować tablicę z atrybutem *ALIAS* i wykonać dynamiczne odwzorowanie na tę tablicę tablicy bazowej przy użyciu instrukcji *IDENTIFY*. Obie tablice muszą być tego samego typu. Instrukcja *IDENTIFY* daje znacznie większe możliwości tworzenia wycinków tablicy, np.:

```
REAL, ARRAY(1:10, 1:10) : : A
REAL, ARRAY(:), ALIAS : : DIAG
IDENTIFY (DIAG(I) = A(I,I), I=1:10)
```

W przykładzie tym tworzy się jednowymiarową tablicę *DIAG*, która zawiera wszystkie elementy przekątnej dwuwymiarowej tablicy *A*. Tablica *DIAG* przyjmuje konkretny kształt dopiero po wykonaniu instrukcji *IDENTIFY*, kojarzącej się z tablicą bazową.

Istnieje możliwość zmiany wartości tych elementów tablicy, które spełniają określone warunki. Operacja ta zwana przypisaniem maskowanym jest wykonywana instrukcją *WHERE*, o postaci zbliżonej do instrukcji *IF*:

```
REAL, ARRAY(1:10, 1:20) :: TEMP, DIFF TEMP, PRESS, DIFF
                           PRESS
...
WHERE (TEMP<100.0) TEMP = TEMP -DIFF TEMP
...
WHERE (TEMP>80.0)
PRESS = PRESS + DIFF PRESS
TEMP = TEMP -DIFF TEMP
ELSEWHERE
PRESS =1.0
END WHERE
```

W pierwszej postaci instrukcji *WHERE* te elementy tablicy *TEMP*, dla których warunek w nawiasach jest spełniony, przybierają nową wartość. Możliwa jest także zmiana wartości odpowiednich elementów innych tablic, które muszą jednak mieć te same wymiary co tablica testowana. Blok *ELSEWHERE* umożliwia wykonanie przypisań również dla tych elementów, dla których warunek nie jest spełniony.

DOKŁADNOŚĆ NUMERYCZNA

Nowa norma Fortranu przewiduje, że w każdej implementacji będą zapewnione przynajmniej dwie metody aproksymacji liczb rzeczywistych. Każdą taką metodę charakteryzują dwa parametry: efektywna precyzja dziesiętna i efektywny zakres wykładnika. Przy definiowaniu obiektów typu *REAL* można dodatkowo podawać dwa parametry w celu określenia minimalnej precyzji i zakresu wykładnika:

```
REAL (PRECISION=8, EXPONENT RANGE=150) :: X(100)
```

Powyższa dyrektywa oznacza, że każdy element tablicy *X* ma precyzję co najmniej 8 cyfr dziesiętnych i zakres wykładnika –150..150. Kompilator wykorzystuje te dane, aby wybrać najbardziej odpowiednią maszynową reprezentację zmiennych.

Według normy kompilator musi wybrać taką samą metodę aproksymacji, która zapewnia co najmniej taką samą precyzję i co najmniej taki sam zakres wykładnika, jak wyspecyfikowano. Jeśli istnieje wiele takich metod, to należy wybrać tę, która przekracza pożądane wartości jak najmniej. Stosowanie nowego sposobu określania precyzji ma na celu ułatwienie pisania w Fortranie programów przenośnych, co jest szczególnie ważne dla bibliotek podprogramów matematycznych.

Efektywną precyzję zastosowaną przez kompilator można odczytać funkcją *EFFECTIVE PRECISION*, a efektywny zakres wykładnika funkcją *EFFECTIVE EXPONENT RANGE*. W celu podania stałej rzeczywistej z określoną precyzją i zakresem wykładnika, należy użyć dyrektywy *EXPONENT LETTER*:

```
EXPONENT LETTER (8,150) L
...
CALL SUB (10.987654L23)
...
SUBROUTINE SUB (PARAM)
REAL (8,150) PARAM
```

W powyższym przykładzie użyto dyrektywy *EXPONENT LETTER* do określenia precyzji i zakresu wykładnika stałej przekazywanej jako parametr typu *REAL* do podprogramu.

Aktualna norma przewiduje, że jeśli dla danego parametru formalnego określono jawnie precyzję, to parametry aktualne muszą posiadać tę samą precyzję. Ponieważ jest to duże ograniczenie, wprowadzono możliwość określania precyzji parametru formalnego jako *REAL (*, *)*. Parametr taki może być zastąpiony parametrem aktualnym o dowolnej precyzji, jednakże wszystkie określone w ten sposób parametry formalne podprogramu muszą mieć zawsze tę samą precyzję podczas wywołania.

```
FUNCTION BELKA (X,Y)
REAL(*,*)::X, Y
...
```

W tym wypadku można wywołać funkcje *BELKA* z dwoma argumentami typu *REAL* o dowolnej precyzji, lecz tej samej dla obu parametrów.

Jako dodatkowy typ wprowadzono struktury danych. Struktury definiuje się w blokach *TYPE*, podając nazwę struktury oraz nazwy i typy jej składowych, np.:

```
TYPE OSOBA
    INTEGER WIEK
    CHARACTER(LEN=50) NAZWISKO
END TYPE OSOBA
```

W powyższym przykładzie definiuje się strukturę o nazwie *OSOBA*, zawierającą składowe *WIEK* i *NAZWISKO*. Można wykorzystywać uprzednio zdefiniowane struktury do definiowania nowych. Definicja struktury może zawierać również listę parametrów, np.:

```
TYPE STRING (MAX LEN)
    INTEGER LENGTH
    CHARACTER(LEN=MAX LEN) TEXT
END TYPE STRING
```

W powyższym przykładzie użyto parametru *MAX LEN* do określenia rzeczywistego rozmiaru składowej *TEXT*. Zawarte w liście parametrów nazwy są parametrami formalnymi typu *INTEGER* i mogą być wykorzystane jako wyrażenia pierwotne do deklarowania składowych. Gdy deklaruje się obiekt typu strukturalnego z parametrami, należy podać wartości tych parametrów. Dla listy parametrów istnieją dwa słowa zastrzeżone: *PRECISION* i *EXPONENT RANGE*. Określają one precyzję i zakres wykładnika dla składowych typu *REAL* lub *COMPLEX:*

```
TYPE MATRIX (PRECISION, EXPONENT RANGE, ORDER)
REAL (PRECISION, EXPONENT RANGE), ARRAY (ORDER,
      ORDER):: A
END TYPE MATRIX
```

Deklaracja obiektu strukturalnego następuje przez użycie dyrektywy specyfikacji *TYPE*, przy czym możliwe jest również tworzenie tablic struktur, np.:

```
TYPE (OSOBA) :: OSOBA1
TYPE (OSOBA), ARRAY (1:10) :: OSOBY
TYPE (STRING (MAX LEN=20)) :: DANA
```

OSOBA1 jest skalarem typu *OSOBA; OSOBY* jest tablicą o elementach typu *OSOBA*, a *DANA* jest strukturą typu *STRING* o maksymalnie 20 znakach w składowej *TEXT*.

Struktura może zawierać jedną lub więcej składowych. Dostęp do składowych struktury następuje przy użyciu operatora %, np.:

```
OSOBA % WIEK
OSOBY(J) % NAZWISKO
```

Przez podanie ciągu wartości (jednej) dla każdej składowej tworzy się wartość strukturalną, np.:

```
OSOBA (21,'JAN NOWAK')
STRING (20) (16, "Hello, I am JanB")
```

Spośród operacji standardowych dla struktur dostępne jest jedynie przypisanie. Można jednak definiować własne operatory realizujące inne działania.

WIESŁAW L. NOWIŃSKI
Instytut Podstaw Informatyki
Polska Akademia Nauk
Warszawa

Algorytmy rekonstruowania obrazów metodami rozwinięcia w szereg

W poprzednich artykułach [4,5] opisano ogólny schemat systemu do rekonstruowania obrazów oraz scharakteryzowano metody transformacyjne i oparte na tych metodach algorytmy rekonstruowania. Celem niniejszego artykułu jest scharakteryzowanie metod rekonstruowania opartych na rozwinięciu w szereg oraz odpowiadających im algorytmów.

PODSTAWY METOD REKONSTRUOWANIA OPARTYCH NA ROZWINIĘCIU W SZEREG

Metody rekonstruowania obrazów oparte na rozwinięciu w szereg charakteryzują się sformułowaniem problemu od samego początku w postaci dyskretnej, w przeciwieństwie do metod transformacyjnych, w których dopiero w celu zaimplementowania należy przedstawić operatory ciągłe w postaci dyskretnej. Ponadto metody te, w odróżnieniu od metod transformacyjnych, nie ograniczają się do pomiarów wykonywanych wyłącznie wzdłuż prostych.

W metodach tych przyjmuje się, że obraz **f** można rozwinąć w szereg względem odpowiednio dobranych obrazów bazowych. Dlatego też zakłada się istnienie zbioru liniowych i ciągłych funkcjonałów $\{R_i\}$, $i = 1..I$, odwzorowujących obraz na liczbę rzeczywistą (zauważmy, że w metodach transformacyjnych R_i jest dla wszystkich i, czyli wszystkich promieni, transformacją Radona). Ponadto zakłada się znajomość zbioru obrazów bazowych $\{b_j\}$ $j = 1..J$, których kombinacja liniowa daje odpowiednią aproksymację **f** obrazu **f**, czyli istnienie takich liczb rzeczywistych, że

$$\mathbf{f} \approx \hat{\mathbf{f}} = \sum_{j=1}^{J} x_j b_j \tag{1}$$

Wektor $\mathbf{x} = [x_1, x_2, ..., x_J]^T$ nazywa się wektorem obrazu. Uwzględniając liniowość i ciągłość operatorów R_i z równania (1) otrzymuje się

$$y_i \approx R_i\mathbf{f} \approx R_i\hat{\mathbf{f}} = \sum_{j=1}^{J} x_j R_j b_j \approx \sum_{j=1}^{J} x_j r_{i,j} \tag{2}$$

We wzorze (2) y_i równa się w przybliżeniu $R_i\mathbf{f}$, gdyż uwzględniono błędy pomiarowe; ponadto może zaistnieć potrzeba aproksymacji $R_i b_j$ przez $r_{i,j}$ ze względu na efektywność obliczeń lub brak możliwości dokładnego obliczenia wartości $R_i b_j$.

Oznaczając przez **R** macierz $[r_{i,j}]_{I \times J}$ oraz przez **e** – I-wymiarowy wektor kolumnowy, którego i-ty element jest różnicą między lewą a prawą stroną zależności (2), otrzymujemy

$$\mathbf{y} = \mathbf{R}\,\mathbf{x} + \mathbf{e} \tag{3}$$

Macierz **R** nazywa się macierzą projekcji, wektor **e** zaś – wektorem błędów. Stosując metodę rozwinięcia w szereg otrzymuje się następujące sformułowanie problemu rekonstruowania obrazu:

Dla danego wektora pomiarów y znaleźć, korzystając z równania (3), przybliżenie $\hat{\mathbf{x}}$ wektora obrazu x.

Tak sformułowany problem nazywa się dyskretnym problemem rekonstruowania. W wyniku rozwiązania dyskretnego problemu rekonstruowania otrzymuje się przybliżenie $\hat{\mathbf{x}}$ wektora obrazu **x**, a następnie oblicza przybliżenie $\hat{\mathbf{f}}$ obrazu **f** z zależności

$$\hat{\mathbf{f}} = \sum_{j=1}^{J} \hat{x}_j b_j \tag{4}$$

W równaniu (3) wektor błędów jest nieznany. Najprostsza próba rozwiązania równania (3), polegająca na przyjęciu, że **e** jest wektorem zerowym, nie gwarantuje uzyskania odpowiedniego rozwiązania problemu rekonstruowania, gdyż równanie $\mathbf{y} = \mathbf{R}\mathbf{x}$ może nie mieć rozwiązań lub może mieć wiele rozwiązań, z których żadne nie musi być dobre z punktu widzenia rekonstrukcji. Dlatego też sformułowano wiele różnych kryteriów określających, który z otrzymanych wektorów obrazu powinien być przyjęty jako rozwiązanie równania (3). Kryteria te są zwykle formułowane w postaci zadania optymalizacji statycznej:

Jako rozwiązanie równania (3) należy wybrać ten wektor obrazu, który minimalizuje pewną funkcję $\varphi_1(\mathbf{x})$ (kryterium podstawowe), a jeśli istnieje wiele różnych wektorów minimalizujących funkcję $\varphi_1(\mathbf{x})$, to należy z nich wybrać ten wektor, który minimalizuje inną funkcję $\varphi_2(\mathbf{x})$ (kryterium uzupełniające).

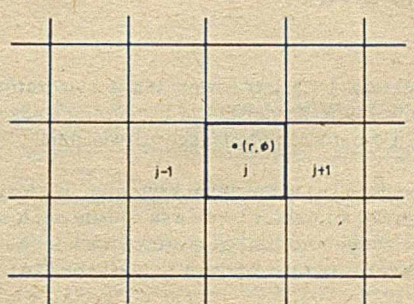
Stosuje się dwie podstawowe grupy metod rozwiązywania dyskretnego problemu rekonstruowania obrazów: metody iteracyjne i nieiteracyjne. W metodach nieiteracyjnych dobiera się obrazy bazowe tak, aby zdekomponować rozwiązywany układ równań liniowych; algorytmy rekonstruowania obrazów oparte na tych metodach sprowadzają się do rozwiązywania układów równań liniowych. Metody iteracyjne można zastosować do dowolnego zbioru obrazów bazowych; algorytmy oparte na tych metodach zależą od wybranych obrazów bazowych i przyjętych kryteriów: podstawowego i, ewentualnie, uzupełniającego.

OBRAZY BAZOWE

Jest wiele różnych sposobów definiowania obrazów bazowych. Niektóre z nich są bardzo proste, intuicyjnie oczywiste; inne mogą wynikać ze złożonych rozważań matematycznych, jak np. obrazy bazowe będące funkcjami własnymi transformaty Radona. Przedstawione zostaną dwa typy obrazów bazowych: dyskretyzujące obraz i pierścieniowe.

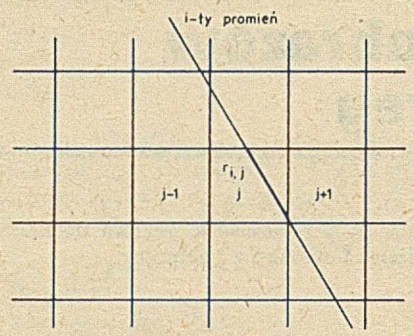
Niech J będzie liczbą punktów obrazu (pikseli). Wówczas obrazy bazowe definiuje się następująco (rys. 1):

$$b_j(r,\psi) = \begin{cases} 1 \text{ gdy punkt o współrzędnych biegunowych } (r,\psi) \text{ leży wewnątrz } j\text{-tego piksela;} \\ 0 \text{ w przeciwnym razie} \end{cases}$$



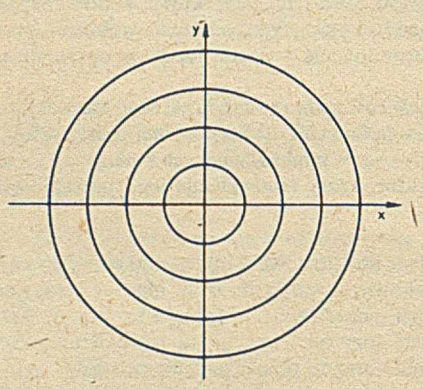
Rys. 1. Obrazy bazowe dyskretyzujące obraz

Aproksymując obraz **f** z użyciem tak zdefiniowanych obrazów bazowych otrzymuje się zdyskretyzowaną wersję obrazu **f**. Zaletą tak określonych obrazów bazowych jest łatwość wyznaczania wartości elementów macierzy projekcji: $r_{i,j}$ jest równe długości przecięcia i-tego promienia z j-tym pikselem (rys. 2), tym bardziej, że w typowych zastosowaniach liczba niezerowych elementów tej macierzy stanowi mniej niż 1% wszystkich jej elementów.



Rys. 2. Wyznaczanie wartości elementów macierzy projekcji dla obrazów bazowych dyskretyzujących obrazu

Wadą tych obrazów bazowych jest tworzenie brzegu obrazu w postaci schodków: w celu otrzymania gładszego (wizualnie) brzegu należy zwiększać wartość J, a tym samym wymiar macierzy projekcji.



Rys. 3. Pierścieniowe obrazy bazowe

Pierścieniowe obrazy bazowe otrzymuje się przez podział obszaru rekonstrukcji na pierścienie o jednakowej grubości i określenie w każdym pierścieniu pewnej funkcji harmonicznej (rys. 3). Dzięki zastosowaniu pierścieniowych obrazów bazowych macierz $\mathbf{R}^T\mathbf{R}$ (por. dalszą część artykułu) można zdekomponować na odpowiednie podmacierze dzięki ortogonalności szeregów harmonicznych.

KRYTERIA OPTYMALIZACJI

Kryteria podstawowe i uzupełniające są sformułowane w postaci zadania optymalizacji statycznej, w którym zbiór rozwiązań dopuszczalnych jest zbiorem rozwiązań układu równań (3).

Kryterium minimalizujące wektor błędów jest następujące:

$$\min \| \mathbf{e} \|^2 = \min \| \mathbf{y} - \mathbf{R}\mathbf{x} \|^2 \tag{5}$$

gdzie norma wektora $\| \cdot \|$ jest równa sumie kwadratów wszystkich elementów tego wektora. Kryterium to ma istotną wadę – nie zawiera żadnej informacji o naturze pożądanego rozwiązania.

Odmienne podejście jest w kolejnym kryterium, w którym przyjmuje się, że wektor obrazu **x** i wektor błędu **e** są zmiennymi losowymi X i E, oraz zakłada się znajomość funkcji gęstości prawdopodobieństwa p_X i p_E. Wówczas wektor obrazu wybiera się następująco

$$\max[p_E(\mathbf{y} - \mathbf{R}\mathbf{x}) p_X(\mathbf{x})] \tag{6}$$

Zgodnie z tym kryterium wybiera się najbardziej prawdopodobny wektor obrazu, który będzie jednocześnie najbardziej zgodny z danymi pomiarowymi.

Okazuje się, że te i inne kryteria są szczególnymi przypadkami następującego kryterium kwadratowego:

$$\min[a(\mathbf{y} - \mathbf{R}\mathbf{x})^T \mathbf{A} (\mathbf{y} - \mathbf{R}\mathbf{x}) + (\mathbf{x} - \mathbf{x}_O)^T (b\mathbf{B} + c\mathbf{C}^{-1})(\mathbf{x} - \mathbf{x}_O)] \tag{7}$$

gdzie **A**, **B**, **C** są odpowiednimi macierzami, $\mathbf{x}_0$ wektorem, a wagi a, b, c – nieujemnymi liczbami rzeczywistymi.

W dotychczasowych rozważaniach założona informacja o rekonstruowanym obrazie była zawarta w kryterium podstawowym. Bardzo często informacja ta dotyczy przedziałów, wewnątrz których powinny leżeć wartości elementów wektora obrazu (np. być nieujemne przy stosowaniu wektorów bazowych dyskretyzujących obraz). Można również zażądać, aby dla wszystkich rozwiązań układu równań (3) wartość każdego elementu wektora błędów należała do założonego przedziału. Prowadzi to do zastąpienia układu równań (3) z nieokreślonym wektorem błędów układem nierówności

$$\mathbf{N}\,\mathbf{x} \leqslant \mathbf{q} \tag{8}$$

Wobec tego rozwiązanie dyskretnego problemu rekonstruowania z nieokreślonym wektorem błędów sprowadza się do poszukiwania wektora obrazu spełniającego układ nierówności (8). W celu jednoznacznego rozwiązania układu nierówności (8) wprowadza się dwa kryteria uzupełniające: kryterium minimalnej normy wektora obrazu i kryterium maksymalnej entropii wektora obrazu.

ITERACYJNE METODY REKONSTRUOWANIA OBRAZÓW

Przyjmijmy, że przy wyznaczaniu wektora pomiarowego **y** wykonano 512 rzutów, zawierających po 512 promieni każdy. Wówczas liczba równań układu (3) wynosiłaby 256 K. Nawet przy ograniczeniu pomiarów do 128 rzutów po 128 promieni liczba równań pozostanie ogromna – 16 K. Złożoność problemu (a tym samym zapotrzebowanie na moc obliczeniową) dodatkowo zwiększa możliwość istnienia wielu rozwiązań układu (3). Z tego też powodu, aby rozwiązanie dyskretnego problemu rekonstruowania obrazów było rozwiązywalne w realnym czasie i z realnym kosztem, zastosowano metody iteracyjne.

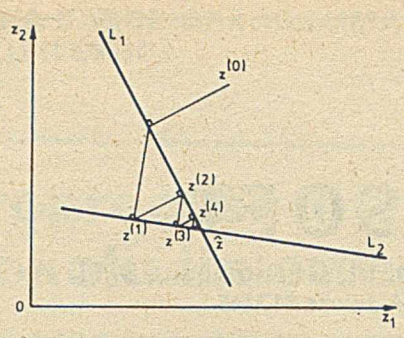
Istnieją dwie podstawowe grupy interacyjnych metod rekonstruowania obrazów: metody algebraiczne i metody symultacyjne. W każdym kroku iteracyjnym dla metod algebraicznych wykorzystuje się pewien (możliwie mały) podzbiór równań układu (3) lub nierówności układu (8), dla metod symultacyjnych natomiast – wszystkie równania układu (3) lub nierówności układu (8).

W algebraicznych metodach rekonstruowania korzysta się ze specyficznej metody rozwiązywania układów równań i nierówności, zwanej metodą relaksacyjną. Metoda relaksacyjna zastosowana do rekonstruowania obrazów okazała się identyczna z metodą rozwiązywania układów algebraicznych równań liniowych, przedstawioną w 1937 r. przez Stefana Kaczmarza [2].

Metoda relaksacyjna

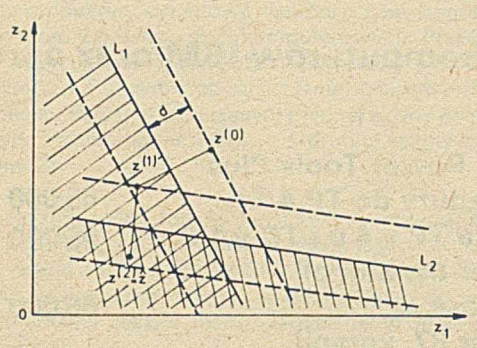
Metoda relaksacyjna jest iteracyjną metodą rozwiązywania układów nierówności: można ją również zaadaptować do rozwiązywania układów równań. Aby ją zilustrować, warto rozważyć układ składający się z dwóch równań o dwóch niewiadomych z_1 i z_2. Na płaszczyźnie $Oz_1 z_2$ równania te wyznaczają dwie proste L_1 i L_2 (w ogólnym wypadku są to hiperpłaszczyzny), których punkt przecięcia $\hat{\mathbf{z}}$ jest poszukiwanym rozwiązaniem układu równań (rys. 4). Rzutując (począwszy od dowolnie wybranego punktu startowego $\mathbf{z}^{(0)}$) na proste L_1, L_2, L_1, L_2,... kolejne punkty otrzymuje się ciąg $\mathbf{z}^{(0)}$, $\mathbf{z}^{(1)}$, $\mathbf{z}^{(2)}$,... zbieżny do $\hat{\mathbf{z}}$.

Metoda relaksacyjna jest pewną odmianą powyższego postępowania. Przemieszczenie danego punktu odległego od hiperpłaszczyzny o wielkość d odbywa się w kierunku hiperpłaszczyzny ograniczającej daną półprzestrzeń, nowo generowany punkt nie musi natomiast leżeć na



Rys. 4. Iteracyjne rozwiązywanie układu równań

hiperpłaszczyźnie, lecz w warstwie o grubości d otaczającej tę hiperpłaszczyznę (rys. 5). Tak generowany ciąg punktów jest zbieżny do punktu $\hat{\mathbf{z}}$ spełniającego dany układ nierówności, jeśli zbiór rozwiązań tego układu nie jest zbiorem pustym.



Rys. 5. Interpretacja metody relaksacyjnej

Metodę relaksacyjną można zaadaptować do rozwiązywania układu równań, zastępując każde równanie równoważnym mu układem dwóch nierówności.

Algebraiczne metody rekonstruowania obrazów

Algorytmy algebraicznych metod rekonstruowania obrazów na ogół są oparte na metodzie relaksacyjnej.

Najprostszy algorytm tego rodzaju otrzymuje się stosując metodę relaksacyjną do rozwiązywania układu równań (3) z zerowym wektorem błędów. Wówczas metoda relaksacyjna daje rozwiązanie układu równań $\mathbf{y} = \mathbf{Rx}$ (o ile zbiór rozwiązań tego układu nie jest zbiorem pustym); ponadto jeśli się odpowiednio wybierze punkt startowy, to rozwiązanie spełnia kryterium minimalnej normy. Wadą tego algorytmu jest to, że rozwiązania układu $\mathbf{y} = \mathbf{Rx}$ nie muszą istnieć, a nawet jeśli istnieją, to ich przydatność do rekonstruowania obrazów może być wątpliwa.

Wady tej jest pozbawiony algorytm rozwiązywania układu nierówności $\mathbf{Nx} \leqslant \mathbf{q}$ metodą relaksacyjną. Ma on jednak inną wadę: generowane przez niego rozwiązanie nie spełnia żadnego kryterium.

Z tego też powodu powstały inne algorytmy, w których korzysta się z modyfikacji metody relaksacyjnej zachowujących zalety dwóch poprzednich algorytmów i niwelujących ich wady kosztem zwiększonych wymagań dotyczących czasu i pamięci komputera [3].

Symultacyjne metody rekonstruowania obrazów

W metodach symultacyjnych, podobnie jak w metodach algebraicznych, generuje się ciąg punktów zbieżny do pewnego punktu $\hat{\mathbf{x}}$. Zaletą tych metod jest to, iż rozwiązanie $\mathbf{x}$ spełnia kryterium kwadratowe, określone związkiem (7). Płaci się za to większymi niż w metodach algebraicznych wymaganiami dotyczącymi czasu i pamięci komputera. W każdym kroku iteracyjnym uwzględnia się bowiem wszystkie równania układu (3), co wymaga wielu operacji na macierzach; ponadto wielkość tego kroku (zapewniająca zbieżność metody) zależy od parametru będącego maksymalną wartością własną macierzy o złożonej postaci.

W metodach algebraicznych rezerwuje się obszar pamięci dla jednego punktu (którego elementy są w każdym kroku iteracyjnym uaktualniane), w metodach symultacyjnych natomiast w celu wykonania kroku iteracyjnego należy pamiętać oba punkty. Zwiększa to znacznie zapotrzebowanie na pamięć komputera. Przykładowo, jeśli wartości elementów wektora obrazu o wymiarach 512×512 pikseli ograniczyć tylko do liczb całkowitych (2 bajty), co może nie zapewnić zbieżności algorytmu, to na zapamiętanie wektora obrazu potrzeba 512 KB pamięci; do wykonywania dokładnych obliczeń z użyciem liczb rzeczywistych o podwójnej precyzji (8 bajtów) potrzeba natomiast aż 2 MB pamięci.

NIEITERACYJNE METODY REKONSTRUOWANIA OBRAZÓW

Innym sposobem mającym na celu zmniejszenie liczby obliczeń w metodach opartych na rozwinięciu w szereg jest dekompozycja układu równań. Wiąże się to z wyborem odpowiednich obrazów bazowych.

Niech wektor obrazu spełnia kryterium minimalnego wektora błędów. Z warunku na minimum ze związku (3) otrzymuje się

$$\mathbf{R}^T\mathbf{Rx} = \mathbf{R}^T\mathbf{y} \tag{9}$$

Jeśli macierz $\mathbf{R}^T\mathbf{R}$, zawierająca $J \times J$ elementów, jest nieosobliwa, to można wyznaczyć wektor $\mathbf{x}$ rozwiązując układ równań (9) np. metodą eliminacji Gaussa. Jednak koszty obliczeniowe tego rozwiązania byłyby bardzo duże, gdyż np. dla obrazu o wymiarach 512×512 pikseli J jest równe 256 K, czyli macierz $\mathbf{R}^T\mathbf{R}$ składa się z 256 K wierszy.

Okazuje się jednak, że wybierając pierścieniowe obrazy bazowe i dzieląc obszar rekonstrukcji na U pierścieni zawierających po V funkcji harmonicznych, otrzymuje się macierz $\mathbf{R}^T\mathbf{R}$ w postaci blokowej macierzy diagonalnej złożonej z $2V+1$ macierzy o wymiarach $U \times U$ [3]. W ten sposób problem rekonstruowania z macierzą $J \times J$ zostaje zdekomponowany na $2V+1$ podproblemów z macierzami $U \times U$, gdzie $J = U(2V+1)$. Stosując kwadratowe prawo nakładów obliczeniowych (tzn. przyjmując, że nakłady te są proporcjonalne do kwadratu liczby równań), otrzymuje się dla tej metody $(2V+1)$-krotne zmniejszenie nakładów obliczeniowych.

* * *

W artykule przedstawiono problemy rekonstruowania obrazów metodami rozwinięcia w szereg. Algorytmy tych metod wymagają znacznie większych mocy obliczeniowych i znacznie pojemniejszych pamięci niż algorytmy oparte na metodach transformacyjnych. Ponadto w przeciwieństwie do metod transformacyjnych, metody rozwinięcia w szereg nie dają możliwości zrekonstruowania części obrazu.

W stosunku do metod transformacyjnych, metody oparte na rozwinięciu w szereg mają jednak wiele zalet. Są bardzo elastyczne i mają większy zakres zastosowań; dane pomiarowe mogą być zbierane nie tylko wzdłuż prostych, lecz również wzdłuż dowolnej krzywej (jak np. w rekonstruowaniu z rzutów ultradźwiękowych [1]) lub pasa. Otrzymany układ równań (3) jest niezależny od geometrii wiązki promieni; wobec tego opracowane algorytmy rozwiązywania dyskretnego problemu rekonstruowania mogą być wykorzystywane przy różnych konfiguracjach pomiarowych, również i w rekonstrukcji trójwymiarowej. W sformułowaniu dyskretnego problemu rekonstruowania uwzględnia się błędy pomiarowe i ograniczoność liczby danych pomiarowych; umożliwia to stosowanie tych metod przy rzutach obarczonych błędami pomiarowymi (najodporniejsze na te błędy są metody symultacyjne), przy niekompletnych rzutach i przy małej liczbie rzutów (a więc mniejszym naświetlaniu pacjenta szkodliwym promieniowaniem).

Wszystkie te zalety metod rozwinięcia w szereg, przy ciągłej tendencji do zwiększania mocy obliczeniowej produkowanego sprzętu komputerowego, przypuszczalnie spowodują, że będą one w niedalekiej przyszłości powszechnie stosowane.

LITERATURA

[1] Johnson S.A., et al.: Reconstruction of tree-dimensional velocity fields and other parameters in acoustic ray tracing. Proc. IEEE Ultrasonics Symposium, 1975

[2] Kaczmarz S.: Angenäherte Auflösung von Systemen Linearer Gleichungen. Bull. Acad. Polon. Sci. Lett. A., Vol. 35, 1937

[3] Nowiński W. L.: Metody rekonstrukcji obrazów w tomografii komputerowej. Prace IPI PAN, nr 597, Warszawa 1986

[4] Nowiński W.: Rekonstruowanie obrazów w tomografii komputerowej. Informatyka, nr 3, 1989

[5] Nowiński W.: Algorytmy rekonstruowania obrazów za pomocą metod transformacyjnych. Informatyka, nr 4, 1989.

KRYSTYNA KANDZIORA-BLAUMAN
JERZY KISILEWICZ
Wrocław

Mikroprocesorowa realizacja protokołu liniowego w systemie SM

Budowane obecnie sieci komputerowe wykorzystują do transmisji danych przede wszystkim istniejącą sieć telefoniczną. Muszą więc być wyposażone w urządzenia umożliwiające współpracę komputera komunikacyjnego z siecią telefoniczną, tzn. w modemy i adaptery liniowe.

W artykule omówiono sprzętową realizację poszczególnych poziomów protokołu liniowego sieci komputerowych. Przedstawiono inteligentny adapter, realizujący poziom pakietowy protokołu, zbudowany z układów o dużym stopniu scalenia. Adapter jest urządzeniem komunikacyjnym umożliwiającym z jednej strony połączenie ze wspólną szyną minikomputera SM, a z drugiej strony ze stykiem modemowym S2.

Adapter realizuje część funkcji protokołu liniowego. Im więcej realizuje on funkcji, tym mniej czasu wymaga jego obsługa programowa przez węzeł (minikomputer SM) i tym większą sumaryczną szybkość transmisji można uzyskać w dołączonych do węzła sieci kanałach. Minimalny zestaw realizowanych przez adapter funkcji protokołu zawiera:

- konwersję równoległo-szeregową i szeregowo-równoległą bajtów,
- generowanie i rozpoznawanie sekwencji specjalnych,
- generowanie i sprawdzanie sum kontrolnych oraz sygnalizowanie błędów transmisji,
- przekazywanie sygnałów sterujących do modemu i z modemu.

Funkcje te są związane ze sterowaniem łączem fizycznym i dotyczą pierwszego poziomu protokołu liniowego.

REALIZACJA NIŻSZYCH POZIOMÓW PROTOKOŁU

Do grupy adapterów realizujących minimalny zestaw funkcji (tzn. poziom piewszy) protokołu liniowego należy adapter DUP 11 firmy DEC oraz kompatybilny z nim adapter ALS 11, którego model wykonano w Instytucie Sterowania i Techniki Systemów Politechniki Wrocławskiej [1]. Adaptery te mogą współpracować z protokołami bajtowymi, np. BISYNC, lub bitowymi, np. HDLC. Adapter ALS 11 sygnalizuje przerwaniami konieczność odczytu i zapisu każdego transmitowanego bajtu. Podprogramy obsługi przerwań mają około 15 instrukcji, zatem czas ich wykonywania na minikomputerze SM 3 wynosi około 120 μs. Wynika stąd, że sumaryczna szybkość transmisji we wszystkich kanałach danego węzła nie przekroczy 64 kb/s. Czas potrzebny na przetwarzanie transmitowanej informacji dodatkowo tę szybkość ograniczy. Do wielu zastosowań, gdzie wymagana jest większa szybkość transmisji lub obsługa większej liczby kanałów, adapter tego rodzaju już nie wystarcza. Konieczne stało się poszukiwanie rozwiązań, które umożliwiłyby sprzętową realizację funkcji protokołu, dotychczas programowo realizowanych przez minikomputer węzła. Dotyczy to funkcji sterowania połączeniem logicznym, a więc drugiego poziomu protokołu liniowego.

Firma DEC skonstruowała urządzenie COMM-IOP-DUP wykorzystujące pomocniczy komputer komunikacyjny KMC 11A jako integralny sterownik bezpośredniego dostępu do pamięci DMA dla adapterów DUP 11. Komputer KMC 11A ma przez UNIBUS dostęp w trybie DMA do pamięci komputera węzłowego i do sterowanych przez siebie adapterów. Testuje on periodycznie rejestry stanów adapterów, sprawdzając gotowość do zapisu i odczytu danych. Zapisuje też i odczytuje bufory danych w adapterach oraz zapisuje i odczytuje zawartość przydzielonych buforów w komputerze węzłowym. Realizacja ta ma kilka istotnych wad.

1. Częstotliwość strobowania rejestrów jest jednakowa dla wszystkich adapterów i zależy od szybkości danych w najszybszej linii. W związku z tym maksymalna szybkość transmisji danych w liniach obsługiwanych przez COM-IOP-DUP wynosi tylko 19,2 kb/s.

2. Przy maksymalnym obciążeniu wykorzystuje on 10% czasu magistrali UNIBUS, ponieważ przesłanie jednego bajtu danych między adapterem a pamięcią komputera węzłowego wymaga co najmniej trzech transmisji NPR (ang. *non-processor request*).

Rozwiązaniem alternatywnym jest realizacja nowej wersji adaptera rozszerzonej o moduł DMA. W tym wypadku transmisja jednego bajtu wymaga tylko jednego cyklu NPR. Jest to rozwiązanie bardziej niezawodne, ponieważ awaria jednego adaptera powoduje wyłączenie z sieci tylko jednej linii, natomiast w wypadku awarii komputera KMC 11A wyłączony zostaje cały węzeł sieci.

Przykładem takiego rozwiązania jest adapter ALS 11M [2]. Spełnia on wszystkie funkcje adaptera ALS 11 i jest tak skonstruowany, że może być użyty jako ALS 11, jeżeli układy związane z DMA nie zostaną uaktywnione. Adapter ten generuje przerwania wektorowe tylko w celu odczytania adresu i pojemności bufora danych, a więc znacznie rzadziej niż czynił to adapter bez cechy DMA. Przy buforach długości 256 bajtów czas obsługi programowej adaptera maleje około 256 razy i nie stanowi już ograniczenia dla szybkości transmisji. Istotnego znaczenia nabiera natomiast czas związany z programową realizacją pozostałych funkcji protokołu liniowego, ponieważ ogranicza on przepustowość węzła sieci. W celu zwiększenia tej przepustowości należy albo zastosować szybsze komputery węzłowe, co często nie jest możliwe, albo przenieść realizację innych funkcji protokołu liniowego do adapterów.

REALIZACJA POZIOMU PAKIETOWEGO PROTOKOŁU

Programowa realizacja wielu funkcji protokołu liniowego (np. X.25) ma tę zaletę, że umożliwia elastyczne rozszerzenie i modyfikowanie realizowanych funkcji. Uzasadnia to zastosowanie mikroprocesorów do realizacji protokołu liniowego w adapterach. Przykładem takiej realizacji może być przedstawiony w niniejszym artykule inteligentny adapter ALS 11μp.

Program realizujący protokoł liniowy w adapterze może być umieszczony albo w pamięci EPROM, albo wprowadzany do pamięci RAM w fazie inicjowania pracy węzła. Zastosowanie pamięci EPROM upraszcza konstrukcję i obsługę programową adaptera i z tego względu wydaje się korzystniejsze. Adapter musi też zawierać pamięć RAM, przy czym jej pojemność nie rzutuje w sposób zasadniczy na koszt i może wynosić np. 64 KB. Daje to możliwość buforowania całych ramek (w celu ich retransmisji, w wypadku braku potwierdzenia odbioru od odległego węzła, lub odrzucenia przy odbiorze przez adapter po stwierdzeniu błędu sumy kontrolnej).

W wypadku realizacji protokołu X.25 CCITT dla łącza symetrycznego z asynchronicznym trybem odpowiedzi, adapter powinien analizować lub zapisywać pola sterujące i adresowe transmitowanych ramek informacyjnych *I* oraz generować i analizować ramki sterujące nadzorcze *S* i nienumerowane *U*. Parametry zmienne protokołu muszą być przekazane do adaptera przez program węzła za pomocą rejestrów wejściowych w fazie inicjowania pracy węzła.

Podane powinny być w szczególności:
- czas oczekiwania na odpowiedź (tj. przeterminowania, ang. *time-out*),
- wielkość okna (<8),
- maksymalna liczba retransmisji,
- maksymalna liczba oktetów w ramce (<1024).

Adapter po zaprogramowaniu nawiązuje połączenia w kanałach do niego dołączonych. W wypadku braku połączenia określony kanał przechodzi w stan nieaktywny.

Poziom pakietowy protokołu komunikuje się z adapterem przesyłając z buforów informację do nadania lub rezerwując bufory puste, przeznaczone do odbioru informacji (przesyłany jest adres i pojemność bufora przez rejestry pośrednie adaptera). Informacja jest buforowana do wielkości okna dla każdego kanału i kierunku transmisji. Przy odbiorze zwalniane są bufory adaptera tych ramek, które zostały odebrane przez poziom wyższy i dla tych ramek wysyłane są potwierdzenia. Przy nadawaniu zgłaszane jest do minikomputera SM zakończenie transmisji bufora dopiero wtedy, gdy zostanie odebrane potwierdzenie z odległego urządzenia *DCE* lub *DTE*. Oznacza to podwójne buforowanie nadawanej informacji, co zapobiega konieczności jej zwrotu z buforów adaptera do SM w wypadku załamania się transmisji.

O wszelkich nieprawidłowościach transmisji (np. awaria linii) minikomputer SM może być poinformowany przez rejestry pośrednie adaptera, odczytywane w odpowiedzi na przerwanie generowane przez adapter.

BUDOWA ADAPTERA

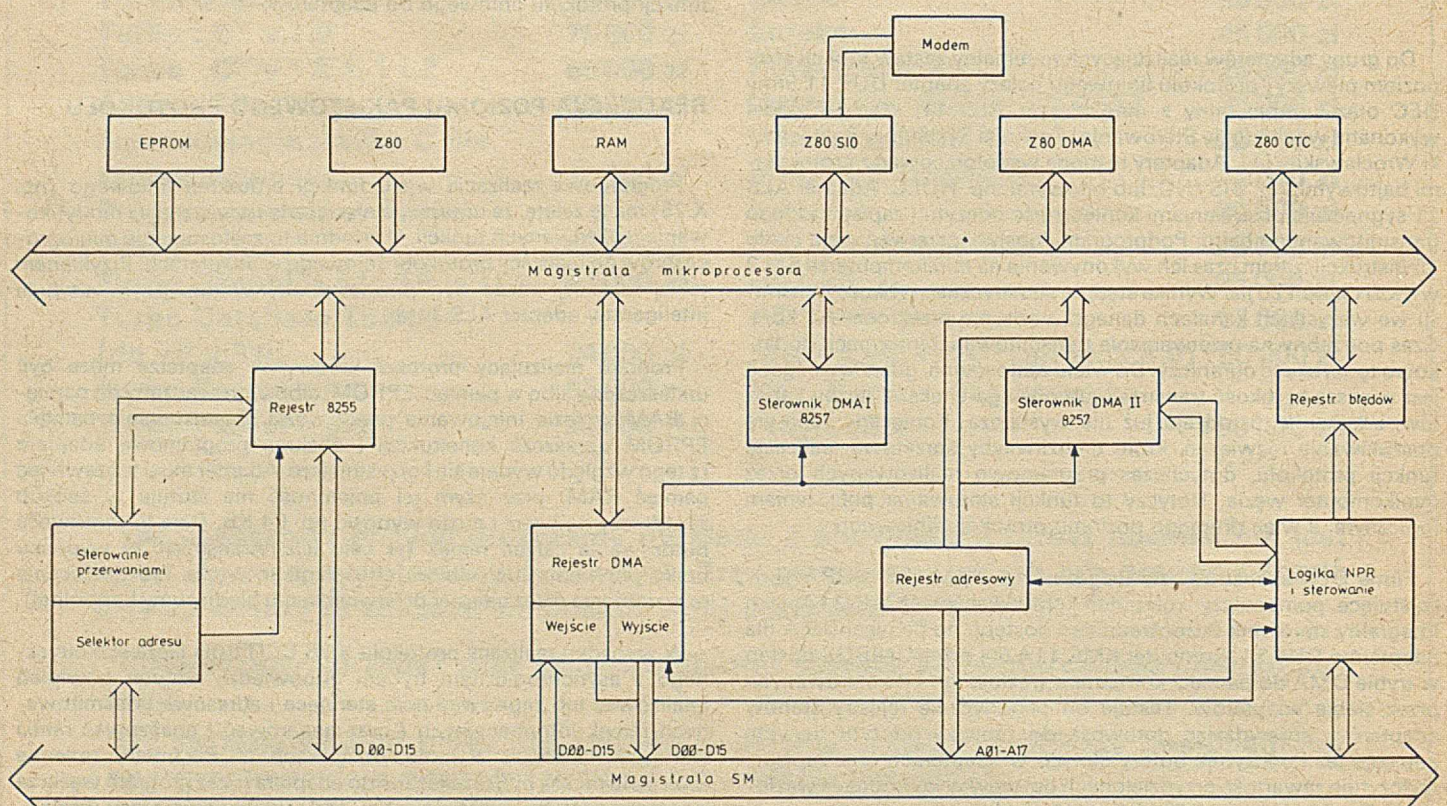
Inteligentny adapter liniowy ALS 11μP zrealizowany w technice mikroprocesorowej przedstawiono na rysunku. Podstawowe funkcje konwersji szeregowo-równoległej, zabezpieczenia kodowego i współpracy z modemami są realizowane przez układ *Z80-SIO*, sterowany mikroprocesorem Z80. Przepływ danych pomiędzy *Z80-SIO* a pamięcią RAM odbywa się w trybie DMA za pomocą sterownika DMA firmy Zilog. Wymiana danych między pamięcią RAM a pamięcią minikomputera SM odbywa się przez rejestry pośrednie adaptera i jest nadzorowana przez dwa sterowniki 8257 firmy Intel. Pierwszy z nich steruje transmisją między pracującymi w trybie porozumienia (ang. *handshaking*) rejestrami pośrednimi adaptera z pamięcią RAM. Dla każdego kierunku transmisji przewidziano odrębne rejestry. Dane wyjściowe z pamięci adaptera są w dwóch cyklach DMA wprowadzane do 16-bitowego rejestru wyjściowego, opartego na dwóch układach 8212. Podobnie dane wejściowe zawarte w 16-bitowym rejestrze wejściowym DMA są w dwóch cyklach DMA wpisywane do pamięci RAM. Zadaniem drugiego sterownika 8257 jest przepisywanie 16-bitowych słów danych z pamięci minikomputera SM do wejściowych rejestrów DMA adaptera oraz przepisywanie danych z wyjściowych rejestrów DMA do pamięci minikomputera SM. Transmisja ta odbywa się w trybie DMA. W tym celu sterownik ten współpracuje z układami przerwań NPR i Master Control typowymi dla systemu SM.

Magistrala adresowa systemu SM zawiera 18 linii adresowych *A00-A17*. Linia *A00* nie jest wykorzystywana, ponieważ założono transmisję 16-bitowych słów. Kolejnych 16 bitów *A01-A16* jest podawane ze sterownika DMA 8257, natomiast bit *A17* z oddzielnego rejestru . Sterownik nadzoruje cykle DMA między minikomputerem SM a rejestrami pośrednimi DMA i nie ma bezpośredniego wpływu na pracę mikroprocesora Z80. Aby możliwe było jednoczesne używanie szyn danych oraz szyn adresowych *A0-A3* przez mikroprocesor i sterownik, zastosowano układy separujące 8226. Mikroprocesor ma możliwość obserwacji linii *HLDA* drugiego sterownika (informującej, czy aktualnie trwa cykl NPR) oraz możliwość zablokowania żądania cyklu NPR. Pozwala to uniknąć konfliktu na liniach danych i liniach adresowych *A0-A3* podczas zapisu lub odczytu wewnętrznych rejestrów drugiego sterownika 8257.

Informacje sterujące między minikomputerem SM a adapterem są przekazywane w trybie przerwań wektorowych za pośrednictwem rejestrów pośrednich, zbudowanych z czterech układów sprzęgających Intel 8255. Dwa z nich są zaprogramowane (w trybie 1) jako wejściowe, a pozostałe dwa jako wyjściowe. Układy te współpracują z typowym dla systemu SM selektorem adresów i układem przerwań *BR*.

. . .

Zastosowanie inteligentnych adapterów w węźle sieci w znacznym stopniu upraszcza system operacyjny węzła i umożliwia uzyskanie dużych szybkości transmisji danych w dołączonych kanałach. Przedstawiony w tym artykule inteligentny adapter może obsługiwać jednocześnie dwie linie dupleksowe.



Schemat funkcjonalny inteligentnego adaptera ALS 11μP

Adapter został tak zbudowany, że jego dwa fizyczne rejestry wejściowe i wyjściowe tworzą osiem wirtualnych rejestrów wejściowych i wyjściowych. Uzyskano w ten sposób prostszą budowę urządzenia, przy zachowaniu łatwej obsługi programowej przez węzeł minikomputera SM. Konstrukcja adaptera jest tak pomyślana, że właściwie jest on uniwersalnym urządzeniem sprzęgającym między wspólną szyną systemu SM i magistralą systemu mikroprocesorowego. Urządzenie to pracuje jako adapter dzięki wyposażeniu go w układ o dużym stopniu scalenia Z80-SIO i odpowiedni program w pamięci EPROM.

Przedstawiony inteligentny adapter ALS 11 μP realizuje protokół X.25 CCITT, choć jego struktura umożliwia realizację różnych protokołów liniowych w zależności od programu umieszczonego w pamięci EPROM. Dalsze rozszerzanie funkcji protokołu liniowego realizowanych przez adapter liniowy doprowadziłoby do tego, że stałby się on węzłem sieci, co wydaje się niecelowe.

LITERATURA

[1] Batycki T. i in.: Adapter liniowy synchroniczny ALS 11. Raport SPR 3/84, ISTS, Politechnika Wrocławska, 1984

[2] Batycki T. i in.: Adapter liniowy synchroniczny ALS 11M. Raport SPR 3/86, ISTS, Politechnika Wrocławska, 1986

[3] CCITT Yellow Book, Vol. VIII, Rec X.25, Geneva, 1980

[4] Klingman E.E.: Projektowanie systemów mikroprocesorowych, WNT, Warszawa, 1982.

Wychodząc naprzeciw zapotrzebowaniu środowiska redakcja nasza oferuje łamy INFORMATYKI na bezpłatne zamieszczanie anonsów o imprezach informatycznych organizowanych na terenie kraju.

Zwracamy się do organizatorów konferencji, wystaw, konkursów i sympozjów z prośbą o przysyłanie do redakcji krótkich (pół strony maszynopisu) informacji o planowanych imprezach. Teksty powinny zawierać datę i miejsce imprezy, krótką charakterystykę jej tematyki oraz adres organizatorów.

Oczekujemy na zgłoszenia.

NIE DAJEMY RECEPT – SPRZEDAJEMY NARZĘDZIA

ELEKTRONIKA FILM KOMPUTER
Agencja Usługowo-Handlowa
Spółdzielni Pracy UNICUM

ul. Barska 3/20, 02–315 Warszawa
tel. 23-67-57, teleks 816955 efcom pl

BIURO HANDLOWE: ul. Dzika 4
Warszawa
tel. 635-23-36, 635-23-37
teleks 816075 medi pl

OFERUJE USŁUGI W DZIEDZINIE KOMPUTERYZACJI PRZEDSIĘBIORSTW

PODEJMUJE SIĘ KOMPLEKSOWEJ OBSŁUGI KONTRAHENTÓW:

- sprzedaż sprzętu mikrokomputerowego (kompletacja, dostawa, serwis gwarancyjny i pogwarancyjny)
- opracowanie oprogramowania użytkowego (wdrożenie, szkolenia)
- instalacja systemów wielodostępnych (SCO XENIX 286, 386) oraz systemów sieciowych (Novell) – licencyjnych

Służymy Państwu:

- doradztwem organizacyjnym
- projektowaniem, oprogramowaniem oraz wdrażaniem systemów dedykowanych konkretnemu użytkownikowi
- projektowaniem unikalnych programów wraz z nadzorem autorskim.

EO / 1299 / 88

Nowe kompilatory Ady dla mikroprocesorów

Pojawiające się obecnie na rynku kompilatory Ady są przeznaczone głównie dla komputerów VAX i generują kod na różne mikroprocesory stosowane w systemach wbudowanych.

SofTech Ada-86

Firma SofTech stworzyła kompletny zestaw narzędzi, o nazwie Ada-86, do wytwarzania w Adzie oprogramowania generującego kod dla rodziny mikroprocesorów 16-bitowych firmy Intel (8086, 186 i 286). Działają one na komputerach VAX z systemem operacyjnym VMS i wraz z systemem wykonawczym stanowią środowisko do tworzenia programów aplikacyjnych czasu rzeczywistego dla systemów wbudowanych.

Komputerem macierzystym dla kompilatora Ada-86 może być dowolny komputer z rodziny VAX (seria 780, 8000 lub MicroVAX) pracujący pod kontrolą systemu VMS. Biblioteka czasu wykonania dla wytwarzanego kodu zastępuje system operacyjny. Udostępnia funkcje zarządzania pamięcią, obsługi przerwań, wielozadaniowości, obsługi wejścia–wyjścia i inne, realizujące semantykę Ady. Specjalne pragmy do szybkiej obsługi przerwań gwarantują czas reakcji odpowiedni do pracy w czasie rzeczywistym. Moduły wynikowe wytwarzane przez eksporter mają format stosowany przez firmę Intel, są więc zgodne z systemami uruchomieniowymi tej firmy, programatorami pamięci PROM i programami usługowymi, włącznie z symbolicznym programem uruchomieniowym.

W skład systemu Ada-86 wchodzą:
– skrośny kompilator Ady,
– biblioteka czasu wykonania, niezbędna do wykonywania programów w Adzie na mikroprocesorach Intela (zawierająca opcje minimalną i pełną),
– konsolidator wiążący rozłącznie skompilowane fragmenty programu,
– importer przekształcający kod wytworzony przez asembler Intela na postać zgodną z konsolidatorem,
– eksporter wytwarzający moduły wynikowe,
– zarządca biblioteki (ekranowy lub rozkazowy) wspomagający przechowywanie kodu wynikowego podczas całego procesu wytwarzania oprogramowania.

Ada dla Intela 386

Pakiet kompilacji skrośnej o nazwie Ada-386, sprzedawany przez firmę Intel, jest przeznaczony dla mikroprocesorów 80386, używanych szczególnie w tzw. zastosowaniach krytycznych. Został on opracowany wspólnie z firmą TeleSoft, przy użyciu jej kompilatora TeleGen 2 i jest napisany całkowicie w Adzie. Działa na komputerach VAX z systemem operacyjnym VMS i umożliwia wytwarzanie oprogramowania dla systemów czasu rzeczywistego wprost z terminala VAX-a. Pakiet korzysta ze wszystkich zasobów tego komputera, umożliwiając użycie go także jako maszyny docelowej, co znacznie ułatwia wczesne testowanie jednostek programowych Ady, przed wytworzeniem kodu dla mikroprocesora 80386.

Oprócz kompilatora generującego kod dla dziewiczego mikroprocesora, w skład pakietu wchodzą następujące narzędzia:
– symboliczny program uruchomieniowy, ułatwiający uruchomianie na poziomie źródłowym kodu wykonywanego na komputerze docelowym,
– globalny optymalizator, zmniejszający rozmiar generowanego kodu i zwiększający szybkość jego wykonania,
– konsolidator łączący moduły skompilowane rozłącznie i biblioteki wykonawcze Ady w jeden wykonywalny obraz,
– importer dokonujący konwersji modułów wynikowych, w językach ASM, PL/M i C, na dołączalne moduły Ady, które można wywoływać za pomocą pragmy INTERFACE,
– system wykonawczy, konfigurowalny dla różnych środowisk sprzętowych,
– inne, jak analizator odwołań zewnętrznych (ang. *cross referencer*) i formater programów źródłowych.
Kompilator umożliwia, oczywiście, specyfikowanie reprezentacji i wplatanie kodu maszynowego, a także generowanie kodu dla koprocesora arytmetycznego 80387, zgodnego z normą IEEE.

Ada dla transputera

Czołowy europejski producent kompilatorów Ady, firma Alsys, opracowuje kompilator dla transputera firmy Inmos. Pierwsza wersja będzie działać jako kompilator skrośny na komputerze VAX oraz kompilator macierzysty na procesorze IMS T800 zainstalowanym na karcie IBM PC, pracującej pod kontrolą systemu operacyjnego DOS. Projekt kompilatora oparto na badaniach wstępnych, przeprowadzonych łącznie przez firmy Alsys, Inmos i Meiko Ltd.

Choć prace mają być zakończone dopiero w czerwcu 1989 roku, to już obecnie można tworzyć oprogramowanie w Adzie dla transputerów, korzystając z innych kompilatorów Ady dostępnych na maszynach serii VAX. Opracowane w ten sposób programy użytkowe będzie można później skompilować ponownie, za pomocą kompilatora generującego kod dla transputera.

Ada w połączeniu z transputerem znacznie poszerzy możliwości tworzenia systemów wbudowanych do zastosowań w takich dziedzinach, jak lotnictwo, łączność, obronność, technika radarowa, sterowanie procesami, rozpoznawanie mowy, a także – w stacjach roboczych i superkomputerach. Kompilator Ady wzbogaci dotychczas stosowany zestaw narzędzi programowych dla transputerów, zawierający przede wszystkim translator języka Occam, ale także translatory Fortranu, Pascala i języka C.

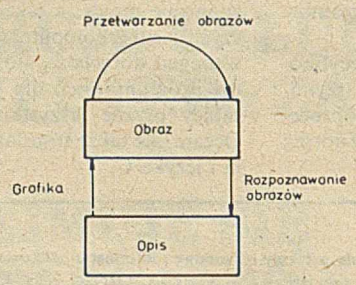
MK/JZ

Słownik pojęć i terminów z dziedziny rozpoznawania i przetwarzania obrazów (1)

Niniejszy słownik, opracowany na podstawie projektu normy IEEE P-610.4 *Glossary of Image Processing and Pattern Recognition Terms* oraz – artykułu *The Full Computing Reviews Classification Scheme, Computing Reviews, January 1983*, jest uzupełnieniem i rozszerzeniem *Słownika pojęć i terminów z dziedziny grafiki komputerowej*, opublikowanego w Informatyce, nr 2, 3, 4 i 8 z 1985 r. Część terminów używanych w rozpoznawaniu i przetwarzaniu obrazów odnosi się również do grafiki komputerowej i terminy te znajdują się we wspomnianym słowniku. Dlatego nie ma ich w niniejszym opracowaniu. Są również takie terminy, które w przetwarzaniu lub rozpoznawaniu obrazów mają nieco inne znaczenie, ich znaczenie uległo w ciągu kilku ostatnich lat pewnej zmianie, bądź są terminami bardzo często używanymi w tych dziedzinach. Takie terminy, mimo że były objaśniane we wspomnianym słowniku, zostały umieszczone również w niniejszym. Terminy objaśniane są pisane dużymi literami, a w nawiasach podano odpowiedniki angielskie.

Od Redakcji. Związki między trzema wymienionymi dziedzinami wyraził w bardzo ciekawy sposób Theo Pavlidis w przetłumaczonej na język polski książce „Grafika i przetwarzania obrazów – algorytmy", WNT, Warszawa, 1987 (rys. 1). Według niego:

- *grafika polega na tworzeniu obrazów na podstawie informacji nieobrazowej,*
- *przetwarzanie obrazów dotyczy zagadnień, w których dane wejściowe i wyjściowe mają postać obrazów,*
- *rozpoznawanie obrazów polega na tworzeniu opisu obrazu wejściowego lub zakwalifikowaniu go do jakiejś szczególnej klasy.*



Rys. 1. Związki między grafiką a rozpoznawaniem i przetwarzaniem obrazów (wg Pavlidisa)

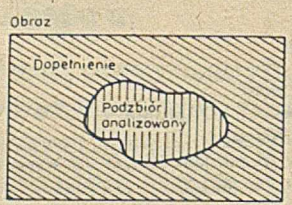
ANALIZA SKUPIEŃ (*cluster analysis*) – wykrywanie i opisywanie SKUPIEŃ w OBRAZIE CYFROWYM (PRZETWARZANIE OBRAZÓW), bądź w PRZESTRZENI CECH OBRAZU (ROZPOZNAWANIE OBRAZÓW).

ANALIZA OBRAZU (*image analysis*) – proces opisywania lub definiowania OBRAZU za pomocą jego składowych, właściwości i zależności.

BRZEG, OBRZEŻE (*border, boundary*) – zbiór tych DWELI OBSZARU na OBRAZIE, które przylegają do DWELI DOPEŁNIENIA tego OBSZARU (przec. WNĘTRZE).

CECHA (*feature*) – atrybut WZORCA, np. rozmiar, kształt, TEKSTURA, wykorzystywany w KLASYFIKOWANIU OBRAZÓW.

DOPEŁNIENIE (*complement*) – wszystkie te DWELE OBRAZU, które nie należą do danego podzbioru OBRAZU (rys. 2).



Rys. 2. Ilustracja dopełnienia

DWEL, GREL, PIKSEL (*pixel, pel, picture element, resolution cell*) – najmniejszy element OBRAZU CYFROWEGO, który może mieć przypisany POZIOM JASNOŚCI, różniący się od POZIOMÓW JASNOŚCI elementów sąsiednich. Dwie pierwsze nazwy są skrótami zwrotów *dwuwymiarowy element* i *graficzny element* a trzecia pochodzi od angielskiego określenia *picture element*.

DWELOCIĄG, GRELOCIĄG, PIKSELOCIĄG (*run*) – ciąg DWELI mających ten sam POZIOM JASNOŚCI.

DYSKRETYZACJA, DIGITALIZACJA (*discretization, digitization*) – proces przekształcania (ciągłego) OBRAZU na OBRAZ CYFROWY.

DZIURA (*hole*) – spójna składowa DOPEŁNIENIA OBSZARU otoczona przez ten OBSZAR (rys. 3).

Rys. 3. Ilustracja dziury

FILTRACJA SKRAJNOPOZIOMOWA (*high-emphasis filtering*) – technika WYOSTRZANIA OBRAZU, polegająca na intensyfikowaniu ekstremalnych różnic w POZIOMACH JASNOŚCI DWELI.

JASKRAWOŚĆ, LUMINANCJA (*brightness*) – wielkość związana z DWELEM, reprezentująca ilość światła padającego ze SCENY w określonym kierunku.

KLASA OBRAZU, KATEGORIA OBRAZU (*pattern class, pattern category*) – jedna ze zbioru wzajemnie wykluczających się grup, do których dany WZORZEC może być przydzielony w wyniku KLASYFIKOWANIA.

KLASYFIKOWANIE OBRAZÓW, IDENTYFIKOWANIE OBRAZÓW (*pattern classification, pattern identification*) – proces przydzielania WZORCÓW do określonych KLAS.

KLASYFIKATOR, REGUŁA DECYZYJNA (*classifier, decision rule*) – stosowany w KLASYFIKOWANIU OBRAZÓW przepis na przydzielanie obserwowanej jednostki OBRAZU do określonej KLASY WZORCÓW na podstawie CECH wydzielonych z OBRAZU.

KODOWANIE OBRAZU, KOMPRESJA OBRAZU (*image coding, image compression*) – proces eliminowania nadmiarowości lub aproksymowania OBRAZU w celu przedstawienia go w bardziej zwartej postaci.

KODOWANIE ADAPTACYJNE (*adaptive coding*) – zastosowanie dwóch lub kilku sposobów KODOWANIA do jednego OBRAZU, opartych na zależnościach między różnymi częściami OBRAZU.

KODOWANIE DWELOCIĄGOWE (*run length encoding*) – technika KODOWANIA OBRAZU, w której rzędy (paski poziome) OBRAZU są sekwencjami DWELOCIĄGÓW o określonych długościach i POZIOMACH JASNOŚCI.

KODOWANIE MIĘDZYRAMKOWE (*interframe coding*) – technika KODOWANIA OBRAZÓW, w której ciąg OBRAZÓW jest zagęszczany przez wykorzystanie nadmiarowości między kolejnymi OBRAZAMI.

KODOWANIE WYDAJNE (*efficient encoding*) – technika KODOWANIA OBRAZU, zapewniająca zwięzłą reprezentację POZIOMÓW JASNOŚCI z możliwością dokładnego ODTWORZENIA OBRAZU.

KODOWANIE Z WYPRZEDZENIEM (*predictive coding*) – technika KODOWANIA OBRAZÓW wykorzystująca POZIOMY JASNOŚCI poprzedzających DWELI do przewidzenia POZIOMU JASNOŚCI bieżącego DWELA, tak że wystarczy kodować jedynie różnicę między przewidzianą i zamierzoną wartością JASNOŚCI.

KONTRAST (*contrast*) – różnica między przeciętną JASKRAWOŚCIĄ dwóch podzbiorów OBRAZU.

KONTUROWANIE, KODOWANIE KONTUROWE (*contour encoding*) – technika KODOWANIA OBRAZU, w której OBSZAR mający stały POZIOM JASNOŚCI jest kodowany za pomocą BRZEGU tego OBSZARU.

KOREKCJA GEOMETRYCZNA (*geometric correction*) – technika ODNOWIENIA OBRAZU, w której wykonuje się przekształcenia geometryczne OBRAZU w celu skompensowania jego geometrycznych zniekształceń.

KRAWĘDŹ (*edge*) – zbiór DWELI na KRZYWEJ mających taką właściwość, że DWELE w ich SĄSIEDZTWIE, lecz po przeciwnych stronach KRZYWEJ mają różne POZIOMY JASNOŚCI.

KRZYWA (*curve*) – ślad znaczony przez punkt poruszający się w przestrzeni lub – skończony zbiór DWELI reprezentujący ten ślad.

KRZYWA ZAMKNIĘTA (*closed curve*) – KRZYWA, która przecina siebie dokładnie raz w ten sposób, że punkt początkowy KRZYWEJ jest równocześnie punktem końcowym KRZYWEJ.

KWANTOWANIE (*quantization*) – proces, w którym każdy DWEL w OBRAZIE jest oznaczony jednym POZIOMEM JASNOŚCI ze skończonego zbioru tych POZIOMÓW.

KWANTOWANIE LINIOWE (*linear quantizing, equal interval quantizing*) – technika KWANTOWANIA, w której zakres POZIOMÓW JASNOŚCI w OBRAZIE jest dzielony na pewną liczbę przedziałów takiej samej długości, a poziom KWANTOWANIA przypisany każdemu DWELOWI jest taki sam dla wszystkich DWELI, których początkowe POZIOMY JASNOŚCI zawierają się w tym samym przedziale.

KWANTOWANIE RÓWNOPRAWDOPODOBNE (*equal probability quantizing*) – technika KWANTOWANIA, w której zakres POZIOMÓW JASNOŚCI w OBRAZIE jest dzielony na ciągłe przedziały, takie że częstotliwość wystąpienia każdego poziomu KWANTOWANIA jest taka sama.

ŁUK (*arc*) – nie przecinająca się część KRZYWEJ lub skończony zbiór DWELI reprezentujących tę część KRZYWEJ.

ŁUK ELEMENTARNY (*simple arc*) – nie przecinająca się część KRZYWEJ.

MANIPULOWANIE SKALĄ JASNOŚCI (*gray scale manipulation*) – technika WZMACNIANIA OBRAZU, w której wygląd OBRAZU jest poprawiany przez zastosowanie OPERATORA PUNKTOWEGO do każdego DWELA w OBRAZIE CYFROWYM.

MASKOWANIE NIEOSTROŚCI (*unsharp masking*) – technika WYOSTRZANIA OBRAZU, w której celowo rozmazana wersja OBRAZU jest odejmowana od tego OBRAZU.

OBIEKT SCENY (*scene object*) – zbiór TRELI mających takie same CECHY.

OBRAZ (1) (*image, picture*) – dwuwymiarowa reprezentacja SCENY.

OBRAZ (2), (*pattern*) → WZORZEC.

OBRAZ DWÓJKOWY (*binary image*) – OBRAZ CYFROWY, w którym każdy DWEL przybiera wartość 0 lub 1.

OBRAZ CYFROWY (*digital image, digitized image*) – OBRAZ powstały wskutek przekształcenia OBRAZU ciągłego na tablicę DWELI, którym przypisano wektory CECH, takich jak POZIOM JASNOŚCI, barwa, nasycenie lub JASKRAWOŚĆ.

OBRAZ KRAWĘDZIOWY (*edge image*) – OBRAZ CYFROWY, w którym każdy DWEL jest oznaczony jako krawędziowy bądź jako niekrawędziowy.

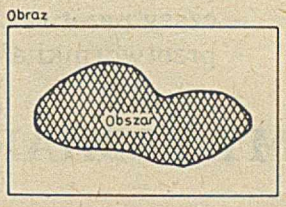
OBRAZ OPTYCZNY (*optical image*) – wynik rzutowania SCENY na powierzchnię, np. OBRAZ SCENY utworzony na filmie za pomocą soczewki kamery.

OBRAZ SYMBOLICZNY (*symbolic image*) – OBRAZ CYFROWY, w którym wartość związana z każdym DWELEM jest pewnym symbolem a nie POZIOMEM JASNOŚCI.

OBRAZ WIELOWIDMOWY (*multi-band image*) – zbiór OBRAZÓW jednej SCENY utworzonych za pomocą techniki radiacyjnej, w różnych zakresach widma.

OBRAZY ZAREJESTROWANE (*registered images*) – dwa lub więcej OBRAZÓW tej samej SCENY, umiejscowionych względem siebie w ten sposób, że odpowiednie punkty na OBRAZACH reprezentują ten sam punkt na SCENIE.

OBSZAR (*region*) – SPÓJNY podzbiór OBRAZU (rys. 4).



Rys. 4. Ilustracja obszaru

OBWÓD (*perimeter*) – liczba DWELI w BRZEGU OBSZARU.

ODNOWIENIE OBRAZU (*image restoration*) – proces przywodzenia OBRAZU do jego pierwotnego stanu przez odwrócenie skutków znanych lub ocenionych uproszczeń.

OPERATOR KRAWĘDZIOWY (*edge operator*) – OPERATOR SĄSIEDZTWA określający, które DWELE w OBRAZIE są DWELAMI krawędziowymi.

OPERATOR OBRAZOWY (*image operator*, image transform, image transform operator) – funkcja odwzorowująca OBRAZ wejściowy na OBRAZ wyjściowy.

OPERATOR PUNKTOWY (*point operator*) – OPERATOR OBRAZOWY, który przypisuje POZIOM JASNOŚCI każdemu DWELOWI wyjściowemu na podstawie POZIOMU JASNOŚCI odpowiadającego mu DWELA wejściowego (przec. OPERATOR SĄSIEDZTWA).

OPERATOR SĄSIEDZTWA (*neighborhood operator*) – OPERATOR OBRAZOWY, który przypisuje POZIOM JASNOŚCI każdemu DWELOWI wyjściowemu na podstawie POZIOMÓW JASNOŚCI DWELI sąsiadujących z odpowiednim DWELEM wejściowym (przec. OPERATOR PUNKTOWY).

OSTRY (*sharp*) – dotyczący elementów OBRAZU, które są dobrze zdefiniowane i łatwo rozróżnialne (przec. ROZMAZANY).

WOJCIECH MOKRZYCKI

Iszkowski W.: Przyspieszenie obliczeń przez ich rozpraszanie (1). Rozpraszanie sprzętowe

INFORMATYKA 1989, nr 5, s. 1

Pierwsza część przeglądu obecnie stosowanych metod zwiększenia szybkości działania komputerów, zawierająca omówienie rozwiązań sprzętowych.

Iszkowski W.: Acceleration of computations through scattering (1). Hardware scattering

INFORMATYKA 1989, No. 5, p. 1

First part of survey of the to-day used methods for computer speed increasing, which includes discussion of hardware solutions.

Iszkowski W.: Beschleunigung von Berechnungen durch Zerstreung (1). Hardwarezerstreung

INFORMATYKA 1989, Nr. 5, S. 1

Erster Teil eines Übersichts von heutzutage angewendeten Methoden für Computergeschwindigkeitssteigerung, der eine Besprechung von Hardwarelösungen umfasst.

Simon H.K.: Procesy przeszukiwania i wnioskowania w rozwiązywaniu problemów (3)

INFORMATYKA 1989, nr 5, s. 5

Dokończenie artykułu z dziedziny badań nad sztuczną inteligencją, zawierające omówienie programów transformacji problemu postawionego w języku naturalnym na jego reprezentację formalną.

Simon H.K.: Search and reasoning in problem solving (3)

INFORMATYKA 1989, No. 5, p. 5

Termination of the paper from the field of AI research, which includes discussion of programs for transformation of in natural language formulated problems to its formal representation.

Simon H.K.: Suchungs- und Schlussfolgerungsprozesse bei Lösung von Problemen (3)

INFORMATYKA 1989, Nr. 5, S. 5

Beendigung des Artikels aus dem Bereich der künstlichen Intelligenz, die eine Besprechung der Programme für Umwandlung des in Natursprache gestellten Problems zu seiner Formaldarstellung umfasst.

Deminet J.: Badanie przepustowości sieci lokalnych

INFORMATYKA 1989, nr 5, s. 8

Wyniki przeprowadzonych w Polsce badań przepustowości lokalnych sieci mikrokomputerowych Transnet, Ethernet i Arcnet.

Deminet J.: Research on local network capacity

INFORMATYKA 1989, No. 5, p. 8

Results of the Polish research on capacity of Transnet, Ethernet and Arcnet local networks.

Deminet J.: Durchsatzleistung der lokalen Netze

INFORMATYKA 1989, Nr. 5, S. 8

Ergebnisse der in Polen durchgeführten Forschung über Durchsatzleistung von Transnet-, Ethernet- und Arcnet--Lokalnetzen.

Kaczmarczyk K.: Charakterystyka modułu procesora mikrokomputerów rodziny Mera 600

INFORMATYKA 1989, nr 5, s. 11

Szczegółowa charakterystyka rozwiązań mikroprocesora M6, stosowanego przez zakłady MERASTER w Katowicach do budowy mikrokomputerów rodziny Mera 600.

Kaczmarczyk K.: Characteristics of the microprocessor module for Mera 600 microcomputer family

INFORMATYKA 1989, No. 5, p. 11

Detailed characteristics of solutions of the M6 microprocessor, which is used by MERASTER-Works in Katowice for building microcomputers of the Mera 600 family.

Kaczmarczyk K.: Charakteristik des Prozessormoduls für Mera 600-Mikrorechnerfamilie

INFORMATYKA 1989, Nr. 5, S. 11

Eine detaillierte Charakteristik der Lösungen von M6--Mikroprozessor, der von MERASTER-Werke in Katowice für Mikrorechner der Mera 600-Familie angewendet wird.

Fuglewicz P., Korniak T.: Oprogramowanie baz danych dla systemów mikrokomputerowych rodziny Mera 600

INFORMATYKA 1989, nr 5, s. 15

Charakterystyka rozwiązań organizacyjnych i programowych trzech różnych, opracowanych dla mikrokomputerów rodziny Mera 600, systemów baz danych.

Fuglewicz P., Korniak T.: Data base software for microcomputers of the Mera 600 family

INFORMATYKA 1989, No. 5, p. 15

Characteristics of organizing and program solutions of the three different, for Mera 600 microcomputer family elaborated data base systems.

Fuglewicz P., Korniak T.: Datenbanksoftware für Mera 600-Mikrorechnerfamilie

INFORMATYKA 1989, Nr. 5, S. 15

Eine Charakteristik von Organisations- und Programmlösungen der drei verschiedenen und für Mera 600--Mikrorechnerfamilie erarbeiteten Datenbanksysteme.

Skrzymowski J.: Nowa norma Fortranu (1)

INFORMATYKA 1989, nr 5, s. 18

Pierwsza część charakterystyki języka Fortran 88, zawierająca omówienie podstawowych rozszerzeń tej wersji w porównaniu do normy Fortran 77.

Skrzymowski J.: New standard of Fortran (1)

INFORMATYKA 1989, No. 5, p. 18

First part of characteristics of the Fortran 88 language, which includes discussion of this version basic extensions in comparison with the Fortran 77 standard.

Skrzymowski J.: Neue Fortran-Norm (1)

INFORMATYKA 1989, Nr. 5, S. 18

Erster Teil einer Charakteristik von Fortran 88-Sprache, der eine Besprechung von grundlegenden Erweiterungen dieser Version im Vergleich zu Fortran 77-Norm umfasst.

Nowiński W.: Algorytmy rekonstruowania obrazów metodami rozwinięcia w szereg

INFORMATYKA 1989, nr 5, s. 21

Przegląd różnych metod rekonstruowania obrazów, opartych na rozwinięciu w szereg, oraz podstawy teoretyczne tych metod.

Nowiński W.: Algorithms for image reconstruction using series expansion methods

INFORMATYKA 1989, No. 5, p. 21

Survey of different, on series expansion based methods for image reconstruction and theoretic principles of these methods.

Nowiński W.: Algorithmen für Bildrekonstruktion mit Anwendung von Reihenentwiclungsmethoden

INFORMATYKA 1989, Nr. 5, S. 21

Ein Übersicht von verschiedenen, auf Reihenentwicklung basierten Methoden für Bildrekonstruktion und theoretische Grundlagen für diese Methoden.

Kandziora-Blauman K., Kisilewicz J.: Mikroprocesorowe realizacje protokołu liniowego w systemie SM

INFORMATYKA 1989, nr 5, s. 25

Charakterystyka rozwiązań inteligentnych adapterów sieci, usprawniających działanie i ułatwiających obsługę węzła minikomputera SM.

Kandziora-Blauman K., Kisilewicz J.: Microprocessor realization of the line protocol to SM-system

INFORMATYKA 1989, No. 5, p. 25

Characteristics of intelligent network adapter solutions, which improve work and simplifies operating of SM minicomputer node.

Kandziora-Blauman K., Kisilewicz J.: Mikroprozessorrealisation des Linearprotokolls im SM-System

INFORMATYKA 1989, Nr 5, S. 25

Eine Charakteristik von Lösungen der intelligenten Nenzadaptere, die Arbiet und Bedienung von SM-Minicomputerknoten rationalisieren und erleichtern.

Kolorowa drukarka laserowa

Firma Colores z Norcross (stan Georgia) wprowadziła do swej kopiarki kolorowej podzespół drukujący, co pozwoliło na uzyskanie kolorowego wydruku laserowego w ciągu jednego przesuwu papieru. Dotychczas drukarki tego rodzaju dawały obrazy jednobarwne. Próby zaprojektowania laserowej drukarki kolorowej polegały na przesyłaniu papieru przez kilka stanowisk drukujących w różnych kolorach, co znacznie zwiększało złożoność mechaniczną urządzenia i koszt.

W nowym rozwiązaniu wykorzystuje się pas fotoprzewodzący, na którym wytwarza się obraz elektrostatyczny, który następnie przesuwa się pod modułami tonerów, naelektryzowanych proszków o różnych barwach, odpowiednio przyciąganych lub odpychanych przez poszczególne elementy obrazu. Moduły tonerów są identyczne i umieszczone obok siebie. Obrazy w poszczególnych kolorach są nakładane na pas. Złożony obraz jest drukowany jednofazowo, przy czym papier przechodzi przez stanowisko oczyszczające, gdzie usuwa się resztki tonerów. Dobry wydruk wymaga prostoliniowego przesuwu na drodze około 45 cm. W końcowej fazie absorbowane cząstki tonera są podgrzewane w utrwalaczu (ang. *fuser*) w celu trwałego złączenia z papierem. Drukarka oparta na tej nowej technice może drukować obrazy (teksty) czarno-białe z szybkością 45 stron na minutę, obrazy trójkolorowe – z szybkością 7,5 stron na minutę i czterokolorowe – z szybkością 6 stron na minutę. Pas fotoprzewodzący jest wykonany z firmowego materiału polimerowego, który zapewnia odpowiednie właściwości mechaniczne i elektryczne.

Oprogramowanie sieciowe 10-NET

Firma 10-NET Communications, która jest oddziałem Digital Communications Associates rozpoczęła sprzedaż nowego oprogramowania sieciowego, które może być wykorzystane w kartach IBM Token-Ring i PC Network. Jest ono związane z NETBIOS-em i dostosowane do protokołów określanych jako SMB (ang. *Server Message Block* – blok komunikatów usługodawcy). Programy usługowe obejmują pocztę elektroniczną, kalendarz, spooling, łączność między użytkownikami, rozgłaszanie wiadomości, zdalne kierowanie zadaniami. Stacje usługodawców mogą być wyspecjalizowane lub pełnić tę rolę obok swoich zadań; każdy komputer osobisty może też dzielić dane i urządzenia peryferyjne. Cena za węzeł wynosi 395 dolarów, a łącznie z dokumentacją oprogramowania, kablem i kartą sprzęgu – 695 dolarów.

Nowe modele PS/2

Firma IBM rozszerzyła ofertę mikrokomputerów PS/2 o siedem nowych modeli. Model 70-386 jest dostępny w trzech konfiguracjach. Najszybsza z nich, 70-A21, zawiera mikroprocesor 80386 z zegarem o częstotliwości 25 MHz, pamięć notatnikową o pojemności 64 KB wraz ze sterownikiem 82386, pamięć operacyjną o czasie dostępu 80 ns i pojemności 2 MB, którą można powiększyć do 8 MB i stację dyskową o pojemności 120 MB i czasie dostępu 23 ms. Następna konfiguracja, 70--121, ma zegar o częstotliwości 20 MHz, a pamięć operacyjną o czasie dostępu 85 ns i pojemności 2 MB, którą można rozszerzyć do 6 MB, oraz taką samą jak poprzednia konfigurację stację dyskową. Ostatnia konfiguracja, 70-E61, ma najniższą częstotliwość zegara – 16 MHz, pamięć operacyjną o czasie dostępu 100 ns i pojemności 1 MB, którą można rozszerzyć do 6 MB, oraz stację dysków o pojemności 60 MB i czasie dostępu 27 ms.

Wszystkie konfiguracje zawierają ponadto pamięć stałą o pojemności 128 KB, powodującą automatyczne testowanie podzespołów systemu po włączeniu urządzenia, stację dysków elastycznych o średnicy 3,5 cala i pojemności 1,44 MB, wbudowany sterownik, kartę graficzną VGA, magistralę Micro--Channel, port szeregowy i równoległy oraz trzy dodatkowe miejsca na pakiety. Ceny poszczególnych konfiguracji wynoszą odpowiednio 11 295, 7995 i 5995 dolarów.

Oprócz dotychczasowego modelu 50 pojawił się nowy Model 50Z, zawierający zegar o częstotliwości 10 MHz, pamięć operacyjną o czasie dostępu 85 ns i pojemności 1 MB, którą można rozszerzyć do 2 MB, pamięć stałą, stację dysków elastycznych, kartę VGA, trzy wolne miejsca dodatkowe i wbudowane sterowniki. Model ten sprzedawany jest jako 50-031 ze stacją dyskową o pojemności 30 MB i czasie dostępu 39 ms, za 3995 dolarów, i jako 50-061, ze stacją o pojemności 60 MB i czasie dostępu 27 ms – za 4595 dolarów.

Najtańszy model 25 to stanowisko robocze, zawierające zmodyfikowany adapter sieci Token-Ring. Zawiera on mikroprocesor 8086 z zegarem o częstotliwości 8 MHz, pamięć operacyjną o pojemności 640 KB na karcie systemowej, stację dysków elastycznych o średnicy 3,5 cala i pojemności 720 KB, pamięć stałą o pojemności 64 KB, kartę MCGA, porty szeregowy i równoległy, usprawnioną klawiaturę, port wskaźnikowy i złącze do urządzenia dźwiękowego. Model z monitorem monochromatycznym (25-L01) kosztuje 2139 dolarów, a z monitorem kolorowym (25-L04) – 2848 dolarów.

Francuskie mikrokomputery AT

Firma Getek, znana z opracowań kompatybilnych z wyrobami DEC, oferuje mikrokomputery GT 286 i GT 386. Pierwszy z nich, oparty na mikroprocesorze 80286 pracuje z częstotliwością zegara 6-12 MHz i ma pamięć operacyjną o pojemności 1 MB, którą można ośmiokrotnie powiększyć. W zależności od konfiguracji, pamięć masowa obejmuje stację dysków elastycznych o średnicy 3,5 cala i pojemności 1,2 MB oraz dysków Winchester o pojemności od 20 do 140 MB, oraz zabezpieczającą pamięć taśmową 60 lub 120 MB. Istnieją wersje komputera z ekranem monochromatycznym i kolorowym. GT 286 zawiera sterownik EGA/Hercules, klawiaturę 102-przyciskową i pakiet RS 232/Centronics.

Jeśli chodzi o GT 386, to istnieją dwie wersje konstrukcyjne: płaska i „wieżowa" (pionowe rozmieszczenie podzespołów). Procesor 32-bitowy może tu działać ze stanem oczekiwania lub bez tego stanu (odpowiednio przy częstotliwości 16 lub 21 MHz). Standardowa pamięć operacyjna ma pojemność 2 MB, a rozszerzenie do 8 MB można zrealizować bezpośrednio na pakiecie głównym. Pojemność dysku stałego może sięgać 320 MB.

Oba mikrokomputery mogą zawierać pakiet sieci Ethernet, a GT 286 również emulator systemu Tektronics 41XX, co znacznie rozszerza jego możliwości.

Nowa rodzina skomputeryzowanych stanowisk pracy

Przedsiębiorstwo Sun Mikrosystems Inc. z Montain View w Kalifornii oferuje nową serię stanowisk roboczych o nazwie Sun 386i, które mają 32-bitową jednostkę centralną Intel 80386, a także koprocesor arytmetyczny 80387, pamięć o pojemności 4 MB, sterownik sieci Ethernet, magistralę AT/XT z czterema miejscami na dodatkowe karty typu PC, sterownik sprzęgu SCSI (ang. *Small Computer System Interface*), porty RS--232 i Centronics, stację dysków elastycznych o średnicy 3,5 cala i pojemności 1,44 MB, dodatkową stację dysków stałych o pojemności 327 MB, klawiaturę i mysz. Częstotliwość zegara wynosi 20 lub 25 MHz. Mogą być wykorzystane monitory kolorowe o przekątnej 19 lub 16 cali i rozdzielczości 1152 × 900 elementów, lub 14-calowe o rozdzielczości 1024 × 768, a monochromatyczne o przekątnej 19 lub 15 cali i rozdzielczości 1152 × 990.

System Sun 386i DOS Window pozwala na jednoczesną pracę wielu zadań. Użytkownicy mogą wymieniać i łączyć dane i teksty zarówno w systemie DOS jak i Unix. Ponadto, stanowisko posiada udoskonaloną wersję firmowego systemu operacyjnego SUN OS. Ponad 75 innych firm opracowało już oprogramowanie użytkowe i urządzenia dostosowane do tego stanowiska. W zależności od wyboru monitora i dodatkowej stacji dysków cena jego waha się od 7990 do 13 990 dolarów.

Wystawa Comdex

Poprzednia wystawa COMDEX w Las Vegas pokazała, że w 1987 r. system IBM PS/2 był już dość rozpowszechniony, zarówno jeśli chodzi o sprzęt, jak i oprogramowanie. Wersja 1.0 systemu operacyjnego OS/2 zapowiedziana była na początek roku ubiegłego i to tylko za 325 dolarów. W tym samym czasie miały być też dostępne, przystosowane do PS/2 takie znane programy, jak 1-2-3, Symphony, SGBD, Paradox i Quatro.

Jeśli chodzi o PC, to dominowały tu systemy oparte na 80386 z częstotliwością zegara 20 MHz. Wśród mikrokomputerów przenośnych warto wymienić Amstrada PPC 640, który przy cenie poniżej tysiąca dolarów, zawiera mikroprocesor 8086, klawiaturę kompatybilną z AT, mały ekran ciekłokrystaliczny, dwa czytniki 3,5 calowych dysków elastycznych i modem kompatybilny z układami firmy Hayes. Systemy oparte na 80386, a także jeszcze 80286, demonstrowały firmy Samsung, NEC, Grid, Zenith i Canadian Ogivier. Posiadały one zwykle pamięć zewnętrzną o pojemności 10-40 MB i kosztowały 4--5 tys. dolarów.

Urządzenia peryferyjne bardzo się rozwinęły i oferowane są modele dostosowane do różnych rodzajów konfiguracji. Przebojem są wciąż drukarki laserowe, wśród których dominują firmy Oki, Ricoh i Canon ze znanymi już na ogół rozwiązaniami. Widać tu walkę o język, w jakim oprogramowane są drukarki. DEC, Quic, Qume, NEC i IMB stosują Postscript, opracowany przez firmę Adobe, a Hewlett-Packard stosuje swój własny język PCL. Szeroko eksponowane są 24-igłowe drukarki mozaikowe uderzeniowe (IBM, Okidata, Olimpia, PCPI itd.). Wśród drukarek kolorowych dominują termiczne z taśmą barwiącą i strumieniowe. Pojawienie się kolorowych drukarek laserowych było przewidywane na przełomie 1988 i 1989 roku.

Pokazano bogaty wybór pamięci magnetycznych. Niektóre z nich wykorzystują kasety dźwiękowe typu ECMA firmy Philips o pojemności 10-150 MB, inne – minikasety DC 1000 i 2000 firmy 3M (20-80 MB), a jeszcze inne wideokasety VHS (np. Digidata 2,5 GB) i kasety 8 mm (Exabyte 2,2 GB). Na połowę 1988 roku zapowiedziano tzw. pamięć DAT-MSS (*Digital Audio Tape – Mass Storage System*, dźwiękowa taśma cyfrowa – system pamięci masowej). Taka kaseta firmy Hitachi zawierająca 60 m taśmy będzie miała pojemność 1 GB.

Jeśli chodzi o dyski, to przy średnicy 5,25 cala firma Maxtor uzyskała pojemność 760 MB, Micropolis 1600 – 765 MB, a Siemens Magafile – 777 MB. Dyski o średnicy 3,5 cala osiągają pojemność 180 MB (Maxtor) i 200 MB (Control Data). Firma Seagate, ze swymi modelami ST 157 i 158 specjalizuje się w mniejszych pojemnościach, ale za to lżejszych i z mniejszym poborem mocy. Stacje firmy Miniscribe o symbolach 8425F i 8438F posiadają pojemność odpowiednio 21,4 i 32,7 MB i średni czas dostępu 40 ms.

Dyski optyczne są wytwarzane między innymi przez firmę ATG. Model 1002, kompatybilny z poprzednim 1001, ma bufor o pojemności 64 KB i charakteryzuje się średnim czasem dostępu do sektora 145 ms. Weryfikowany zapis może być na nim dokonywany z szybkością 175 KB/s. Gigadisc 2000 tej firmy, o średnicy 12 cali, ma pojemność 3,2 GB. Większość jednak znanych firm jak Hitachi, LMS Philips i Control Data, Optotech i Sony oferuje modele o średnicy 5,25 cala. Znana z produkcji aparatów fotograficznych firma Olimpus ma wprowadzić do sprzedaży dysk z wielokrotnym zapisem oparty na zjawisku magnetooptycznym ze zmianą fazy.