12 1983

# informatyka

Systemy mikrokomputerowe
iAPX 432
Procesy współbieżne

Spis treści rocznika 1983

WYDAWNICTWO SIGMA

# mikroKLAN

# Mikrokomputer nasz codzienny

**w następnym numerze**
**początek**
**potem co miesiąc**
**na środkowych**
**stronach**

**mikroinformatyka**
**— programy**
**— konstrukcje**
**— eksploatacja**
**— „rynek"**
**— kontakty, terminy...**
**— ludzie KLANU**

**dla**
**hobbystów**
**profesjonalistów**
**użytkowników**
**a**
**nawet**
**niegroźnych**
**fanatyków**

**forum koncepcji**
**poglądów**
**propozycji**
**postaw**

**nowa droga**
**informatyki**

**zapraszamy**

Domek, samochód, kolorowy telewizor. Synonimy życia na wysokim poziomie... Magiczne atrybuty pozwalające zapomnieć o miejscu i czasie, czuć się Europejczykiem. Wszystko wskazuje, że niebawem do kolekcji dołączy następny element: mikrokomputer. Skoro prawie każdy Brown i co drugi Schmidt używa go na co dzień, to czemu kilkudziesięciu Kowalskich ma się czuć gorzej przez to, że nie ma w domu mikrokomputera (przykrytego serwetką). Brown'owi komputer pomaga w planowaniu wydatków, uczy dzieci i pilnuje domu. Kowalski wydzwania po ogłoszeniach w poszukiwaniu gier, aby ostatecznie pogrążyć Malinowskich. U Kowalskiego komputer nie steruje ogrzewaniem szklarni. Trudno jest dostać zwyczajny zawór, cóż więc marzyć o elektrycznym. Nie prowadzi też buchalterii — posunięć władz podatkowych nie sposób przewidzieć.

Mikrokomputer to „szczytowe osiągnięcie myśli technicznej". Nikt więc nie zarzuci Kolwalskiemu, że żyje na zapyziałych antypodach cywilizowanego świata. Gdyby Brown wiedział jak relatywnie olbrzymią sumę utopił Kowalski w zabawce, pewnie umarłby ze śmiechu. Kilkudziesięcioletni byt wykształcił Kowalskiemu specyficzną świadomość — jak być Europejczykiem. I w taki właśnie sposób dotychczas stosowano u nas informatykę.

Trudno mi było opanować zdziwienie, gdy zadzwonił do mnie pewien „prywaciarz" z pytaniem, czy na ZX SPECTRUM można założyć bazę danych dla ewidencji klientów i materiałów. Sprawa się jednak szybko wyjaśniła: „prywaciarz" był absolwentem Politechniki. Nie jest on odosobniony. Frustracja młodej inteligencji, zmuszonej także sytuacją materialną do przejścia do sektora prywatnego, owocuje... otwarciem na techniczne nowości. A ci ludzie nie inwestują w kwiatek do kożucha. Nie dziwiłem się już widząc tłumy na pokazach klubu mikrokomputerowego ABAKUS.

Na początek mamy polski mikroprocesor (leciwy, ale jest!). Brak nam natomiast dobrych i tanich konstrukcji oraz oprogramowania użytkowego, na miarę krajowych potrzeb (nie tylko „prywaciarskich"). A ponad wszystko — brakuje nam swobodnego obiegu informacji. Tak aby nie wszyscy musieli startować od początku.

A więc do stycznia — mikroKLAN.

**ANDRZEJ J. PIOTROWSKI**
**red. prowadzący mikroKLAN**


WYDAWNICTWO SIGMA
NACZELNA ORGANIZACJA TECHNICZNA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004





**Redakcja: 00-941 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40, dyżury redakcji 10.00—12.00**

Zakł. Graf. „Tamka". Zam. 2319. Obj. 5,0 ark. druk. Nakład 3550 egz. M-78.

**Cena egzemplarza zł 75.—** **INDEKS 36124** **Prenumerata roczna zł 900.—**

# Informatyka

zastosowania w gospodarce, technice i nauce

Nr 12
MIESIĘCZNIK
1983
ROK XVIII
Grudzień




ORGAN KOMITETU INFORMATYKI MINISTERSTWA NAUKI, SZKOLNICTWA WYŻSZEGO I TECHNIKI ORAZ KOMITETU NAUKOWO-TECHNICZNEGO NOT DS. INFORMATYKI

**W NUMERZE:** **Strona**

ANDRZEJ SKORUPSKI
Instytut Informatyki
Politechnika Warszawska

# Rozwój systemów mikrokomputerowych

Szybki rozwój systemów mikrokomputerowych jest spowodowany wielką podażą elementów VLSI i odwrotnie — podaż układów rośnie wraz z możliwością coraz bardziej masowych zastosowań systemów. Technika mikroprocesorowa doprowadziła do zwiększenia gęstości upakowania elementów logicznych, zwiększenia niezawodności i zmniejszenia ceny urządzeń cyfrowych, a zatem umożliwiła także zastosowanie ich tam, gdzie do tej pory było to bądź niemożliwe, bądź nieopłacalne. Liczba wyprodukowanych systemów mikrokomputerowych dawno już przekroczyła milion i stale wzrasta. Zmieniają się właściwości systemów, obszary ich zastosowań, ale — co najważniejsze — zmienia się technologia ich wykonywania. Na pakietach o powierzchni 300 cm² mieszczą się komputery 16-bitowe wraz ze złączami urządzeń zewnętrznych. Technologia wykonania pakietów i połączeń staje się zagadnieniem coraz trudniejszym.

## KLASYFIKACJA UKŁADÓW VLSI

Układy VLSI można podzielić na dwie grupy:
— układy przeznaczone do zastosowań masowych (ang. custom logic), takich jak układy kalkulatorowe, gry telewizyjne i inne, procesory samochodowe, układy sterowania magnetofonami, pralkami itp.
— układy uniwersalne do projektowania urządzeń i systemów cyfrowych, a w szczególności systemów komputerowych.

Rozwój pierwszej grupy układów jest uzależniony od projektantów sprzętu powszechnego użytku, którzy chcą i potrafią zwiększyć jego atrakcyjność, polepszyć jego właściwości oraz ułatwić posługiwanie się nim.

W drugiej grupie układów można wyróżnić dwie klasy:
— układy ogólnego przeznaczenia
— układy do projektowania systemów komputerowych.

Do układów klasy pierwszej można zaliczyć pamięci, programowane tablice logiczne (ang. programmable logic array — PLA) oraz rodziny układów mikroprocesorów segmentowych (ang. slice microprocessor). Te ostanie służą głównie do budowy urządzeń wykorzystujących struktury mikroprogramowane.

Do układów klasy drugiej zalicza się rodziny układów mikroprocesorowych ze stałą listą rozkazów. Są to układy do projektowania centralnych procesorów, systemów pamięciowych oraz sterowników urządzeń zewnętrznych. Do tej klasy układów zalicza się także komputery jednoukładowe (ang. single chip microcomputer), które powstały wskutek scalenia kilku układów wymienionych wyżej. Rodziny układów mikroprocesorowych ze stałą listą rozkazów wzbudzają obecnie największe zainteresowanie projektantów.

## KLASYFIKACJA SYSTEMÓW MIKROKOMPUTEROWYCH

Pod względem konstrukcji i technologii wykonania systemy mikrokomputerowe można podzielić na:
— systemy jednopłytkowe (ang. single board computer — SBC)
— systemy modułowe.
Systemy jednopłytkowe są na ogół systemami o ograniczonych możliwościach, tj. wyposażone są w pamięć o stosunkowo niewielkiej pojemności i minimalny zestaw urządzeń zewnętrznych. Systemy modułowe mają najczęściej bogate oprogramowanie i możliwość rozbudowy.

Pod względem zastosowań systemy mikrokomputerowe można podzielić na:
— uniwersalne, jak np. komputery powszechnego użytku zwane komputerami osobistymi (ang. personal computer)
— „dedykowane" (ang. custom system) przeznaczone do określonych zastosowań.

Systemy uniwersalne są wyposażone w pamięć wewnętrzną o maksymalnie dużej pojemności oraz w jak najtańszy zestaw urządzeń zewnętrznych. Ponadto komputer taki jest wyposażony w oprogramowanie ułatwiające współpracę operatora z systemem. W skład oprogramowania wchodzą najczęściej popularne języki wysokiego poziomu, jak np. BASIC lub PASCAL. Przykładowym takim komputerem jest APPLE COMPUTER, którego cena (ok. 2000 dol.) ma umożliwić w tym roku sprzedaż ponad 1 mln egzemplarzy [1]. Jest on wyposażony w klawiaturę alfanumeryczną, monitor, pamięć zewnętrzną na minidyskach lub magnetofonie kasetowym powszechnego użytku oraz drukarkę dla tworzenia trwałych kopii (ang. hard copy).

Mikrokomputerowe systemy przeznaczone dla określonych użytkowników mogą być zarówno systemami „zamkniętymi" — przeznaczonymi dla jednego zastosowania, jak również systemami „otwartymi" — służącymi pracom badawczym zmierzającym do opracowania nowych systemów (ang. development system).

Najczęściej spotyka się następujące rodzaje systemów zamkniętych:
- systemy pomiarowe, jak np. systemy rejestracji i przetwarzania danych cyfrowych i analogowych, systemy testujące czy aparatura pomiarowa
- systemy sterowania — procesami technologicznymi, ruchem, maszynami, magazynami, rozdziałem energii itp.
- systemy telekomunikacyjne, np. centrale telefoniczne i teleskopowe, sieci telekomunikacyjne, nadajniki radiowe i telewizyjne
- systemy przetwarzania tekstów — redagowanie i składanie tekstów, gazety telewizyjne, systemy agencyjne, itp.
- systemy biurowe — systemy księgowania, sekretarskie, informacyjne, bankowe i inne
- systemy wspomagające sprzęt powszechnego użytku, np. kasy elektroniczne, wagi, kuchnie, gry i zabawy elektroniczne
- systemy cyfrowej rejestracji fonii i wizji, analizatory i syntezatory dźwięków i obrazów
- systemy protez — stymulatory serca, układy sterowania protezami kończyn i inne.

Systemy zamknięte są budowane w oparciu o różne rodzaje układów VLSI, w różnych technologiach najczęściej przy użyciu struktury zaprojektowanej dla danego zastosowania. Wynika to z faktu, że systemy te powiela się w ilościach uzasadniających opracowanie dla nich specjalnej konstrukcji oraz specjalnego oprogramowania.

Inaczej projektuje się systemy otwarte, gdzie zachowuje się możliwość wyboru i zmiany konfiguracji, a także dobudowania nowych zespołów funkcjonalnych. Są one wyposażone w pamięci wewnętrzne o możliwie największej pojemności oraz w urządzenia zewnętrzne ułatwiające opracowanie nowych systemów (programatory pamięci stałych, emulatory zespołów funkcjonalnych, itp.). Systemy takie wyposaża się ponadto w oprogramowanie narzędziowe, ułatwiające tworzenie oprogramowania innych systemów. Najczęściej systemy otwarte wykonuje się jako systemy modułowe.

Obecnie w kraju jesteśmy na etapie wdrażania do produkcji systemów modułowych. W pierwszych dwóch tegorocznych numerach INFORMATYKI opisano cztery takie systemy: RTOS, MSWP, MIKROSTER i MSM. Wiadomo

---

Życiorys dr. inż. Andrzeja Skorupskiego przedstawiliśmy w numerze 2/1983

przy tym, że istnieje więcej takich systemów, już zaprojektowanych. Jednocześnie, w artykule wstępnym skrytykowano rozproszenie prac w tej dziedzinie. Czy jednak unifikacja i standaryzacja na obecnym poziomie technologicznym w kraju przyspieszy rozwój systemów? Odpowiedź na to pytanie nie może być jednoznaczna. Poniżej przedstawiam pewne przesłanki, które mogą dopomóc w odpowiedzi na to pytanie.

## MODUŁOWE SYSTEMY MIKROKOMPUTEROWE

Modułowy system mikrokomputerowy składa się z następujących elementów:
- szyny standardowej (ang. bus) zwanej również magistralą
- procesora, ewentualnie z systemem przerwań i układem bezpośredniego dostępu do pamięci (DMA)
- bloków pamięci wewnętrznej (operacyjnej), tj. bloków pamięci oprogramowania (pamięci stałej — ROM) i bloków pamięci danych (RAM)
- sterowników urządzeń wejścia-wyjścia.

### Szyna standardowa

Najistotniejszym kryterium wyboru standardu szyny i podziału systemu na moduły jest przeznaczenie tego systemu, a pośrednio podział zadań pomiędzy sprzęt i oprogramowanie. Wynika z niego wiele innych kryteriów, jak np.: szybkość działania, rozdział dostępu do szyny pomiędzy moduły systemowe, docelowa pojemność pamięci wewnętrznej i docelowa liczba urządzeń zewnętrznych, łatwość projektowania modułów sprzętowych, a także oprogramowania systemu, możliwość rozbudowy systemu i łączenia systemów ze sobą, czy możliwości technologiczne.

Na świecie dają się zauważyć dwie tendencje. Jedna — zmierzania do standaryzacji szyn, a co za tym idzie — i modułów [3]. Druga — zmierzania do budowy systemów wykorzystujących kilka różnych standardów szyn, aby ewolucyjnie dochodzić do rozwiązań optymalnych — zwłaszcza w czasie, kiedy gwałtownie rozwija się produkcja układów VLSI, a także technologia wykonywania systemów [6, 7].

Oba podejścia mają swoje wady i zalety. Pierwsze z nich narzucają producenci sprzętu bardziej uniwersalnego, drugie zaś — wyspecjalizowane firmy produkujące sprzęt dedykowany, np aparaturę naukowo-badawczą czy inne urządzenia unikalne. W samej Europie istnieje obecnie ok. 140 różnych standardów szyn [7]. Ponieważ do tej pory w kraju nie dominuje żadne z zastosowań — przyjęcie jednego standardu wydaje się niecelowe lub przedwczesne. Wynika to z faktu, że żaden z opracowanych dotychczas systemów nie umożliwia optymalnego rozwiązania systemowego dla dowolnych zastosowań [7]. Wydaje się, że dalsza konkurencja pomiędzy ośrodkami krajowymi doprowadzi do ewolucyjnych zmian optymalizujących rozwiązania systemów dedykowanych. Natomiast przyczyną podobieństwa krajowych systemów modułowych jest krajowy poziom technologiczny i ograniczona baza elementowa.

### Procesory

Procesory krajowych systemów modułowych oparto na mikroprocesorze typu 8080 lub 8085 (jedynie system MSM ma wersje dla mikroprocesorów M6800 i Z-80). Są to zatem systemy 8-bitowe. W najbliższej przyszłości powstaną systemy 16-bitowe i znowu przy wyborze konstrukcji będą decydować możliwości technologiczne.

Na świecie obserwuje się coraz większą złożoność technologiczną konstrukcji, mimo coraz większego stopnia scalenia układów. Wynika to ze zwiększenia się długości przetwarzanego słowa, ale także z przejmowania przez sprzęt funkcji oprogramowania. Przykładowo — procesor iAPX 432 firmy INTEL wykonuje sprzętowo wiele funkcji wykonywanych dotąd przez oprogramowanie, tzn. zarządza procesami współbieżnymi, organizuje komunikację pomiędzy procesorami, a także adresuje $2^{40}$ bajtów pamięci wirtualnej za pomocą $2^{24}$ adresów fizycznych [4].

Pamięci wewnętrzne krajowych systemów modułowych zbudowane są na ogół jako pamięci półprzewodnikowe statyczne, przy użyciu układów o pojemności 1 K × 1 bit. Jedynie system MSM ma pamięć dynamiczną zbudowaną w oparciu o układy o pojemności 16 K × 1 bit. W przyszłości trzeba będzie opracować bloki szybkiej pamięci statycznej przy wykorzystaniu układów 8-bitowych, jak również bloki pamięci dynamicznej w oparciu o układy o pojemności 64 K × 1 bit. Współczesne tendencje światowe przedstawiono w tabeli [2].

W dziedzinie pamięci zewnętrznych dostępne są w kraju jedynie pamięci na 8-calowych dyskach elastycznych (prod. KFAP) oraz pamięci kasetowe (prod. MERAMAT). Wszystkie systemy krajowe są przystosowane do współpracy z jednostką dyskową SP45DE, a tylko system MIKROROSTER i MSM są przystosowane do pracy z pamięcią kasetową. Zaś jedynie system MSM wyposażono w sterownik pamięci na magnetofonie powszechnego użytku. Najbliższy etap w rozwoju tych systemów to opracowanie sterowników minidysków elastycznych (dyskietka 5-calowa) oraz sterowników dysków podwójnej gęstości (takie dyski są już produkowane w Bułgarii).

Współczesną tendencją światową w dziedzinie systemów mikrokomputerowych jest stosowanie dysków sztywnych typu Winchester [2]. Pamięci na takich dyskach 8-calowych mają już pojemności ok. 300 MB, a firma CENTURY DATA SYSTEM projektuje dysk z wymiennym nośnikiem. Natomiast mniejsze dyski 5-calowe mają pojemność od 20 do 50 MB.

### Sterowniki urządzeń

Urządzenia wejścia-wyjścia stosowane w systemach mikrokomputerowych można podzielić na cztery grupy:
— urządzenia z nośnikiem papierowym (czytnik taśmy dziurkowanej, perforator taśmy papierowej, drukarka)
— urządzenia operatorskie (klawiatury i monitor ekranowy)

**Współczesne tendencje rozwoju pamięci półprzewodnikowych**

| Rodzaj pamięci | Produkcja za 2—3 lata | | | | Produkcja za 3—10 lat | | | |
|---|---|---|---|---|---|---|---|---|
| | Długość słowa | | | | | | | |
| | 1 | 4 | 8 | 16 | 1 | 4 | 8 | 16 |
| Dynamiczna RAM | 256 K<br>64 K | 64 K | 64 K<br>256 K | — | 1 Mbit | 256 K<br>512 K | 256 K<br>512 K | — |
| MOS statyczna RAM | — | 64 K<br>128 K | 64 K<br>128 K | 64 K | — | — | 256 K | 128 K<br>256 K |
| Bipolarna statyczna RAM | 16 K | 16 K | — | — | 64 K | 64 K | — | — |
| Bipolarna PROM | — | 32 K | 32 K<br>64 K | — | — | — | 128 K<br>256 K | 64 K<br>128 K |
| MOS ROM | — | — | 256 K<br>1 M | — | — | — | 2 M<br>4 M | — |
| Ultrafiolet EPROM | — | — | 128 K<br>256 K | — | — | — | — | — |
| EPROM | — | — | 64 K | — | — | — | 128 K<br>256 K | — |
| Pamięci domenowe | 1—4 M | | | | 4—16 M | | | |

— urządzenia graficzne
— analogowe urządzenia wejścia-wyjścia lub inne urządzenia specjalne (wejścia binarne z obiektów, wejścia częstotliwościowe, wejścia i wyjścia głosowe).

Wszystkie krajowe systemy mogą współpracować z urządzeniami pierwszej grupy. Podobnie jest z urządzeniami drugiej grupy dostępnymi w kraju, jak monitory ekranowe MERA 7952 lub VIDEOTON. Istnieje także inne rozwiązanie (tańsze) polegające na tym, aby jako monitor wykorzystać telewizor powszechnego użytku. Zastosowano je jedynie w systemie MSM.

Urządzenia graficzne są w kraju praktycznie niedostępne i żaden z wymienionych systemów nie jest wyposażony w sterowniki takich urządzeń, ani też w oprogramowanie do przetwarzania graficznego.

Pod względem wyposażenia w urządzenia ostatniej grupy, należy wyróżnić system MIKROSTER. Zawiera on kilka modułów specjalizowanych do współpracy z takimi urządzeniami.

Współczesne tendencje światowe polegają na rozwoju nowych technik wprowadzania i wyprowadzania informacji. Oprócz bardzo nowoczesnych monitorów i drukarek graficznych są już dostępne urządzenia do wprowadzania informacji głosem oraz głosowe urządzenia wyjściowe [5]. Opracowuje się ponadto coraz nowsze drukarki (np. termiczne) o programowanych formatach znaków. Od dłuższego czasu trwają także poszukiwania optymalnego sposobu wprowadzania informacji z klawiatury. Polegają one na dążeniu do opracowania tanich i niezawodnych technologii, dzięki którym będzie można produkować dużą liczbę urządzeń.

* * *

Można zatem stwierdzić, że rozwój systemów mikrokomputerowych w kraju jest uzależniony od ogólnego poziomu technologicznego. Obecny stan opracowań i produkcji systemów zależy od bazy elementowej. Różnorodność opracowanych systemów umożliwi ukierunkowania ich zastosowań i rozpoczęcie produkcji systemów przeznaczonych do określonych zastosowań. Jednocześnie trzeba dążyć do koordynacji prac nad nie rozwiązanymi problemami, jak np. budowa systemów 16-bitowych lub systemów przeznaczonych dla masowych odbiorców.

LITERATURA


[1] Computer Design Pioneers. Computer Design, Vol. 21, No. 12, pp. 83 (1982)
[2] Eidsmore D.: Memory System Technology. Computer Design, Vol. 21, No. 12, pp. 133—156 (1982)
[3] Jankowski M. T.: KRABUS — sprzęg wewnątrzkasetowy dla systemów mikrokomputerowych. INFORMATYKA, nr 1/1983
[4] Killmon P., Asec J.: Processor Technology. Computer Design, Vol. 21, No. 12, pp. 101—132 (1982)
[5] Parker R.: Input/Output Technology. Computer Design, Vol. 21, No. 12, pp. 157—186 (1982)
[6] Tri-bus Minicomputer Advances Data Acqusition, Processing and Presentation Functions. Computer Design, Vol. 21, No. 12, pp. 49—52 (1982)
[7] Warren C.: A Mix of Standard and Proprietary Buses Marks the Latest μC Systems. Electronic Design, Vol. 31, No. 6, pp. 101—113 (17 March 1983)
[8] Waśniewska M. (oprac.): Pamięci dyskowe typu Winchester. INFORMATYKA, nr 6/1983.



**ANDRZEJ JACEK PIOTROWSKI (oprac.)**
**Warszawa**


# System iAPX 432

**Opracowany przez firmę INTEL system iAPX 432 stanowi przełom w technice mikroprocesorowej. Dotychczas architektura układów mikroprocesorowych umożliwiała stosowanie ich jedynie w mikrokomputerach i sterownikach. System iAPX 432 został zaprojektowany z myślą o wykorzystaniu w systemach komputerowych. Użytkowe właściwości systemu odzwierciedlają — jak twierdzi producent — najnowsze tendencje w organizacji oprogramowania i systemów operacyjnych.**

**System iAPX 432 wprowadzono na rynek w 1981 roku, trudno więc jeszcze o jego obiektywną ocenę. Można jednak oczekiwać, że pionierskie rozwiązania staną się źródłem standardów, podobnie jak to było w przypadku rodziny mikroprocesorów typu 8080. Należy podkreślić, że architekturę systemu iAPX 432 oparto na podstawach zbliżonych do tych, które stosują twórcy nowoczesnych języków programowania (ADA, CLU, CONCURRENT PASCAL OBJECT PROGRAMMING LANGUAGE).**

Poniżej zestawiono najbardziej spektakularne cechy systemu.

• iAPX 432 jest pierwszym systemem zapewniającym całkowicie przezroczystą dla oprogramowania pracę wielozadaniową. W skład systemu może wchodzić dowolna liczba procesorów, z których każdy może wykonywać niezależne zadania. Przydział zadań dla poszczególnych procesorów odbywa się automatycznie. Od liczby zastosowanych procesorów zależy moc obliczeniowa systemu. Rozwiązano w ten sposób często spotykany problem polegający na tym, że zazwyczaj po pewnym okresie eksploatacji systemu użytkownik uważa posiadany komputer za „zbyt mały". Dotychczas skazany był albo na ograniczenie swoich aspiracji, albo na zakup nowego systemu o większej mocy obliczeniowej, z czym wiązały się problemy adaptacji już posiadanego oprogramowania. W przypadku systemu iAPX 432 tego rodzaju zmiana wymaga jedynie dołączenia kolejnego modułu (modułów) procesora.

• System iAPX 432 ma zdolność sprzętowej detekcji błędów — począwszy np. od błędu dzielenia przez zero, a skończywszy na złożonych błędach wynikających ze współpracy wielu procedur. W przypadku wykrycia błędu (w tym także błędu wynikającego z uszkodzenia sprzętowego) realizacja zadania zostaje zawieszona. Moduł, w którym wykryto nieprawidłowość, realizuje procedurę diagnostyczną, wskutek czego powstaje szczegółowy opis błędu. Pozostała część systemu wieloprocesorowego może kontynuować pracę, jednak ze zmniejszoną mocą obliczeniową.

• Oprogramowanie użytkowe w systemie iAPX 432 wykonywane jest w językach wysokiego poziomu. W celu uproszczenia kompilatorów, lista rozkazów asemblera jest symetryczna, tzn. wszystkie rodzaje adresowania mogą być używane dla argumentu dowolnego typu, a istniejących operatorów można używać dla danych dowolnego typu. Zastosowano ustalony przez IEEE standard dla arytmetyki zmiennoprzecinkowej.

• Do dyspozycji programisty jest niemalże nieograniczony obszar adresowy. Uzyskano to przez zastosowanie pamięci wirtualnej o wielkości $2^{40}$ bajtów (!). Mechanizmy sprzętowe umożliwiają wykorzystanie takiego obszaru przy zastosowaniu pamięci fizycznej o pojemności $2^{24}$ bajtów (16 MB). Obszar pamięci dzielony jest na segmenty (ponad 16 mln) — tak, że praktycznie każda znacząca procedura

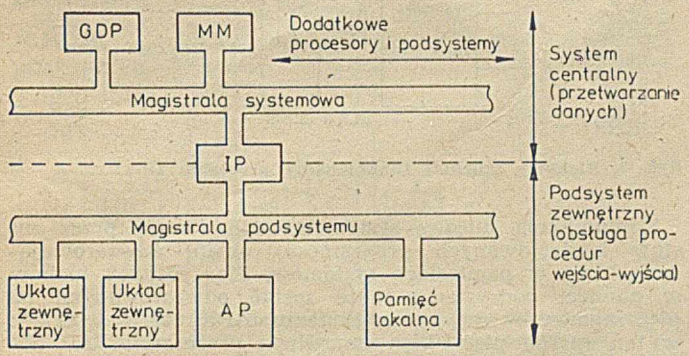
może mieć przydzielony własny segment pamięci. Umożliwia to zastosowanie sprzętowych mechanizmów ochrony plików.

• Zwiększone zadania stawiane przed oprogramowaniem narzucają jego specyficzną organizację. Programy dzieli się na bloki funkcjonalne. Ułatwia to z jednej strony łączenie w całość programów opracowywanych przez wiele osób, a z drugiej — umożliwia zastosowanie rozbudowanych systemów zabezpieczających dane i oprogramowanie przed zniszczeniem oraz dostępem przez nieuprawnionych użytkowników.

• System iAPX 432 realizuje sprzętowo funkcje wykonywane zazwyczaj przez systemy operacyjne:
— aktywacja i deaktywacja procesów
— alokacja zadań pomiędzy procesory
— dynamiczna alokacja pamięci
— komunikacja między procesami.

## ARCHITEKTURA SYSTEMU iAPX 432

W systemie wykorzystane są następujące zespoły:
GDP (General Data Processor) — zasadniczy procesor danych
IP (Interface Processor) — procesor sprzęgający
AP (Attached Processor) — dodatkowy procesor sterujący podsystemem wejścia-wyjścia
MM (Main Memory) — podstawowa pamięć systemu
MCU (Memory Control Unit) — układ kontrolny bloku pamięci
BIU (Bus Interface Unit) — układ kontrolny magistrali.




Rys. 1. Organizacja systemu iAPX 432

Na rysunku 1 przedstawiono konfigurację systemu iAPX 432. W celu zwiększenia możliwości systemu zastosowano oddzielne procesory: GDP do przetwarzania danych oraz IP do obsługi wprowadzania i wyprowadzania danych. GDP dekoduje program, dokonuje obliczeń i generuje adresy, a IP jest odpowiedzialny za komunikację z urządzeniami zewnętrznymi. Współpraca GDP, IP i pamięci (MM) odbywa się przez magistralę systemową.

IP jest równocześnie połączony z podsystemem zewnętrznym. Na podsystem ten przerzucono realizację wszelkich operacji związanych ze sterowaniem urządzeń zewnętrznych, obsługą przerwań i buforowaniem danych. Podsystem jest więc autonomicznym komputerem sterowanym przez procesor AP, z którym współpraca odbywa się za pośrednictwem IP. Procesor IP spełnia zarazem rolę bariery zabezpieczającej pamięć centralną przed nieautoryzowanym dostępem.

Funkcję AP spełnia konwencjonalny mikroprocesor (np. typu 8086), dobrany w zależności od wymagań nakładanych na podsystem zewnętrzny.

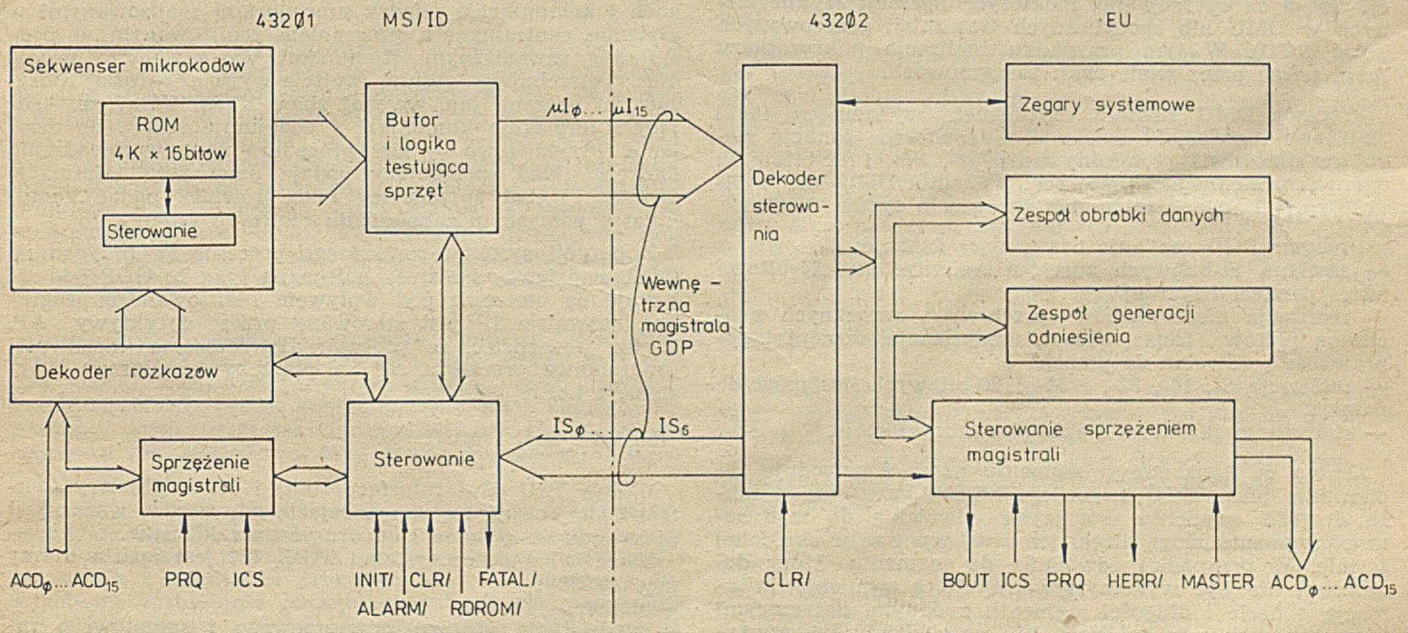
Jak wynika z rysunku 1, system iAPX 432 zawierać może dowolną liczbę układów GDP i IP. Architektura systemu umożliwia przetwarzanie wieloprocesorowe w sposób całkowicie przezroczysty dla oprogramowania, gdyż każdy z procesorów ma możliwości komunikowania się z pozostałymi w sposób automatyczny lub pod kontrolą programową. Ponadto zastosowano mechanizmy umożliwiające komunikację między programami realizowanymi na tym samym procesorze lub innych procesorach.

Trudno byłoby zdefiniować uniwersalną magistralę systemową dla systemu opartego na iAPX 432, która uwzględniałaby różne możliwe konfiguracje. Dlatego też określono jedynie standardowy protokół komunikacyjny między jednostkami GDP, IP i MM. Koncepcję protokołu oparto na przekazywaniu pakietu informacji, którego długość uzależniona jest od wykonywanej w systemie operacji. Umożliwia to minimalizację liczby linii magistrali i ich efektywne wykorzystanie. Konstruktorom systemów pozostawiono w ten sposób dużą swobodę w dopasowaniu magistrali systemowej do potrzeb konkretnej konstrukcji.

Układy MCU i BIU poszerzają możliwości systemu, szczególnie w zakresie odporności na uszkodzenia sprzętowe. Możliwe jest zlokalizowanie i wydzielenie uszkodzonych części systemu, jak również wyeliminowanie błędów wynikających z zaistniałego uszkodzenia.

## ARCHITEKTURA PROCESORA GDP

Procesor GDP składa się z dwóch układów VLSI, jednak funkcjonalnie można go podzielić na trzy człony:
ID (Instruction Decoder) — dekoder rozkazów




Rys. 2 Architektura procesora danych (GDP) systemu iAPX 432

MS (Microinstruction Sequencer) — generator mikrorozkazów
EU (Execution Unit) — człon realizacji programu.
Poszczególne człony można traktować jako niezależne procesory ułożone w kanał przetwarzający. Na rysunku 2 przedstawiono schemat blokowy GDP.

Jednostki ID i MS umieszczono w układzie typu 43201, a EU — w układzie typu 43202. Łączy je wewnętrzna magistrala procesora składająca się z linii $\mu_{15}$..., $ul_0$ oraz $IS_6$, ..., $IS_0$.

Do podstawowych zadań ID należy:
— odbiór dyrektyw programowych
— przetwarzanie informacji zawartej w rekordach programowych o różnej długości
— wydzielanie adresów logicznych
— generacja adresów początkowych dla sekwencji mikrorozkazów
— generacja mikrorozkazów dla prostych operacji.

Rozkaz w systemie iAPX 432 składa się z rekordów zawierających różne liczby bitów. ID określa długość rekordu i steruje pobraniem odpowiedniej liczby słów z pamięci, a następnie rozgranicza poszczególne pola rozkazu, zabezpieczając przechowanie niezbędnych informacji do chwili wykonania rozkazu.

W rozkazie może wystąpić odwołanie do konkretnej komórki pamięci. Adres logiczny musi wtedy zostać przesłany do zespołu generacji odniesienia (RGU — Reference Generation Unit) w celu wytworzenia odpowiedniego adresu fizycznego. Podobnie jak w przypadku pozostałych pól rozkazu, na adres logiczny może składać się różna liczba bitów. Jednostka ID tworzy adresy logiczne i przechowuje je do chwili, kiedy będą potrzebne w RGU. Ponadto ID ma zdolność szybkiego przeskoku do dowolnego bitu w segmencie pamięci, co jest szczególnie ważne w programach o dużej liczbie rozgałęzień.

Do podstawowych zadań jednostki MS należy:
— pobieranie mikrorozkazów z wewnętrznej pamięci ROM
— przesyłanie mikrorozkazów do członu realizującego (EU)
— obsługa sygnałów kontrolno-sterujących magistrali systemowej
— obsługa komunikacji między procesorami
— sygnalizacja błędów.

Mikrorozkazy są przekazywane przez dekoder do jednego z niezależnych procesorów tworzących EU:
DMU (Data Manipulation Unit) — zespół obróbki danych
RGU (Reference Generation Unit) — zespół generacji odniesienia.

Zazwyczaj mikrorozkazy są wykonywane w okresie jednego cyklu maszynowego. Każdy z procesorów zawiera jednak lokalny sekwenser, wykorzystywany przy realizacji złożonych operacji (np. niektórych operacji arytmetycznych w DMU lub specyficznych transmisji generowanych przez RGU). W tym przypadku realizacja mikrorozkazu może trwać przez wiele cykli maszynowych.

Procesor DMU zawiera arytmometr wykonujący 16- i 32-bitowe mnożenie i dzielenie. Odpowiednie funkcje sterujące umożliwiają wykonywanie 32-, 64- i 80-bitowych operacji zmiennoprzecinkowych. Ponadto DMU ma zdolność sprzętowego rozróżniania dziewięciu typów danych.

Procesor RGU realizuje następujące funkcje:
— zamiana 40-bitowych adresów logicznych na 24-bitowy adres pamięci fizycznej
— realizacja mechanizmów sprzętowych związanych z ochroną plików (zapis, odczyt, modyfikacja, dostęp) i poprawnością odwołań do pamięci
— realizacja 8-, 16-, 32-, i 64- i 80-bitowych dostępów do pamięci
— sterowanie rejestrem wskaźnika szczytu stosu.

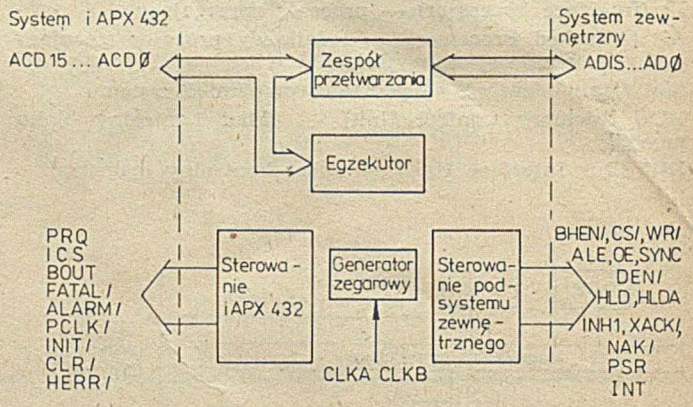
Dostęp do konkretnego segmentu pamięci powoduje zachowanie przez RGU fizycznego adresu bazowego, a także długości segmentu. Późniejsze odwołania do tego samego segmentu mogą dzięki temu występować w skróconej formie. W przypadku odwołania do segmentu, który dotychczas nie był wykorzystywany, RGU wymienia informację o bazie i długości segmentu z danymi dotyczącymi segmentu, który do tej pory był najrzadziej wykorzystywany.

## ARCHITEKTURA PROCESORA IP

Procesor IP umieszczono w układzie typu 43203. Z pozycji podsystemu zewnętrznego IP spełnia rolę procesora podległego (ang. slave) procesorowi AP. IP odwzorowuje część obszaru adresowego podsystemu zewnętrznego w centralnej pamięci systemu iAPX 432, z zachowaniem mechanizmów zabezpieczających, analogicznych jak dla GDP.

Podsystem zewnętrzny realizuje koncepcję kanału logicznego wykorzystywaną w systemach komputerowych, gdyż w sposób autonomiczny przejmuje obsługę urządzeń wejścia-wyjścia. Duża swoboda pozostawiona w konfiguracji podsystemu zewnętrznego umożliwia dopasowanie go do specyficznych potrzeb użytkownika w najbardziej ekonomiczny sposób.

Ze sprzętowego punktu widzenia procesor IP tworzy dwa sprzęgi:
— standardowy do magistrali systemu iAPX 432
— uniwersalny do wykorzystania przez dowolny 8- lub 16-bitowy system mikrokomputerowy.
Na rysunku 3 przedstawiono blokowy schemat funkcjonalny IP.




**Rys. 3. Blokowy schemat funkcjonalny procesora IP**

Komunikacja między systemami odbywa się przez zestaw kontrolowanych programowo okien, odwzorowujących bloki w pamięciach systemów. Pięć różnych bloków w pamięci podsystemu może zostać odwzorowanych na pięć segmentów w pamięci systemu iAPX 432. Powstające w ten sposób pięć okien umożliwia procesorowi AP odwołania do pamięci centralnego systemu z wykorzystaniem adresów logicznych lub — w specyficznych sytuacjach — bezpośrednio przez 24-bitowy adres fizyczny.

Przewidziany zestaw funkcji umożliwia operacje na blokach i komunikację między procedurami realizowanymi w systemie centralnym a programem realizowanym w podsystemie zewnętrznym. Komunikat wysłany przez procedurę potrzebującą obsługi wejścia-wyjścia określa rodzaj żądanej operacji (np. wczytaj zbiór XYZ). Układ sprzęgający interpretuje komunikat i podejmuje stosowane działanie. Przykładowo — gdy procedura żąda wprowadzenia danych, układ sprzęgający podaje dane jako informację zwrotną. Układ sprzęgający może również podać komunikat o pomyślnym zakończeniu realizacji żądania.

Z pozycji systemu centralnego procesor IP przypomina GDP. Podstawowa różnica polega na tym, że GDP oddziaływuje na otoczenie pod wpływem realizowanego programu, natomiast IP jest sterowany przez dyrektywy AP. Można powiedzieć, że procesor IP rozszerza możliwości programowe procesora AP na potrzeby współpracy z systemem iAPX 432.

## UKŁADY BIU I MCU

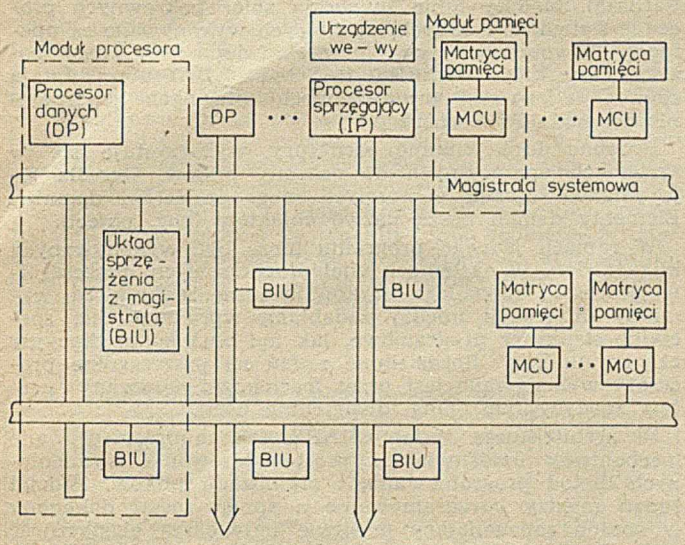
Układy BIU (Bus Interface Unit) i MCU (Memory Control Unit) stosuje się w sytuacjach, gdy bardzo ważna jest niezawodność systemu i odporność na zakłócenia.

Niezawodna praca systemu iAPX 432 jest oparta o trzy mechanizmy:
— detekcji błędów
— wydzielania obszaru programowego i sprzętowego zagrożonego błędem

— eliminacji skutków wystąpienia błędu.
Układy MCU i BIU opierają realizację wspomnianych mechanizmów na koncepcji barier rozgraniczających. Tego rodzaju podejście, zabezpieczające przed propagacją błędów, umożliwia nie tylko konstruowanie prostych, niezawodnych systemów, lecz nawet systemów, które w trakcie nieprzerwanej pracy zastępują uszkodzone bloki.

Poza wykryciem i izolowaniem punktowych uszkodzeń w złożonych systemach, układu BIU i MCU określają także rodzaj i miejsce uszkodzenia sprzętu oraz inicjują odpowiednią procedurę eliminującą skutki błędu. Działanie „naprawiające" podejmowane jest natychmiast po wykryciu błędu — tak, że realizacja procedur programowych może być kontynuowana.

Na rysunku 4 przedstawiono strukturę rozbudowanego systemu iAPX 432 z uwzględnieniem układów BIU i MCU. Poszczególne moduły procesora i pamięci oraz szyny systemowe mogą być powielane w celu stworzenia systemu o architekturze rozbudowanej stosownie do wymagań na moc przetwarzania i stopień niezawodności.




**Rys. 4. Architektura złożonego systemu iAPX 432. Zastosowanie BIU i MCU umożliwia powiększenie magistrali systemowej i zastosowanie wielu modułów pamięci w systemie o dużej niezawodności**

Układy BIU i MCU umożliwiają realizację systemów o różnym stopniu odporności na uszkodzenia. Na poziomie podstawowym (system samonaprawiający się) testowana jest parzystość kodów przesyłanych magistralami i stosowane są kody korekcyjne w matrycy pamięci. Czas stracony wskutek wystąpienia uszkodzenia określany jest przez okres potrzebny na ponowne zainicjowanie i wykonanie programu diagnostycznego, który lokalizuje usterkę i rekonfiguruje system — tak, aby izolować uszkodzony moduł lub magistralę od reszty systemu. Procedury przerwane przez wykrycie uszkodzenia muszą zostać ponownie rozpoczęte.

Drugi poziom asekuracji stwarza niemalże 100% pewności w wykryciu i skorygowaniu uszkodzenia w systemie, dzięki zastosowaniu wspomnianych barier rozgraniczających. Każdy proces komunikacyjny przez układ sprzęgający modułu (procesora czy pamięci) lub magistralę systemową jest sprawdzany przy użyciu specjalnego mechanizmu detekcji.

Każdy z modułów jest zbudowany z dwóch identycznych części, z których jedna wykonuje zasadniczą pracę modułu, a druga wykonuje te same operacje i porównuje ich wynik z rezultatem generowanym przez pierwszą. Jakakolwiek rozbieżność powoduje wystosowanie przez BIU lub MCU raportu o błędzie do pozostałych układów systemu i odłączenie modułu.

Systemy stosujące drugi poziom asekuracji niewątpliwie gwarantują dużą niezawodność, jednak wystąpienie uszkodzenia wymaga ponownej realizacji przerwanych procedur. Niezbędna jest ponadto rekonfiguracja systemu, co automatycznie obniża moc obliczeniową.

Trzeci poziom asekuracji polega na zdublowaniu każdego z modułów. W takim podwójnym module, w przypadku wystąpienia usterki, kontynuacja zadania jest podejmowana natychmiast przez moduł do tej chwili nieczynny. Wykrycie uszkodzenia i jego eliminacja prawie nie wpływa na realizację procedury. Dla utrzymania trzeciego stopnia asekuracji niezbędne jest jednak przydzielenie przez system operacyjny nowego „zapasowego" modułu do obwodu, w którym wystąpiło uszkodzenie.

Układy MCU i BIU łączy specjalna sieć wykorzystywana do przekazywania informacji o prawidłowości zachodzących w systemie procesów. Przykładowo — gdy procesor generuje odczyt z pamięci, BIU sprawdza, czy odpowiednie sygnały są obecne na magistrali systemowej. Natępnie MCU sprawdza, czy po przebyciu magistrali dotarły one do matrycy pamięci i czy wywołały poprawną reakcję. Wreszcie BIU weryfikuje prawidłowość odpowiedzi docierającej do procesora.

Przewidziana została ponadto możliwość programowego inicjowania testów rzadko wkorzystywanych linii i mechanizmów. Przesłanie przez program odpowiedniego rozkazu do BIU lub MCU powoduje generację sygnału błędu, co umożliwia sprawdzenie jego propagacji w systemie. Sprawdzane są również bufory oraz parzystość bitów przesyłanych magistralą. Testowanie pamięci odbywa się w ramach cyklu odświeżającego. Informacja jest przechowywana w pamięci z zastosowaniem specjalnego kodowania umożliwiającego korekcję w przypadku przekłamania pojedynczego bitu. W cyklu odświeżającym odczytywana jest zawartość poszczególnych komórek pamięci, a wykryte błędy są korygowane. Detekcja błędu powoduje wystosowanie odpowiedniego komunikatu przez MCU.

## ORGANIZACJA PAMIĘCI

W systemie iAPX 432 przyjęto segmentową organizację pamięci. Proces odwzorowywania pamięci odbywa się w dwóch etapach, co umożliwia oddzielenie procesu alokacji pamięci od kontroli dostępu. Kontrola dostępu do pamięci uwzględnia typ segmentu (np. dane, program, stos), jak również atrybuty ograniczające dostęp przez różne procedury. Każdy niezależny moduł programowy jest wyposażony w plik zawierający numery segmentów, które mogą być potrzebne w trakcie realizoacji programu. Ponadto plik zawiera spis atrybutów skojarzonych z poszczególnymi segmentami, obowiązujących dla danego modułu programowego. Niezależnie od pliku opisującego warunki dostępu programu do segmentów pamięci, istnieje tablica zawierająca informacje o położeniu segmentu w pamięci fizycznej, jego wielkości i — inne dane wykorzystywane przy alokacji.

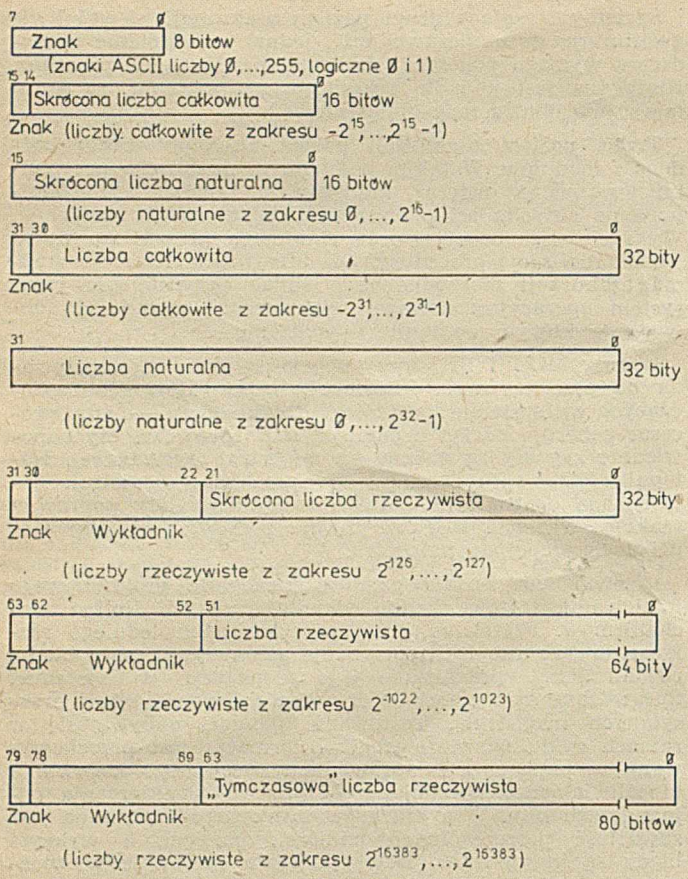
Wielkość pamięci przydzielonej dla modułu nie musi być koniecznie określona w czasie kompilowonia programu. W systemie iAPX 432 zastosowano mechanizmy umożliwiające dynamiczną zmianę wykorzystywanego obszaru.

Plik zawierający numery segmentów wykorzystywanych przez moduł programowy, sam również jest pewnym segmentem pamięci i jego numer może być elementem innego piku zawierającego numery segmentów. Umożliwia to tworzenie złożonych struktur segmentów składających się z segmentów. Struktury takie mogą być wykorzystywane m.in. przez mechanizmy ochrony plików.

## OPERACJE NA DANYCH

System iAPX 432 wyposażono w obszerny zestaw operacji umożliwiających wykonywanie działań na różnych typach danych. Typy te określane jako elementarne — w odróżnieniu od złożonych struktur danych, zestawiono na rysunku 5. Stosowane operatory można podzielić na następujące grupy:

— arytmetyczne (np. dodawanie, dzielenie, pierwiastkowanie)
— logiczne (np. AND, OR, XOR)
— warunkowe (np. równe, większe niż...)
— konwersje (np. zamiana na liczbę całkowitą)




**Rys. 5. Elementarne typy danych w systemie iAPX 432**

— przeniesienia (np. przenieś do..., zapamiętaj)
— operacje bitowe (np. wyodrębnienie)

Mianem danych strukturalnych określono zespoły danych elementarnych. W systemie iAPX 432 wykorzystuje się dane strukturalne w postaci tablic i rekordów. Tablice zawierają elementy jednakowego typu, podczas gdy rekordy mogą zawierać elementy różnych typów. W systemie nie przewidziano operacji na danych strukturalnych, możliwy jest jednak łatwy dostęp do poszczególnych elementów danych strukturalnych przez wykorzystanie licznych trybów adresowania.

Kody rozkazów są przechowywane w segmentach pamięci, którym przypisano atrybut segmentu programowego. Procesor traktuje segment programowy jako nieprzerwany ciąg bitów. Instrukcje mogą zawierać różną liczbę bitów i mogą rozpoczynać się w dowolnym miejscu ciągu. Kody rozkazów określono tak, aby najczęściej używane zawierały najmniejszą liczbę bitów. Najkrótszy kod rozkazu zawiera 6 bitów, a na najdłuższy składają się 344 bity.

Zbiór rozkazów systemu iAPX 432 jest całkowicie symetryczny (często określa się taki zbiór mianem regularnego). Znaczy to, że dla danej każdego typu można zastosować każdy użyteczny operator. Każdy z trybów adresowania może być wykorzystany w odniesieniu do dowolnego argumentu, jeżeli ma to sens. Jest to duża zaleta przy stosowaniu języków wysokiego poziomu, znacznie upraszczająca kompilatory. Przykładowo — każda z poniższych operacji wymaga pojedynczego rozkazu:

A = B*C — mnożenie wartości skalarnych
A (I) = B (J) : C (K) — dzielenie elementów tablicy
P.N = A (I)*C — umieszczenie na N-tym miejscu w rekordzie P wyniku mnożenia I-tego elementu tablicy przez skalar C. Każda z tych operacji wymaga podania trzech adresów.

W wielu przypadkach operacje mogą być wykonywane znacznie szybciej, jeżeli do przechowywania danych wykorzystuje się stos. Przykładowo — każda z poniższych operacji wymaga pojedynczego rozkazu:

A = $+B — dodanie argumentu znajdującego się na szczycie stosu do wartości w B i umieszczenie wyniku w A
$ = A*$ — mnożenie wartości z A przez argument znajdujący się na szczycie stosu i umieszczenie wyniku na stosie
$ = $+$ — dodanie dwóch argumentów ze szczytu stosu i umieszczenie wyniku na stosie.

## METODY PROGRAMOWANIA

Przy tworzeniu rozbudowanego oprogramowania należy dokonać dekompozycji zadania na mniejsze bloki funkcjonalne. Istnieje szereg kryteriów pomagających w wydzieleniu optymalnych modułów programowych (tzw. modułów Parnasa). Moduły takie zawierają zbiór pokrewnych procedur i struktury danych, na których wykonywane są operacje. Nowoczesne języki programowania wysokiego poziomu (np. CONCURRENT PASCAL, ADA) uwzględniają stosowanie struktur programowych zbliżonych koncepcyjnie do modułów Parnasa.

Podobne funkcjonalnie struktury wykorzystuje się w systemach operacyjnych do ochrony plików. Określa się je mianem domen ochronnych (ang. protection domain). Elementy domen noszą nazwę obiektów (ang. object).

W ramach jednego programu może być wykonywanych kilka zadań. W systemach wieloprocesorowych, poszczególne procesory równolegle wykonują różne zadania. Do wymiany informacji między zadaniami wprowadza się specjalne struktury programowe, jak np. port komunikacyjny czy komunikaty. Rozdzielanie zadań na poszczególne procesory wykonywane jest przez mechanizm nazywany „portem spedycyjnym" (ang. dispatching port).

W architekturze systemu iAPX 432 zaimplementowano mechanizmy umożliwiające proste stosowanie wymienionych metod programowania i organizacji plików. Podział zadań między oprogramowanie a sprzęt został dokonany w sposób zapewniający możliwie największą elastyczność systemu i maksymalne uproszczenie oprogramowania. Przykładowo — rozdział zadań między procesory jest wykonywany automatycznie, jednak programowo można ustalić technikę dokonywania rozdziału (np. stosowanie systemu priorytetów). Krytyczne czasowo i często wykorzystywane funkcje są wykonywane z reguły sprzętowo, natomiast złożone, rzadko wykorzystywane funkcje — pozostawiono do realizacji programowej. Ponieważ większość funkcji wykonywanych przez systemy operacyjne zrealizowano sprzętowo, firma INTEL reklamuje system iAPX 432 jako zawierający „krzemowy system operacyjny".

LITERATURA

[1] Microprocessor and Peripherial Handbook, 210844, Intel Corporation, Santa Clara, CA, 1982
[2] Introduction to the iAPX 432 Architecture. Intel Corporation, Santa Clara, CA, 1981
[3] Peterson C. B. et al.: Two Chips Endow 32-bit processor with Fault-tolerant Architecture. Electronics, 7 April 1983.

JERZY GRABOWSKI
Warszawa

# Porównanie właściwości mikroprocesorów 16-bitowych (2)

Występowanie poszczególnych rozkazów w różnych mikroprocesorach zilustrowano w tabeli 1. Przy typowych rozkazach, występujących dla wszystkich mikroprocesorów podane wartości oznaczają — odpowiednio — liczbę rozkazów różnych funkcjonalnie i formalnie, według opisanego kryterium komercjalnego.

## PPRZESŁANIA

Grupa przesłań jest liczna w przypadku mikroprocesorów I-8086 i Z8000. Mikroprocesor Z8000 dysponuje rozkazami wymiany zawartości rejestrów, pobrania danych oraz przesyłania informacji między rejestrem a stosem. INTEL 8086 ma dodatkowo rozkaz pobrania adresu efektywnego. Liczba rozkazów tej grupy wynika z przyjętej dla tych mikroprocesorów architektury typu rejestr-pamięć (np. rozróżnianie rozkazów ładowania, zapamiętywania przesyłania na stos itd.), a także — zarezerwowania oddzielnych kodów dla różnych wariantów argumentu (brak ogólności trybów adresowania). Zauważamy, że w przypadku architektury typu pamięć-pamięć wszystkie te rozkazy można sprowadzić do jednego (por. tab. 1 dla LSI-11).

Tabela 1. Dostępność rozkazów w mikroprocesorach 16-bitowych

| Grupa rozkazów | INTEL 8086 | ZILOG 8000 | LSI — 11 | TMS 9900 | MC 68000 | NS 16032 |
|---|---|---|---|---|---|---|
| Przesłania | 12/23 | 14/27 | 1/2 | 4/5 | 10/14 | 7/15 |
| Przesłania ciągów danych | + | + | – | – | – | + |
| Porównania i przeszukiwania ciągów | + | + | – | – | – | + |
| Odwzorowania ciągów (tłumaczenie) | – | + | – | – | – | + |
| Przesłanie na stos grup rejestrów | – | – | – | – | – | + |
| Arytmetyczne | 20/32 | 13/29 | 12/19 | 9/13 | 15/37 | 20/58 |
| Zmiennoprzecinkowe | – | – | 25/45 | – | – | – |
| Logiczne | 5/13 | 6/54 | 5/9 | 10/12 | 4/12 | 7/21 |
| Przesunięcia i obroty | 8 | 12/28 | 6/10 | 4 | 8/24 | 3/9 |
| Operacje bitowe | – | + | – | – | + | + |
| Operacje na danych nietypowych | – | – | – | – | – | + |
| Skoki | 8/24 | 8/71 | 8/25 | 5/17 | 9/74 | 12/29 |
| Rozkazy wejścia-wyjścia | + | + | – | + | – | – |
| Przerwania programowe | + | + | + | + | + | + |

Mikroprocesor MC 68000 ma w grupie rozkazów przesyłania rozkazy nowe funkcjonalnie, bardzo użyteczne przy skoku (powrocie) do (lub z) procedury — rezerwujące (lub zwalniające) pole pamięci na stosie, które jest przydatne do przechowywania zmiennych lokalnych treści procedury. Oprócz rozkazów przesłań, wymienionych przy omawianiu Z8000, mikroprocesor MC 68000 ma rozkaz zapisania adresu efektywnego na stos oraz rozkaz wymiany zawartości rejestru z pamięcią. Mikroprocesor NS 16032 ma podobne rozkazy, z tym że dodatkowo dysponuje możliwością przechowania danych na stosie (odtworzenia ze stosu) przez dowolną grupę rejestrów (rozkaz przechowania i odtworzenia zawartości grupy rejestrów). Mikroprocesor TMS 9900 ma ogólny rozkaz przesyłania, charakterystyczny dla architektury typu pamięć-pamięć, oraz operacji na rejestrach zrealizowanych układowo.

## ROZKAZY ARYTMETYCZNE

W grupie rozkazów arytmetycznych mikroprocesora I-8086 wyróżniają się rozkazy do wykonywania operacji na danych kodowanych (BCD), tak jak w przypadku Z8000. Wszystkie mikroprocesory mają podobne rozkazy arytmetyczne obejmujące: dodawanie, odejmowanie zwykłe i z przeniesieniem, operację rozszerzania znaku i — na ogół — rozkazy jednostkowych zmian argumentu. Ten ostatni rozkaz występuje czasami (Z8000, TMS 9900) w wariancie z określeniem wartości zmiany, która przestaje być „jednostkowa", a staje się „mała". Wszystkie mikroprocesory mają operacje mnożenia i dzielenia różniące się długością dozwolonych argumentów. Tak więc mikroprocesory mogą mnożyć liczby 8- i 16-bitowe. NS 16032 jednak mnoży także dane 32-bitowe uzyskując 64-bitowy wynik, który może umieścić bezpośrednio w pamięci; natomiast inne mikroprocesory wykorzystują do tego tylko rejestry. Mikroprocesor TMS 9900 ma operację mnożenia o tyle zubożoną, że wynik nie może przekroczyć 16 bitów, a argumenty są traktowane jak liczby bez znaku. Mikroprocesor MC 68000, z kolei, nie może mnożyć bajtów z 16-bitowym wynikiem.

Do grupy rozkazów arytmetycznych należą także rozkazy operacji zmiennoprzecinkowych. Rozkazy zmiennoprzecinkowe ma mikroprocesor LSI-11/23, zaś mikroprocesory z rodziny NS 16000 mogą współdziałać z układem NS 16081, który jest procesorem zmiennoprzecinkowym. NS 16081 może pracować w systemie wieloprocesorowym jako procesor podległy lub na prawach urządzenia zewnętrznego. Mikroprocesor LSI-11/23 ma 20 podstawowych rozkazów operacji zmiennoprzecinkowych wykorzystujących 64-bitowe akumulatory. Lista rozkazów LSI-11/23 obejmuje zmiennoprzecinkowe mnożenie, dzielenie, dodawanie, konwersje z pojedynczej precyzji (mantysa 24-bitowa, wykładnik 8-bitowy) na podwójną (mantysa 56-bitowa) oraz konwersje liczb zmiennoprzecinkowych obu precyzji na liczby całkowite (i odwrotnie), a także pewne operacje na wykładniku i mantysie. Podobną listę rozkazów ma mikroprocesor NS 16081.

Firma MOTOROLA proponuje, z kolei, pamięć stałą MC 68341, która zawiera układowe procedury (ang. firmware routines) obsługujące operacje zmiennoprzecinkowe dodawania, odejmowania, mnożenia, dzielenia, obliczania pierwiastka kwadratowego, porównywania, wyznaczania wartości bezwzględnej, itp. Procedury są współużywalne (ang. reetrant, wielobieżne) i przesuwne ang. position independent).

## ROZKAZY LOGICZNE

Grupa rozkazów logicznych ma cechy wspólne dla wszystkich mikroprocesorów: obejmuje ona rozkazy sumy, iloczynu, różnicy symetrycznej i negacji. Tylko mikroprocesor LSI-11 nie wykonuje operacji iloczynu logicznego, która musi być składana z dwóch rozkazów. LSI-11 umożliwia natomiast zerowanie logiczne wyniku według drugiego argumentu, który pełni funkcję maski. Podobny rozkaz mają mikroprocesory TMS 9900 i NS 16032. Pierwszy z nich, zresztą — podobnie jak LSI-11 — ma pewne niekon-

sekwencje w budowie rozkazów podstawowych operacji logicznych. Mikroprocesory mogą też, na ogół, badać (porównywać z zerem) dane, przekazując wynik do rejestru stanu lub — jak w przypadku Z8000 — używać jednocześnie parametryzującego rozkaz warunku do sprawdzania jego zgodności z wynikiem testu, umieszczając wynik porównania na najmniej znaczącym bicie argumentu wynikowego. Zwykle można też porównywać dwa argumenty.

## PRZESUNIĘCIA I OBROTY

Grupa rozkazów przesunięcia i obrotu jest najliczniejsza w przypadku mikroprocesora Z8000. Jednak I-8086 ma unikalną właściwość wykonywania przesunięć i obrotów parametryzowanych długością danych przesuwanych i nie będących wielokrotnością długości bajtu (można przesuwać dane o długości od 1 do 16 krokiem co 1).

## ROZKAZY BLOKOWE

Rozkazy działań na ciągach i blokach danych — realizowane jako operacje wielokrokowe, przerywalne i funkcjonalnie równoważne niebanalnym procedurom dla procesorów nie posiadających tych rozkazów — mają mikroprocesory I-8086, Z8000 i NS 16032. Do ich wykonywania używa się rejestrów uniwersalnych, które w kontekście rozkazów tej grupy stają się rejestrami specjalizowanymi, zawierającymi liczniki kroków lub adresy.

Zbiór typowych rozkazów w tej grupie ma mikroprocesor NS 16032. Należą do nich przesłania, porównania, wyszukiwanie elementów i przekształcenie ich według wzorca. Warunkiem zakończenia operacji może być rozpoznanie w pamięci elementu o zadanej wartości (równej lub różnej od wzorca) lub zmniejszenie ustawionego wcześniej licznika. Mikroprocesor NS 16032 ma też rozkazy przesyłania i porównania blokowego, działające na ciągach danych o długości ustalonej w rozkazie. Istnieje 5 podstawowych rozkazów operacji blokowych, wśród których można wyróżnić 27 wariantów.

Mikroprocesor Z8000 ma 6 podstawowych rozkazów tego rodzaju, ale porównania są parametryzowane przez 14 warunków według rejestru stanu (NS 16032 używa tylko warunków równości i nierówności). Po uwzględnieniu wariantów, związanych z typem danych i z warunkami, otrzymuje się 120 różnych rozkazów. Ponadto każdy z tych rozkazów występuje w wersji jednokrokowej. Porównując tę liczbę rozkazów z całkowitą (por. tab. 5 w części 1 artykułu, INFORMATYKA 11/83 str. 9) można uważać układ Z8000 za mikroprocesor rozkazów warunkowych. Mikroprocesor I-8086 ma pięć rozkazów operujących ciągami danych. Używa się przy tym rozkazu powtarzania, który zmienia interpretację następnego rozkazu z jednokrokowej na wielokrokową.

## OPERACJE BITOWE

Rozkazy operacji na adresowalnych bitach istnieją dla mikroprocesorów Z8000, MC 68000 i NS 16032. Do typowych rozkazów tej grupy należą: testowanie (wynik w rejestrze stanu), kasowanie i ustawianie bitu. Bit adresuje się przez podanie adresu w określonym trybie oraz przemieszczenia lokalizującego bit w słowie pamięci lub rejestrze. Przemieszczenie może być podane dynamicznie (wartość w rejestrze) lub statycznie (argument prosty). Mikroprocesor MC 68000 ma jeszcze dodatkowy rozkaz inwersji adresowanego bitu z odwzorowaniem jego wartości początkowej w rejestrze stanu. Mikroprocesor NS 16032 ma najbogatszy zbiór rozkazów z tej grupy. Rozkazy zmiany stanu i testowania bitu w pamięci umożliwiają blokadę adresowanej komórki (np. dla DMA) na czas wykonywania rozkazu, co ma istotne znaczenie dla pracy programów w środowisku wieloprocesorowym ze wspólną pamięcią. Oprócz tego, NS 16032 ma rozkaz wyszukiwania pierwszego ustawionego bitu, rozkazy odwzorowywania warunków z rejestru stanu na adresowany bit oraz rozkazy inwersji bitów.

## OPERACJE NA DANYCH NIETYPOWYCH

Rozkazami, które ma tylko mikroprocesor NS 16032, są operacje na strukturach danych o nietypowej długości (ang. field instructions), czyli na danych, których kwant nie jest bitem ani wielokrotnością bajtu (np. maksymalnie upakowana tablica przedstawiająca znaki kodu 6-bitowego). Pojedyncze elementy upakowanych tablic takich danych trafiają kawałkami o różnej długości w sąsiednie komórki pamięci. Wagę problemu może docenić programista, który np. na bajtowej maszynie próbował dokodować magnetyczne taśmy z maszyny o nietypowej długości słowa i przekonał się, że wyniki rozwiązania tego prostego zadania przypominają — zamiast oczekiwanych — tworzone przez generator liczb pseudolosowych. Mikroprocesor NS 16032 ma cztery rozkazy do rozwiązywania takich problemów.

## SKOKI

Rozkazy skoków obejmują skoki zwykłe, warunkowe, wywołania procedur i powroty z procedury zwykłej lub obsługującej przerwanie. Skoki zwykłe i wywołania procedur umożliwiają przekazywanie sterowania w dowolne miejsce pamięci. Dla mikroprocesorów I-8086, Z8000, TMS 9900, MC 68000 rozkazy te mają ograniczony zbiór trybów adresowania. We wszystkich przypadkach dozwolony jest tryb bezpośredni lub względny przez licznik rozkazów. Mikroprocesor I-8086 może używać trybu rejestrowego z rejestrem segmentu, definiując zamiar skoku w obrębie bieżącego segmentu lub poza segment. Przy wywoływaniu procedur wszystkie mikroprocesory 16-bitowe — oprócz TMS 9900 — używają stosu do przechowywania adresu powrotu. Mikroprocesor TMS 9900 używa w tym celu wyróżnionych rejestrów roboczych, a jego architektura w ogóle nie przewiduje stosu. Jest to niewątpliwa wada tego mikroprocesora.

Interesujący sposób przechowania adresu powrotu ma mikroprocesor LSI-11. Stosu używa się do pamiętania adresu powrotu tylko w szczególnym przypadku użycia dwuargumentowego rozkazu wywołania podprogramu, wymagającego dodatkowo określenia rejestru. W ogólności adres powrotu jest umieszczany w rejestrze, którego początkową zawartość zapisuje się na stosie. Sekwencje działania rozkazów wywołania i powrotu są tak zaprojektowane, że określenie w rozkazie wywołania (powrotu) licznika rozkazów sprowadza ich działanie do klasycznego rozwiązania z pamiętaniem (odtwarzaniem) zawartości licznika rozkazów przez stos. Rozwiązanie z tzw. rejestrem łączącym (ang. linkage register) ułatwia dostęp do stałych, które mogą być interpretowane jako parametry aktualne, a są umieszczane po rozkazie wywołania procedury. Rozwiązanie takie jest stosowane przy przekazywaniu parametrów aktualnych do procedury napisanej w języku FORTRAN.

Z rozkazami wywołania (powrotu z) procedury związane bywają ważne rozkazy rezerwowania (zwalniania) pamięci zajmowanej na stosie przez dane lokalne w treści procedury. Rozkazy tego rodzaju mają mikroprocesory MC 68000 i NS 16032. Są one ważne z punktu widzenia pisania procedur rekurencyjnych czy — ogólniej — współużywalnych, a więc tworzenia np. bibliotek procedur rezydujących w pamięci i obsługujących wielozadaniowe środowisko programowe. Ułatwiają one również tworzenie kodu wynikowego przy tłumaczeniu programów z języków wysokiego poziomu (np. PASCAL, ADA), gdzie wymaga się, by procedury były rekurencyjne, a obszary pamięci dla zmiennych lokalnych i parametrów były powoływane dynamicznie tylko na czas działania procedury. Dla mikroprocesora Z8000 rozkazy powrotu mogą być warunkowe.

Skoki warunkowe stosują adresowanie względem licznika rozkazów ze stałą lub zmienną (tylko MC 68000 i NS 16032) długością wyrównania. Do skoków warunkowych zalicza się, przydatne do organizowania wyliczanych pętli, rozkazy zmniejszające o 1 stan wskazanego rejestru i warunkujące efektywność skoku stanem zera w rejestrze. Mają je wszystkie omawiane mikroprocesory oprócz TMS 9900. W przypadku MC 68000 rozkaz ten jest o tyle ogólniejszy, że można użyć w nim dowolnego warunku uzależniającego efektywność skoku od stanu modyfikowanego rejestru. Mikroprocesor TMS 9900 ma unikalny rozkaz wykonania operacji oderwanej od naturalnej sekwencji programu i wskazanej argumentem adresowym, co jest przydatne w czasie uruchamiania programów.

Szczególnie obszerny i uniwersalny jest zbiór rozkazów skoków mikroprocesora NS 16032. Podobnie jak w przypadku LSI-11 — przy skokach prostych i wywołaniach procedur można używać dowolnego trybu adresowania. Mikroprocesor NS 16032 ma też specjalny rozkaz ułatwia-

jący organizowanie tablic skoków; tablice te są niezbędne przy tworzeniu kodu wynikowego w czasie tłumaczenia instrukcji wyboru (np. języka PASCAL). Rozkazy skoków warunkowych mogą używać wyrównania o zmiennej długości, co z kolei umożliwia optymalizowanie długości programu. Lista rozkazów tego mikroprocesora zawiera też wywołania i powroty z tzw. procedury zewnętrznej, które przechowują (odtwarzają) dodatkowo stan rejestru potrzebnego przy adresowaniu w trybie zewnętrznym. Rozkazy te są związane z dostępem do przechowywanych w pamięci pojedynczych kopii procedur bibliotecznych, używanych przez programy środowiska wielozadaniowego.

## ROZKAZY WEJŚCIA-WYJŚCIA

Rozkazy wejścia-wyjścia mają następujące mikroprocesory: I-8086, Z8000 i TMS 9900. W pozostałych mikroprocesorach zastosowano metodę odwzorowania rejestrów urządzeń zewnętrznych na obszar adresowy pamięci (ang. memory mapped i/o). Przy takim podejściu nie ma specjalizowanych rozkazów wejścia-wyjścia, gdyż ich funkcję przejmują dowolne rozkazy (najczęściej przesłania) zmieniające stan rejestrów sprzęgu, adresowanych tak jak normalne komórki pamięci.

W przypadku mikroprocesora LSI-11 każde urządzenie — z punktu widzenia programu — jest reprezentowane przynajmniej przez dwa rejestry: rejestr stanu i rejestr danych.

Adresy tych rejestrów są umieszczane na końcu obszaru adresowego w zarezerwowanym polu o długości 4 K słów. Zaleca się określone przyporządkowanie sprzęgów typowych urządzeń do ustalonych adresów obszaru komunikacji ze światem zewnętrznym. Rejestr stanu ma pola do wpisywania funkcji lub trybu obsługi urządzenia (zapytywanie lub przerwania), a także bity tylko odczytywane, które przechowują informację o aktualnym stanie operacji (urządzenie gotowe, operacja wykonywana, błąd itd.). Dzięki temu można obsługiwać urządzenia metodą programowego zapytywania (ang. polling) lub korzystając z przerwań. Rejestr danych służy do wysyłania danej lub do jej odbioru w obu metodach pracy.

Inne urządzenia mogą działać w trybie bezpośredniego dostępu do pamięci. Uaktywnienie operacji przez sprzęg DMA wymaga wpisania do rejestrów parametrów transferu i jego zapoczątkowania stosowaną funkcją. Urządzenia wykorzystujące ten sposób dostępu mają zwykle więcej rejestrów niezbędnych z punktu widzenia ich programowej obsługi.

W mikroprocesorach I-8086 i Z8000 do obsługi urządzeń zewnętrznych stosuje się metodę komunikacji przez rejestr (ang. accumulator i/o). Wymaga ona specjalizowanych rozkazów, które przesyłają dane między rejestrem mikroprocesora a rejestrem sprzęgu urządzenia. Adresowanie rejestrów urządzeń odbywa się przez obszar adresowy wyodrębniony z pamięci.

Najbogatszy zbiór rozkazów specjalizowanych do obsługi urządzeń zewnętrznych ma mikroprocesor Z8000. Zawarto w nim rozkazy przesłań blokowych, które mogą być wykorzystane do obsługi szybkich urządzeń — stanowiąc rozwiązanie pośrednie między metodą DMA a programowym przesyłaniem danych. Mikroprocesor ma 12 grup rozkazów wejścia-wyjścia występujących w 40 różnych wariantach. Komunikacja przez rejestr roboczy urządzeniem zewnętrznym występuje także dla mikroprocesora I-8086. Zbiór rozkazów wejścia-wyjścia tego mikroprocesora jest jednak znacznie uboższy niż w przypadku Z8000. Lista rozkazów I-8086 nie zawiera przesłań blokowych.

Odmienną technikę obsługi urządzeń wejścia-wyjścia zrealizowano w mikroprocesorze TMS 9900. Jest to tzw. metoda CRU ang. control unit register i/o technique). Jej istotą jest adresowanie i przesyłanie pojedynczych bitów, w przeciwieństwie do słów lub bajtów, jak w poprzednich dwóch metodach. Rozkazy wejścia-wyjścia TMS 9900 umożliwiają grupowanie przesyłanych bitów w porcje, czyli — bajty lub słowa. Do adresowania obiektów w obszarze adresowym sprzętu używa się jednego z rejestrów roboczych.

## ROZKAZY KONTROLI ZASOBÓW SYSTEMU

Do rozkazów kontroli zaliczono te, za pomocą których można wpływać na globalny stan systemu, a więc na programy współbieżne lub inne procesory zestawu wieloprocesorowego. Należą do nich rozkazy zatrzymania lub zawieszenia programu, kasowania zewnętrznej magistrali, maskowania przerwań, zmiany praw dostępu do segmentu pamięci oraz specjalizowane rozkazy przełączające sterowanie na inne procesory. Są to na ogół rozkazy uprzywilejowane, a więc przeznaczone do wykonywania przez nadrzędne programy systemu.

Wszystkie omawiane mikroprocesory mają rozkazy zatrzymania programu, po których układ przechodzi w stan oczekiwania na przerwanie zewnętrzne. Większość z nich ma rozkaz inicjowania zewnętrznej magistrali, a rozkazy maskowania przerwań występują we wszystkich przypadkach. Różne są jednak szczegóły ich realizacji i komunikowania się z innymi procesorami systemu.

Mikroprocesor I-8086 ma rozkaz związany z kontrolą systemu wieloprocesorowego. Jest to tzw. rozkaz ESC (ang. escape), który umożliwia przełączenie przetwarzania na procesor podległy. Procesor nadrzędny wykonując rozkaz ESC wysyła na magistralę jego adres efektywny i zawiesza działanie. Procesor podległy próbkując stan magistrali rozpoznaje skierowane do niego żądanie, odczytuje przesłany adres i używa go do zainicjowania swego działania. Po zakończeniu operacji, procesor podległy wyprowadza procesor nadrzędny ze stanu oczekiwania, a ten kontynuuje swój program. Również mikroprocesor Z8000 ma specjalne rozkazy kontroli środowiska wieloprocesorowego. Może blokować i odblokowywać wybrane kombinacje przerwań, oddzielenie dla przerwań wektorowych i niewektorowych. Wśród rozkazów kontroli ma też rozkazy przeznaczone do inicjowania stanu rejestru sterowania, kontroli układów odświeżania pamięci dynamicznej RAM oraz rejestru wskaźnika pola wektorów przerwań.

Mikroprocesor TMS 9900, wśród rozkazów kontroli środowiska programowego, ma rozkaz inicjowania rejestru wskaźnika obszaru roboczego oraz rozkaz przełączania kontekstu, czyli skoku do podprogramu z ustawieniem nowej wartości wskaźnika obszaru roboczego. Rozkaz ten, będący odmianą wywołania podprogramu, definiuje nowy zbiór rejestrów roboczych do używania w obrębie podprogramu (czy raczej — procesu). Rozkaz powrotu do poprzedniego kontekstu pozwala wrócić do programu głównego z odtworzeniem starego wskaźnika.

Mikroprocesor LSI-11 ma rozkazy operacji na rejestrze stanu. Dostęp do układu zarządzania pamięcią odbywa się na tej samej zasadzie jak do urządzenia zewnętrznego — bez używania specjalizowanych rozkazów i przez przesłania do rejestrów układu odwzorowanych w obszarze adresowym pamięci. Specjalizowane rozkazy do bloku zarządzania pamięcią, dostępne tylko w trybie pracy uprzywilejowanej, potrafią przełamać zabezpieczenia układu zarządzania i pobrać lub przesłać dane przez lokalny stos do (lub z) „obcego" obszaru adresowego.

Mikroprocesor NS 16032 ma rozkaz definiowania konfiguracji, który wpisuje odpowiednie bity do rejestru stanu. Ustalenie konfiguracji przez tylko zapisywalne (!) bity rejestru stanu umożliwia odwoływanie się do zewnętrznych procesorów przez kody rozkazów dodatkowych, niewidocznych dla procesora głównego i kierowanych do obsługi procesora podległego. Takimi rozkazami mogą być np. operacje zmiennoprzecinkowe kierowane do układu NS 16081 lub rozkazy zarządzania pamięcią obsługiwane przez układ NS 16082. Także mikroprocesor TMS 9900 używa tzw. kodów dodatkowych dla rozkazów nie zrealizowanych.

## PRZERWANIA PROGRAMOWE

Przerwania programowe można podzielić na kilka grup wyróżnionych według przyczyn (źródeł) przerwania:

— zaprojektowana pułapka
— błąd rozkazu arytmetycznego
— złamanie praw dostępu do segmentu pamięci
— nielegalny rozkaz
— rozkaz o nie zrealizowanym kodzie.

Wszystkie przerwania wewnętrzne są niemaskowalne.

Przeznaczeniem pułapek jest możliwość odwołania się programu nieuprzywilejowanego do obsługi systemowej lub do zapoczątkowania procedury uruchomieniowej. Można je zaliczyć do grupy skoków ze skróconą częścią adresową. Jest to szczególnie zasadne w przypadku mikroprocesorów nie mających stanu pracy uprzywilejowanej, gdyż każde

przerwanie przeprowadza procesor w ten stan. Procesor I-8086 ma dwa takie rozkazy. Jeden z nich może określić 8-bitową wartość definiującą procedurę obsługi przerwania i w tym sensie można traktować ten rozkaz jako wywołanie o skróconym kodzie. Mikroprocesor LSI-11 ma cztery przerwania tej grupy kierowane do procedur obsługi przez cztery różne wektory przerwań. Dwie pułapki są parametryzowane stałą 8-bitową, ale nie ma ona wpływu na sprzętową obsługę przerwania, tak jak w przypadku I-8086. Może być natomiast odczytana programowo, wprowadzając dodatkowy parametr obsługi. Mikroprocesor MC 68000 ma jeden rozkaz w tej podgrupie, a NS 16032 dwa rozkazy, z których jeden jest parametryzowany programowo.

Przerwanie arytmetyczne jest realizowane często jako szczególny rodzaj testu na przepełnienie pamiętane w rejestrze stanu. Źródło przerwania może być też uaktywniane bezpośrednio w trakcie wykonywania rozkazu tworzącego błędny wynik. Pierwsze rozwiązanie występuje w mikroprocesorze I-8086. Do rozkazów wewnętrznych przerwań arytmetycznych można zaliczyć też występujące dla MC 68000 sprawdzenie zakresu argumentu, które jest używane do kontroli poprawności adresowania tablic. Mikroprocesor ten ma też rozkaz przerwania wewnętrznego uaktywnianego bitem przepełnienia. Przerwania wewnętrzne zgłaszane rozkazem generującym błędny wynik ma mikroprocesor NS 16032 (dzielenia przez zero) i LSI-11/23, dla którego źródłem przerwania są układy arytmetyki zmiennoprzecinkowej.

Rozkazy przerwań wewnętrznych, będące skutkiem dostępu do segmentu pamięci niezgodnie z określonymi prawami, mają te mikroprocesory (mikrokomputery), które posiadają układy zarządzania pamięcią. Są to: LSI-11/23, Z80001 (z układem Z8010), NS 16032, NS 16016 (z układem NS 16081) i MC 68000 (z układem MC 68451).

Przerwania wywoływane przez rozkazy nielegalne mają wszystkie mikroprocesory z wyjątkiem I-8086. Brak zdefiniowanej reakcji na kod rozkazu nie zrealizowanego jest bardzo nieprzyjemną cechą układu. Mikroprocesor TMS 9900 używa tych przerwań do rozszerzenia listy rozkazów o rozkazy obsługiwane programowo. To samo podejście zastosowano w mikroprocesorze NS 16032, z tą różnicą, że przy obsłudze rozkazu nielegalnego — w realizowanym sprzętowo pierwszym kroku — próbuje się zidentyfikować procesor podległy, który mógłby wykonać nieobsługiwany kod rozkazu.

Rozkazy uprzywilejowane, wykonywane w trybie pracy normalnej procesora wywołują przerwania wewnętrzne, tak jak nielegalne kody. Takimi rozkazami uprzywilejowanymi są rozkazy zatrzymania procesora, kasowania magistrali zewnętrznej, działania na rejestrze stanu i rejestrach układów zarządzania pamięcią, a w przypadku LSI-11 także wszystkie odwołania do obszaru pamięci zawierającej rejestry sprzęgów urządzeń zewnętrznych. To ostatnie przerwanie ma źródło w układach zarządzania pamięcią.

## EFEKTYWNOŚĆ PROGRAMOWANIA

W tabeli 2 przedstawiono zaczerpnięte z książki C. A. Titusa i in., wnioski liczbowe dotyczące szybkości działania i długości realnych programów dla omawianych mikroprocesorów (oprócz NS 16032, gdyż w trakcie pisania wspomnianej książki mikroprocesory z serii NS 16000 nie były dostępne jej autorom). Tabela zawiera wyniki testów dla czterech prostych problemów:

— sortowanie 600-elementowej tablicy danych 16-bitowych

— wyszukiwanie w tablicy bajtowej 6-znakowego napisu umieszczonego na 86 miejscu od jej początku

— obliczanie pierwiastka kwadratowego klasyczną metodą iteracji

— wyznaczanie funkcji sinus i cosinus na podstawie tablicy wartości.

Dane do sortowania są na wejściu programu uporządkowane w sposób najmniej korzystny. Funkcja sinus jest stabelaryzowana w zakresie od 0 do 90°, a funkcje sinus i cosinus mają być wyznaczane w zakresie od 0 do 360°.

Test sortowania ma ocenić działanie rozkazów przesyłania; wyszukiwanie napisu ma badać efektywność rozkazów przeszukiwania blokowego lub zastępujących je; obliczanie pierwiastka kwadratowego testuje efektywność mnożenia i dzielenia, a funkcja trygonometryczna — właściwości dostępu do tablic. Wyniki testów zawarte w tabeli obejmują pomiar czasu i długości programu dla każdego zagadnienia i uwzględnionego procesora. W testach dla różnych mikroprocesorów brano pod uwagę identyczne dane wejściowe. Czytelnik łatwo zauważy, że wyniki testów są szczególnie korzystne dla mikroprocesorów Z8000. Mikroprocesory nie dysponujące rozkazami operacji blokowych mają wyraźnie dłuższy czas rozwiązywania problemu wyszukiwania napisu. Także długości programów dla Z8000 są najkorzystniejsze. Należy jednak przypuszczać, że nie uwzględniony w tabeli mikroprocesor NS 16000 będzie miał parametry porównywalne lub lepsze. Tabela wykazuje także, że — po teoretycznej normalizacji czasów względem częstotliwości zegara — parametry mikroprocesora TMS 9900 znacznie odbiegają od pozostałych na jego niekorzyść. Dowodzi to nieskuteczności doboru rozkazów w jego liście, co uwidacznia się też w długości programów dla tego mikroprocesora. Wybrane przykłady nie upoważniają do wysnuwania ostatecznych wniosków o jakości mikroprocesorów, ale dają jednak pewne rozeznanie w ich możliwościach.

* * *

Przedstawiony cykl artykułów (INFORMATYKA nr 3—12/83) w głównej mierze oparto na książce C. A. Titusa i in. [6]. Zawiera ona bardzo staranny przegląd mikroprocesorów 16-bitowych. Każdy z mikroprocesorów jest rozpatrywany według jednakowego schematu, który uwzględnia: architekturę, listę rozkazów, system przerwań, układy do konfigurowania zestawu mikrokomputerowego wraz z przykładami takich zestawów oraz przykłady zaprogramowanych i przetestowanych problemów jednolicie dobrane z punktu widzenia porównania czasu działania i długości kodu. W książce Dullhoffa [2] można znaleźć bardzo ogólny przegląd mikroprocesorów 16-bitowych z uwzględnieniem nie opisanych w prezentowanym opracowaniu (odpowiedniki minikomputerów NOVA firmy DATA GENERAL, mikroprocesor PACE INS 8900 firmy NATIONAL SEMI-

**Tabela 2. Porównanie czasów wykonania i długości programów wzorcowych**

| Czynności | | INTEL 8086 | ZILOG 8000 | LSI 11/02 | LSI 11/23 | TMS 9900 | MC 68000 |
|---|---|---|---|---|---|---|---|
| Częstotliwość taktowania | | 8 MHz | 10 MHz | | | 2 MHz | 8 MHz |
| Sortowanie | sekundy | 2,1 | 1,6 | 10,5 | 4,2 | 33 | 2,0 |
| | bajty | 46 | 38 | 34 | 34 | 76 | 40 |
| Wyszukiwanie napisu | mikrosekundy | 235 | 94,5 | 979 | 392 | 2250 | 424 |
| | bajty | 56 | 38 | 40 | 40 | 52 | 50 |
| Pierwiastek kwadratowy*) | mikrosekundy | 195/147 | 51/71 | 458/628 | 183/251 | 680/860 | 88/121 |
| | bajty | 52 | 28 | 66 | 66 | 70 | 42 |
| Funkcja trygonometryczna*) | mikrosekundy | 11/13 | 4,6/7,1 | 64/85 | 26/34 | 160 | 7,8/10,3 |
| | bajty | 52 | 28 | 78 | 78 | 62 | 38 |

*) Czasy: minimalny/maksymalny

CONDUCTOR) i bardzo dokładne studia nad mikroprocesorem TMS 9900 — głównie od strony programowania. Podręcznik [5] wydany przez firmę DIGITAL EQUIPMENT CORPORATION jest opisem technicznym sprzętu o wielkim stopniu scalenia zbudowanym wokół mikroprocesora LSI-11. Pozostałe pozycje mają charakter przeglądów.

Zbiór zagadnień opisanych w niniejszym cyklu nie wyczerpuje oczywiście problematyki związanej z mikroprocesorami 16-bitowymi. Rzeczywistą przydatność układów można ocenić dopiero w konfrontacji konkretnego problemu z dostępnymi środkami. Jeśli jednak te środki nie są dostępne, wszelkie rozważania o jakości czy możliwościach mają sens działań pozornych lub zastępczych. Należy więc życzyć inżynierom i programistom, by mieli oni jednak szansę stosować te układy w praktyce zawodowej. I to nie tylko na egzemplarzach przywiezionych w kieszeni.

**LITERATURA**


[1] Davis H. A.: Comparing Architectures of Three 16-Bit Microprocesors. Computer Design, Vol. 18, No. 7, p. 91, 1979

[2] Dullhoff T.: 16-Bit Microprocessors Architecture. Prentice-Hall. Englewood Cliffs (NY), 1979

[3] Heering J.: A Comparative Analisis of the INTEL 8086, the ZILOG Z8000 and the MOTOROLA MC 68000 Microprocessors. Euromicro Journal, Vol. 6, p. 135, 1980

[4] Manner R., Deluigi B.: 16-Bit-Prozessoren im Vergleich. Elektronik, 5, 6, 7/1981

[5] Microcomputers and Memories. Digital Equipment Corporation, Maynard (MA), 1981

[6] Titus C. A., Titus J. A., Baldwin A., Hubin W. N., Scanlon L.: 16-Bit Microprocessors. Howard W. Sams and Co., Indianapolis, 1981.


# Przemysłowy FORTRAN czasu rzeczywistego.

## Część 3

W ostatniej części omówienia projektu normy przemysłowego FORTRANU Czasu Rzeczywistego przedstawiono wywołania dotyczące operacji na ciągach dwójkowych i bitach, przesyłania informacji do/z urządzeń wejścia-wyjścia procesu technologicznego oraz zarządzania plikami danych.

### PRZETWARZANIE CIĄGÓW DWÓJKOWYCH I BITÓW

Wywołania funkcji opisane poniżej dostarczają mechanizmów operowania na ciągach bitów, jak również na poszczególnych bitach wewnętrznego przedstawienia zmiennych całkowitych. Liczby całkowite traktuje się tak, jak by nie miały znaku (z wyjątkiem operacji przesunięcia arytmetycznego). Zakłada się, że bit numer 0 (bit położony najbardziej na prawo) jest bitem najmniej znaczącym.

#### Operacje logiczne na ciągach bitów

Możliwe są następujące operacje logiczne: **OR, AND, EOR** i **NOT** (odpowiednio: LUB, I, ALBO, NIE) — zrealizowane jako funkcje całkowite. Domyślny typ dla **OR, AND** i **EOR** jest wskazany użyciem **I,** jako pierwszej litery nazwy funkcji. Parametry funkcji, **j** i **k,** są wyrażeniami arytmetycznymi całkowitymi. Po wykonaniu funkcji parametry pozostają niezmienione. Operacje wykonuje się na odpowiadających bitach dwóch argumentów i otrzymuje wartość odpowiedniego bitu wyniku. Wartości podane w poniższych tablicach przedstawiają pojedyncze bity argumentów i wartości funkcji.

Sposób obliczania wartości funkcji logicznych: sumy logicznej **IOR (j, k),** iloczynu logicznego **IAND (j, k),** dopełnienia logicznego **NOT (j)** i różnicy symetrycznej **IEOR (j, k)** przedstawiono w odpowiednich tablicach poniżej.

| | | | | |
|---|---|---|---|---|
| j | 0 | 1 | 0 | 1 |
| k | 0 | 0 | 1 | 1 |
| Suma logiczna | 0 | 1 | 1 | 1 |

| | | | | |
|---|---|---|---|---|
| j | 0 | 1 | 0 | 1 |
| k | 0 | 0 | 1 | 1 |
| Iloczyn logiczny | 0 | 0 | 0 | 1 |

| | | |
|---|---|---|
| j | 0 | 1 |
| Dopełnienie logiczne | 1 | 0 |

| | | | | |
|---|---|---|---|---|
| j | 0 | 1 | 0 | 1 |
| k | 0 | 0 | 1 | 1 |
| Różnica symetryczna | 0 | 1 | 1 | 0 |

#### Operacje przesunięć

Możliwe są operacje przesunięć logicznych, arytmetycznych i cyklicznych, zrealizowane jako funkcje całkowite. Funkcje mają parametry **j** i **n,** które uważa się za wyrażenia arytmetyczne całkowite. Parametr **j** określa ciąg bitów podlegający przesunięciu, a **n** — licznik przesunięcia, w następujący sposób:

**n > 0** wskazuje przesunięcie w lewo
**n = 0** wskazuje brak przesunięcia
**n < 0** wskazuje przesunięcie w prawo.

Jeżeli bezwzględna wartość licznika przesunięcia jest większa niż liczba bitów w numerycznej jednostce pamięci, to wynik jest niezdefiniowany, tzn. zależny od procesora. Operacje przesunięć nie zmieniają wartości parametrów.

Przesunięcie logiczne ma postać **ISHL (j, n).** Wszystkie bity przedstawiające parametr **j** przesuwa się o **n** miejsc. Bity wysuwane z lewego lub prawego końca są tracone. Z przeciwnego końca wprowadza się zera.

Przesunięcie arytmetyczne ma postać **ISHA (j, n).** Argument **j** i wartość funkcji są traktowane jako liczby całkowite ze znakiem. Wszystkie bity przedstawiające parametr **j** przesuwa się o **n** miejsc. W przypadku przesunięcia w prawo **(n < 0)** jeżeli j jest dodatnie, wprowadza się zera na lewym końcu; jeżeli natomiast **j** jest ujemne wprowadza się jedynki. Bity wysuwane z prawego końca są tracone. W przypadku przesunięcia w lewo **(n > 0),** wprowadza się zera na prawym końcu, zaś bity wysuwane z lewego końca są tracone. Przy przesunięciu w lewo może wystąpić nadmiar arytmetyczny.

Przesunięcie cykliczne ma postać **ISHC (j, n).** Wszystkie bity przedstawiające parametr **j** przesuwa się cyklicznie o **n** miejsc tzn. bity wysuwane z jednego końca są wprowadzane na drugi. Nie traci się żadnych bitów. Liczba bitów przedstawiających **j** jest zależna od procesora.

**Przetwarzanie pojedynczych bitów**

Funkcje operacji bitowych mają dwa parametry, **j** i **k**, które są wyrażeniami arytmetycznymi całkowitymi:
**j** — określa ciąg bitów
**k** — określa wybrany bit, numerowany jak podano we wstępie.
Jeżeli **k** jest ujemne, równe lub większe niż liczba bitów używanych do przedstawienia wartości całkowitej, to wynik funkcji jest niezdefiniowany. Omawiane funkcje nie zmieniają wartości parametrów.

Funkcja typu logicznego **BTEST (j, k)** bada **k**-ty bit parametru **j**. Jeżeli jest on równy **1**, to wartością funkcji jest PRAWDA, natomiast jeżeli jest równy **0**, wartością funkcji jest FAŁSZ.
Funkcja całkowita **IBSET (j, k)** powoduje ustawianie bitu. Wartość funkcji jest równa wartości parametru **j** z **k**-tym bitem ustawionym na **1**.

Zerowanie bitu odbywa się przy użyciu funkcji całkowitej **IBCLR (j, k)**. Wartość funkcji jest równa wartości parametru **j** z **k**-tym bitem ustawionym na **0**.

Funkcja całkowita **IBCHNG (j, k)** powoduje zmianę wartości bitu. Wartość funkcji jest równa wartości parametru **j** z **k**-tym bitem zastąpionym przez jego dopełnienie.

## WEJŚCIE-WYJŚCIE Z/DO KONTROLOWANEGO PROCESU

Użytkownik FORTRANU musi mieć możliwość komunikacji z urządzeniami specyficznymi dla swego zastosowania. Ponieważ większość systemów wejścia-wyjścia zależy od komputera, można to unormować w sposób uniwersalny tylko przez znormalizowane wywołania programów sterujących urządzeń, specjalnie napisanych dla każdego systemu. Niezależne od komputera systemy wejścia-wyjścia są na ogół znormalizowane (CAMAC, GPIB itp.). Znormalizowane wywołania wejścia-wyjścia dla tych systemów są równie ważne jak wywołania podane poniżej, jednak wykraczają poza zakres niniejszej normy.

**Zakres i ogólna budowa procedur wejścia-wyjścia**

Urządzenie peryferyjne procesu jest łączem pomiędzy procesem (lub jego urządzeniami koncowymi) a centralną jednostką przetwarzającą komputera. Dane opisujące zachowanie się procesu w czasie i przestrzeni są odbierane przez jednostkę przetwarzającą i przystosowywane tak, by mogły być przesłane do centralnej jednostki przetwarzającej przez sprzęg wejściowo-wyjściowy. Różnorodność zadań, wymagana przez proces, zaowocowała wielką liczbą urządzeń peryferyjnych pochodzących od różnych producentów. Jednak w ciągu wielu lat ewolucji sprzętu ustalono w dużym stopniu zgodne i ogólnie przyjęte kierunki rozwoju. Poniżej podano ich krótką charakterystykę.

Bramy wejściowo-wyjściowe (ang. I/O ports) rozróżnia się przez ich indywidualne adresy. Identyfikatory bram (adresy) używane w wywołaniach procedur będą prawdopodobnie w większości systemów identyczne z poszczególnymi adresami sprzętowymi, ale nie jest to obowiązkowe. Związek ten uważa się za cechę zależną od procesora, pozostającą poza zakresem niniejszej normy.

Norma FORTRANU określa, że jedna instrukcja musi się skończyć, zanim rozpocznie się przetwaraznie następnej. Standardowe wejścia-wyjścia opisane poniżej stosują się do tej zasady. Zadanie wywołujące będzie oczekiwało zakończenia i ten tryb pracy jest zaznaczony ostatnią literą **W** (ang. waiting, oczekiwanie) w nazwach wszystkich procedur.

Poniższe oznaczenia parametrów stosuje się w większości wywołań. Jeżeli dokładny sens identyfikatorów odbiega od tego, co opisano niżej, będzie to specjalnie zaznaczone w szczegółowym opisie wywołania. Jeżeli znaczenie jest dokładnie takie, jak poniżej, to opis parametru będzie pominięty.

Procedury procesowego wejścia-wyjścia mają zazwyczaj cztery, a w specjalnym przypadku — pięć parametrów, które dalej będą oznaczone przez **i, j, k, m** (i ewentualnie **n**). Takie wywołanie ma ogólną postać:

CALL procio (i, j, k, m)

gdzie:
**procio** — wskazuje jeden z podprogramów
**i** — określa liczbę wartości do przesłania (wyrażenie arytmetyczne całkowite)
**j** — ckreśla nazwę tablicy całkowitej lub elementu tablicy, która zawiera niezbędną informację dla opisu bram, tj. adresy i informacje o sposobie przekształcenia danych (prawidłowa postać informacji jest zależna od procesora)
**k** — określa nazwę tablicy całkowitej lub elementu tablicy, która zawiera wartości wejściowe lub wyjściowe
**m** — jest wskaźnikiem, którego wartość charakteryzuje powodzenie wywołania (argument ten musi być zmienną całkowitą lub elementem tablicy całkowitej)
$m \leqslant 0$ — niezdefiniowane
$m = 1$ — wszystkie dane zostały przesłane
$m \geqslant 2$ — sytuacje błędne.

**Wprowadzanie i wyprowadzanie wartości analogowych**

W przypadku operacji wejścia rozróżnia się sprzętowo zrealizowane wprowadzanie sekwencyjne oraz wprowadzanie w dowolnej kolejności. Dla wprowadzania sekwencyjnego, wejściowy parametr **j** zawiera adres pierwszego wejścia analogowego; dalsze adresy są generowane automatycznie. W drugim przypadku, w tablicy **j** należy podać pełny ciąg adresów. Dla operacji wyjścia postać parametru jest zawsze „w dowolnej kolejności", tj. w tablicy **j** umieszcza się wszystkie adresy. Na ogół postać **j** jest zależna od systemu.

Relacja pomiędzy zakresem bramy wejściowej lub wyjściowej i odpowiadającym jej elementem w **k** jest zależna od procesora.

Wykonanie wywołania podprogramu AISQW powoduje odczytanie ciągu analogowych bram wejściowych o kolejnych adresach. Postać wywołania sekwencyjnego wprowadzania danych analogowych jest następująca:

CALL AISQW (i, j, k, m)

gdzie:
**i** — określa liczbę analogowych bram wejściowych do odczytu (parametr jest wyrażeniem całkowitym)
**j** — jest opisem sprzętowej bądź programowej informacji dla pobrania i przetworzenia danych z kolejnych bram analogowych (jest to nazwa tablicy całkowitej lub elementu)
**k** — jest tablicą całkowitą lub elementem tablicy całkowitej dla zapamiętania przetworzonych wartości analogowych.

Podprogram AIRDW czyta ciąg wejściowych bram analogowych w zadanym porządku. Postać wywołania jest następująca:

CALL AIRDW (i, j, k, m)

gdzie:
**j** — jest opisem sprzętowej lub programowej informacji dla pobrania i przetworzenia poszczególnych wartości analogowych, a znaczenie pozostałych parametrów — takie, jak dla podprogramu AISQW.

Podprogram AOW wyprowadza ciąg wartości analogowych na zbiór analogowych bram wyjściowych w zadanej kolejności. Wywołanie ma postać:

CALL AOW (i, j, k, m)
gdzie:
**i** — określa liczbę analogowych bram wyjściowych (wyrażenie całkowite)
**j** — zawiera informacje dla przetworzenia i przesłania danych (nazwa tablicy całkowitej)
**k** — jest tablicą całkowitą, z której wyprowadza się wartości analogowe.

**Wprowadzanie i wyprowadzanie wartości cyfrowych**

W tym przypadku zakłada się, że jakkolwiek efektywna informacja może być czasami przedstawiona pojedynczym bitem, to jednak trzeba będzie przesyłać do lub z tablicy całkowitej wartości cyfrowe (uważane za obiekty wykorzystujące całe numeryczne jednostki pamięci lub słowa, np. 16 bitów dla każdej numerycznej jednostki pamięci).

Wywołanie wprowadzania cyfrowego ma postać:

CALL DIW (i, j, k, m)

gdzie:
i — określa liczbę wprowadzanych wartości cyfrowych (wyrażenie całkowite)
j — zawiera sprzętową, a w niektórych przypadkach programową, informację dla przekształcenia i przesłania (jest to nazwa tablicy całkowitej; j może także zawierać ewentualną specyfikację dotyczącą zerowania)
k — jest tablicą całkowitą w której będą zapamiętane wartości cyfrowe.

W przypadku wyprowadzania odróżnia się wyprowadzanie impulsowe (ang. digital output momentary) od wyprowadzania z wartością utrzymywaną stale (ang. digital output latching).

Wykonanie wywołania podprogramu DOMW realizuje wyprowadzenie impulsowe na zbiór bram wyjściowych. Impuls pojawi się na bitach wybranych przez jedynkę odpowiedniego bitu i elementu tablicy k. Zaden impuls nie pojawi się na bitach wybranych przez binarne zero. Czas trwania impulsu wskazuje się, w odpowiedniej postaci, parametrem n. Impuls rozpocznie się w chwili wykonania DOMW lub wraz z następnym taktem zegara. Zależy to (jak również postać impulsu, np. polaryzacja czy napięcie) od zastosowanego sprzętu.

Wywołanie ma postać:

CALL DOMW (i, j, k, n, m)

gdzie:

i — określa liczbę wyprowadzanych wartości cyfrowych (wyrażenie całkowite)

j — zawiera informację sprzętową dla przesłania każdej z wartości cyfrowych (nazwa tablicy całkowitej)

k — jest tablicą całkowitą przedstawiającą wartości cyfrowe do wyprowadzenia

n — jest liczbą jednostek czasu zegara zależną od zastosowanego sprzętu; jeżeli procesor nie dopuszcza wyboru czasu trwania impulsu, argument ten jest ignorowany, ale musi występować (musi być wyrażeniem całkowitym).

W przypadku wyprowadzania cyfrowego statycznego DOLW, oprócz pola wyjściowego wymaga się też pola maski dla wskazania bitów które mają być zmienione na wyjściu. Wywołanie ma postać:

CALL DOLW (i, j, k1, k2, m)

gdzie:

i oraz j — mają znaczenie jak dla podprogramu DOMW
k1 — jest tablicą całkowitą, przedstawiającą wartości cyfrowe do wyprowadzenia
k2 — oznacza tablicę, której wartości określają wyjścia cyfrowe, zmieniane przez podprogram; bit ustawiony w tablicy k2 wskazuje, że wyjście cyfrowe będzie zmienione do stanu zdefiniowanego przez odpowiednią pozycję bitu w odpowiadającym elemencie tablicy całkowitej k1 (porządek elementów w k1 i k2 powinien odpowiadać porządkowi w j; argument ten musi być nazwą tablicy całkowitej lub elementem tablicy całkowitej).

## ZARZĄDZANIE PLIKAMI

Opisane wywołania zewnętrznych procedur umożliwiają zarządzanie dostępem do plików, a także rozstrzyganie problemów spornych żądań dostępu do pliku w środowisku wielozadaniowym lub wieloprocesorowym. W takim środowisku należy się spodziewać, że zadania współbieżne będą próbowały dokonać operacji na tym samym pliku w jednej chwili. Dlatego zdefiniowane poniżej wywołania zewnętrznych procedur dostarczają informacji niezbędnej dla procesora do rozplanowania równoczesnego dostępu w sposób uporządkowany. Metoda rozplanowania zarządzania dostępem pozostaje w gestii procesora.

Celem wywołań procedur z tego punktu jest dostarczenie metod, którymi zadanie może powiadomić procesor o sposobie, w jaki zamierza ono użyć pliku. Jednak celem tych wywołań nie jest żądanie związania określonych właściwości lub atrybutów ze wskazanymi plikami. Wywołania umożliwiają uniknięcie problemów związanych z konfliktami, jeżeli będą użyte łącznie z prawidłową konstrukcją programu. Realizacja niniejszej normy nie stanowi jednak gwarancji, że takie problemy nie powstaną.

Pliki istnieją w większości systemów obliczeniowych i mogą mieć różne artybuty i właściwości.

- Plik może zawierać dane, programy lub informację katalogową.
- Mogą istnieć różne sposoby dostępu[1]) do pliku, takie jak: sekwencyjny, bezpośredni i strumieniowy.
- Plik może być utworzony lub skasowany przez zadanie, przez sytemową procedurę pomocniczą, albo w czasie generacji systemu.
- Plik może mieć artybuty ochrony w celu zapewnienia jego tajności.
- Gdy plik jest połączony z zadaniem, połączenie to może być ograniczone przez procesor z powodu tajności.
- Plik może być połączony z grupą powiązanych zadań współbieżnych i to połączenie może być ograniczone dla zapewnienia uporządkowanego rozwiązania problemów związanych z konfliktami pomiędzy zadaniami współbieżnymi.
- Plik może być dla zadania wewnętrzny albo zewnętrzny.
- Plik może znajdować się na nośnikach stale dołączonych lub wymiennych.
- Plik może znajdować się w pamięci głównej lub pomocniczej.
- Ograniczenia z powodu tajności lub konfliktów dostępu mogą dotyczyć pliku lub części składowej pliku, takiej jak rekord lub poszczególne dane.

### Środowisko systemu plików

W przemysłowych systemach komputerowych czasu rzeczywistego powszechnym zjawiskiem jest współbieżne działanie zadań ze współdzielonymi zasobami, takimi jak pliki. Niniejsza norma nie obejmuje wszystkich dziedzin zarządzania plikami, ale dotyczy problemów, które najczęściej powstają w przemysłowych systemach komputerowych czasu rzeczywistego. W poniższej tabeli przedstawiono te cechy, które są objęte normą, jak i te, które z niej wyłączono. Tym niemniej cechy wyłączone mogą wpłynąć na wynik żądania połączenia współbieżnego zadania z plikiem. Takie ograniczenia połączenia są zależne od procesora i nie wchodzą w zakres niniejszej normy.

Cechy i atrybuty plików

| Pliki objęte normą | Pliki wyłączone z normy |
|---|---|
| Pliki, których zawartość uważa się za dane | Pliki, których zawartość nie jest uważana za dane przez zadanie współbieżne dokonujące dostępu do pliku |
| Pliki istniejące tylko na stale dołączonych nośnikach lub na wymiennych, których nie usunięto | Pliki istniejące na wymiennych nośnikach, które usunięto |
| Pliki położone w głównej pamięci lub w pamięci pomocniczej | |
| Pliki zewnętrzne dla zadania współbieżnego | Pliki wewnętrzne dla zadania współbieżnego |
| Tworzenie i kasowanie plików przez zadanie współbieżne | Tworzenie i kasowanie plików przez systemową procedurę pomocniczą lub przy generacji systemu |
| Połączenie pliku z zadaniem współbieżnym, tak dla plików utworzonych przez system, jak i utworzonych przez zadanie współbieżne | Metody dostępu do pliku |
| Ograniczenia dostępu do pliku dotyczące całego pliku | Ograniczenia dostępu do pliku dotyczące części pliku |
| Połączenie pliku z zadaniem współbieżnym niezależne od sposobu dostępu (np. bezpośredniego, sekwencyjnego lub strumieniowego) | Atrybuty pliku mające na celu zapewnienie jego tajności |

### Procedury zarządzania dostępem do pliku

Wywołania procedur opisane poniżej są nieprzerywalne, tzn. procesor będzie wypełniał jedno takie wywołanie w danej chwili. Wymaganie to gwarantuje, że opisane operacje są wykonywane w sposób uporządkowany.

---

[1]) Ang. „access" tłumaczone przez „dostęp" w tym rozdziale często oznacza „przetwarzanie", tj. zapis lub odczyt

Argumentowi **m** nadaje się wartość równą lub większą niż 2, gdy żądanie nie zostanie przyjęte przez system nadzorczy. Poszczególne realizacje mogą określić konkretne wartości **m** w dopuszczalnym zakresie dla oznaczenia specyficznej przyczyny, dla której żądanie zostało odrzucone.

Tworzenie plików

Wykonanie wywołania podprogramu CFILW utworzy nazwany plik, ale nie otworzy go. Pliki utworzone przez CFILW nie mają żadnego atrybutu tajności, który ograniczałby jakiemuś zadaniu współbieżnemu dostęp do pliku. Zawartość nowo utworzonego pliku jest niezdefiniowana.

Wywołanie ma postać:

CALL CFILW (j, n1, n2, m)

gdzie:
**j** — określa plik i może być wyrażeniem całkowitym, nazwą tablicy całkowitej, nazwą procedury lub wyrażeniem znakowym (konkretny procesor ustala, które z powyższych czterech postaci są dopuszczalne)
**n1** — określa liczbę znakowych jednostek pamięci w rekordzie w tym pliku (argument musi być wyrażeniem całkowitym)
**n2** — określa maksymalną liczbę rekordów w pliku (argument musi być wyrażeniem całkowitym)
**m** — przybiera wartość przy powrocie do wywołującego zadania dla wskazania skutków żądania i musi być równe 1 lub więcej (1 — plik utworzony pomyślnie, 2 lub więcej — plik nie utworzony); argument ten musi być nazwą zmiennej całkowitej lub nazwą elementu tablicy całkowitej.

Kasowanie plików

Wykonanie wywołania podprogramu DFILW usunie plik z systemu plików. Każdy plik utworzony mechanizmem z poprzedniego punktu może być skasowany przez wykonanie wywołania DFILW, z tym jednak, że kasowanie nie dojdzie do skutku, jeżeli plik jest aktualnie otwarty dla jakiegoś zadania.

Wywołanie ma postać:

CALL DFILW (j, m)

przy czym parametry **j** oraz **m** mają znaczenie takie, jak dla procedury CFILW.

Otwieranie plików

Wykonanie wywołania podprogramu OPENW połączy jednostkę określoną przez zadanie z nazwanym plikiem i określi pożądany tryb dostępu tego zadania do pliku.

Wywołanie ma postać:

CALL OPENW (i, j, k, m)

gdzie:
parametry **j** oraz **m** — mają znaczenie takie, jak powyżej
**i** — określa jednostkę logiczną, przez którą zbiór nazwany argumentem **j** jest identyfikowany w zadaniu (argument ten musi być wyrażeniem całkowitym)
**k** — określa tryb dostępu pożądany przez zadanie; jest to deklaracja zamierzonego sposobu użycia pliku przez zadanie (argument ten musi być wyrażeniem całkowitym).
Określa się następujące wartości **k**:
1 — (OTWARTY), zadanie wywołujące żąda czytania i pisania lub tylko pisania, inne zadania mają prawo do takiego samego dostępu.
2 — (CHRONIONY ODCZYT), dostęp do czytania jest wymagany przez zadanie wyowłujące i dozwolony dla innych zadań.
3 — (ZAMKNIĘTY), dostęp do czytania i pisania lub tylko do pisania jest wymagany przez zadanie wywołujące; zadanie wywołujące wyklucza jakikolwiek dostęp do pliku przez inne zadania.

Jeżeli plik jest aktualnie otwarty dla innego zadania, to decyzje wobec żądania konkretnego trybu dostępu będą następujące:

**OTWARTY** — żądanie nie spełni się, jeżeli plik jest aktualnie otwarty dla innego zadania w trybie ZAMKNIĘTY lub CHRONIONY; w przeciwnym razie jest spełniane
**CHRONIONY ODCZYT** — żądanie nie spełni się, jeżeli plik jest aktualnie otwarty dla innego zadania w trybie ZAMKNIĘTY lub OTWARTY
**ZAMKNIĘTY** — żądanie nie spełni się.

Każda próba otwarcia pliku powiedzie się tylko wtedy, gdy plik istnieje. Jeżeli plik został utworzony przez mechanizm nie objęty niniejszą normą, to atrybuty nadane plikowi przy tworzeniu mogą ograniczyć udzielenie jakiegoś trybu dostępu danemu zadaniu.

Zamykanie plików

Wykonanie wywołania podprogramu CLOSEW zakończy połączenie zadania za pośrednictwem określonej jednostki logicznej z nazwanym plikiem. Wywołanie ma postać:

CALL CLOSEW (i, m)

gdzie: parametry **i** oraz **m** mają znaczenia, takie jak powyżej, odpowiednio do zamknięcia pliku.

Zmiana trybu dostępu

Wykonanie wywołania podprogramu MODAPW zmieni tryb dostępu wywołującego zadania do uprzednio otwartego przez to zadanie pliku, bez zamykania i powtórnego otwierania pliku. Jeżeli zadanie wywołujące nie ma żadnego trybu dostępu do pliku, to żądanie nie zostanie spełnione. Jeżeli żądanie zmiany nie może być przyjęte, poprzedni tryb dostępu pozostaje w mocy.

Wywołanie ma postać:

CALL MODAPW (i, k, m)

gdzie:
**i** oraz **k** — mają znaczenie takie, jak w punkcie dotyczącym otwierania plików, a **m** — przybiera wartość przy powrocie do zadania wywołującego dla wskazania skutków żądania i musi być równe 1 lub więcej (1 — żądany tryb dostępu został udzielony zadaniu, 2 lub więcej — tryb dostępu istniejący przed żądaniem pozostaje w mocy); argument ten musi być nazwą zmiennej całkowitej lub nazwą elementu tablicy całkowitej.

Jeżeli plik jest aktualnie otwarty dla innego zadania, to żądanie zmiany trybu nie spełni się. Jeżeli plik został utworzony przez mechanizm nie objęty normą, to atrybuty nadane plikowi przy tworzeniu mogą ograniczyć udzielenie zadaniu jakiegoś trybu dostępu.

**Tłumaczył:**
**KAZIMIERZ MALISZEWSKI**

## *KSIĄŻKI NADESŁANE*

### WYDAWNICTWO POLITECHNIKI WROCŁAWSKIEJ

Moszkowicz M.: Ekonomiczne problemy rozwoju produkcji i zastosowań komputerów w gospodarce. Seria Monografie, nakład 200 egz., 185 str., cena 122 zł, Wrocław, 1983.

**Autor podejmuje problematykę związaną z komputeryzacją różnych dziedzin gospodarki krajowej. W kolejnych rozdziałach znajduje się analiza zjawiska komputeryzacji, zastosowań komputerów w gospodarce i ekonomii oraz opisane są podstawy sterowania rozwojem komputeryzacji. Przedstawiona jest także pewna koncepcja sterowania rozwojem produkcji i zastosowań maszyn cyfrowych w Polsce.**

### PAŃSTWOWE WYDAWNICTWO EKONOMICZNE

Dziedziczak I.: Organizacja bazy danych księgowych. Seria Informatyka w praktyce, nakład 2500 egz., 171 str., cena 110 zł, Warszawa, 1983.

**Wychodząc z ogólnych problemów baz danych, autor przedstawia specyficzne uwarunkowania struktur danych księgowych, Księgowe Obiekty Informacyjne i formy ich występowania, modelowe konstrukcje księgowania oraz organizację zawartości informacyjnej rozważanych baz danych.**

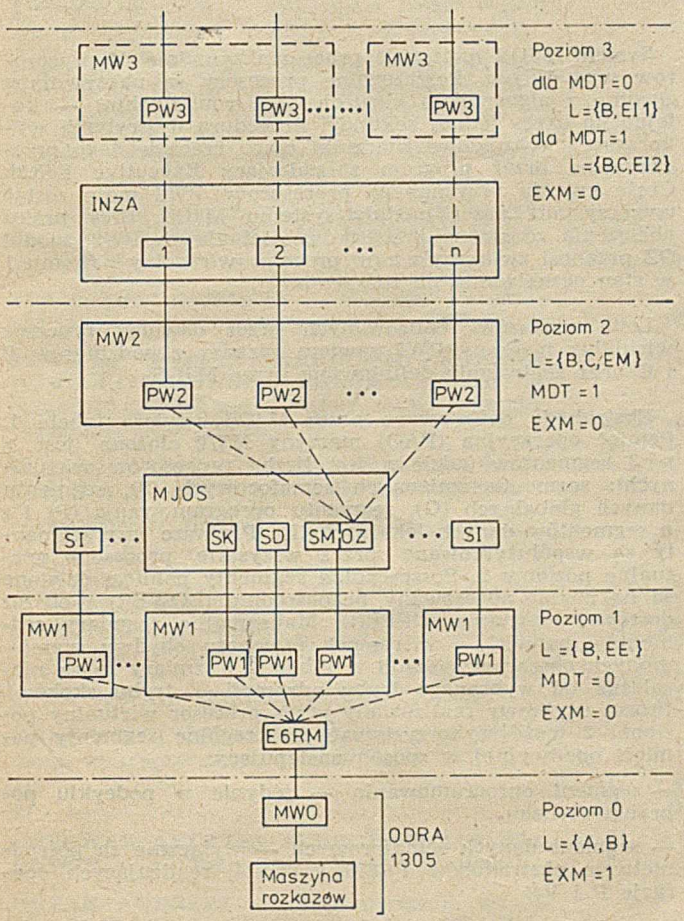
JAN KNIAT
ZYGMUNT LISZYŃSKI
Środowiskowy Ośrodek Informatyki
Politechnika Poznańska

Programowanie procesów współbieżnych (2)

# Środowisko programowe

**W pierwszej części artykułu (INFORMATYKA nr 7—8/1983) przedstawione zostały modyfikacje sprzętowe dokonane w procesorze centralnym ODRA 1305, umożliwiające w operacjach dostępu do pamięci realizację przeadresacji rozłącznej. Po zakończeniu modyfikacji procesora podjęto dalsze prace mające na celu zbudowanie systemu maszyn wirtualnych, umożliwiającego równoczesną realizację wielu użytkowych, interakcyjnych systemów informatycznych. Przyjęto założenie, że dowolny z systemów użytkowych może mieć charakter wieloprocesowy, a procesy tego systemu mogą być wzajemnie związane. Liczebność zbioru systemów realizowanych współbieżnie może się dynamicznie zmieniać, stosownie do decyzji podejmowanych przez użytkowników w chwili rozpoczynania seansu.**

Docelowy system maszyn wirtualnych uzyskano konstruując hierarchię programów zarządzających, definiujących kolejne zbiory tych maszyn (rys. 1).




Rys. 1. Hierarchia maszyn wirtualnych

Maszyna wirtualna, którą udostępnia poziom 0, definiowana jest przez zmodyfikowaną jednostkę centralną maszyny cyfrowej ODRA-1305.

W zbiorze realizowanych przez tę maszynę rozkazów wyróżnić można dwa rozłączne podzbiory:
A — rozkazów dostępnych wyłącznie przez EXM = 1
B — rozkazów dostępnych przy MDT = 0, których definicje nie zależą od stanu EXM.

Zmodyfikowany program zarządzający Executive E6RM kreuje zbiór maszyn wirtualnych udostępnianych przez poziom 1. Procesory wirtualne poziomu 1 działają w trybie EXM = 0, stąd lista rozkazów tych procesorów nie zawiera podzbioru A. Występują w niej natomiast definiowane przez program E6RM ekstrakody (EE).

Maszyna wirtualna MW1 może posiadać od jednego do czterech procesorów wirtualnych — każdy procesor wirtualny realizuje pojedynczy proces obliczeniowy. Algorytm rozdziału procesora sprzętowego pomiędzy procesory PW1 wykorzystuje zasadę „najwyższy priorytet najpierw".

Pojedyncza maszyna wirtualna, udostępniana przez poziom 2, tworzona jest przez system operacyjny MJOS. Poszczególne procesory tego poziomu realizują procesy jednego wieloprocesorowego systemu informatycznego. Procesy są przetwarzane przez MJOS cyklicznie, otrzymując kolejno przydział określonego kwantu czasu.

W liście rozkazów procesorów wirtualnych poziomu 2 występują rozkazy należące do podzbioru B, ekstrakody definiowane przez MJOS (EM) oraz rozkazy podzbioru C, mające realizację sprzętową określoną jedynie w trybie MDT = 1 (grupa 14).

Na ostatnim z omawianych poziomów (3) kreowane są docelowe maszyny wirtualne, z których korzystają programiści konstruujący użytkowe systemy informatyczne. Maszyny udostępniane przez ten poziom definiowane są przez wieloprocesorowy podsystem zarządzający INZA (realizowany w trybie MDT-1). Oprogramowanie tego podsystemu ma charakter inwariantny i wszystkie procesy współużytkują jedną jego kopię. Procesory wirtualne poziomu 3 (PW3) mogą pracować zarówno w trybie MDT=0 jak i MDT=1. Dowolna liczba tych procesorów może wchodzić w skład wybranej maszyny wirtualnej MW3, realizującej pojedynczy system informatyczny. Równocześnie istnieć może wiele maszyn MW3, możliwa jest zatem na tym poziomie współbieżna realizacja wielu różnych systemów informatycznych.

Z każdym procesorem wirtualnym poziomu 2 związane jest na stałe jedno konwersacyjne urządzenie zewnętrzne (monitor ekranowy, dalekopis lub drukarka mozaikowa z klawiaturą). Urządzenie to jest współużytkowane również przez pochodne maszyny wirtualne wyższych pozomów.

Procesory wirtualne z poziomu 2 (i ich pochodne z wyższych poziomów) umożliwiają dostęp do plików dyskowych, zapewniając współbieżną aktualizację tych plików. Procesy realizowane przez te procesory mogą także współużytkować dane i oprogramowanie oraz mogą wymieniać komunikaty.

Poniżej przedstawione zostaną niektóre aspekty realizacyjne i charakterystyka użytkowa maszyn wirtualnych z poziomów 1, 2 i 3. Poziom zerowy, realizowany przez zmodyfikowany procesor centralny maszyny cyfrowej ODRA 1305, przedstawiono w pierwszej części artykułu.

## PROGRAM ZARZĄDZAJĄCY EXECUTIVE E6RM

Udostępnienie właściwości zmodyfikowanego procesora centralnego ODRA 1305 osiągnięto w wyniku niezbędnej, lecz ograniczonej co do zakresu, redefinicji niektórych funkcji programu E6RM. Modyfikacje te wynikają z konieczności odwzorowania w języku maszyn wirtualnych MW1 wszystkich nowych cech sprzętu. W związku z tym dokonano:

— rozszerzenia definicji bloków informacyjnych (PIB) systemów informatycznych wykonywanych przez MW1 o informacje dotyczące segmentów pamięci operacyjnej maszyny MW2

— rozszerzenia zbioru ekstrakodów EE o rozkazy definiujące segmenty pamięci

— redefinicji procedury podziału czasu procesora (BATM) i procedur współpracujących w celu ustalania stanu rejestrów MDT i BM oraz zawartości rejestrów odniesień

— rozszerzenia procedury inicjacji przerwań, umożliwiającego pracę w trybie BM i legalizację rozkazu MODE w trybie normalnym

— optymalizacji niektórych funkcji systemu E6RM oraz korekty zauważonych błędów.

Modyfikacje te mają postać krótkiego programu, wprowadzanego podczas fazy uruchamiania programu zarządzającego E6RM.

Dla realizacji maszyn wirtualnych z wyższych poziomów wykorzystano również standardowe właściwości programu zarządzającego Executive E6RM. Zasadnicze — to mechanizm lokalnego wieloprogramowania (subprogramowanie), działanie urządzeń zewnętrznych w trybie odpowiedzi bezpośredniej oraz możliwość nadawania systemom użytkowym statusu zaufania.

Mechanizm subprogramowania pozwala na zdefiniowanie w systemie użytkowym maksymalnie czterech procesów obliczeniowych, realizujących oprogramowanie zawarte w tzw. członach programu użytkowego i wykorzystujących spójny obszar pamięci operacyjnej. Procesy wybierane są do przetwarzania zgodnie z algorytmem „najwyższy priorytet najpierw" — mogą one zawieszać się i być ponownie wznawiane przez procesy innych członów lub na skutek zdarzeń zachodzących w urządzeniach zewnętrznych.

Współpraca z urządzeniem zewnętrznym w trybie odpowiedzi bezpośredniej charakteryzuje się tym, że natychmiast po wydaniu ekstrakodu transmisyjnego przekazywana jest programowi użytkowemu informacja o przyjęciu względnie odrzuceniu, ekstrakodu. Właściwość ta umożliwia zwiększenie zakresu zrównoleglenia operacji obliczeniowych oraz operacji związanych z nadzorowaniem transmisji do i z urządzeń zewnętrznych.

Proces obliczeniowy realizowany przez procesor wirtualny poziomu 1 może otrzymać status zaufania klasy R i jest wówczas nazywany procesem zaufanym. Proces taki ma możliwość kreowania nowej maszyny wirtualnej wyższego poziomu (lub zbioru takich maszyn). Proces zaufany dopuszcza współużytkowanie części swoich zasobów (pamięci operacyjnej, czasu procesora i urządzeń zewnętrznych) przez powstałe w ten sposób maszyny wirtualne. Decyzje o zakresie współużytkowania tych zasobów komunikowane są przez proces zaufany programowi E6RM. Zdarzenia programowe (np. ekstrakody), zachodzące w utworzonych maszynach wirtualnych, oraz zdarzenia zewnętrzne, związane z tymi maszynami, są nadzorowane przez proces zaufany.

## SYSTEM OPERACYJNY MJOS

System MJOS jest systemem użytkowym zmodyfikowanego programu zarządzającego Executive E6RM, korzystającym ze statusu zaufania klasy R. System ten posiada konstrukcję modularną — można w nim wyróżnić cztery moduły funkcjonalne:

**SK** — zarządzający współpracą ze zdalnymi końcówkami (za pośrednictwem systemu komunikacyjnego)

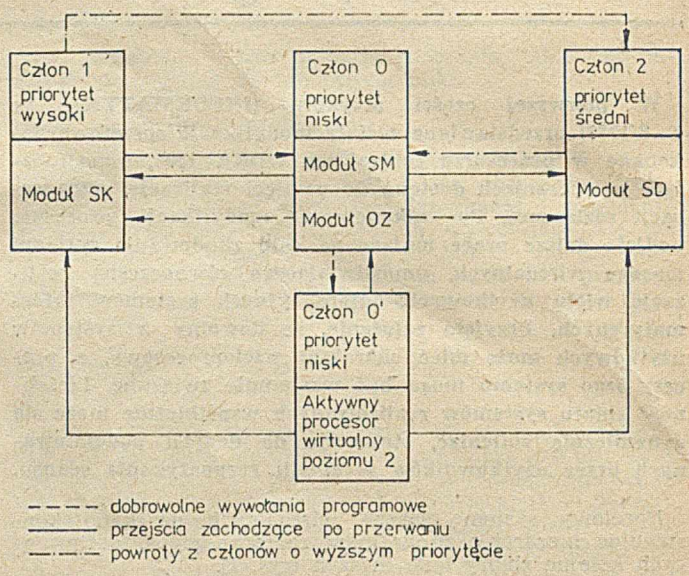
**SD** — realizujący odwołania do wspólnej bazy danych, traktowanej jako zbiór zasobów niejednorodnych z dokładnością do porcji pliku

**SM** — przeznaczony do zarządzania wymianą komunikatów pomiędzy procesorami wirtualnymi poziomu 2

**OZ** — zawierający procedury ogólnego zarządzania pracą systemu MJOS.

Oprogramowanie każdego z modułów SK, SD oraz pary modułów SM-OZ definiuje niezależny proces obliczeniowy realizowany przez pojedynczy procesor wirtualny poziomu 1 (rys. 1). Moduły SK, SD i SM są wzajemnie niezależne i nie wymieniają pomiędzy sobą żadnych informacji. Określone są natomiast powiązania OZ-SK, OZ-SD i OZ-SM. Wszystkie moduły są krotne i posiadają procedury szeregowania i kolejkowania zleceń.

Rysunek 2 ilustruje powiązania istniejące pomiędzy modułami systemu MJOS i kreowanymi przez ten system procesorami wirtualnymi PW2.




**Rys. 2. Powiązania modułów systemu MJOS i kreowanych przez ten system procesorów wirtualnych**

System MJOS nadzoruje pracę maksymalnie 64 procesorów wirtualnych. Poszczególne procesory są uaktywniane zgodnie z algorytmem karuzelowym (round-robin) — kolejno na okres jednej sekundy. Procedura dystrybucji wykorzystuje sekundowe znaczniki czasu rzeczywistego przekazywane przez program zarządzający Executive E6RM. Część kwantu przyznanego procesorowi PW2 może zostać wykorzystana przez moduły systemu MJOS. Przy braku obciążenia ze strony procesorów wirtualnych PW2, moduł OZ przenosi związany z nim procesor wirtualny poziomu 1 w stan oczekiwania na zdarzenie.

Lista rozkazów realizowanych przez dowolny procesor wirtualny maszyny PW2 zawiera rozkazy z podzbiorów B i C oraz ekstrakody definiowane przez MJOS.

Ekstrakody definiowane przez MJOS zawiera tabela 1. Pamięć operacyjna (PAO) maszyny MW2 złożona jest z n+3 segmentów (gdzie n jest liczbą procesorów wirtualnych): segmentu zmiennych semaforowych (S), segmentu danych globalnych (G), segmentu oprogramowania (P) i z n segmentów danych lokalnych (L). Pierwsze trzy segmenty są współużytkowane przez wszystkie procesory wirtualne poziomu 2. Poszczególne segmenty pamięci dzielone są na obszar adresowany bezpośrednio (LOWER) i obszar dostępny pośrednio (UPPER). Maksymalne rozmiary segmentów pamięci, a w ramach segmentu obydwu wymienionych obszarów, podano w tabeli 2 (rozmiary te są niezależne od wybranego trybu adresacji i trybu skoków). Proces użytkowy realizowany przez procesor wirtualny poziomu 2 może wykorzystywać poszczególne segmenty pamięci operacyjnej w sposób następujący:

— segment oprogramowania — jedynie w podcyklu pobrania rozkazu

— segment danych semaforowych — wyłącznie za pośrednictwem ekstrakodów PSEM i VSEM, realizujących operacje P i V

— segment danych globalnych — za pomocą rozkazów podzbioru C i pewnych ekstrakodów

# Spis treści rocznika 1983

## ARTYKUŁY PROBLEMOWE

## RECENZJE



## TERMINOLOGIA



## POGLĄDY



## GRY



## KONFERENCJE

Tabela 1. Lista ekstrakodów systemu operacyjnego MJOS

| Kod | X | Nazwa | Definicja |
|---|---|---|---|
| 163 | 0 | PSEM | wykonaj operację P nad wskazanym w N(M) semaforem |
| | 1 | VSEM | wykonaj operację V nad wskazanym w N(M) semaforem |
| 164 | 1 | SUSIN | zawieś wykonywanie procesu do czasu zakończenia dowolnej transmisji, lecz nie dłużej niż na sekundę |
| | 2 | | zawieś wykonywanie procesu do czasu zakończenia dowolnej transmisji |
| | 7 | | uruchom maszynę wirtualną wyższego poziomu |
| 165 | 0 | GIVE | podaj w N(M) datę binarnie |
| | 1 | | podaj w N(M) datę znakowo |
| | 2 | | podaj w N(M) czas znakowo |
| | 3 | | podaj w N(M) bieżący rozmiar wskazanego segmentu pamięci operacyjnej |
| | 4 | | ustal rozmiar wskazanego segmentu pamięci operacyjnej według stanu N(M) |
| | 5 | | podaj w N(M) systemowy identyfikator użytkownika |
| | 6 | | podaj w N(M) czas bieżący w sekundach |
| | 7 | | ustal nowy identyfikator użytkownika |
| 166 | 0 | TRAN | zainicjuj transmisję komunikatu do/z innego procesu |
| | 1 | | zainicjuj transmisję do/z końcówki |
| | 2 | | zainicjuj transmisję do/z pliku dyskowego |
| 167 | 0 | PUCL | przejmij definicję maszyny wirtualnej wyższego poziomu |

Tabela 2. Maksymalne rozmiary segmentów i obszarów

| Segment | Obszar | Rozmiar |
|---|---|---|
| Oprogramowanie | LOWER | 32 K |
| | UPPER | MP |
| Zmienne semaforowe | LOWER | 4 K |
| | UPPER | — |
| Dane globalne | LOWER | 4 K |
| | UPPER | MP |
| Dane lokalne | LOWER | 4 K |
| | UPPER | 64 K — LOWER |

MP — maksymalna wielkość pamięci operacyjnej udostępnianej przez program zarządzający Executive E6RM

— segment danych lokalnych — wykorzystując pozostałe rozkazy i ekstrakody wymagające kontaktu z pamięcią.

Ekstrakody realizowane przez system MJOS umożliwiają również wymianę komunikatów z innymi procesorami wirtualnymi poziomu 2, kontakt z przypisaną na stałe do każdego procesora końcówką konwersacyjną oraz dostęp do plików dyskowych. Wyróżnia się przy tym pliki lokalne danego procesora (PL) oraz pliki globalne (PG) współużytkowane przez wszystkie procesory wirtualne poziomu 2. Możliwa jest współbieżna aktualizacja plików globalnych — moduł SD realizujący odwołania do tych plików zapewnia utrzymanie spójności logicznej bazy danych oraz nie dopuszcza do powstania blokad 2.

System operacyjny MJOS umożliwia (wzorem programu E6RM) nadawanie statusu zaufania klasy R procesom obliczeniowym realizowanym przez maszyny wirtualne poziomu 2.

System MJOS akceptuje komendy języka sterowania, przekazywane przez operatora za pomocą centralnej konsoli sterującej maszyny. Umożliwiają one wprowadzanie i usuwanie systemu użytkowego oraz uruchamianie i wstrzymywanie procesów użytkowych. System MJOS wyprowadza również na konsolę komunikaty o stanach szczególnych.

## PODSYSTEM INZA

Procesy realizowane przez poszczególne procesory wirtualne poziomu 2 zdefiniowane są przez wspólne, inwariantne oprogramowania podsystemu nadzorczo-sterującego INZA. Podsystem ten kreuje maszyny wirtualne poziomu 3, z których korzystają użytkownicy konstruujący i eksploatujący użytkowe systemy informatyczne. Zdefiniowano dwa typy takich maszyn: WIMAK 1 i WIMAK 2.

Maszyna WIMAK 1 posiada jeden procesor wirtualny (PW3) współpracujący z pojedynczą, spójnie adresowaną pamięcią operacyjną (PAO) oraz z końcówką konwersacyjną, a także z plikami lokalnymi i z współużytkowanymi z innymi maszynami plikami globalnymi. Maszyna ta może ponadto wymieniać komunikaty z innymi maszynami poziomu 3.

Maszyna WIMAK 2 ma konfigurację podobną do konfiguracji maszyny MW2 z dynamicznie zmienną liczbą procesorów wirtualnych PW3, z których każdy ma dostęp do własnej pamięci lokalnej, własnej końcówki i własnych plików lokalnych. Wszystkie procesory wirtualne współużytkują ponadto segmenty pamięci operacyjnej pozostające w dyspozycji tej maszyny i zawierające zmienne semaforowe, dane globalne i oprogramowanie oraz posiadają dostęp do współbieżnie aktualizowanych plików dyskowych. Poszczególne procesory realizują pojedyncze procesy obliczeniowe i mogą wymieniać komunikaty z dowolnymi innymi procesorami danej maszyny wirtualnej, a także z procesorami innych maszyn wirtualnych MW3. Maszyna WIMAK 2 może być również eksploatowana w trybie pojedynczym, w którym posiada ona tylko jeden procesor wirtualny i nie istnieje możliwość dołączenia dalszych procesorów.

Ekstrakody realizowane przez podsystem INZA są w większości semantycznymi odpowiednikami ekstrakodów systemu MJOS. Procesy realizowane przez procesory wirtualne poziomu 3 mogą mieć nadany status zaufania klasy R, co stwarza możliwości dalszego rozwijania struktury hierarchicznej.

Równocześnie istnieć może n maszyn WIMAK ($n \leqslant 64$) realizujących współbieżnie n systemów informatycznych, przy czym i-ta maszyna ($1 \leqslant i \leqslant n$) dysponuje liczbą $m_i$ procesorów wirtualnych, a ponadto: $\sum_{i=1}^{n} m_i \leqslant 64$.

Podsystem INZA świadczy liczne usługi na rzecz użytkowników maszyn WIMAK. Pierwszą z funkcji podsystemu jest identyfikacja i ewidencjonowanie użytkowników. Podczas pierwszego seansu użytkownik przekazuje swoje dane personalne, które składowane są w pliku ewidencji. Następne seanse rozpoczynają się od podania identyfikatora (imię, nazwisko i data urodzenia), na podstawie którego odtworzony zostaje dotychczasowy stan prowadzonych przez użytkownika obliczeń. Możliwe jest usuwanie użytkowników z ewidencji.

Podsystem INZA wyróżnia dwie klasy użytkowników: zaufanych i standardowych. Użytkownicy standardowi współpracują niezmiennie z tym samym, wybranym podczas pierwszego seansu, konwersacyjnym systemem informatycznym. Odtworzenie stanu obliczeń na początku seansu i składowanie go na końcu seansu realizowane jest automatycznie.

Użytkownicy zaufani mogą wprowadzać, modyfikować, uruchamiać, wstrzymywać i usuwać swoje programy. Mogą oni korzystać z następujących komend języka sterowania (przekazywanych z końcówki):
— wyboru systemu informatycznego i klasy procesów w tym systemie czyli definiowania procesu użytkowego
— odczytywania i modyfikowania zawartości segmentów pamięci maszyny wirtualnej
— uruchamiania i wstrzymywania procesu użytkowego
— składowania i odtwarzania stanu obliczeń
— likwidowania procesu użytkowego i kończenia seansu.

* * *

Przedstawiona hierarchiczna struktura maszyn wirtualnych utworzona została dzięki trzykrotnemu zastosowaniu jednego schematu, polegającego na wyposażeniu zaufanego procesu obliczeniowego w środki umożliwiające kreację pochodnych maszyn wirtualnych wyższego poziomu. Proces zaufany realizowany przez maszynę wirtualną na i-tym poziomie ($i = 0, 1, 2$) redefiniuje i ewentualnie (dla $i = 0, 1$) zwielokrotnia tę maszynę, tworząc następną generację maszyn wirtualnych poziomu $i+1$. Zasoby fizyczne, czyli przestrajalny funkcjonalnie procesor sprzętowy (realizujący rozkazy z podzbiorów A, B i C) oraz pamięć operacyjna,

są w odpowiednich częściach i zgodnie z określonymi algorytmami szeregowania przydzielane poszczególnym maszynom wirtualnym.

Możliwa jest dowolna rozbudowa omawianej hierarchii maszyn — występują tu jednak dwa ograniczenia. Końcówka konwersacyjna jest związana z pojedynczym procesorem wirtualnym poziomu 2 i musi być współużytkowana przez wszystkie pochodne procesory wirtualne wyższych poziomów. Ponadto wzrost liczby poziomów hierarchii powoduje wzrost kosztu odwołań pomiędzy sąsiadującymi poziomami.

Obecnie trwają prace rozwojowe wzbogacające możliwości udostępnianych użytkownikom maszyn wirtualnych. Tworzony jest uniwersalny system plików (zapewniający również wirtualizację znakowych urządzeń zewnętrznych za pomocą plików dyskowych), konstruowany jest wielozadaniowy edytor i opracowywana jest procedura dynamicznej rekonfiguracji pamięci współużytkowanej.

Dla opisu oprogramowania, systemów informatycznych realizowanych przez maszyny wirtualne WIMAK 1 i WIMAK 2 opracowano język adresów symbolicznych o nazwie PLAN5. Język ten stanowi rozszerzenie PLANU4 dokonane w celu udostępnienia programiście środków językowych służących do opisu procesów współbieżnych.

Podstawowym obiektem definiowanym za pomocą wyrażeń języka PLAN5 jest system informatyczny złożony z pewnej liczby klas procesów jednorodnych. Każda klasa definiowana jest przez opis pojedynczego procesu należącego do tej klasy. Modularyzację oprogramowania umożliwia podział opisu procesu na segmenty. Lista dyrektyw języka zawiera dyrektywy strukturalne, nadrzędne główne i pomocnicze. Zbiór instrukcji dostępnych w języku odpowiada liście rozkazów maszyny wirtualnej WIMAK 2. Dane współużytkowane przez procesy niejednorodne mogą być definiowane w obszarach wspólnych.

Kompilator języka PLAN5 (XPL5) uzyskano przez modyfikację kompilatora XPLT. Z kompilatorem tym współpracuje konsolidator XPC5.

System MJOS/INZA wykorzystywany jest przez system nauczający programowania w języku FORTRAN oraz przez tworzony obecnie konwersacyjny system programowania w języku LISP. Maszyny WIMAK 1 wykorzystywane są przez studentów w trakcie laboratorium systemów operacyjnych, maszyny WIMAK 2 eksploatowane będą podczas laboratorium programowania procesów współbieżnych.

**LITERATURA**

[1] Liszyński Z.: Środowisko wirtualne dla jednodziedzinowych dialogowych systemów informatycznych. Rozprawa doktorska, Środowiskowy Ośrodek Informatyki Politechniki Poznańskiej, Poznań, 1978

[2] Małuszyński J.: Sterowanie współbieżną aktualizacją bazy danych. Rozprawa doktorska, Środowiskowy Ośrodek Informatyki Politechniki Poznańskiej, Poznań 1980

[3] Prace dyplomowe i raporty Środowiskowego Ośrodka Informatyki Politechniki Poznańskiej.

Przegląd języków wysokiego poziomu (2)

# Publikacje i organizacje

Istnieje wiele publikacji i organizacji zajmujących się wyłącznie jednym lub kilkoma językami programowania. Poniżej zostaną podane jedynie najważniejsze, pominięto zaś te, które są tylko marginesowo związane z językami programowania. W pierwszej kolejności zostaną omówione takie, które zajmują się ogólniejszą problematyką, a potem dopiero charakteryzujące się węższą specjalizacją.

## PUBLIKACJE

Istnieją dwa czasopisma, które zajmują się niemal wyłącznie językami programowania. Jednym z nich jest ACM TRANSACTIONS ON PROGRAMMING LANGUAGES AND SYSTEMS (TOPLAS) wydawane przez ACM (Association for Computing Machinery), drugim — COMPUTER LANGUAGES wydawane przez Pergamon Press. Oba te czasopisma przyjmują i publikują materiały dotyczące ogólnych aspektów języków programowania, a także poszczególnych języków; najchętniej publikują materiały nowatorskie lub prace o charakterze naukowym. Wiele z pozostałych czasopism publikuje zarówno oryginalne materiały na temat języków programowania, jak i prace z innych dziedzin informatyki.

Oprócz wymienionych czasopism istnieją dwa typy publikacji: biuletyny ogólnoprogramistyczne oraz dotyczące poszczególnych języków.

Jedynym biuletynem o charakterze ogólnym, wysoko cenionym w środowisku informatycznym, jest ACM SIGPLAN NOTICES wydawany przez sekcję ACM „Języki programowania" — ACM Special Interest Group on Programming Languages (SIGPLAN). Pismo to zawiera bieżące informacje, omówienia, ogłoszenia i sporo materiałów prywatnych (nie weryfikowanych), a także pośredniczy w kontaktach pomiędzy wszystkimi zainteresowanymi tą dziedziną wiedzy. Ponadto wykorzystywane jest również do publikowania sprawozdań z licznych konferencji organizowanych przez ACM SIGPLAN, a czasem nawet do publikowania definicji nowych języków (np. języka EUCLID czy propozycji definicji języka ADA). Autorka uważa, że SIGPLAN NOTICES jest najbardziej wartościowym źródłem informacji o bieżącym stanie dziedziny języków programowania.

Jak wspomniano, istnieje również pewna liczba biuletynów dotyczących poszczególnych języków. Wydają je często nieformalne grupy i organizacje (np. użytkowników) lub prywatne osoby. Tego typu pisma istnieją dla następujących języków: ALGOL (60 i 68), APL, COBOL, FORTRAN, JOVIAL, MUMPS i PASCAL. Celem tych biuletynów jest informowanie o technicznych i organizacyjnych problemach poszczególnych języków.

## ORGANIZACJE

Istnieje kilka różnych organizacji związanych z językami programowania, w tym zajmujące się ogólnymi problemami rozwoju języków programowania lub standaryzacją. Jedyną organizacją zajmującą się rzeczywiście ogólnymi problemami języków jest wspomniany już ACM SIGPLAN, który w 1980 roku skupiał ponad 10 tys. członków. Głównym zadaniem tej organizacji jest przygotowywanie konferencji i wydawanie wspomnianego biuletynu SIGPLAN NOTICES.

Istnieje również wiele niezależnych organizacji zajmujących się rozwojem języków programowania. Najstarszą i najważniejszą z nich jest CODASYL (Conference on Data Systems Languages), utworzony w 1959, który był odpowiedzialny za opracowanie COBOLU i w dalszym ciągu jest z nim związany, aczkolwiek obecnie w kręgu jego zainteresowań leżą także inne problemy języków programowania, takie jak bazy danych i języki do sterowania systemami operacyjnymi.

Istnieje ponadto Międzynarodowa Federacja Przetwarzania Informacji — IFIP (International Federation for Information Processing), w ramach której opracowywano ALGOL 68 (Grupa Robocza 2.1 — IFIP Working Group 2.1).

Przykładem może być też grupa użytkowników komputerów IBM, skupiona w organizacji SHARE, która od 1963 pracowała nad językiem PL/I.

Na początku lat siedemdziesiątych Departament Obrony USA prowadził prace nad językiem JOVIAL 73, a pod koniec tej dekady rozpoczęto tam prace nad językiem ADA.

I wreszcie ANSI (American National Standards Institute), instytut który zajmuje się standaryzacją języków programowania. ANSI jest jedną z najważniejszych organizacji związanych z językami programowania. Zostały tam opracowane prawie wszystkie amerykańskie normy językow programowania. Jedynym językiem, którego oficjalną normę opracowano poza ANSI był JOVIAL (norma Sił Powietrznych USA).

## STANDARYZACJA JĘZYKÓW PROGRAMOWANIA

Autorka zauważyła, że gdy w dyskusji wspomina się o standaryzacji języków programowania, u wielu osób, zajmujących się technicznymi aspektami tych języków natychmiast słabnie zainteresowanie rozmową. Niestety, często nie zdajemy sobie sprawy z tego, że standaryzacja ma w wielu przypadkach dużo większe znaczenie ekonomiczne (a nawet techniczne) dla języków programowania, niż inne rodzaje działalności.

Standaryzacja języków jest procesem długim, nudnym, frustrującym, pełnym trudności i dosyć ryzykownym. Mimo to — jest tym rodzajem działalności, który chroni dziedzinę języków programowania przed szkodliwym indywidualizmem rozwiązań. Zapewnia użytkownikom otrzymywanie odpowiednich narzędzi do pracy, ponieważ producentom oprogramowania dostarcza charakterystyk, które powinny być uwzględniane przy opracowywaniu ich produktów, natomiast użytkownikom daje kryteria oceny tych produktów. Normy upraszczają kształcenie ludzi, opracowywanie programów, wymienność programów i wykonawcow, a także opracowywanie kompilatorów i innych narzędzi programowania.

Głównym problemem standaryzacji języków jest właściwy moment opracowania normy. Normy powstają zazwyczaj zbyt wcześnie lub zbyt późno. Jeśli są ogłaszane za wcześnie, tzn. zanim jeszcze język mógł być szeroko stosowany i modyfikowany w oparciu o doświadczenia, wtedy ze względu na brak tych doświadczeń również słabsze cechy języka zostaną z pewnością ujęte w normie. Z drugiej strony — jeśli norma jest opracowywana dopiero wtedy, gdy język jest już szeroko stosowany, wówczas zwykle istnieje już wiele jego wersji („dialektów") i trzeba toczyć długie boje z ludźmi reprezentującymi odmienne punkty widzenia. Należy bowiem brać pod uwagę prestiż firm, różne rozwiązania techniczne oraz poniesione zazwyczaj duże nakłady finansowe.

Ogólna koncepcja standaryzacji języków ulegała zmianom. Kiedy na początku lat sześćdziesiątych utworzono Komitet ANSI do spraw standaryzacji języków programowania (nazwany potem X3.4) okazało się, że właściwie dla wszystkich był to zupełnie nowy problem. W praktyce okazało się bowiem, że języki programowania są bardziej skomplikowane do znormalizowania niż najbardziej nawet złożone obiekty, których standaryzacją zajmował się dotąd ANSI.

Początkowo zastanawiano się, czy właściwe będzie opracowanie tylko jednej normy języków programowania. Ten pomysł został szybko porzucony jako niepraktyczny z technicznego i taktycznego punktu widzenia, natomiast poważnie zastanawiano się nad stworzeniem norm języków dla poszczególnych dziedzin zastosowań, przy czym pierwszymi miały być dwa obszary zastosowań: obliczenia naukowe i przetwarzanie danych. Z przetwarzaniem danych nie było problemu, ponieważ COBOL był jedynym kandydatem do standaryzacji. Jednak dla obliczeń naukowych istniały już FORTRAN oraz ALGOL 60 i każdy z nich miał swoich zwolenników, którzy byli zdecydowani walczyć o jego standaryzację. Standaryzacja jednego z tych języków nie spowodowałaby oczywiście automatycznie zniknięcia drugiego, ale zorientowano się, że gdy powstanie norma jednego z tych języków, to będzie on cieszył się znacznie większym zainteresowaniem użytkowników niż drugi, bez normy. Ponieważ żadna ze stron nie chciała zrezygnować, Komitet X3.4 — jakkolwiek niechętnie — zdecydował ostatecznie, że należy opracować normy obu języków. Decyzja ta doprowadziła do tego, że w chwili obecnej w ANSI ciągle brak jest odpowiedzi na pytanie, czy słuszne jest dążenie do opracowywania normy konkretnego języka z pominięciem faktu, że istnieją także inne języki przeznaczone dla tej samej dziedziny zastosowań.

W ANSI istnieją odpowiednie grupy robocze, do których mogą być zgłaszane propozycje norm języków. Grupy te bardzo wnikliwie rozważają każdą przedstawioną propozycję. Większość zainteresowanych nie zdaje sobie jednak sprawy, jak bardzo kosztowna jest standaryzacja języka. W związku z tym, zanim Komitet X3 podejmie decyzję o standaryzacji języka, musi mieć pewność, że norma będzie naprawdę użyteczna.

ANSI nie jest jedyną organizacją zajmującą się normalizacją w dziedzinie informatyki. Poza nią istnieją także inne organizacje — zarówno w USA (np. National Bureau of Standards, ISA — Instrument Society of America), jak i innych krajach. Działa także międzynarodowa organizacja ISO (International Standards Organization).

Istnieją trzy sposoby opracowywania normy ANSI.

Jednym z nich jest powołanie przez Komitet X3 grupy roboczej, pracującej zwykle w oparciu o istniejące dokumenty definicji języka. Rezultaty pracy takiej grupy są publicznie ogłaszane jako projekt wstępny normy (ang. Draft Proposed Standard). Następnie grupa zbiera i opracowuje uwagi na temat projektu, ewentualnie go poprawia i kieruje do Komitetu X3 celem przekazania do komisji oceny norm ANSI (Board of Standards Review), która wydaje ostateczną decyzję.

Drugą metodą standaryzacji jest tzw. metoda „dyskusji" (ang. „canvass"). Mamy z nią do czynienia wtedy, gdy jakaś organizacja poza Komitetem X3 podejmuje się opracowania normy języka. Gdy projekt jest gotowy wysyła się go kolejnym opiniodawcom według tzw. „listy uczestników dyskusji" (ang. „canvass list"), aby na podstawie otrzymanych uwag dokonać ewentualnych zmian. Skorygowany projekt jest wysyłany, podobnie jak w pierwszym przypadku, do ANSI, która dodatkowo bada, czy przyjęta „lista uczestników dyskusji" jest dostatecznie reprezentatywna.

Trzecia możliwość tworzenia norm dotyczy organizacji akredytowanych przy ANSI, które mogą zaproponować własną metodę prowadzącą do zatwierdzenia normy przez ANSI.

Przy tworzeniu norm języków: APT, BASIC, COBOL, FORTRAN, PL/I, APL, PASCAL stosowano pierwszy sposób. Do języków MUMPS i ADA zastosowano metodę „dyskusji", natomiast z trzeciej drogi standaryzacji żadna z organizacji akredytowanych przy ANSI dotychczas nie skorzystała. Tabela podaje numery istniejących norm ANSI i ISO oraz lata ich powstania.

**Wykaz norm języków programowania ANSI oraz ISO**

| Język | Symbol normy ANSI[1]) | Symbol normy ISO[1]) |
|---|---|---|
| ADA | rozpoczęto prace | — |
| ALGOL 60 | — | [2]) |
| APL | rozpoczęto prace | — |
| APT | X3.37 — 1977 | — |
| BASIC | X3.60 — 1978 | ISO 6373 — 1980 |
| COBOL | X3.23 — 1968[2]) | ISO 1989 — 1972 |
| | X3.23 — 1974 | ISO 1989 — 1978 |
| FORTRAN | X3.9 — 1966[2]) | ISO 1539 — 1972 |
| | X3.9 — 1978 | ISO 1539 — 1979 |
| BASIC FORTRAN | X3.10 — 1966[2]) | ISO 1539 — 1972 |
| MUMPS | X11.1 — 1977 | — |
| PASCAL | — | DP 7185 — ? |
| PL/I | X3.53 — 1976 | ISO 6160 — 1979 |
| Podzbiór PL/I | X3.74 — 1981? | ISO 6522 — 1981? |

[1]) Końcowy człon symbolu oznacza rok zatwierdzenia
[2]) norma wycofana

W 1980 roku istniały grupy robocze ds. standaryzacji języków APL i PASCAL, a także ds. standaryzacji następujących kategorii języków: języka definicji danych, języka graficznego oraz języka do sterowania systemami operacyjnymi. Najlepiej by było, oczywiście, aby raz opra-

cowana norma pozostawała ciągle aktualna. W praktyce się to nie zdarza, a nawet procedura ANSI przewiduje konieczność rewizji normy co pięć lat (może się to sprowadzić tylko do potwierdzenia prawidłowości istniejącej wersji). Także użytkownicy języka są zainteresowani doskonaleniem norm, ponieważ zdają sobie sprawę z tego, że dobre normy będą wdrażane i szeroko stosowane. Niezbędne są również modyfikacje języka, a więc i normy, uwzględniające rozwój sprzętu (np. obsługę monitorów ekranowych). Zdarza się też, że nowi członkowie grup roboczych reprezentują inne poglądy na temat treści języka niż ich poprzednicy, a w związku z tym wprowadzają zmiany odpowiednie do swoich wymagań.

Jako przykład tego zjawiska autorka omówiła sprawę propozycji, najbardziej chyba kontrowersyjnej zmiany w normie języka FORTRAN. W 1980 roku zaproponowano, aby w następnej wersji tej normy spacje stały się znakami znaczącymi. Była to bardzo radykalna propozycja, ponieważ od momentu powstania FORTRANU główną jego cechą był właśnie brak znaczących spacji, co zdaniem autorki jest fundamentalną cechą składni tego języka. Autorka uważa, że intencją pomysłodawców jest późniejsze wprowadzenie do języka innych zmian, które bez wspomnianej modyfikacji nie są możliwe. Gdyby jednak tę poprawkę zaakceptowano, wówczas każdy istniejący obecnie program można by było uważać za niepoprawny. Zwolennicy FORTRANU odkrywają to, co twórcy PL/I stwierdzili już w 1963, a mianowicie, że FORTRAN nie może być poprawiony bez rezygnacji z pewnych istotnych jego cech.

Procesy standaryzacji języków zostały bardzo skomplikowane przez dążenie do ciągłego ich rozszerzania w kierunku dostosowania do coraz obszerniejszych klas problemów. Najbardziej oczywistymi przykładami tego były BASIC i FORTRAN. Pierwszy z nich był początkowo bardzo prostym językiem, lecz w trakcie jego normalizacji dodano tak wiele elementów, że obecnie trudno uważać BASIC za język bardzo prosty. To samo odnosi się do FORTRANU, a także do COBOLU.

Można oczywiście uważać, że zmiany są robione w imię postępu, bo przecież gdy pojawiają się nowe możliwości i nowe potrzeby, to należy je uwzględnić w podstawowych językach. Według autorki, języki te osiągną jednak z czasem rozmiary i złożoność języków od początku projektowanych jako języki bardzo rozbudowane, a mianowicie PL/I, ALGOL 68 i ADA. Spowoduje to oczywiście konieczność tworzenia nowych, prostych języków. Poza tym proste języki (np. BASIC) rozrastają się zwykle w sposób spontaniczny, a tą drogą nie otrzymuje się języków tak efektywnych, jak te, które od początku projektowano jako języki o szerokich możliwościach.

Za kilka lat należy oczekiwać nowej normy COBOLU oraz rozszerzonego BASICA. Natomiast do 1985 roku nie będzie raczej nowej normy FORTRANU. Gdyby przewidywać na podstawie czasu jaki upłynął pomiędzy opracowaniem pierwszej normy FORTRANU i jej pierwszą rewizją (12 lat), to można spodziewać się nowej wersji normy tego języka dopiero ok. 1990 r. Dla PL/I najważniejszą sprawą jest formalne zatwierdzenie wybranego jego podzbioru, co prawdopodobnie zostanie dokonane w najbliższym czasie. Za kilka lat pojawi się prawdopodobnie nowa norma PL/I, ale nie należy spodziewać się po niej radykalnych zmian, chociaż mogą zostać tam uwzględnione dodatkowe ważne funkcje. W USA i w Wielkiej Brytanii tworzona jest norma PASCALA, której projekt powinien być wkrótce opublikowany. Natomiast dopiero w połowie 1980 została utworzona w USA grupa, która zajęła się opracowaniem normy języka APL, więc rezultatów jej prac można się spodziewać nie wcześniej niż za kilka lat.

W przypadku języków sterowania systemami operacyjnymi, to brak jest podstawowego dokumentu, który można by było traktować jako punkt wyjściowy, nie mówiąc już o bardzo poważnych wątpliwościach, czy tego typu opracowanie może być w ogóle zrealizowane.

Prace nad normami PASCALA i APL rozpoczęto w Europie, po czym ISO wysłało projekt tych norm do ANSI, a dalsze prace nad nimi przejęły komitety amerykańskie. Normy pozostałych języków, oprócz ALGOLU 60, były inicjowane w USA. Przy języku ADA, jako że od początku był on „publicznie" opracowywany, wybrano metodę „dyskusji", której uwieńczeniem powinna być norma ANSI.

**Opracowały:**
**TERESA WÓJCIEKIAN, HALINA CIECHOMSKA**
**na podstawie referatu Jean E. Sammet przedstawionego na konferencji SEAS AM'82**

**W dwóch poprzednich numerach przedstawiliśmy zarys działalności najstarszych przedsiębiorstw sieci ETOB — warszawskiego i krakowskiego, obchodzących obecnie swoje trzydziestolecie. Poniżej prezentujemy jedną z ważniejszych sfer pracy katowickiego ETOBU, dwudziestolatka. (Red.)**

# Oprogramowanie narzędziowe komputera R-32

Katowickie Przedsiębiorstwo Informatyki Przemysłu Budowlanego ETOB dysponuje dużym parkiem maszynowym, w którym króluje stale rozbudowywany i eksploatowany pod nadzorem systemu operacyjnego OS/JS komputer R-32.

Odmienny system operacyjny, w porównaniu do systemów dotychczas eksploatowanych komputerów MIŃSK -32 oraz ODRA-1305, spowodował konieczność odpowiedniego przestawienia technik programowania i projektowania systemów przetwarzania, a co istotniejsze — stworzył dogodniejsze warunki zastosowania programów uniwersalnych realizujących powszechnie występujące funkcje powtarzalne.

Sekcja Technologii katowickiego ETOBU opracowała siedem takich programów. Mimo różnych zastosowań, programy te mają szereg cech wspólnych, a mianowicie:

- wszystkie zachowują standardy programów pomocniczych OS/JS
- ich funkcjami kierują zdania sterujące umieszczane zawsze w zbiorze opisanym zdaniem SYSIN
- charakteryzują się dużą szybkością wykonywania i małą zajętością pamięci operacyjnej (przy średnio skomplikowanych parametrach pojemność wymaganej pamięci rzadko przekracza 30 KB)
- przygotowanie zdań sterujących nie wymaga umiejętności programowania — wystarcza tu znajomość reguł języka opisu zadań
- w przypadku, gdy funkcje programu nie są dla użytkownika wystarczające, ma on możliwość ingerencji w działanie programu przez wprowadzenie własnych procedur.

### Konwertery

Pierwszym problemem, jaki wyniknął z faktu posiadania przez przedsiębiorstwo trzech rodzajów komputerów, była konieczność przenoszenia pomiędzy nimi zbiorów. Spowodowało to powstanie serii następujących programów konwersji zbiorów:

IEBCNVRO (R-32 — ODRA-1305)
IEBCNVOR (ODRA-1305 — R-32)
IEBCNVMR (MIŃSK-32 — R-32).

Struktura logiczna tych konwerterów jest identyczna. Zdania sterujące są przekształcone na ciągi rozkazów (na zasadzie generowania), po czym sterowanie oddawane jest do wygenerowanej części programu.

Rekordy są poddawane konwersji polami, których kolejność i miejsce w rekordzie wyjścia określa użytkownik. Z wyjątkiem zmiennego przecinka, wszystkie typy pól zbioru wejściowego mogą być przekształcone na dowolne formaty pól, dopuszczalne na komputerze, dla którego dany zbiór jest formowany. Ponadto, co wynika z układu zdań sterujących, rekordy mogą być skracane, wydłużane, wypełniane stałymi albo ciągami tego samego znaku (zerowanie, spacjowanie itp.).

Zastosowanie generowania ciągów rozkazów umożliwiło uzyskanie znacznych szybkości przetwarzania, mimo że nośnikiem jednego ze zbiorów (wejścia lub wyjścia) musi być taśma magnetyczna. Konwersja 10 tys. rekordów o 20—30 polach każdy trwa przeciętnie 6—8 minut, co praktycznie jest szybkością samego kopiowania.

Dwa pierwsze wymienione programy zostały opisane już cztery lata temu w INFORMATYCE (nr 12/79). W porównaniu do wówczas przedstawionych rozwiązań, możliwości obu programów istotnie rozszerzono. Główną zmianą w programie IEBCNVRO jest wprowadzenie możliwości przenoszenia programów w językach COBOL i PLAN w formie wymaganej przez kompilatory tych języków na komputerze ODRA. W przypadku pakietu IEBCNVOR, istotną zmianą jest dopuszczenie do konwersji zbiorów na taśmach magnetycznych nie zapisanych w standardzie JS (wcześniejsze modele ODRY nie miały tej możliwości wskutek braku CRC).

### Uniwersalne programy przetwarzania danych

W każdym typowym systemie przetwarzania danych gospodarczych występuje grupa czynności powtarzalnych, które mogą być wykonywane przez te same, odpowiednio plastyczne programy. Czynności te można usystematyzować według następujących rodzajów operacji:

— wczytywanie i kontrola danych
— aktualizacje zbiorów
— wydruki
— selekcja rekordów
— sortowania.

Zgodnie z powyższą systematyzacją, opracowano grupę programów realizujących poszczególne rodzaje operacji. Oczywiście, nie w każdej sytuacji programy uniwersalne mogą znaleźć zastosowanie, dostosowano je bowiem do rozwiązań typowych. Poniżej zostaną podane zwięzłe charakterystyki poszczególnych programów.

### Program IEBCRDR

Jego główną funkcją jest czytanie, kontrola i wstępne przetwarzanie 80-bajtowych rekordów zapisanych znakowo. Ograniczenie rekordu wejściowego do 80 bajtów występuje tylko w przypadku kart dziurkowanych. Jeśli dane znajdują się na innym nośniku, dopuszcza się rekordy do 256 bajtów.

Zbiór wejściowy może zawierać do 25 typów rekordów, zaś każdy rekord można rozpisać na 255 pól. Możliwość ta wynika z faktu, że pola mogą nakładać się na siebie, zaś to samo pole może być definiowane na dowolnie wiele sposobów.

Dla pojedynczego pola można określić:
— postać (numeryczne, tekst)
— sposób wypełniania (pełne, częściowe, pominięte)
— sposób adiustacji
— sposób zapisu znaku liczby (jeśli występuje)
— dopuszczalne wartości (zawarte w tablicy wartości)
— zakres wartości
— relację z innym polem
— podwójną relację z innymi polami lub literalami.

Każde pole rekordu wejściowego może być przeniesione do rekordu wyjściowego w dowolnej postaci (z wyjątkiem zmiennego przecinka), na polach można wykonywać proste operacje arytmetyczne, a ponadto w rekordzie wyjściowym można umieszczać wartości stałe lub wypełniać pola ciągami znaków. Sformowane rekordy mogą być wpisywane do 25 zbiorów wyjściowych. Rekordy, w których stwierdzono błędy, są wyprowadzane na drukarkę (opcjonalnie), ze wskazaniem rodzaju błędu oraz numerem bajtu stanowiącego początek pola, w którym wykryto błąd. Niezależnie od stwierdzenia wystąpienia błędu, rekord jest sprawdzany do końca, lecz nie jest umieszczany w zbiorze wyjściowym. Co najwyżej — również opcjonalnie — można go przenosić do specjalnego zbioru rekordów błędnych.

### Program IEBUPDTN

Jest on przeznaczony do aktualizacji zbiorów sekwencyjnych. Na zbiorze aktualizowanym można przy użyciu tego programu wykonywać następujące operacje:
— dodawanie rekordów
— zamianę całych rekordów

— zamianę w rekordzie pojedynczych pól stałych, albo pól zmiennych, odpowiednio wskazanych w rekordzie aktualizującym
— usuwanie rekordów
— wielokrotne usuwanie rekordów (w przypadku, gdy pola-klucze aktualizacji powtarzają się, jeden rekord aktualizujący usuwa ze zbioru aktualizowanego wszystkie rekordy o tych samych kluczach).

Zbiory aktualizowany i aktualizujący muszą być posortowane według pól zawierających klucze aktualizacji. Program dopuszcza do 255 kluczy aktualizacji, a ponadto — z wyjątkiem zmiennego przecinka — mogą one mieć dowolne formaty. Ich miejsca w rekordzie, długość oraz postać określa sam użytkownik. Także użytkownik definiuje miejsce i wartości wyróżników aktualizacji. Nie jest konieczne, aby rekordy — aktualizowany i aktualizujący — miały identyczą budowę. Odpowiednim zdaniem sterującym można sprowadzić strukturę jednego z nich (lub obu) do tej samej postaci.

W przypadku, gdy liczba rekordów aktualizujących jest niewielka, można pominąć sortowanie obu zbiorów, zaznaczając to w zdaniach sterujących.

Rekord po aktualizacji może również ulegać wydłużaniu, skracaniu, przebudowie, a ponadto można wprowadzać do niego wartości stałe lub wypełniać ciągami znaków.

**Program IEBPRINT**

Jest to uniwersalny program wydawniczy, a jego zestaw zdań sterujących umożliwia realizację większości typowych wydruków jakie występują w systemach EPD.

Poza stroną tytułową i nagłówkami użytkownik definiuje rozmieszczenie pól rekordu, liczbę kluczy sortowania zbioru wejściowego (do 7), rozmieszczenie sum przypadających na pojedynczy klucz (do 20), a — ponadto decyduje o szerokości wydruku, liczbie wierszy na stronie itp.

Istotną zaletą tego programu jest krótki czas przygotowania wydawnictwa. Zredagowanie średnio skomplikowanego tabulogramu, przy pewnej wprawie pisania zdań sterujących, zajmuje około dwóch godzin — co w porównaniu z opracowaniem programu wydawniczego w językach wyższego poziomu daje znaczne oszczędności czasu programowania, a także eliminuje czas kompilacji. Nie bez znaczenia jest również to, że czas realizacji wydawnictwa opracowanego za pomocą programu IEBPRINT na komputerze jest krótszy od czasu realizacji identycznego wydawnictwa przygotowanego programem specjalnie w tym celu napisanym.

**Program IEBSELEX**

Przeznaczony jest do wybiórczego kopiowania rekordów z dowolnego zbioru. Użytkownik definiuje warunki logiczne, jakie ma spełniać wyselekcjonowany rekord, oraz czynności, jakie na nim mają być wykonane. Operatory logiczne OR i AND są przyjmowane przez program automatycznie według określonych zasad. Dopuszczalna liczba zagłębień w warunki logiczne wynosi 25, natomiast maksymalna liczba zestawów warunków — 10. W jednym zestawie warunków — liczba użytych warunków oraz badanych pól i zestawów literali są limitowane wyłącznie pojemnością pamięci maszyny cyfrowej.

Cały zbiór, jego fragmenty lub pojedyncze, wyselekcjonowane rekordy można przenosić do 25 zbiorów wyjściowych. Ponadto rekordy te można wydłużać, skracać, wypełniać stałymi i ciągami znaków oraz zmieniać ich strukturę.

**MULTISORT**

Program ten jest stosowany wówczas, gdy jeden i ten sam zbiór jest kilkakrotnie sortowany według różnych kluczy (np. przy realizacji wydawnictw). Zastosowanie MULTISORTU zamiast programu pomocniczego IERRCO , który w jednym przebiegu może przesortować zbiór według tylko jednego klucza daje znaczne oszczędności czasu komputera. W czasie jednego przebiegu program ten może przesortować zbiór według dziesięciu kluczy, co nie oznacza, że zapisuje on dziesięć zbiorów wyjściowych. Tworzony jest tylko jeden zbiór wyjściowy w specjalnym trybie zapisu, a jego odczyt wymaga procedury dołączanej do programu wydawniczego. Procedurę taką zastosowano w programie IEBPRINT i można ją stosować w programach opracowanych w językach wyższego poziomu.

Zdania sterujące MULTISORTU są takie same, jak zdania SORT programu IERRCO , przy czym — co wynika z powyższego opisu — w zbiorze SYSIN można umieścić do dziesięciu zdań.

* * *

Korzyści wynikające ze stosowania programów uniwersalnych są niewątpliwe, ponieważ:

- w przypadku jakichkolwiek zmian (np. struktur rekordów, wydawnictw itp. odpada czas ponownej kompilacji programów, a ponadto zmiany w zdaniach sterujących są znacznie prostsze, niż korekty w programach
- czas realizacji programów uniwersalnych jest na ogół krótszy niż przy wyspecjalizowanych programach w językach wyższego poziomu
- opracowanie zdań sterujących nie wymaga umiejętności programowania.

**JERZY NOWOSAD**

**Katowickie Przedsiębiorstwo Informatyki Przemysłu Budowlanego ETOB**

# Małe—realnie

Niejednokrotnie na tych łamach postulowano istnienie w polskiej informatyce małych, dynamicznych przedsiębiorstw usługowych. My też uwierzyliśmy, że „małe jest piękne" i — korzystając z Uchwały Rady Ministrów nr 169 — zdecydowaliśmy się (we dwóch) na odejście z pracy w przedsiębiorstwie państwowym i założenie własnej firmy. Po pokonaniu trudności w uzyskaniu koncesji na prowadzenie działalności w dziedzinie oprogramowania, przystąpiliśmy do pracy, w sierpniu 1982.

Zaczęliśmy od reklamy — wysyłając oferty do kilkudziesięciu zakładów przemysłowych dysponujących komputerami RIAD, IBM bądź minikomputerami MERA 300 i MERA 100. Na oferty te nie dostaliśmy odpowiedzi — widocznie wybrane przez nas ośrodki mają informatyków, którzy mogą zaspokoić potrzeby własnego przedsiębiorstwa. Odwiedziliśmy też wiele dużych ośrodków, m.in. URSUS i ZSM—MERA, polecając swoje usługi. Z podobnym rezultatem.

Pomimo naszych konkurencyjnych cen (np. wdrożenie systemu OS na R-32 wyceniliśmy na 125 tys. zł) i krótkiego czasu zrealizowania usługi (np. ORBIS-owi zaproponowaliśmy wykonanie systemu — począwszy od założeń, a skończywszy na próbnej eksploatacji — w terminie ok. trzech miesięcy) nie zdecydowano się na współpracę z nami.

Jednemu z zakładów zaproponowaliśmy zrealizowanie sytemu gospodarki materiałowej za 150 tys. zł. Zakład ten miał wprawdzie własny niewykorzystany sprzęt, jednak dzierżawił system i czas na maszynie w innym przedsiębiorstwie. Za te usługi płacił ok. 30 tys zł miesięcznie. Ale jednorazowy wydatek na proponowany przez nas system „nie opłacał się" kierownictwu.

Jeszcze jednym świadectwem „czasu" jest historia naszej oferty wykonania kontrolera dyskowego i zamiany jednostki dyskowej produkcji zachodniej na podobną, lecz produkcji bułgarskiej. Awarie dotychczasowej jednostki powodowały przerwy w eksploatacji systemu informatycznego, temat więc — wydawałoby się — pilny, a warunki proponowanej przez nas umowy — korzystne dla przedsiębiorstwa. Niestety, i tym razem naszej oferty nie przyjęto.

Mówiąc krótko — w sytuacji obowiązującej obecnie reformy gospodarczej i wynikającego z niej sposobu rozliczania wydatków na informatykę, widoczny jest brak zainteresowania kierownictwa przedsiębiorstw tą działalnością. Szczególnie gdy konieczne są nakłady finansowe. Nawet w sytuacjach zdawałoby się pilnych, kierownictwo zakładu wykazuje powściągliwość — wie, że i tak komputery można zastąpić ludźmi, bądź sprzętem zmechanizowanym.

W dotychczasowej naszej pracy znajdowaliśmy zrozumienie jedynie w małych instytucjach pozbawionych własnych programistów, a mających mało lub wcale nie wykorzystane komputery. Ale i w takich ośrodkach spotykaliśmy się z różnym podejściem ze strony kierownictwa — w zależności od wcześniejszych, często negatywnych doświadczeń z informatyką. Ci szefowie, którzy mieli w przeszłości pozytywne doświadczenia z informatyką z reguły wiedzieli, czego chcą i jakie informacje są i będą im potrzebne; naginali nawet na potrzeby systemów informatycznych istniejącą w zakładzie organizację pracy czy obieg dokumentów. W tych przypadkach byli oni jednak zainteresowani przede wszystkim systemami zunifikowanymi, które mogą być wdrażane i sprzedawane w innych jednostkach gospodarczych.

Najsmutniejsze zaś refleksje o sposobie gospodarowania w odwiedzanych przez nas przedsiębiorstwach nasuwa sposób, w jaki nieoficjalnie uzasadniano niechęć do naszych ofert — to co proponujecie odpowiada nam, ceny i terminy są korzystne, wolimy jednak zapłacić nawet 10 razy więcej firmie państwowej — bo, wiecie, kontakty z prywaciarzami, potem jakaś kontrola, NIK...

Nie istnieje zatem dzisiaj zbyt duże zapotrzebowanie na usługi informatyczne w zakresie oprogramowania. I — jak się wydaje — w tej dziedzinie w ciągu najbliższych lat będziemy mieli do czynienia z postępującą stagnacją. Znowu stracimy kilka lat.

**PIOTR SZEMPLIŃSKI**



# Najnowsze tendencje w dziedzinie komputerowych systemów sterowania

**Kontynuując doroczny przegląd ciekawszych materiałów konferencyjnych dotyczących sterowania przy użyciu komputerów [1]) omówimy ważniejsze zagadnienia przedstawione na kilku konferencjach międzynarodowych, które odbyły się w ubiegłym roku:**

- **IFAC Symposium on Theory and Application of Digital Control (TADC), w New Delhi, 5—7 stycznia**
- **4t IFAC Workshop on Distributed Computer Control Systems (DCCS), w Tallinie, 24—26 maja**
- **1982 CERN School of Computing (CSC), w Zinal, Szwajcaria), 29 sierpnia — 11 września**
- **3rd IFAC/IFIP Symposium on Software for Computer Control (SOCOCO), w Madrycie, 5—8 października**
- **2nd International Symposium on Real Time Data Handling and Process Control (RTD), w Wersalu, 3—5 listopada.**

W poniższym przeglądzie, zwrócono uwagę przede wszystkim na prace mające bliższe odniesienie do praktyki krajowej. Omówiono kolejno zagadnienia architektury, tj. zestawów mikrokomputerowych i systemów rozłożonych przestrzennie, oprogramowania oraz zastosowań.

## ZESTAWY MIKROKOMPUTEROWE

Tematyka ta jest bardzo rozległa i obejmuje zarówno skupione zestawy z pojedynczym procesorem, jak i wielokomputerowe zestawy rozłożone przestrzennie — włącznie z sieciami komputerowymi — omówione w następnym punkcie. Jednym z podstawowych zagadnień w budowie zestawów jest dobór magistrali na poziomie mikrokomputera (jak również na poziomie sieci). Coraz częściej jednak przechodzi się nad tym do porządku dziennego — chyba dlatego, że powszechna dostępność i mały koszt elementów elektronicznych o dużym stopniu scalenia (w krajach wysokouprzemysłowionych) wystarcza do szybkiego zaprojektowania i wykonania dowolnych systemów według określonych założeń — nawet przez samego użytkownika.

Powstaje coś w rodzaju bariery opłacalności normalizacji. Skoro systemy znormalizowane zawierają — ze swej istoty — pewną nadmiarowość, to opłaca się je wytwarzać dopiero przy większym zapotrzebowaniu. Jest to uzasadnione np. przy zestawach instalowanych w Europejskim Centrum Badań Jądrowych CERN, pod Genewą, gdzie — według szacunkowych danych — w najbliższym pięcioleciu będzie potrzeba ok. 2000 kaset i 37000 modułów. W specjalnym raporcie [2]), liczącym ok. 100 stron, zaproponowano stosowanie znormalizowanej magistrali G64 (brak bliższych danych dotyczących budowy i rozkładu sygnałów) dla mikroprocesorów 8-bitowych i magistrali VME [3]) dla 16-bitowych.

Wydaje się, że wybór standardu VME, spośród 5—6 najbardziej popularnych magistrali mikrokomputerowych, jest względnie rozsądny nawet dla mikrokomputerów 8-bitowych. Magistrale S-100 i MULTIBUS (P-696 i P-796) są już przestarzałe. Z kolei prace nad magistralą P-896 [4]) (i jej europejskim wariantem I-896) przeciągają się ponad miarę. Pewną alternatywą byłaby magistrala STD-bus [5]), ale niestety — jest za mało rozpowszechniona w Europie. Poza nowoczesną i nieskomplikowaną budową (cztery szyny: przełączana adresowa, arbitrażowa, przerwaniowa i użytkowa), silną stroną magistrali VME jest poparcie kilku producentów (MOTOROLA, MOSTEK, SIGNETICS-PHILIPS). Jednakże jej wadą w warunkach polskich jest ukierunkowanie na układy mikroprocesora MC

[1]) Poprzednie artykuły z tego cyklu były opublikowane w numerach 9/1980, 11—12/1981 i 2/1983 INFORMATYKI

[2]) Final Report of the Steering Committee for Microprocessor Standardization, CERN, Geneva, 10 December 1982

[3]) Rudyk M.: VME-Bus. Elektronik, nr 10, ss. 90—96, 21 Mai 1982

[4]) Borrill P.: IEEE P896 — the Futurebus Project. Microprocessors and Microsystems, Vol. 6, No. 9, pp. 489—495 (1982)

[5]) Titus C. A., Titus J. A., Larsen D. G.: STD Bus Interfacing. Howard W. Sams, Indianapolis (IN), 1982

68000. Warto wszakże zwrócić uwagę, że już obecnie powstaje szereg układów sprzęgających tę magistralę z innymi systemami, np. CAMAC (E. Barrelet i in., RTD).

Choć możliwe jest prównanie różnych magistrali (E. V. Chernykh, RTD, R. W. Dobinson, CSC), zawsze można wziąć pod uwagę tylko parametry techniczne, bez uwzględnienia czynników pozatechnicznych (dostępność elementów, możliwości projektowe i wykonawcze, zgodność międzynarodowa itp.), które w naszych warunkach odgrywają istotną rolę. Ci, którzy nie mają tych zmartwień, konstruują magistrale do wąskiej klasy zastosowań, jak FASTBUS przeznaczona prawie wyłącznie do automatyzacji eksperymentów w fizyce wielkich energii (H. Muller, RTD), a nawet — „jednorazowe" magistrale eksperymentalne (R. Patzelt, H. Böck, RTD).

Warto uświadomić sobie, że problem wyboru magistrali, choć podstawowy, nie jest jedyny, ani nawet najważniejszy. Nie można tego problemu zlekceważyć, niemniej rozwiązanie go nie załatwi wszystkiego. W cytowanym opracowaniu CERN-u zalecono podjęcie wielu innych działań dotyczących standaryzacji użycia mikroprocesorów, wymieniając przede wszystkim kompleks zagadnień z dziedziny oprogramowania:

— stworzenie standardowych stanowisk roboczych (ang. workstations)

— opracowanie jednolitego oprogramowania skrośnego

— znormalizowanie funkcji jądra czasu rzeczywistego

— przyjęcie jednego, nowoczesnego języka wysokiego poziomu (np. MODULA-2)

— zapewnienie przenośności oprogramowania.

Reasumując — zagadnieniom standaryzacji sprzętu mikroprocesorowego poświęca się wprawdzie nadal wiele uwagi, jednak na wymienionych konferencjach nie były dostatecznie reprezentowane. Przyczyną tego stanu wydaje się być przekonanie, że w tej dziedzinie nie można już sformułować zbyt wielu odkrywczych poglądów lub myśli. W referatach przeglądowych, dotyczących zastosowań mikroprocesorów w systemach sterowania (D.W. Clarke, H. Unbehauen, TADC) również nie podjęto tych zagadnień, poprzestając na pobieżnym omówieniu najdogodniejszych właściwości mikroprocesorów i wskazując na konieczność stosowania języków wysokiego poziomu, standaryzacji sieci i badań nad współpracą człowiek-maszyna.

## SYSTEMY ROZŁOŻONE PRZESTRZENNIE

Problematyka systemów sterowania rozłożonych przestrzennie jest tak ważna, że poświęcono jej odrebne sympozjum (DCCS), a na pozostałych też zajęła wiele miejsca. Jednym z podstawowych problemów dotyczących lokalnych sieci komputerowych jest ich standaryzacja (L. Pouzin, CSC, R. Popescu-Zeletin, RTD, G. G. Wood, RTD/DCCS, T. J. Harrison, DCCS). Choć powstanie znormalizowanej sieci komputerowej, przeznaczonej m.in. do sterowania, jest konieczne — wydaje się, że twórcy norm nie zdawali sobie sprawy, jak wiele kwestii trzeba tu rozstrzygnąć. Do chwili obecnej dokonano stosunkowo wielu uzgodnień ale też wiele pozostaje jeszcze do zrobienia.

Przypomnijmy, że główne prace koncentrują się wokół sieci PROWAY (International Electrotechnical Commission) i P-802 (Institute of Electrical and Electronics Engineers, USA), na poziomie najniższych warstw modelu komunikacji OSI (ang. open system interconnection) opracowanego przez ISO (International Organization for Standardization). Obecnie wiadomo już, na podstawie pierwszych doświadczeń z sieciami lokalnymi, że spośród dwóch wariantów dostępu do ośrodka transmisji według projektu P-802, tzn. dostępu wspólnego z wykrywaniem kolizji i dostępu oznakowego (ang. carrier sense multiple access with collision detection, CSMA/CD i — odpowiednio — token access), dla rozłożonych systemów sterowania lepsza jest druga metoda (R. Hainich, RTD).

Metoda typu CSMA/CD wzorowana na sieci ETHERNET jest niedeterministyczna i nie gwarantuje uporządkowanego szeregowania dostępu, a przez to nie zapewnia wystarczającej niezawodności przy chwilowych przeciążeniach, często spotykanych w przemysłowych systemach czasu rzeczywistego. Jest natomiast bardzo dogodna do zastosowań biurowych. Jednak w przypadku systemów sterowania można względnie łatwo dostosować sieć typu ETHERNET do dostępu oznakowego, wprowadzając dodatkowy protokół na poziomie łącza fizycznego (V. Tschammer, K. Wawer, RTD).

Sieć tego rodzaju, nazwaną DANUBE, zbudowano we współpracy między zachodnioberlińskim Hahn-Meitner-Institut a francuskim centrum badawczym INRIA w Rocquencourt [6]. Ośrodkiem transmisji jest kabel współosiowy o maksymalnej długości 1 km, do którego można dołączyć 255 stacji. Szybkość transmisji wynosi 2 Mb/s. Interesujące jest, że protokoły warstwy fizycznej i warstwy łącza zrealizowane są przez standardowo wytwarzane układy (odpowiednio — tzw. transceiver i sterownik DCB-100 oparty na procesorze typu 8085 i magistrali MULTIBUS), a protokoły wyższych warstw przez mikrokomputer LSI-11/23. Planuje się utworzenie tzw. bramy (ang. gateway) do połączenia sieci lokalnej DANUBE z siecią powszechną HMINET [7] stosującą protokoł X.25.

Siecią lokalną typową dla systemów sterowania jest HYDRA (M. G. Rodd, DCCS) — eksperymentalna, lokalna sieć komputerowa, służąca do celów badawczych i przeznaczona do zastosowania w przemyśle. Ma ona kształt pętli, którą tworzą pary skręcane, przenoszące informacje z szybkością 38,4 kboda. Każdy węzeł sieci składa się z procesora nadrzędnego (mikrokomputer INTEL SDK-80 lub minikomputer VARIAN 620/1) i z procesora komunikacyjnego (INTEL 8080A). Opracowano protokoły dla różnych poziomów transmisji, lecz tylko dwa najniższe są zgodne z modelem OSI/ISO.

Powstanie sieci HYDRA stanowi najlepszy dowód, jak wielka jest potrzeba normalizacji w tym kierunku. W różnych ośrodkach rozwiązuje się te same zagadnienia, co prowadzi do absurdalnej sytuacji — ile ośrodków, tyle rozwiązań. Pewne nadzieje wiąże się z powstaniem dokładnej specyfikacji magistrali PROWAY (W. Ansaldi i in., R. Popescu-Zeletin, RTD), która zapewnia komunikację na obszarze 2 km, z szybkością 1—2 Mb/s i możliwością dołączenia 100 stacji. Dostęp do ośrodka transmisji jest oznakowy (ang. token), tzn. deterministyczny, jednak proponuje się nawet wprowadzenie dostępu kontrolowanego przez użytkownika, czego nie zapewnia żadna z metod projektu P-802.

Ważną cechą tej sieci jest tzw. obsługa bezpołączeniowa (ang. connectionless service), która nie wymaga otwierania połączenia między komunikującymi się procesami, ani potwierdzenia ze strony odbiorcy wiadomości. Warstwowa architektura sieci PROWAY jest w zasadzie zgodna z modelem ISO. Warstwie łącza (ang. link layer) odpowiadają dwa poziomy — ścieżki (ang. path) i magistrali (ang. highway), która spełnia najważniejsze funkcje komunikacyjne. Główne z nich, tzw. funkcje nadzorcy i zarządcy (ang. supervisor, manager) są dublowane w celu uzyskania pożądanej niezawodności. Na poziomie łącza proponuje się przyjęcie standardowego protokołu HDLC, który można rozszerzyć na warstwy ścieżki oraz magistrali — i wydaje się, że każde inne rozwiązanie byłoby szkodliwe dla popularyzacji i rozpowszechnienia sieci PROWAY jako normy. Do komunikacji z sieciami powszechnymi najwygodniejsze byłoby przyjęcie normy X.25.

Wśród tradycyjnych rozwiązań sieci komputerowych przeznaczonych do sterowania zwraca uwagę renesans szeregowego systemu CAMAC, który w Polsce jest, niestety, prawie nieznany. Warto na to zwrócić uwagę, gdyż zestawy camacowskie służą do sterowania niezwykle złożonymi urządzeniami (U. Beyschlag i in., V. Schmidt, RTD), jak wielki akcelerator cząstek, tzw. SPS (ang. Super Proton

---

[6]) Martin M. et al.: Les résaux locaux: définition et exemple de réalisation — DANUBE. Bull. de Liaison de la Recherche en Informatique et Automatique, no. 70, p. 8—12 (1981)

[7]) Baum D.: HMI-Networks I and II. Report HMI-B 325, Hahn-Meitner-Institut, Berlin, Mai 1980

Synchrotron), pod Genewą, czy urządzenie do prowadzenia kontrolowanej syntezy jądrowej, tzw. JET (ang. Joint European Torus), w pobliżu Oksfordu. Obszar, na którym rozłożone są obiekty i sterujące nimi komputery wynosi w pierwszym przypadku 1000 × 200 m² (29 inteligentnych kaset szeregowych w 4 pętlach), a w drugim — ma promień ok. 500 m (30 minikomputerów dołączonych przez 100 kaset CAMAC). Inteligencja sterowników i komputerów musi być dostateczna dla niezawodnego działania systemu, dlatego w obu przypadkach zapewniono dublowanie funkcji, rezerwowe magistrale i odpowiednie oprogramowanie.

W wielu sieciach komputerowych coraz częściej i śmielej stosuje się łącza światłowodowe (V. Schmidt, E. Querasser i in., RTD). Choć w większości przypadków są to sieci eksperymentalne, nie ulega wątpliwości, że włókna światłowodowe mają znaczną przewagę — w sensie parametrów technicznych — nad tradycyjnymi ośrodkami przesyłania informacji.

Mimo że w dziedzinie lokalnych sieci komputerowych nie ma obecnie jednolitych, standardowych rozwiązań, niektóre normy narzucają się same, jak np. kodowanie bifazowe (ang. Manchester coding) na poziomie warstwy fizycznej, protokół HDLC (ang. High-level Data Link Control) dla warstwy łącza logicznego, czy zalecenia ECMA [8]) (European Computers Manufacturers Association) dla wyższych warstw (J. M. Ayache i in, SOCOCO).

## ERA OPROGRAMOWANIA

„Żyjemy w erze oprogramowania. Dla specjalistów jest to oczywiste od kilku lat, jednak obecnie staje się jasne także dla użytkowników, dyrektorów zakładów i ogólnie dla przedsiębiorców" — stwierdził prezydent Stanford Research Institute, instytucji zaangażowanej bardzo poważnie w badania w dziedzinie produkcji oprogramowania (W. F. Miller, TADC). W zakładach Toshiba ok. 2000 inżynierów jest zatrudnionych przy produkcji oprogramowania dla przemysłowych systemów sterowania (K. Takezawa, DCCS). Za przyczynę tej sytuacji uznaje się fakt, że oprogramowanie sprzętu komputerowego jest kluczem do zwiększenia produktywności, zarówno w przemyśle — jak i usługach.

Programowanie komputerów nie jest już tym, czym było dawniej, tzn. kodowaniem programu. Na programowanie składa się obecnie przede wszystkim specyfikacja i projektowanie programów (I. C. Pyle, RTD: programming equals specifying plus designing). Coraz większy nacisk kładzie się na traktowanie programów tak, jak innych wyrobów produkowanych przemysłowo, poddając je dobrze określonym operacjom w trakcie procesu technologicznego. Ten proces nazywa się już powszechnie cyklem rozwojowym oprogramowania. Kodowanie zajmuje w nim względnie mało miejsca. Tak jak w każdym procesie produkcyjnym, na każdym etapie cyklu rozwojowego oprogramowania potrzebne są narzędzia ułatwiające poszczególne operacje. W tym przypadku są to głównie narzędzia programowe. Wiele z nich — kompilatory, asemblery, programy ładujące, systemy operacyjne — stosuje się od lat. Obecnie, wraz z gwałtownym rozwojem metod produkcji oprogramowania, ich liczba niesłychanie wzrasta. Niektóre trudno nawet zidentyfikować na podstawie nazwy. Któż potrafi — na przykład — powiedzieć, co to jest configuration manager?

Według najnowszych badań [9]) liczba zarejestrowanych narzędzi wynosiła 362, jednak ich rzeczywista liczba jest o wiele większa. Pełen zestaw tych narzędzi tworzy to, co określa się nazwą środowiska produkcji oprogramowania (ang. software production environment) lub środowiska programowego języka (J. N. Buxton, CSC). Próbę stworzenia takiego środowiska dla języka FORTRAN, ukierunkowanego na twórców oprogramowania matematycznego i nazwanego TOOLPACK, podjęło konsorcjum (L. Osterweil, RTD) złożone z ośmiu instytucji, w większości przyszłych użytkowników tych narzędzi. Podstawowe idee [10]) tego projektu, dalekiego jeszcze od zakończenia, można tu scharakteryzować jedynie w wielkim uproszczeniu. Dla użytkowników narzędzia powinny stanowić łatwy w użyciu zestaw do automatycznego przetwarzania plików. Dlatego zasadnicze znaczenie, obok innych cech budowy tego środowiska, ma system plików i język poleceń (ang. command language).

Wyjaśnienie konieczności stosowania narzędzi programowych na każdym etapie cyklu rozwojowego programu nie jest trudne. Skoro samo użycie języka PASCAL, zamiast języków asemblera, zmniejsza koszty produkcji programu około pięciokrotnie (W. F. Miller, TADC), to nie ma wątpliwości — że użycie bardziej skomplikowanych narzędzi do automatyzacji programowania ułatwi nam pracę wielokrotnie.

Rzeczywiście, spora liczba rozszerzeń PASCALA (H. Kopetz i in., E. Zarándy, E. Schoitsch, RTD, H. Elmqvist, SOCOCO) do zastoswań w czasie rzeczywistym wskazuje na znaczną potrzebę jednolitego języka wysokiego poziomu, mającego bardziej nowoczesne cechy strukturalne i przystosowanego do wykonywania tego rodzaju zadań. Jednakże, mimo gwałtownego rozwoju języków spełniających te warunki, jak ADA, MODULA-2, PEARL (J. Zalewski, G. Maier, E. Joho, SOCOCO), jest prawie pewne, że przez kilka najbliższych lat w dziedzinie komputerowych systemów sterowania będzie nadal dominował FORTRAN. Przyczyna tego stanu jest łatwa do wyjaśnienia. Doskonałość budowy języka nie jest w żadnym stopniu czynnikiem decydującym o jego użyciu w konkretnym zastosowaniu. Najważniejszym bodaj czynnikiem, który o tym decyduje, jest łatwość użycia języka, a pod tym względem FORTRAN ma kolosalną przewagę.

Dokładnie połowa wszystkich istniejących narzędzi jest napisana w FORTRANIE [9]) (druga połowa przypada na wszystkie inne języki). Nie należy więc dziwić się stwierdzeniom, że „FORTRAN będzie z nami jeszcze przez wiele lat" (M. C. Crowley-Milling, CSC: FORTRAN will be with us for many years to come) i to używany właśnie w systemach sterowania (F. Gagliardi i in., RTD). Charakterystyczne, że nawet dla tak złego języka jak BASIC tworzy się oprogramowanie narzędziowe, bo trudno znaleźć inną alternatywę ze względu na jego prostotę (D. Williams, RTD).

Oczywiście, stosowanie metod inżynierii oprogramowania do produkcji programów już od fazy specyfikacji (L. Motus, K. Kääramees, SOCOCO/DCCS, C. Kohler, RTD) musi uwzględniać bardziej formalne podejście do każdej z faz cyklu rozwojowego oprogramowania. W granicznym przypadku może to całkowicie wyeliminować pisanie programów dla systemów sterowania — tak jak rozumie się je dzisiaj. Dominująca stanie się faza specyfikacji wymagań, które sformalizowane posłużą do wygenerowania programu dla komputera docelowego. Jeden z takich systemów, nazwany EPOS (Engineer and Process-Oriented Support) powstał dla języka PEARL (V. Scheub, RTD).

Referaty wygłoszone na wymienionych konferencjach dotyczyły, oczywiście, także wielu problemów bardziej szczegółowych i to niezwykle ważnych w całej dziedzinie programowania. Tutaj można je jednak tylko wymienić. Przykładowo — poszczególne sesje sympozjum SOCOCO były poświęcone w całości zagadnieniom oprogramowania robotów (J. C. Latombe, CSC), komputerowemu projektowaniu układów sterowania (J. Davidson, DCCS), współpracy człowieka z maszyną (L. David i in., RTD) czy programowaniu systemów rozłożonych przestrzennie (H. Kopetz, RTD, R. A. Trakhtengerts, DCCS). Dostrzega się niezwykle ważną rolę baz danych w komputerowych systemach sterowania (H. Schweppe, RTD, M. C. Crowley-Milling, CSC) i kontynuuje się prace dotyczące różnych aspektów niezawodności tych systemów (J. R. Tayler, RTD, M. C. Rodd, B. Krzykacz, SOCOCO).

---

[8]) European Computer Manufacturers Association: Transport Protocol. Standard ECMA-72, January 1981, Geneva

[9]) Houghton Jr., R. C.: Software Development Tools — A. Profile. Computer, Vol. 16, No. 5, pp. 63—70 (1983)

[10]) Osterweil L.: Software Environment Research — Directions for the Next Five Years. Computer, Vol. 14, No. 4, pp. 35—43 (1981)

**ZASTOSOWANIA (na przykładzie sterowania eksperymentami jądrowymi)**

Powyższe omówienie światowych tendencji może rodzić wątpliwości, czy magistrale mikrokomputerów, lokalne sieci komputerowe lub inżynieria oprogramowania są tak ściśle związane ze sterowaniem, aby wypełnić treść całego artykułu. Odpowiedź dają zastosowania. Wskutek gwałtownego rozwoju techniki komputerowej i zmniejszania się kosztów sprzętu, znacznie zwiększyła się skala problemów. Zagadnienia komputerowego sterowania pojedynczych układów stopniowo ustępują miejsca problematyce sterowania przy użyciu bardzo złożonych systemów wieloprocesorowych. Z drugiej strony — nawet stosunkowo nieskomplikowany, mikrokomputerowy układ sterowania może być bardzo trudny do zrealizowania ze względu na wymagania niezawodnościowe (D. Tabak, SOCOCO).

Okazuje się, że w przypadku zwiększenia skali problemów zawodzą dotychczasowe metody projektowania komputerowych systemów sterowania. Typowym przykładem, z którym się zetknąłem, jest system sterowania spektrometrem neutronów przy użyciu komputera, którego standardowe oprogramowanie składało się z kilku procedur wykonujących poszczególne funkcje: obliczanie położeń i ustawianie ramion spektrometru, wykonywanie pomiaru i przedstawianie wyników. Takim samym spektrometrem sterowano w innym ośrodku (D. G. Dimmler, DCCS) przy użyciu pięciu mikrokomputerów, stanowiących węzły jednego ze skupień (ang. cluster) większej sieci. Oprogramowanie mikrokomputerów składało się z niezależnych modułów, oddzielnie napisanych w FORTRANIE. W obu przypadkach należałoby napisać oprogramowanie niezależne od fizycznej struktury układu sterowania, określone jedynie przez wymagania funkcjonalne użytkownika. Poszczególnym funkcjom odpowiadałyby np. zadania w języku ADA, a odpowiednie narzędzia programowe dokonałyby alokacji tych zadań na zadaną konfigurację (np. od 1 do 5 komputerów) w sposób „przezroczysty" dla użytkownika.

Niemal wszystkie problemy wymienione w poprzednich punktach są na porządku dziennym przy projektowaniu niezwykle skomplikowanego systemu sterowania eksperymentalnym zespołem urządzeń syntezy jądrowej NOVA, zainstalowanym w Lawrence Livermore Laboratory (USA), którego celem jest wskazanie możliwości przemysłowego generowania energii przy wykorzystaniu kontrolowanej reakcji termojądrowej (G. J. Suski i in., DCCS). Dla uzmysłowienia sobie stopnia złożoności systemu sterowania, warto podać, że jego zadaniem jest kontrola 10 promieni laserowych skupiających energię 100—150 kJ w impulsie o szerokości 3 ns, przebywających drogę ok. 200 m, zanim trafią w cel o średnicy kilku mm.

W celu realizacji zadań sterowania cały system podzielono na cztery podsystemy odpowiadające naturalnemu podziałowi urządzenia na podsystemy — generacji promienia, formowania promienia, diagnostyki plazmy i obserwacji próbki — realizujące np. sterowanie krótkimi impulsami prądowymi o amplitudach 20—30 MA z dokładnością do mikrosekund, sterowanie 1000 silników krokowych, zbieranie danych ze 100 czujników dostarczających cyklicznie ok. 300 tys. bajtów informacji lub — ze 120 przyrządów dostarczających jednorazowo $10^7$ bajtów informacji.

System sterowania, który oprócz wymagań funkcjonalnych spełnia wymagania względnie małego kosztu, niezawodności i modyfikowalności, składa się z ok. 50 mikrokomputerów LSI-11/23, trzech centralnych komputerów VAX-11/780 i procesora macierzowego FPS-120B (produkcji FLOATING POINT SYSTEMS, USA). Na poziomie przyrządów, a więc niższym od poziomu mikrokomputerów, stosuje się magistrale CAMAC i IEEE-488 (GPIB), natomiast komputery połączone przy użyciu włókien światłowodowych tworzą sieć lokalną zwaną NOVANET, w której każdy komputer ma w dowolnej chwili dostęp do dowolnego innego i jego urządzeń. Protokoły sieci spełniają wymagania trzech najniższych warstw modelu OSI/ISO. Oprogramowanie całego systemu, napisane w języku PRAXIS [11]), zawiera ok. 150 tys. linii kodu źródłowego. Jak twierdzą autorzy — nie zostało ono napisane w ADZIE tylko dlatego, że pierwsze narzędzia tego języka nie pojawiły się dostatecznie wcześnie, aby można ukończyć projekt NOVA.

Ogromnej liczby zagadnień, które należało rozwiązać, nie można tu, oczywiście, nawet wymienić. Warto jedynie zwrócić uwagę na niekonwencjonalne problemy współpracy operatora z systemem (centralne funkcje zobrazowano na czterech monitorach sterowni) i zadania stojące przed bazą danych tak złożonego systemu (zrealizowane oczywiście przez system relacyjny). Zarządzanie w tym systemie polega — mówiąc w uproszczeniu — na kontrolowaniu i pielęgnacji poszczególnych wersji modułów oprogramowania oraz na rejestrowaniu i kontrolowaniu zmian konfiguracji sprzętu, jak np. połączeń komputerów sieci, zestawu bloków w kasetach czy kolejności operacji.

Budowa i ewolucja takich systemów trwa na ogół kilka lat, co można prześledzić na przykładzie wielkiego akceleratora SPS (M. C. Crowley-Milling, CSC). Korzystając z tych doświadczeń, w pracach nad systemem sterowania nowego akceleratora LEP opracowano szereg zaleceń, które z różnych względów mogą się wydać interesujące. Dzięki wykorzystaniu mikroprocesorów, moc obliczeniowa całego systemu sterowania ulega rozłożeniu co najmniej na kilkaset urządzeń zainstalowanych na całym obszarze akceleratora. Wskutek znacznego zwiększenia liczby jednostek obliczeniowych, wzmocnieniu ulega wymaganie zgodności i wymienności, a więc także standaryzacji, poszczególnych modułów tych jednostek. Wynika stąd konieczność przyjęcia jednolitej magistrali mikrokomputerowej.

Oczywistą konsekwencją rozprzestrzeniania komputerów jest utworzenie sieci lokalnej, łączącej poszczególne jednostki na kilku poziomach. Obecnie oba wymagania (tzn. standardowa magistrala i sieć lokalna) są zrealizowane przez inteligentne sterowniki w systemie CAMAC. Kilkuletnie doświadczenia współpracy operatorów z komputerami wskazały przydatność uniwersalnych konsoli, bez konieczności zróżnicowania funkcji na kilka monitorów, co stoi w sprzeczności z wymaganiami innych dużych obiektów (zakładów chemicznych, elektrowni jądrowych), gdzie — jak się uważa — poszczególne konsole powinny przedstawiać informacje selektywnie i permanentnie, a nie tylko na żądanie.

Bardzo zaskakującym doświadczeniem z eksploatacji tego systemu sterowania było także stwierdzenie, że najodpowiedniejszym językiem do sterowania jest interpreter. Stworzono nawet specjalny język NODAL [12]), wzorowany na BASICU i FOCALU. Spośród kilku przyczyn mających wpływ na ten fakt, najważniejsze wydaje się żądanie łatwego programowania i testowania zestawów przez użytkowników, tj. fizyków. Ze względu na ogrom zadań w zakresie oprogramowania, nie można w rozsądnym czasie sformułować pełnej specyfikacji systemu, która umożliwiłaby napisanie programów zawodowym programistom. Oczywiście, sam interpreter, systemy operacyjne i całe oprogramowanie podstawowe musi być napisane w nowoczesnym języku wysokiego poziomu, jak np. ADA, choćby ze względu na konieczność zapewnienia niezawodności i przenośności oprogramowania czy na ograniczenia czasowe.

Jednym z ważniejszych osiągnięć autorów tego systemu sterowania wydaje się określenie organizacji bazy danych. Rolę bazy danych rozumie się jako wyodrębnienie danych z programów — tak, aby przy częstych zmianach danych unikać modyfikacji programów. W systemach wieloprocesorowych sytuacja komplikuje się przez konieczność wyboru między scentralizowaną a rozproszoną bazą danych. W systemie SPS opracowano

---

[11]) Greenwood J. R. et all.: An Introduction to PRAXIS, Report UCRL-52957; Programming in PRAXIS, Report UCID-18995, Lawrence Livermore National Laboratory, 1980

[12]) Crowley-Milling M. C., Shering G. C.: The NODAL System for the SPS. Report CERN-78-07, European Organization for Nuclear Research, Geneva, 1978

koncepcję modułów danych[13]), które stanowią elementy bazy danych zawierające programy obsługi (ang. driver) odpowiednich urządzeń i rezydujące w poszczególnych procesorach.

* * *

Może się wydawać dziwne, że w artykule o systemach sterowania nie ma w ogóle mowy o algorytmach. Taka jest jednak tendencja, że — mimo gwałtownego rozwoju teorii sterowania — w praktyce wymagania dotyczące algorytmów, choć nie we wszystkich dziedzinach, stanowią ok. 10% wszystkich wymagań narzuconych na system sterowania. Przykładowo — bezpośrednie sterowanie cyfrowe jest jednym z bardzo wielu elementów systemu sterowania na poziomie użytkowym. Znacznie ważniejsze są zagadnienia podstawowe dla komputerów: systemy operacyjne i języki programowania (umożliwiające niezawodną pracę w czasie rzeczywistym) oraz narzędzia programowe, stwarzające warunki do efektywnego programowania i działania systemów sterowania. Dominująca jest tendencja do decentralizacji funkcji tych systemów, wynikająca z coraz większych możliwości mikrokomputerów, które — choć często rozproszone geograficznie — na ogół stanowią zintegrowane zestawy w postaci sieci komputerowych.

Spośród zagadnień istotnych, lecz nie omówionych powyżej, warto zwrócić uwagę na intensywne prace nad wykorzystaniem procesów wektorowych, macierzowych i innych systemów o niekonwencjonalnej architekturze (W. K. Giloi, RTD, V. Zacharov, CSC). Choć — podobnie jak superkomputerów — w dającej się przewidzieć przyszłości nie będziemy mieli w Polsce tak wielkich systemów sterowania, jak NOVA czy SPS, warto przyjrzeć się im bliżej. A to z tego względu, że przy ich realizacji szczególnie ostrej weryfikacji ulegają dotychczasowe poglądy i koncepcje dotyczące komputerowych systemów sterowania, jak również krystalizują się nowe idee, których przypuszczalnie nie można by tak wyraźnie sformułować rozważając systemy o znacznie mniejszym stopniu komplikacji. Słowem — nawet jeśli skala największych krajowych przedsięwzięć jest w najlepszym razie o rząd wielkości mniejsza, nie wątpię, że z zapoznania się ze światowymi osiągnięciami może płynąć i dla nas ważna nauka.

**JANUSZ ZALEWSKI**

[13]) Crowley-Milling M. C.: The Data Module — the Missing Link in High Level Control Languages, pp. 63—65, Trends in On-line Computer Control Systems, IEE Conf. Publ. 172, London, 1979

# Nauczyć słonia stepować

**Spotkanie było pilnie strzeżoną tajemnicą. W czerwcu 1980 zespół badawczy z placówki IBM w Boca Raton udał się do Seattle na rozmowy z Billem Gatesem, szefem niewielkiego przedsiębiorstwa produkującego oprogramowanie (8 mln dol. obrotu rocznego). Po podpisaniu umowy o dochowaniu ścisłej tajemnicy, ze strony zespołu IBM padła propozycja: koncern zbliża się do końca prac nad komputerem osobistym mającym konkurować z wyrobami APPLE i RADIO SHACK i chciała by Gates (miał wtedy 24 lata) napisał system operacyjny dla nowej maszyny określonej wówczas kryptonimem „Chess". Oferta była bezprecedensowa. Nigdy przedtem gigant (wówczas 26 mld dol. obrotu rocznego) nie zdecydował się, by tak skromna firma odegrała ważną rolę w opracowaniu jednego z jej komputerów. Ryzyko opłaciło się. W ciągu następnych 10 miesięcy MICROSOFT CORP., kierowana przez Gatesa, zaprojektowała cały system operacyjny i podstawowe programy dla IBM PERSONAL COMPUTER. Maszyna zadebiutowała w sierpniu 1981 i natychmiast zawojowała rynek. W bieżącym roku sprzedaż osiągnie poziom 600 tys. sztuk, zaś w 1984 prawdopodobnie przekroczy milion.**

Sukces IBM PC stworzył również nowy rynek dla niezależnych producentów oprogramowania i małych wytwórców sprzętu; trwa wyścig nowych wyrobów dla PC i całych, kompatybilnych z IBM komputerów. Jednocześnie APPLE i RADIO SHACK, które dwa lata temu kontrolowały lwią część rynku komputerów osobistych, coraz szybciej tracą grunt.

Sukces IBM w dziedzinie komputerów osobistych — to tylko jeden z elementów agresywnego ataku na cały rynek. W ciągu ostatnich dwóch lat firma całkowicie zreorganizowała swoje gigantyczne siły w dziedzinie sprzedaży i marketingu, a także gruntownie odnowiła strategię produkcji, badań i cen, by utrzymać swoją pozycję w batalii z amerykańskimi, japońskimi i europejskimi konkurentami o każdą część rynku komputerów. Od 1976 roku IBM zainwestował 25 mld dol. w nowe zakłady i urządzenia, zwiększając swoje możliwości i — jak powiedział jeden z pracowników firmy — „pokrywając Amerykę nowymi fabrykami i siecią punktów sprzedaży". Według szefa koncernu, Johna R. Opla: „reorganizacja miała ten sam cel co inwestycje, których dokonano w ostatnich latach — nadać IBM pozycję pozwalającą osiągnąć korzyści z burzliwie rozwijającego się rynku".

Wyniki świadczą, że cel ten jest osiągany — w ubiegłym roku obroty skoczyły do 34 mld dol., przy zyskach na poziomie 20%. Analitycy rynku przewidują podobny 20-procentowy zysk również w tym i następnym roku. „W zeszłym roku — powiedział Frank Gens z bostońskiej Yankee Group — przemysł był świadkiem narodzin nowego IBM i konkurencja staje się brutalna". Osiągnięte i przewidywane rezultaty koncernu znalazły także swoje odbicie na Wall Street, czyli na giełdzie nowojorskiej — ocenia się nawet, że dotąd żadne akcje nie miały takiego wpływu na kształtowanie się ogólnego wskaźnika akcji przemysłowych (Dow Jones).

Większość agresywnych elementów w nowej strategii IBM pojawiła się w styczniu 1982, gdy Departament Sprawiedliwości umorzył trzynastoletnie śledztwo o łamanie ustawy antytrustowej przeciwko komputerowemu gigantowi. „Śledztwo to miało ogromny wpływ na IBM" — powiedział analityk Ulric Weil z Morgan Stanley. „Większość działalności koncernu była zagrożona. Pod groźbą wyroku o przymusowym podziale, IBM działał ostrożnie i nawet w 1979 roku spadły jego dochody. Ale gdy oskarżenie wycofano, koncern rozpoczął niczym niezmącony marsz na podbój świata. IBM obniżał ceny niektórych wyrobów przed decyzją 28 stycznia, ale robił to w sekrecie. Wiadomość o nich rozchodziła się z ust do ust." — kontynuuje Weil — „Agresywna polityka cenowa wybuchła w całej pełni dopiero po antytrustowym zwycięstwie". Polityka obniżenia cen miała wielki wpływ na konkurentów. W ubiegłym roku AMDAHL CORP., producent kompatybilnych z IBM dużych komputerów, zanotował obniżkę obrotów; MAGNUSON COMPUTER SYSTEMS zbankrutował, a NATIONAL ADVACEND SYSTEMS zawarł porozumienie o sprzedaży japońskiego sprzętu HITACHII — rezygnując z bezpośredniej konkurencji z IBM.

Równolegle z bardziej agresywną strategią cen, IBM wprowadził sporo zmian w ulubionym sposobie działania. Koncern zawsze własnymi siłami projektował, produkował i wprowadzał na rynek swoje wyroby. Natomiast IBM PC zbudowano prawie w całości z podzespołów pochodzących

od innych producentów. Podobnie, sercem procesora tekstowego IBM DISPLAYWRITER jest mikroprocesor pochodzący z firmy INTEL CORP. — czołowego wytwórcy półprzewodników, a IBM INSTRUMENTS COMPUTER SYSTEMS używa procesora MOTOROLA 68000, tego samego, którego APPLE użył w nowym komputerze LISA. „IBM dokonał przełomu psychologicznego" — podsumował analityk Peter Wright z Gartner Group.

Koncern poszedł nawet dalej: ogłoszono niedawno podjęcie trzech wspólnych przedsięwzięć z japońskimi firmami i wspólnego działnia marketingowego z ARTIFICAL INTELLIGENCE CORP. Jednocześnie IBM zakupił 12% udziałów w INTEL CORP. i 15% w ROLM CORP. — głównego producenta najnowocześniejszych central telefonicznych.

Większość ekspertów oczekuje rychłego wchodzenia koncernu w nowe sektory rynku. „IBM chce stać się główną siłą w każdej dziedzinie rynku związanej z telekomunikacją i przetwarzaniem danych" — powiedział Gens z Yankee Group. Koncern zaprezentował swój PC w roli urządzenia sterującego roboty przemysłowe, przygotowuje też nowy, tańszy komputer osobisty (kryptonim „Peanut"), przeznaczony do wykorzystywania w nauczaniu i w domu. Rozwija sprzęt do badań naukowych i projektowania inżynierskiego.

Wkrótce IBM powinien stać się główną siłą na rynku sprzętu telekomunikacyjnego, walcząc szczególnie z firmą AT&T. Rozwój nowych technologii obejmuje m.in. plazmowe ekrany, mające zastąpić dotychczasowe lampy kineskopowe, a także postęp w dziedzinie techniki montażu pakietów w dużych komputerach, tzw. „thermal conduction module" (moduły przewodzące ciepło). W kołach przemysłowych mówi się też o tym, że IBM sformował zespół 25 naukowców, którzy mają zaplanować kontratak przeciwko japońskim planom piątej generacji (por. INFORMATYKA 7/82, 11/83).

Jeszcze przed zwycięstwem w procesie antytrustowym, IBM rozpoczął głębokie zmiany w celu przystosowania się do szybkiego rozwoju rynku. Gdy zdecydowano w czerwcu 1980 o opracowaniu własnego komputera osobistego, koncern powołał w tym celu jednostkę, którą nazwano Independent Business Unit (IBU). Grupie dano nie spotykaną dotąd swobodę, by ominąć biurokrację koncernu; zespół PC miał własne służby projektowe, marketingowe i handlowe. IBU — jak powiedział były jego szef Frank T. Cary — „to odpowiedź IBM na pytanie: jak nauczyć słonia stepować".

Zespół PC w Boca Raton w trakcie swych prac przestudiował sukces firmy APPLE (a jednocześnie używał do tego projektu komputerów APPLE II) i zdecydował powielić jej strategię włączenia niezależnych wytwórców oprogramowania w budowę sukcesu swojego komputera osobistego. „Zauważyli, że jest to dla nich korzystne, jeżeli nie będą wchodzić w drogę dynamicznym, twórczym producentom programów" — powiedział Mitch Kapor, prezes LOTUS DEVELOPMENT CORP., autor programów analizy finansowej LOTUS 1,2 i 3, przeznaczonych dla IBM PC — „Pozwolili działać nam w nasz własny sposób".

Podczas gdy producenci programów dla IBM PC zgarniają pieniądze na nagle stworzonym rynku, konkurentom giganta w dziedzinie sprzętu robi się gorąco. Charles I. Peddle, założyciel i prezes VICTOR TECHNOLOGIES INC. (kompatybilne z IBM PC mikrokomputery VICTOR S1) twierdzi, że aby mówić o równej grze, IBM musiałby mu oddawać jedną trzecią swoich pieniędzy wraz z częścią personelu handlowego i co trzeci dzień użyczać swojej nazwy. Ale nawet bez tego sprzedaż VICTORA gwałtownie wzrasta — z 65 mln dol. w całym zeszłym roku do 55 mln dol. w pierwszym kwartale roku bieżącego. Na razie IBM nie może sam sprostać zamówieniom na swój PC, zaś jego czcigodna obecność w dziedzinie komputerów osobistych nobilitowała cały ten przemysł. Dlatego też oczekuje się, że sprzedaż APPLE przekroczy w tym roku wartość 1 mld dol., a TANDY, HEWLETT-PACKARD i DEC są również „na fali".

Sukces komputera osobistego ma przynieść IBM dodatkowe zyski. „Mamy tu do czynienia z metodą konia trojańskiego" — skonkludował Gens z Yankee Group. Sprzedając duże ilości PC wielkim przedsiębiorstwom, koncern tworzy nowy rynek. „Kierownicy siedzą przy swoich samodzielnych PC, ale już wkrótce odczują potrzebę podłączenia do głównego komputera. Nagle 300 użytkowników chce korzystać z dużej maszyny i przedsiębiorstwo musi powiększyć jej moc" — mówi Gens.

Chociaż APPLE ma nadzieję na opanowanie rynku wprowadzając swój komputer LISA (por. INFORMATYKA 11/83), to jednak wie, że siła IBM jest zbyt duża, by ją po prostu zignorować. W kwietniu APPLE przedstawił program opracowany wspólnie z CULLINET CORP., który umożliwia współpracę komputerów APPLE z dużymi maszynami IBM. APPLE zaprezentował także dwie lokalne sieci, pozwalające na połączenie komputerów biurowych ze sobą i na komunikowanie się ich z dużym komputerem.

Niektórzy konkurenci przyjmują wzrastające znaczenie IBM na rynku mikrokomputerów z nieukrywanym niepokojem. „IBM chce wszystkiego. Jego polityka to odczekanie aż dany rynek stanie się dostatecznie duży i wkroczenie z własnym modus operandi" — twierdzi Gene M. Amdahl, jeden z czołowych konstruktorów komputerów IBM, który opuścił koncern w roku 1970, by założyć firmę AMDAHL, a później TRILOGY ( i obecnie w informatyce światowej działają osobno Amdahl — człowiek i AMDAHL — firma).

Wszyscy jednak pamiętają, że warunki zmieniały się diametralnie od czasów, gdy APPLE zaczynał swą działalność w garażu w Palo Alto. Dziś i IBM, i APPLE produkują komputery osobiste na wysokowydajnych, zautomatyzowanych taśmach montażowych, a najważniejsza jest obniżka kosztów wytwarzania. „Jesteśmy agresywni w realizacji naszych celów, które postawiliśmy przed sobą, szczególnie celu stania się producentem o niskich kosztach" — mówi Allen J. Krowe, wiceprezes IBM ds. finansowych i planowania. Z tego samego powodu IBM usprawnił swoje służby handlowe i marketingowe; obecnie kontakty klienta z koncernem prowadzi stale ten sam zespół.

Sukces komputera osobistego był tak duży, że obecnie IBM zamierza zastosować tę samą strategię w innych burzliwie rozwijających się dziedzinach rynku. Założono już nowe IBU, które będą działać w obszarach systemów biomedycznych, aparatury analitycznej, automatyzacji produkcji, nauczania, nowych usług komputerowych i sprzętu telekomunikacyjnego, a zwiększone wydatki na badania i rozwój już w ubiegłym roku wyniosły 2,6 mld dol.

Potrzeba ciągle nowych opracowań wynika z chęci utrzymania 20% rocznego wzrostu zysków i — aby to stało się możliwe — koncern musi być jeszcze bardziej operatywny i zróżnicowany w strukturze. Jednocześnie wyścig ten jest wspierany także przez coraz szybszy postęp technologiczny i coraz krótszy cykl życia wyrobów. Już w najbliższych miesiącach IBM wprowadzi do sprzedaży nowy model sieci komputerowej, a nieco później — opracowaną przy udziale firmy ROLM nową centralę telefoniczno-komputerową.

W najbliższych latach czeka koncern ostra walka o panowanie w wieku informacji z AT&T, firmami japońskimi i tysiącami innych mniejszych i większych konkurentów. „Trudno zatrzymać szarżującego słonia" — twierdzi Peter Wright z Gartner Group. Chyba nawet trudniej niż nauczyć go stepować.

**Oprac. MAREK SOBCZYK**
**na podst. NEWSWEEK**
**z 11.07.1983**

Targi Lipskie

# Sprzęt informatyczny RWPG

**Sprzęt komputerowy wystawiony na Międzynarodowych Targach Lipskich (wiosna 1983) pozwala sądzić, że kraje socjalistyczne włączają się do światowego rozwoju mikroelektroniki. Przejawiło się to najwyraźniej w ekspozycjach Węgier i NRD.**

**Oferta węgierska obejmowała przede wszystkim nowoczesny wielozadaniowy sprzęt minikomputerowy oraz specjalizowane terminale komputerowe.** Wśród nich najciekawsze pozycje to:

- **Minikomputer SM 52/10** (zależnie od wykonania — kompatybilny z maszynami EC 1011 lub SM-4), zbudowany na bazie mikroprocesora 16-bitowego. Jest wyposażony w następujący zestaw urządzeń zewnętrznych: monitor ekranowy, pamięć taśmowa, pamięć dyskowa (50 M bajtów), dysk stały (5 M bajtów), dysk elastyczny oraz drukarki: wierszową i znakową.
- **Komputer biurowy VT 20/IV** z bogatym oprogramowaniem systemowym i aplikacyjnym, wyposażony w pamięć RAM o pojemności 64 K bajtów oraz ROM o pojemności 6 K bajtów.
- **Terminale VDT serii 52** dostosowane do pracy zarówno w trybie „on-line", jak i autonomicznym. Terminale te składają się z monitora ekranowego i klawiatury. Poszczególne modele przystosowane są do wykonywania określonych funkcji. VDT 52 127 jest specjalizowanym systemem przetwarzania tekstów (urządzenia tego typu są zainstalowane m.in. w radzieckiej agencji prasowej TASS); może on pracować zarówno w oparciu o alfabet łaciński jak i cyrylicę. VDT 52 128 służy do wykonywania analogicznych funkcji jak model 127, z tą jednak różnicą, że wyposażony jest dodatkowo w pamięć na dysku elastycznym. VDT 52 121 jest graficznym systemem wspomagania inżyniera-konstruktora; wyposażony jest w monitor graficzny (340 × 480 punktów) oraz ma możliwość przyłączenia plotera.
- **ORDAS** — system przygotowywania danych na taśmie magnetycznej, wyposażony w osiem stanowisk operatorskich z monitorami ekranowymi typu ADP 2001. Ponieważ wprowadzanie danych z pełną szybkością absorbuje procesor centralny zaledwie w 20—25%, pozostała część mocy obliczeniowej może być wykorzystywana do przetwarzania w trybie lokalnym pod kontrolą systemu operacyjnego OS lub do transmisji danych do dużego komputera.

**Równie bogatą i interesującą, choć nieco odmienną ofertę złożyła NRD.** Do ciekawszych nowości informatycznych z tego kraju niewątpliwie należą:

- **A6471—A6473,** system przetwarzania obrazów przeznaczony do szybkiego, interaktywnego przetwarzania informacji z możliwością bezpośredniego zobrazowania pośrednich i ostatecznych wyników przetwarzania na graficznych, kolorowych lub czarno-białych monitorach ekranowych. System jest szczególnie przydatny do przetwarzania informacji meteorologicznych, kartograficznych obrazów Ziemi (łącznie ze zdjęciami lotniczymi i satelitarnymi), analiz termo-, tomo- i sonograficznych, obrazów mikroskopowych, kontroli jakości materiałów itp.
- **K7823** — urządzenie, bodaj pierwsze w krajach socjalistycznych, przeznaczone do rozpoznawania i identyfikacji mowy ludzkiej. Niestety nie podano, czy i kiedy urządzenie to będzie seryjnie produkowane.
- **K8924** terminal bankowy przeznaczony do obsługi operacji finansowych i pieniężnych w bankach, kasach oszczędnościowych i w urzędach pocztowych.
- **TD40, 1152/255, 6310** — miniaturowe drukarki znakowe i mozaikowe stanowiące uniwersalne urządzenia wyjściowe dla mini- i mikrokomputerów, a także komputerów osobistych.

**Mniej interesujące niestety były ekspozycje informatyczne pozostałych krajów RWPG,** choć i tu można było znaleźć sporo nowości.

**W pawilonie ZSRR wystawiony został EC 1065 — komputer bardzo dużej mocy z serii RIAD II.** Szybkość jednostki centralnej tej maszyny wynosi 4 mln operacji na sekundę, a pojemność pamięci operacyjnej — 16 M bajtów. System ten wyposażony jest w dyski magnetyczne o pojemności 100 i 200 M bajtów.

**W pawilonie bułgarskim, tradycyjnie już dominowały pamięci dyskowe.** Najciekawszym eksponatem tej grupy sprzętu była pamięć dużej pojemności EC 5067 (200 M bajtów). Prezentowano także inne produkty bułgarskiego przemysłu komputerowego, takie jak:

- system przetwarzania danych tekstowych ISOT 1024 C
- terminal bankowy ISOT 1025 C
- system rejestracji danych na taśmie magnetycznej EC 9005 zbudowany w oparciu o mikroprocesor 16-bitowy. Maksymalna konfiguracja systemu obejmuje: 32 monitory ekranowe, 4 jednostki pamięci taśmowej oraz drukarkę.

**Firma ICE—FELIX-ELECTRONUM z Rumunii zaprezentowała minikomputer uniwersalny CORAL 4011.** System operacyjny MINOS oraz pamięć operacyjna o pojemności 256 K bajtów — pozwalają na stosowanie takich podstawowych języków programowania, jak: MACRO-ASSEMBLER, FORTRAN IV, FORTRAN IV PLUS, COBOL, BASIC—PLUS—2 i PASCAL.

**W ekspozycji polskiej żadnych rewelacji niestety nie można było dostrzec.** Tradycyjnie już prezentowaliśmy:

- drukarki wierszowe (tym razem jednak nowy model 401 — EC 7033M)
- minikomputer MERA 60 (wariant 26-B) z pamięcią operacyjną 28 K słów
- drukarki znakowe serii DZM-180
- terminale ekranowe.

Pozostaje tylko wyrazić nadzieję, że nasz przemysł komputerowy zdoła otrząsnąć się z marazmu i na następnych targach zaprezentuje wreszcie systemy konkurencyjne, choćby ze sprzętem pochodzącym z krajów sąsiedzkich.

**Oprac. RYSZARD GRZESIAK**
na podstawie
„Rechentechnik Datenverarbeitung"
nr 5/83

# Komputer osobisty jako system pomiarowy

Jak już niedawno pisaliśmy, każdy szanujący się komputer osobisty ma obecnie specjalne otwory i przyłącza, umożliwiające bezpośrednie dodawanie do niego płytek z układami elektronicznymi. I chociaż zamysłem producenta było rozszerzanie w ten sposób rozmiarów pamięci operacyjnej czy umieszczanie sterownika urządzeń dyskowych, to w tej chwili można powiedzieć, że „dziura zrodziła przemysł". Jednym z ciekawszych rozwiązań tego typu jest produkowany od niedawna przez firmę NORTHWEST INSTRUMENT SYSTEMS zestaw pozwalający przekształcić mikrokomputery APPLE II lub IIe w system pomiarowo-testowy.

Podstawą tego systemu jest moduł aScope 85, który zmienia komputer w programowany oscyloskop z pamięcią i zakresem częstotliwości do 50 MHz. Dodatkowo producent dostarcza odpowiednie oprogramowanie umożliwiające zapisywanie pomiarów w pamięci dyskowej (w formie graficznej), przekształcanie wyników i spełnianie funkcji woltomierza cyfrowego; stanowi ono bazę dla programów pisanych przez użytkownika.

Zastosowania systemu znacznie rozszerzają inne moduły: aGen 65 — programowany generator funkcji (przebiegi sinusoidalne, trójkątne, prostokątne i impulsy o częstotliwości do 5 MHz i amplitudzie do 20 V) oraz aSource 75 i 76 — programowane źródło napięcia wyposażone we własną pamięć kształtu przebiegu (1 lub 4 K po 12 bitów) — częstotliwość do 10 MHz, amplituda do 20 V. Główną zaletą modułów aSource jest dostarczany wraz z nimi Waveform Editor, czyli program kształtujący przebiegi według podanych równań funkcji albo operacji matematycznych na zapamiętanych przebiegach, np. poprzez ich zsumowanie.

Następną generacją przyrządów pomiarowych NORTHWEST INSTRUMENT SYSTEMS stanowi μAnalyst 2000 — analizator stanów logicznych. Rozmiary tego urządzenia nie pozwoliły na bezpośrednie umieszczenie w obudowie komputera, zdecydowano się więc na swoistą rekurencję. Fabryczna „dziura" służy tylko do połączenia z systemem, który zawiera — oprócz własnego zasilacza i płytki interfejsu — miejsce na sześć płytek — modułów pomiarowych, dobieranych stosownie do potrzeb użytkownika.

**Oprac. MkS**

**na podstawie**

**ELECTRONICS z 7.04.1983**

# IFAC

**IFAC, IFIP, NCC, BCS, AFCET — od lat używamy tych skrótów, przyzwyczailiśmy się do nich, ale... właśnie, nie zawsze już wiemy, co się za nimi kryje. A ponieważ pojawiające się od czasu do czasu rozwinięcia nazw też nie zawsze wszystko wyjaśniają, postanowiliśmy w ciągu kilku najbliższych miesięcy zaprezentować najważniejsze organizacje międzynarodowe i krajowe, działające w informatyce lub dziedzinach pokrewnych. (Red.)**

International Federation of Automatic Control (Międzynarodowa Federacja Sterowania Automatycznego), założona we wrześniu 1957, jest międzynarodową federacją krajowych organizacji reprezentujących naukowe i techniczne środowiska, a zajmujących się sterowaniem automatycznym w poszczególnych krajach.

Oficjalnymi językami IFAC są: angielski, francuski, niemiecki, hiszpański i rosyjski, przy czym angielski jest również językiem roboczym.

Celem Federacji jest wspieranie rozwoju nauki i techniki w dziedzinie teorii sterowania automatycznego i szeroko rozumianych zastosowań we wszelkich systemach: technicznych, fizycznych, biologicznych, społecznych i gospodarczych. Zajmuje się ona także wpływem sterowania automatycznego na społeczeństwo. Z założenia wyklucza się wszelkie działania polityczne i gospodarcze. Swoje cele IFAC realizuje poprzez organizowanie konferencji, wydawanie publikacji i wszelkie inne przedsięwzięcia, które pomagają w wymianie i obiegu informacji w tej dziedzinie.

Co trzy lata zwołuje się międzynarodowy kongres, a pomiędzy nimi Federacja organizuje wiele sympozjów i spotkań poświęconych konkretnym dziedzinom sterowania automatycznego. Wszelkie informacje o tych wydarzeniach ukazują się w czasopismie IFAC NEWSLETTER dostarczanym bezpłatnie przez sekretariat Federacji. Głównym oficjalnym pismem IFAC jest AUTOMATICA, którego prenumeratę można zamówić w wydawnictwie Pergamon Press. Publikacje obejmują również zbiory referatów z kongresów, sympozjów i konferencji, monografie i broszury przedstawiające tematykę i organizatorów przyszłych spotkań.

Federacja odgrywa aktywną rolę w życiu międzynarodowym, tworząc wiele ekspertyz dla krajów ONZ i innych organizacji. IFAC ma status konsultacyjny w UNESCO i UNIDO, a także utrzymuje ścisłe związki z Międzynarodową Agencją Energii Atomowej, Europejską Komisją Gospodarczą ONZ, Międzynarodowym Biurem Informatyki, Międzynarodowym Instytutem Stosowanej Analizy Systemów, Międzynarodową Federacją Inżynierii Medycznej i Biologicznej.

IFAC jest również członkiem FIACC (Five International Associations Coordinating Committee) zrzeszającego poza nią Międzynarodową Federację Przetwarzania Informacji (IFIP), Międzynarodową Federację Towarzystw Badań Operacyjnych (IFORS), Międzynarodowe Stowarzyszenie Zastosowań Matematyki i Komputerów w Symulacji (IMACS) oraz Międzynarodową Konfederację Pomiarów (IMEKO).

Najwyższą władzą Federacji jest Zgromadzenie Ogólne składające się z przedstawicieli organizacji krajowych (nasz kraj reprezentuje Polski Komitet Pomiarów i Automatyki NOT), z których każda ma równe prawa i jeden głos. W styczniu 1982 IFAC zrzeszała 40 takich organizacji.

Pomiędzy spotkaniami Zgromadzenia Ogólnego Federacją kieruje Rada, której podlegają: Rada Techniczna i Rada Wykonawcza. Pierwsza z nich koordynuje pracą 14 Komitetów Technicznych (niektóre z nich dzielą się jeszcze na Grupy Robocze), a druga — trzech stałych Komitetów. IFAC reprezentuje Przewodniczący.

Najważniejszymi wydarzeniami w życiu Federacji są organizowane co trzy lata Światowe Kongresy, grupujące za każdym razem 1—2 tys. uczestników. Ich miejscem jest zawsze kraj, z którego pochodzi Przewodniczący urzędujący podczas trzyletniej kadencji poprzedzającej Kongres.

Dotychczasowe Kongresy:
I Kongres, 1960 — Moskwa (ZSRR)
II Kongres, 1963 — Bazylea (Szwajcaria)
III Kongres, 1966 — Londyn (Wlk. Brytania)
IV Kongres, 1969 — Warszawa (Polska)
V Kongres, 1975 — Boston/Cambridge (USA)
VII Kongres, 1978 — Helsinki (Finlandia)
VIII Kongres, 1981 — Kyoto (Japonia).
Przewiduje się odbycie IX Kongresu w Budapeszcie (Węgry) w lipcu 1984, a X — w 1987 w Monachium (RFN).

IFAC Secretariat
Schlossplatz 12
A-2361 Laxenburg
Austria
telefon: (02236) 71-447
teleks: 79248 ifac a

**Oprac. MkS**

# Słownictwo z zakresu inżynierii oprogramowania

Omawiając w poprzednim numerze terminy zdefiniowane w normie „IEEE Standard Glossary of Software Engineering Terminology" — związane z niezawodnością oprogramowania, rozważyliśmy pojęcie **błędu** i terminy z nim związane, nie definiując samego pojęcia **niezawodności.** W normie tej zdefiniowano niezawodność (za ISO 2382/XIX) jako zdolność programu do wykonywania wymaganej funkcji przez określony czas, w określonych warunkach. Takie brzmienie definicji jest konsekwencją jej koniecznej zgodności z definicją niezawodności w ogóle (słowo „program" zastępuje słowo „jednostka funkcjonalna").

Inna definicja, podana w tej samej normie, brzmi następująco: niezawodność oprogramowania (ang. software reliability) jest to prawdopodobieństwo, że oprogramowanie nie spowoduje awarii systemu przez określony czas, w określonych warunkach. Mimo że ta druga definicja wydaje się bardziej skomplikowana od pierwszej, jest bardziej precyzyjna i całkowicie z nią zgodna. Zdolność programu do wykonania funkcji, czyli prawdopodobieństwo jej wykonania, zastąpiono prawdopodobieństwem, że program nie spowoduje awarii, czyli prawdopodobieństwem, że nie dojdzie do niewykonania funkcji.

Z pojęciem **niezawodności** wiąże się szereg innych właściwości oprogramowania zdefiniowanych w omawianej normie. Należy do nich tzw. **dyspozycyjność** (ang. availability), którą określa się jako prawdopodobieństwo, że oprogramowanie będzie zdolne do wykonania funkcji zgodnych z przeznaczeniem systemu, gdy nastąpi żądanie jego użycia. Na niezawodność i dyspozycyjność oprogramowania ma wpływ wiele innych cech, do których należą m.in. poprawność i odporność oprogramowania. Przez **poprawność** (ang. correctness) rozumie się stopień, w jakim oprogramowanie jest wolne od defektów projektowych i defektów kodowania. **Odporność** (ang. robustness) jest to stopień, w jakim oprogramowanie może kontynuować poprawne działanie, pomimo wprowadzenia nieważnych bodźców (ang. inputs).

Choć każdą z wymienionych cech niełatwo określić w sposób ilościowy, wprowadzono kilka innych, jeszcze trudniej poddających się ocenie. Przykładowo — **testowalność** (ang. testability) zdefiniowano jako stopień, w jakim oprogramowanie ułatwia ustalenie kryteriów testowania i ocenę samego oprogramowania ze względu na te kryteria. Natomiast przez **pielęgnowalność** (ang. maintainability) rozumie się łatwość, z jaką oprogramowanie można pielęgnować. Należy dodać, że **pielęgnacją oprogramowania** (ang. software maintenance) nazywa się modyfikację produktu (wyrobu) programowego po jego dostawie, najczęściej w celu poprawienia defektów, a także — w celu polepszenia wydajności (ang. performance) lub w celu dostosowania wyrobu do zmienionego środowiska. **Wydajnością** w węższym sensie — nazywa się zdolność systemu lub podsystemu komputerowego do wykonywania jego funkcji. Inną cecha, która dotyczy w równym stopniu sprzętu i oprogramowania, jest — różnie określane po polsku — **tolerowanie defektów** (ang. fault tolerance), co oznacza wewnętrzną zdolność systemu do nieprzerwanego poprawnego działania w obecności ograniczonej liczby defektów sprzętowych lub programowych.

Dwie cechy, spośród zdefiniowanych w normie, dotyczą szczególnego aspektu użytkowania systemów komputerowych, mianowicie — ochrony dostępu. Przez **nienaruszalność** (ang. integrity) rozumie się stopień, w jakim w systemie komputerowym można kontrolować nieupoważniony dostęp do oprogramowania i danych lub — ich modyfikację. **Zabezpieczenie** (ang. security) oznacza ochronę sprzętu komputerowego i oprogramowania przed przypadkowym lub złośliwym dostępem, użyciem, zmodyfikowaniem, zniszczeniem lub ujawnieniem.

Wszystkie wymienione właściwości oprogramowania — bowiem jego głównie dotyczą — mają wpływ na podstawową cechę tego oprogramowania, tj. jakość (ang. software quality). **Jakość oprogramowania** jest to zespół cech i charakterystyk wyrobu (ang. software product), które wpływają na jego zdolność do spełnienia określonych wymagań. Kontrolą jakości (ang. quality assurance) nazywa się planowe i systematyczne działania niezbędne do stwierdzenia, że wyrób spełnia ustalone wymagania techniczne. Warto dodać, że kontroli jakości oprogramowania poświęcono specjalną normę IEEE Std. 730-1981 „IEEE Standard for Software Quality Assurance Plans", w której określono 13 elementarnych działań tworzących kompleksowy, choć nie wyczerpujący, plan kontroli jakości oprogramowania. Projekt tej normy, prawie nie odbiegający od ostatecznej wersji, omówiono w łatwo dostępnym artykule F. Buckley'a — A. Standard for Software Quality Assurance Plans, Computer, Vol. 12, No. 8, pp. 43—49 (1979).

Na zakończenie należy stwierdzić, że niektóre z podanych powyżej polskich odpowiedników terminów angielskich, np. **pielęgnowalność, nienaruszalność** itd., mogą wzbudzać pewne wątpliwości. Jednakże w sytuacji, gdy znaczenia niektórych pojęć częściowo pokrywają się (np. performance i quality), trudno o dobór trafnych nazw polskich.

**JANUSZ ZALEWSKI**

***Konferencje***

# Inżynieria oprogramowania

Stowarzyszenie informatyków francuskich AFCET (Association Française pour la Cybernétique, Economique et Technique) organizuje w dniach 4—6 czerwca 1984 r. w Nicei drugą konferencję na temat inżynierii oprogramowania. Organizatorzy zapraszają do nadsyłania referatów o objętości do 15 stron maszynopisu (6 kopii) w terminie do 15 listopada 1983 r. Szczegółowy program konferencji zostanie rozesłany w marcu 1984 r.

Pod pojęciem „inżynieria oprogramowania" organizatorzy konferencji rozumieją działalność dotyczącą:
— opracowywania założeń, analizy, projektu, kodowania, testowania, scalania, rozdzielania, eksploatacji i konserwacji programów
— kierowania realizacją oprogramowania (planowanie, organizowanie zespołu, kotrola kosztów i rozliczeń)
— określania i mierzenia jakości produktów programowych
— metod, języków i innych narzędzi programowania oraz tworzenia warunków mających na celu poprawę jakości oprogramowania
— teoretycznych podstaw badania oprogramowania
— problemów ekonomicznych, ergonomicznych, psychologicznych, prawnych i socjologicznych produkcji i sprzedaży oprogramowania.

Szczególnie preferowane są referaty zawierające:
- opis praktycznych doświadczeń z dokładną analizą przyczyn sukcesu lub niepowodzenia
- prezentację nowych koncepcji i pomysłów w dziedzinie inżynierii oprogramowania
- opis zastosowań, których nowatorstwo polega nie tyle na zastosowanej koncepcji, ile na metodzie wdrożenia
- wyniki pomiarów cech oprogramowania
- prace teoretyczne, jeśli ich celem jest pomoc w rozwiązaniu problemów praktycznych.

Jako imprezy towarzyszące przewidziane są:
- wystawa-targi oprogramowania narzędziowego
- seminaria na temat podstawowych problemów inżynierii oprogramowania.

Dalszych informacji udziela:

**2nd CONFERENCE ON SOFTWARE ENGINEERING**
**A.F.C.E.T., 156, Boulevard Péreire, F. 75017 Paris, France.**

■ Dwaj najwięksi użytkownicy komputerów w Wielkiej Brytanii, Urząd Telekomunikacyjny i Ministerstwo Obrony, postanowili sfinansować prace nad pakietem wspomagania produkcji programów w językach CHILL i ADA. CHILL (CCITT High-Level Language) jest językiem oficjalnie zalecanym przez Międzynarodowy Komitet Konsultacyjny Telegrafii i Telefonii, stworzonym do podobnych zastosowań co ADA. CHAPSE (CHILL and ADA Programming Support Environment) jest świadectwem tego, że oba te języki przeznaczone głównie do celów programowania w czasie rzeczywistym cechuje duże podobieństwo, a jednocześnie — że nie musi grozić monokultura ADY.

*

■ Pionierska w dziedzinie oprogramowania dla mikrokomputerów firma VISICORP (San Jose, Kalifornia) obchodzi pięciolecie swojego istnienia. Z tej okazji rozpocznie się sprzedaż opracowywanego od ponad dwóch lat pakietu Visi-On, który pozwala wykorzystywać jednocześnie różne programy Visi (VisiText, VisiCalc itd.) na jednym stanowisku pracy, przy użyciu techniki wielokrotnych „okienek" sterowanych za pośrednictwem „myszki". Pierwsza wersja jest przeznaczona dla komputera osobistego IBM, ale wkrótce pojawią się odpowiedniki dla wyrobów firm TEXAS INSTRUMENTS, WANG, XEROX i HONEYWELL. Ten typ oprogramowania nie będzie nowością tylko dla użytkowników APPLE LISA, w którym uczyniono z niego podstawę całego systemu. Jednocześnie VISICORP obchodzi drugi jubileusz — sprzedano 500 tys. egzemplarzy legendarnego już dziś programu modelowania finansowego VISICALC. Dziś producent proponuje czwartą wersję, o rozbudowanym wyjściu graficznym, nie zmieniając ceny w stosunku do wersji podstawowej.

*

■ Wyścig mikroprocesorów 32-bitowych rozpoczęty. Jako pierwszy dokonał oficjalnej prezentacji swojego modelu Z8000 ZILOG. Jednak prawdziwe dostawy handlowe rozpoczną się dopiero w połowie 1984 r. Nowy procesor będzie zawierał ok. 150 tys. tranzystorów, zaś jego cena — 150 dol. USA przy zakupie co najmniej 100 sztuk. Natomiast pierwszy na rynku — już od stycznia 1984 — pojawi się NATIONAL SEMICONDUCTOR 32032. Również na początku roku oczekiwany jest mikroprocesor MOTOROLA 68020. Te trzy konstrukcje zostaną wykonane w technologii NMOS, natomiast INTEL (najpóźniej na rynku) będzie wykonywał swój 80386 technologią CMOS. Oczekuje się, że za pomocą tych procesorów osiągnie się poziom szybkości minikomputera DEC VAX/750, tzn. 0,5—5 mln operacji na sekundę. Jak zwykle przy wprowadzaniu nowych wyrobów, pojawia się problem zgodności z poprzednimi konstrukcjami i oprogramowaniem dla nich stworzonym. Najprościej rozwiązano tę sprawę dla MC 68020 i NS 32032 — po prostu 16-bitowe poprzedniki tych firm były wewnętrznie 32-bitowe i w związku z tym zmiany są nieznaczne — nie przewiduje się także potrzeby tworzenia nowych rodzin układów towarzyszących, gdyż wystarczą dotychczas istniejące.

*

■ Do tradycji wspomnianych wyżej światowych kongresów IFIP należy łączenie ich z wystawami. Oczywiście we Francji towarzyszył mu XXXIV salon SICOB (21—30 września) — najbardziej prestiżowa, o najbogatszych tradycjach wystawa sprzętu informatycznego i z dziedzin pokrewnych. Oprócz ogólnodostępnej części, od pięciu lat organizuje się w jej ramach dział OEM — przeznaczony wyłącznie dla klientów profesjonalnych. Spotykają się tam przedstawiciele producentów komputerów, urządzeń peryferyjnych czy podzespołów z reprezentantami przedsiębiorstw usług informatycznych i wielkich użytkowników. W tym roku swoje stoiska urządziło 98 wystawców (w porównaniu z 92 — w poprzednim roku) na 3400 m² powierzchni hali wystawowej CNIT pod Paryżem. Najważniejsze wydarzenia salonu wyznaczyły kierunki rozwoju sprzętu w okresie od poprzedniej ekspozycji — były to przede wszystkim nowe konstrukcje pamięci na dyskach magnetycznych (a wśród nich Winchester o średnicy 3,5 cala), a także coraz wygodniejsze, o coraz lepszych parametrach monitory ekranowe i drukarki, których podstawowe własności tłumaczy tytuł relacji „Bądź piękna i milcz" (znane powiedzenie francuskie). Jak co roku, postaramy się niebawem szerzej przedstawić SICOB na naszych łamach.

*

■ Ciągłe poprawianie parametrów układów scalonych dotyczy nie tylko ich szybkości czy pojemności pamięci. W SANDIA NATIONAL LABORATORIES (Albuquergue, USA) konstruktorzy ukończyli pracę nad pierwszym układem scalonym VLSI, który może poprawnie funkcjonować po otrzymaniu dawek promieniowania spotykanych w przestrzeni kosmicznej lub w czasie wybuchu jądrowego. Wykonany technologią CMOS układ statycznej pamięci RAM zawiera ponad 100 tys. tranzystorów przy wymiarach zewnętrznych 4 × 6 mm. Jego projektanci twierdzą, że może on funkcjonować nawet po otrzymaniu dawki 1 mln radów (zazwyczaj dawka 1 tys. radów jest niebezpieczna dla człowieka). Układ, który przeszedł już pomyślnie testy, będzie sprzedawany przez znaną firmę HARRIS.

*

■ Nowa wersja systemu operacyjnego CP/M, zawarta w pamięci ROM, będzie pierwszym wyrobem rynkowym firmy DIGITAL RESEARCH INC. Przeznaczono ją do wykorzystania w 8-bitowych komputerach zbudowanych z procesorów 8080, 8085 lub Z80. „Krzemowy" CP/M, wraz z interfejsem programowym Visual Information Processor (także w pamięci ROM), będzie sprzedawany przez utworzony w sierpniu 1983 oddział produktów rynkowych. Takie rozwiązanie systemu operacyjnego umożliwi jego wykorzystanie w takich komputerach bez pamięci dyskowych, a interfejs programowy pozwoli tworzyć niezależne od typu procesora oprogramowanie zastosowaniowe, zapisywane również w pamięci ROM. PERSONAL CP/M, bo tak nazwano nową wersję, jest bez wątpienia odpowiedzią na powstały we wspólnym przedsięwzięciu firmy MICROSOFT i Japończyków standard MSX.

*

■ Inicjatywę DIGITAL RESEARCH podchwyciła firma ZILOG, ogłaszając, że nowa wersja popularnego mikroprocesora Z80 będzie miała wewnątrz swojej struktury pamięć ROM z zapisanych PERSONAL CP/M. Przewiduje się, że Z80—CP/M będzie wykorzystany nie tylko w tanich komputerach, ale również znajdzie uznanie wśród producentów sprzętu laboratoryjnego i sterującego, którzy będą mogli wyeliminować pamięć dyskową, konieczną dziś do ładowania systemu.

*

■ Przewidując niedobory programistów piszących w języku ADA, firma COULD INC. pokazała na Federalnej Konferencji Informatycznej (Waszyngton, 14—15 września 1983) Learning Environment, czyli komputerowy system wspomagający nauczanie tego języka. Jak twierdzi John L. Donnelly, specjalista z Oddziału Systemów tej firmy, decyzja Departamentu Obrony USA ustalająca, że od lipca 1984 wszystkie nowe programy do celów wojskowych będą pisane w ADZIE, powinna wywołać duże zainteresowanie systemem.

*

■ Gene M. Amdahl, jeden z najsłynniejszych konstruktorów komputerów, m.in. IBM System 360, po opuszczeniu IBM, a potem własnej już firmy AMDAHL, założył w 1980 roku ACSYS (Amdahl Computer Systems). Firma ta, przemianowana rok później na TRILOGY, postawiła sobie za cel opracowanie własnej technologii układów bardzo wielkiej skali integracji (VLSI). Oczywiście w przypadku całkowicie nowego przedsiębiorstwa oznacza to, że przez kilka lat będzie ono potrzebowało pieniędzy, nie przynosząc żadnych zysków. Zakładając ACSYS, Amdahl powiedział: „Moja nowa firma potrzebuje kapitału w wysokości 100 mln dolarów, by w roku 1985 wprowadzić na rynek komputer do zastosowań naukowych o parametrach lepszych niż jakikolwiek system IBM czy innej firmy, zgodny z nim. I dlatego też już w marcu 1981 CII-HB (obecnie po prostu BULL) za sumę 10 mln dol. stał się właścicielem wówczas 10% tej firmy. W roku bieżącym najpierw koncern SPERRY kupił 15% udziałów (co obecnie kosztowało już 42 mln dol.), a potem DIGITAL EQUIPMENT 9% (za 26 mln). Wszyscy ci udziałowcy, oprócz partycypowania w przyszłych zyskach, zapewniają sobie dostęp do nowoczesnej technologii układów VLSI, opracowywanej przez TRILOGY. A uruchomienie ich produkcji jest coraz bliższe — w Irlandii zbudowano już fabrykę (pozyskując nieco funduszy od rządu tego kraju), a DEC ma zamiar wznieść wraz z TRILOGY podobny obiekt w Kalifornii.

**Skorupski A.: Rozwój systemów mikroprocesorowych**

INFORMATYKA 1983, nr 12, s. 3

Klasyfikacja systemów mikroprocesorowych oraz omówienie światowych tendencji ich rozwoju. Bardziej szczegółowo scharakteryzowano mikroprocesorowe systemy modułowe, z uwzględnieniem osiągnięć i zamierzeń krajowych w tym zakresie.

**Скорупски А.: Развитие микропроцессорных систем**

INFORMATYKA 1983, № 12, стр. 2

Классификация микропроцессорных систем и обсуждение мировых направлений их развития. Более подробно охарактеризованы модульные микропроцессорные системы с учетом отечественных достижений и намерений в этой области.

**Piotrowski A. J.: System iAPX 432**

INFORMATYKA 1983, nr 12, s. 4

Charakterystyka systemu mikroprocesorowego iAPX 432, nowego produktu firmy INTEL, stanowiącego przełom w technice mikroprocesorowej. Podkreślono cechy architektury systemu, odróżniające go od dotychczasowych konstrukcji.

**Пётровски А. Й.: Система iAPX 432**

INFORMATYKA 1983, № 12, стр. 4

Характеристика микропроцессорной системы iAPX 432, нового продукта фирмы INTEL, представляющего собой перелом в микропроцессорной технике. Подчеркнуты свойства архитектуры системы, отличающие ее от прежних конструкций.

**Grabowski J.: Porównanie właściwości mikroprocesorów 16-bitowych (2)**

INFORMATYKA 1983, nr 12, s. 9

Druga część porównania mikroprocesorów ZILOG 8001/2, INTEL 8086, MOTOROLA 68000, TEXAS INSTRUMENTS 9900, LSI 11 i NATIONAL SEMICONDUCTOR 16032 z punktu widzenia podstawowych grup rozkazów oraz efektywności programowania.

**Грабовски Й.: Сравнение свойств 16-битовых микропроцессоров (2)**

INFORMATYKA 1983, № 12, стр. 9

Вторая часть сравнения микропроцессоров ZILOG 8001/2, INTEL 8086, MOTOROLA 68000, TEXAS INSTRUMENTS 9900, LSI 11 и NATIONAL SEMICONDUCTOR 16032 с точки зрения основных команд и эффективности программирования.

**Przemysłowy FORTRAN czasu rzeczywistego. Część 3**

INFORMATYKA 1983, nr 12, s. 13

Trzecia, ostatnia część charakterystyki projektu międzynarodowej normy, rozszerzającej zastosowanie języka FORTRAN. Omówiono wywołania operacji na ciągach dwójkowych i bitach, urządzeniach wejścia-wyjścia oraz plikach.

**Промышленный FORTRAN реального времени. Часть 3**

INFORMATYKA 1983, № 12, стр. 13

Третья, последняя часть характеристики проекта международного стандарта расширяющего применения языка FORTRAN. Обсуждены вызовы операций над двоичными последовательностями и битами, а также операций относящихся к устройствам ввода/вывода и файлам.

**Liszyński Z.: Programowanie procesów współbieżnych (2). Środowisko programowe**

INFORMATYKA 1983, nr 12, s. 17

Druga część omówienia prac badawczych prowadzonych w Politechnice Poznańskiej, zmierzających do stworzenia możliwości równoczesnej realizacji wielu interakcyjnych systemów użytkowych na rozbudowanej konfiguracji komputera ODRA 1305. Scharakteryzowano rozwiązania programowe prowadzące do tego celu.

**Лишиньски З.: Программирование параллельных процессов (2). Программная среда**

INFORMATYKA 1983, № 12, стр. 17

Вторая часть обсуждения исследовательских работ, проводимых в Познаньском Политехническом Институте, целью которых является создание возможностей одновременной реализации нескольких интерактивных пользовательских систем для расширенной конфигурации ЭВМ ODRA 1305. Охарактеризованы программные решения, ведущие к этой цели.

**Skorupski A.: Development of microprocessor systems**

INFORMATYKA 1983, No. 12, p. 2

Classification of microprocessor systems and presentation of world-wide trends of their development. Modular microprocessor systems and home achievements and planes in this area are characterized.

**Skorupski A.: Entwicklung der Mikroprozessorsysteme**

INFORMATYKA 1983, Nr. 12, S. 2

Klassifikation der Mikroprozessorsysteme und Besprechung der Welttendenzen ihrer Entwicklung. Genauer wurden modulare Mikroprozessorsysteme mit Rücksicht auf einheimische Ergebnisse und Pläne charakterisiert.

**Piotrowski A. J.: iAPX 432 system**

INFORMATYKA 1983, No. 12, p. 4

Characteristics of the iAPX 432 microprocessor system, a new INTEL product, which is revolutionary in microprocessor technology. New system's architecture features, different from actual constructions, are pointed out.

**Piotrowski A. J.: iAPX 432 Mikroprozessorsystem**

INFORMATYKA 1983, Nr. 12, S. 4

Eine Charakteristik von iAPX 432 Mikroprozessorsystem, einem neuen INTEL-Produkt, das ein Durchbruch in Mikroprozessortechnik ist. Es wurden die Merkmale des Systemsarchitektur betont, die in bisherigen Konstruktionen nicht erscheinen.

**Grabowski J.: 16-bit-microprocessor features comparison (2)**

INFORMATYKA 1983, No. 12, p. 9

Comparison of ZILOG 8001/2, INTEL 8036, MOTOROLA 68000, TEXAS INSTRUMENTS 9900, LSI 11 and NATIONAL SEMICONDUCTOR 16032 microprocessors from basic statements groups and programming effectiveness point of view.

**Grabowski J.: Eigenschaftenvergleich von 16-Bits-Mikroprozessoren (2)**

INFORMATYKA 1983, Nr. 12, S. 9

Zweiter Teil eines Vergleiches von ZILOG 8001/2, INTEL 8086, MOTOROLA 68000, TEXAS INSTRUMENTS 9900, LSI 11 und NATIONAL SEMICONDUCTOR 16032 Mikroprozessoren von dem Standpunkt der Befehlsgrundgruppen sowie der Programmierungseffektivität.

**Real-time FORTRAN for industrial applications. Part 3**

INFORMATYKA 1983, No. 12, p. 13

Third and last part of international standard proposal characteristics for real-time FORTRAN, which includes presentation of operations on binary strings and bits, input/ /output devices and files.

**Echtzeit — FORTRAN für industrielle Anwendungen. Teil 3**

INFORMATYKA 1983, Nr. 12, S. 13

Dritter und letzter Teil einer Charakteristik des internationalen Normentwurfes für FORTRAN-Erweiterung auf Echtzeit-Anwendungen. Es wurden Aufrufe von Operationen auf Dualfolgen und Biten, Eingabe- und Ausgabeeinrichtungen sowie Dateien charakterisiert.

**Liszyński Z.: Concurrent process programming (2). Software environment**

INFORMATYKA 1983, No. 12, p. 17

Second part of presentation of Poznań Technical University's research work oriented for concurrent realization of many interactive data processing systems using extended configuration of ODRA 1305 computer. Software solutions for this purpose are characterized.

**Liszyński Z.: Programmierung der paralellen Prozesse (2). Die Software**

INFORMATYKA 1983, Nr. 12, S. 17

Zweiter Teil einer Besprechung von Forschungsarbeit in der Posener Technischen Universität, um Möglichkeit zur Realisierung mehrerer Dialogsysteme auf erweiterten ODRA 1305 Rechnerkonfiguration zu schaffen. Es wurden die zu diesem Zweck führende Programmlösungen charakterisiert.

# Komu potrzebna PAI

Komitet Informatyki, kiedyś pod przewodnictwem premiera, a od trzech lat — ministra Nauki Szkolnictwa Wyższego i Techniki, za dwa lata będzie mógł obchodzić okrągły jubileusz dziesięcioletniego istnienia. Trwanie będzie — jak można sądzić — największym osiągnięciem Komitetu. Bo przecież katastrofalne krzywe — ilustrujące wszelkie możliwe wskaźniki w informatyce (wprawdzie podobne do tych, jakie obserwujemy w całej gospodarce, ale o ileż efektowniejsze w swoim dramatyzmie) nie są chyba zamysłem istnienia Komitetu Informatyki.

Oczywiście, na tak fatalne wyniki wpływało wiele czynników — analizować je trzeba we wzajemnych związkach i osobno. Ale rzetelnej analizy obejmującej szersze uwarunkowania — jak dotąd — nie ma. Widać jeszcze nie pora na podejmowanie naprawy. Zresztą, kto miałby taką analizę przygotować i korzystać z niej?!

Próby czynione przez ciała społeczne nie wywołują żadnego oddźwięku, ani zainteresowania władz. Poruszanie zaś opinii społecznej nie ma większego sensu i to nie tylko ze względu na obecną niestosowność takich poczynań. Analiza sytuacji przydatna byłaby, przede wszystkim temu, kto miałby za zadanie zająć się serio informatyką w Polsce... A twierdzę, że Komitet Informatyki zrobić tego nie może, musi zaś zrobić — Państwowa Agencja Informatyki (której wprawdzie jeszcze nie ma, ale...).

Zacznę od przykładu, który — mam nadzieję — uda się przekonywająco uogólnić. Dla potwierdzenia, że KI jest tworem niewłaściwym.

Rzecz w malejącym naborze na studia informatyczne, problemie już opisywanym na tej kolumnie INFORMATYKI. Ma on dalszy ciąg.

Grono uczonych, przejęte już nie tylko radykalnym zmniejszeniem limitów na studia informatyczne (w roku 1980 przyjęto ok. 500 osób, a w 1983 — niewiele ponad 200), ale też obserwowanym obecnie brakiem chętnych (odpowiednio już wykształconych) ludzi do pracy — zwróciło się do MNSzWiT z apelem o podjęcie kroków odwracających ten proces.

Argumentacja przedstawiona MNSzWiT opierała się na dwóch przesłankach. Po pierwsze — informatyka jest we współczesnej gospodarce nie do zastąpienia (Polska nie może zostać i nie zostanie dziwolągiem). Po drugie — dzisiejsze decyzje o rekrutacji na studia przyniosą skutki wprawdzie z opóźnieniem, nie będą one jednak przez to mniej groźne.

Odpowiedź przygotowana przez MNSzWiT, uprzejma w tonie, podtrzymała dotychczasowe stanowisko. Nie próbowała go bronić, tak argumentując, jak winien zrobić to urząd Przewodniczącego Komitetu Informatyki — wskazano po prostu na kolegę: Komisję Planowania. Bo tam powstały decyzje o zmniejszeniu liczby studentów w ogóle, a Ministerstwo ustalonym wskaźnikiem zmniejszyło rekrutację równo (na informatyce też) — po 15% na wszystkie kierunki.

Być może istnieją gdzieś głębokie uzasadnienia zmniejszania się kadry wykształconych informatyków i że tego właśnie potrzeba najbardziej polskiej gospodarce. Dziwiłbym się jednak, gdyby te koncepcje rodziły się dla wszystkich branż właśnie w Komisji Planowania, w ciszy gabinetów, bez udziału fachowców spoza urzędu. Przypuszczam, a raczej — mam pewność, że rzecz miała się inaczej. Komisja Planowania w oparciu o prognozy demograficzne i gospodarcze (być może i inne jeszcze przesłanki) zaleciła ogólne zmniejszanie rekrutacji, natomiast ministerstwa (jest kilka nadzorujących kształcenie na poziomie wyższym) miały odpowiednio do swojej branży ustanowić szczegółowe decyzje. Co więc robi Komitet Informatyki (a raczej jego Przewodniczący) z ostatnim już „produkowanym w Polsce zasobem" w informatyce? Ano nic — tnie równo, jak innych (tak przynajmniej wynika ze stanowiska MNSzWiT).

Nie czepiam się Ministerstwa. To że ma informatykę na barkach wcale go pewnie nie bawi. W sytuacji, kiedy wszędzie jest ciężko i trudno jest szukać idealnych rozwiązań, kiedy sytuacja szkolnictwa wyższego i nauki w ogóle jest niełatwa — nie można się właściwie dziwić, że resort odpowiedzialny za wszystkie kierunki kształcenia nie hołubi jednego z nich.

Ktoś kiedyś określił lapidarnie: „Resort na informatyce kariery nie zrobi, jeśli ją zaś zawali, będzie najwyżej niewielka awantura. Gdyby zaś zawalił resztę kosztem informatyki, to z pewnością byłoby trzęsienie ziemi". Podobnie (a trudno odmówić im racji) rozumują też inni ministrowie, członkowie Komitetu Informatyki.

Wydawało się kiedyś, że koncepcja Komitetu, na forum którego ustalane byłyby kierunki rozwoju informatyki, korygowane z zadaniami resortów już bezpośrednio w ramach ministerstw, jest słuszna. Ale od samego początku (może za sprawą strzeżonej przez resorty niezawisłości) Komitet nie był w stanie dokonać choćby niezbędnych regulacji prawnych, np. doprowadzić do uznania nośników maszynowych jako dokumentów (oczywiście pod jasnymi i weryfikowalnymi rygorami).

Mamy zatem nadal objawy braku polityki państwa w informatyce: brak regulacji prawnych, zastój w normalizacji informatyki, brak prób regulowania mechanizmów ekonomicznych stymulujących informatykę (nie tylko bowiem chodzi o środki z budżetu, a głównie — o właściwe ustawienie mechanizmów). Jedyne co się daje zauważyć, to zainteresowanie MPMiH elektroniką, jako jedną z branż produkujących dobra materialne.

Program elektroniki, niedawno przyjęty przez Radę Ministrów, nie jest oszołamiający, być może jednak coś wreszcie „drgnie". Ale jeśli nawet zacznie się w Polsce produkować dużo i to dobrego sprzętu komputerowego — znowu nie będzie on dla Polski bezpośrednio, lecz na eksport. Oczywiście, eksport był, jest i będzie Polsce potrzebny. Ale na tym przecież nie polega informatyzowanie kraju.

Zastrzeżenie to nie dotyczy jednak producentów. Ich rolą nie jest nic innego, jak produkować — byleby dobrze i z pożytkiem dla kraju; tym lepiej, jeśli zyski będą w dewizach. Zastrzeżenie to ponownie należałoby skierować do Komitetu Informatyki, ale przecież jego członkami są ministrowie, którym na pewno zależy na jak najlepiej spełnionej funkcji swoich resortów...

Potrzebne jest inne, niezależne od resortowego spojrzenie. Spojrzenie instytucji i idące za tym wysiłki, aby dziedzina, za którą ta instytucja jest odpowiedzialna, miała szanse spełnić potrzeby społeczne. Teraz i w przyszłości.

W najtrudniejszych nawet czasach nie przestaje się myśleć o jutrze. Doprawdy, nie można sobie tego jutra n o r m a l n e g o wyobrazić, jeśli już nawet nie będzie widać czerwonych świateł oddalających się pociągów z informatyką.

JANUSZ GWIAZDA

# Zasady prenumer

Zamówienia i przedpłaty na prenumeratę INFORMATYKI przyjmuje Zakład Kolportażu Wydawnictwa NOT SIGMA. Adres pocztowy: Wydawnictwo NOT SIGMA — Zakład Kolportażu, 00-950 Warszawa, skr. poczt. 1004. Konto bankowe: 1036-7490-139-11, III O/M NBP w Warszawie.

JEDNOSTKI GOSPODARKI USPOŁECZNIONEJ, INSTYTUCJE I ORGANIZACJE przesyłają zamówienia (w 1 egz.) zawierające: tytuł czasopima, liczbę zamawianych egzemplarzy, okres prenumeraty i pełny adres zamawiającego z kodem pocztowym, oddział i nazwę banku z numerem konta bankowego zamawiającego oraz (ewentualnie) adresy odbiorców, którzy na zlecenie i koszt zamawiającego mają egzemplarze otrzymywać.

Warunkiem realizacji zamówienia jest równoczesne dokonanie odpowiedniej wpłaty na ww. konto Wydawnictwa NOT SIGMA.

Za prenumeratę nie wystawiane są rachunki i nie potwierdzane salda. Prenumeratorzy zbiorowi proszeni są o podawanie na dowodach wpłat (przelewach) znaku kancelaryjnego zamówienia, którego dotyczy wpłata.

Dopisując na zamówieniu PRENUMERATA STAŁA, zamawiający (tylko prenumeratorzy zbiorowi!) nie będą musieli corocznie ponawiać zamówienia, a jedynie dokonywać przedpłaty według aktualnie obowiązujących cen. Wydawnictwo przekazywać będzie co roku potwierdzenie kontynuacji prenumeraty.

PRENUMERATORZY INDYWIDUALNI dokonują wpłaty przekazem na ww. konto, pod powyższym adresem, podają na odwrocie odcinka dla adresata-posiadacza rachunku: tytuł czasopisma, liczbę zamawianych egzemplarzy oraz okres prenumeraty.

Do PRENUMERATY ULGOWEJ upoważnieni są członkowie stowarzyszeń naukowo-technicznych NOT, studenci, uczniowie szkół zawodowych. Warunkiem jej uzyskania jest poświadczenie blankietu dla nabywcy indywidualnego (na odcinku dla adresata) przez właściwe stowarzyszenie NOT, wyższą uczelnię lub szkołę zawodową.

Zamówienia i wpłaty przyjmowane są na okresy kwartalne, półroczne i roczne w terminach:

- do 15 listopada — na I kwartał, I półrocze i cały rok
- do 28 lutego — na III kwartał
- do 31 maja — na III kwartał i II półrocze
- do 31 sierpnia — na IV kwartał.

UWAGA: Przy podawaniu kodu pocztowego i numeru konta bankowego obowiązuje bardzo czytelne pismo. Prenumerata nie wymaga specjalnego przekazu z czerwonym paskiem; wystarczy zwykły przekaz bankowy.

Prenumerata normalna: kwartalna — 225 zł, półroczna — 450 zł, roczna — 900 zł. Prenumerata ulgowa: kwartalna — 150 zł, półroczna — 300 zł, roczna — 600 zł. Prenumerata ze zleceniem wysyłki za granicę jest dwukrotnie droższa.

Dodatkowych informacji o prenumeracie udziela: Zakład Kolportażu, tel. 40-00-21 w. 293, 299 oraz 40-35-89. Egzemplarze archiwalne można nabywać w Klubie Prasy i Informacji Technicznej w Warszawie, ul. Mazowiecka 12, tel. 27-43-65. Zamówienia na egzemplarze archiwalne należy kierować pod adresem Zakładu Kolportażu.

W NUMERZE 1/84 PRZEDSTAWIMY:

pierwsze wydanie mikroKLANU

oraz

Paprzycki M., Urbaniec K.: Komputerowe wspomaganie prac inżynierskich

Fuglewicz P.: MODULA-2 — język lat osiemdziesiątych (1). Charakterystyka języka

Małuszyński J. T.: Odtwarzanie bazy danych techniką dzienników częściowych

Przegląd języków wysokiego poziomu (3). Języki wysokiego poziomu a inżynieria i metodologia programowania

W NAJBLIŻSZYCH NUMERACH:

- Interakcyjny system przetwarzania obrazów
- PROLOG
- Kody kreskowe
- Narzędzia inżynierii oprogramowania
- Robotyka