mikroKLAN 17

5 1985

informatyka

Superkomputer CRAY

ADA

Język programowania C

Koncepcja nauczania mikroinformatyki

WYDAWNICTWA NAUKOWO-TECHNICZNE

Cena 100 zł

informatyka

5 1985

Nr 5
Miesięcznik Rok XX
Maj 1985

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

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40 lub 26-82-61 w. 184

Zakł. Graf. „Tamka". Zam. 0265-1300/85. Obj. 4,0 ark. druk. Nakład 6100 egz. N-26

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza zł 100,—
Prenumerata roczna zł 1200,—

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE:

W NAJBLIŻSZYCH NUMERACH:

- Jerzy Karczmarczuk o efektywności obliczeń numerycznych
- Marian Furman, Zbigniew M. Nowicki i Jerzy Solak o pakiecie do przetwarzania informacji za pomocą MERY 400
- Jacek Irlik o kwalifikacji prawnej programu komputerowego
- Stanisław Waligórski o języku programowania C
- Jan Bielecki o definiowaniu kompilatorów i instrukcji strukturalnych w języku FORTH
- Zbigniew Suraj o problemach nauczania informatyki w szkołach średnich

WACŁAW ISZKOWSKI
Warszawa

Superkomputer CRAY

Zwiększenie mocy obliczeniowej i szybkości obliczeń można osiągnąć dzięki zwielokrotnieniu liczby procesorów, tworząc systemy wieloprocesorowe, lub też przez skonstruowanie komputerów o architekturze innej niż konwencjonalna. Tak więc, obok gwałtownie rozszerzającego się świata mikroinformatyki, następuje rozwój superkomputerów, których podstawowym celem jest wykonanie maksymalnie wielu operacji w najkrótszym czasie. Historia rozwoju maszyn tej kategorii biegnie wraz z całą informatyką, chociaż dopiero w drugiej połowie lat siedemdziesiątych pojawiły się tu instalacje produkowane seryjnie. O wybranych zastosowaniach tego rodzaju maszyn traktuje interesujący esej [3] oraz artykuł w INFORMATYCE [2].

Obecnie najpopularniejszymi superkomputerami są: potokowo działające CRAY i CDC CYBER 205, tablicowy ICL DAP oraz BURROUGHS BSP [3]. O ich protoplastach oraz o niektórych innych rozwiązaniach można przeczytać w książce Enslowa [1]. Oczywiście, nadal są opracowywane nowe koncepcje superkomputerów o coraz większej mocy obliczeniowej. W tym artykule przedstawię bliżej pierwszy z tej rodziny superkomputerów CRAY-maszynę CRAY-1 z 1976 roku.

Uzyskanie maksymalnej szybkości działania maszyny jest możliwe dzięki zaprojektowaniu odpowiedniej architektury oraz zastosowaniu najnowszych technologii sprzętowych. W architekturze maszyny CRAY-1 celowi temu służy praktycznie każdy element, a zwłaszcza:

— wielomodułowa pamięć operacyjna,
— potokowo działające jednostki arytmetyczno-logiczne,
— wewnętrzne pamięci rejestrów skalarnych i wektorowych,
— wewnętrzne pamięci sekwencji instrukcji,
— nakładkowane wykonywanie rozkazów.

Tabela 1. Przetwarzanie potokowe

Klasycznym przykładem przetwarzania potokowego jest sumowanie dwóch wektorów n-elementowych zawierających liczby zmiennoprzecinkowe. Zwykła jednostka dodawania zmiennoprzecinkowego wykona sumowanie par liczb w postaci: $x=e \cdot 2p$, $y=f \cdot 2q$ — w czterech cyklach zegara, dokonując kolejno następujących operacji:
— porównania wykładników (p—q)
— przesunięcia mantysy zgodnie z różnicą wykładników
— dodania mantys
— normalizacji wyniku.
W przypadku wektorów, zakończenie obliczeń nastąpi po 4n cyklach zegara.

Zmiennoprzecinkowa, jednostka dodawania potokowego, działając analogicznie jak poprzednio, po wykonaniu pierwszego kroku dla pierwszej pary liczb przechodzi do wykonywania drugiego kroku dla tej pary i jednocześnie może wykonać pierwszy krok dla następnej pary liczb:

x_1+y_1 1 2 3 4 $=>$ z_1
x_2+y_2 1 2 3 4 $=>$ z_2
x_3+y_3 1 2 3 4 $=>$ z_3
1 2 3 4 5 6

Tak więc pierwszy wynik otrzymamy po czterech cyklach zegara, a każdy następny — w kolejnych cyklach. Dla całego wektora zakończenie obliczeń nastąpi po n+3 cyklach zegara, a więc już dla n=3 obliczenia zostaną wykonane dwa razy szybciej!

Zanim przedstawię dokładniej architekturę maszyny, zwracając szczególną uwagę na sposoby spełnienia podstawowego założenia projektowego, przypomnę zasadę przetwarzania potokowego (ang. pipelining), zilustrowaną w tabeli 1.

ARCHITEKTURA

Podstawowymi parametrami komputera CRAY-1 są: czas cyklu zegara równy 12,5 μs oraz dostępu do pamięci operacyjnej — 50 μs.

Pamięć operacyjna o pojemności 1 M słów (1 048 576 słów) 64-bitowych, uzupełnionych 8-bitowym rozszerzeniem korekcyjnym, jest podzielona na 16 modułów (ang. bank). Adresacja słów przebiega kolejne moduły modulo 16. Właściwość ta (szerokość dostępu) umożliwia równoczesny dostęp do 16 kolejnych słów na każde 50 μs. Szybokość przesłań z (do) pamięci może więc wynosić $16/50 \cdot 10^9 = 320$ M słów/s. Pełna szerokość dostępu jest wykorzystywana jedynie przy pobieraniu instrukcji. Dla danych wykorzystywana jest szybkość równa $1/12{,}5 \cdot 10^9 = 80$ M słów/s (słów 64-bitowych).

Schemat architektury komputera CRAY-1 przedstawia rysunek. W maszynie dostępnych jest 12 różnych jednostek funkcjonalnych, podzielonych na cztery grupy ze względu na rodzaj operacji: adresowych, skalarnych, zmiennoprze-

Tabela 2. Jednostki arytmetyczno-logiczne

JEDNOSTKI ADRESOWE (24-bitowe) — obliczenia modyfikacji adresów, indeksów, operacji na liczbach całkowitoliczbowych o małych adresach przechowywanych w rejestrach adresowych:
dodawanie całkowitoliczbowe — 2 cykle (25 μs)
mnożenie całkowitoliczbowe — 6 cykli (75 μs)

JEDNOSTKI SKALARNE (64-bitowe) — wykonywanie operacji arytmetycznych całkowitoliczbowych i logicznych na wartościach przechowywanych w rejestrach skalarnych:
dodawanie całkowitoliczbowe 3 cykle (7,5 μs)
przesunięcie zawartości:
— pojedynczego rejestru Sx — 2 cykle (25 μs)
— dwóch sąsiednich rejestrów Sx, Sy — 3 cykle (37,5 μs)
logiczne działanie bezpośrednio na rejestrze — 1 cykl (12,5 μs)
wyznaczenie w argumencie liczby bitów równych:
— jedności — 4 cykle (50 μs)
— zero, poprzedzających jedynkę — 3 cykle (37,5 μs)

JEDNOSTKI ZMIENNOPRZECINKOWE (64-bitowe) — wykonywanie operacji zmiennoprzecinkowych na argumentach pobieranych z rejestrów skalarnych i (lub) wektorowych w arytmetyce 48-bitowej mantysy (dokładność 14 cyfr dziesiętnych) i 16-bitowym wykładniku (zakres liczby 10e—2500... 10e+2500):
dodawanie — 6 cykli (74 μs)
mnożenie — 7 cykli (97,5 μs)
aproksymacja odwrotności (dla dzielenia) — 14 cykli (175 μs)

JEDNOSTKI WEKTOROWE (64-bitowe) — wykonywanie potokowe, przy uwzględnieniu zawartości rejestru maski VM i rejestru długości VL:
dodawanie całkowitoliczbowe — 3 cykle (37,5 μs)
przesunięcie w pojedynczych lub w dwóch kolejnych rejestrach według zawartości rejestru Ax — 4 cykle (50 μs)
operacje logiczne na elementach wektora — 2 cykle (25 μs)

cinkowych i wektorowych. Jednostki te mogą działać równocześnie oraz w trybie przetwarzania potokowego, akceptując kolejne argumenty w każdym cyklu zegara. Wykaz tych jednostek oraz liczbę cykli zegara (12,5 μs) potrzebnych do uzyskania pierwszego wyniku podano w tabeli 2.

Aby możliwe było natychmiastowe dostarczenie argumentów i odebranie wyników, z grupami jednostek funkcjonalnych, związano zestawy rejestrów roboczych przechowujących przetwarzane dane. Zestaw ośmiu rejestrów adresowych A0..A7 jest przeznaczony do przechowywania 24-bitowych adresów podlegających modyfikacji. Zestaw ośmiu rejestrów skalarnych S9..S7 przechowuje pojedyncze, 64-bitowe liczby całkowite lub zmiennoprzecinkowe, a zestaw ośmiu rejestrów wektorowych V0..V7 jest przeznaczony do przechowywania wektorów (maksymalnie 64-elementowych) zawierających 64-bitowe liczby całkowite lub zmiennoprzecinkowe.

Bezpośrednie jednokierunkowe powiązanie rejestrów adresowych i skalarnych z pamięcią operacyjną umożliwia przesyłanie danych z maksymalną prędkością 40 M słów/s z opóźnieniem 11 cykli (137,5 μs). Dla zwiększenia elastyczności w przechowywaniu danych, dołączono dodatkową pamięć wewnętrzną utworzoną przez zestaw 64 pomocniczych rejestrów adresowych 24-bitowych B0..B63 oraz zestaw 64 pomocniczych rejestrów skalarnych 64-bitowych T0..T63. Rejestry te mają bezpośrednie połączenie z pamięcią operacyjną; możliwe jest tu jednokierunkowe przesyłanie danych z szybkością 80 M słów/s. Wymiana zawartości między rejestrami roboczymi a pomocniczymi odbywa się w jednym cyklu zegara.

Rejestry wektorowe mają również jednokierunkowe powiązanie z pamięcią operacyjną — z możliwością przesyłania danych z maksymalną szybkością 80 M słów/s, z opóźnieniem 7 cykli (87,5 μs). Dodatkowy 7-bitowy rejestr długości (VL — Vector Length) zawiera liczbę elementów wektora zapisaną w danym rejestrze wektorowym. Drugi 64-bitowy rejestr maski (VM — Vector Mask) specyfikuje, które spośród elementów danego rejestru wektorowego mają być argumentami kolejnej wykonywanej operacji.

Czasy wykonywania operacji w jednostkach funkcjonalnych są podane w tabeli 2. Dla rejestrów adresowych i skalarnych czasy te uwzględniają pobranie argumentów z rejestrów oraz złożenie wyniku w rejestrze. Dla rejestrów wektorowych dodatkowy cykl zegra jest konieczny na przesłanie każdego z argumentów z rejestru na wejście jednostki; konieczny też — dodatkowy cykl na odesłanie wyniku. Interesująca jest realizacja dzielenia zmiennoprzecinkowego pokazana w tabeli 3.

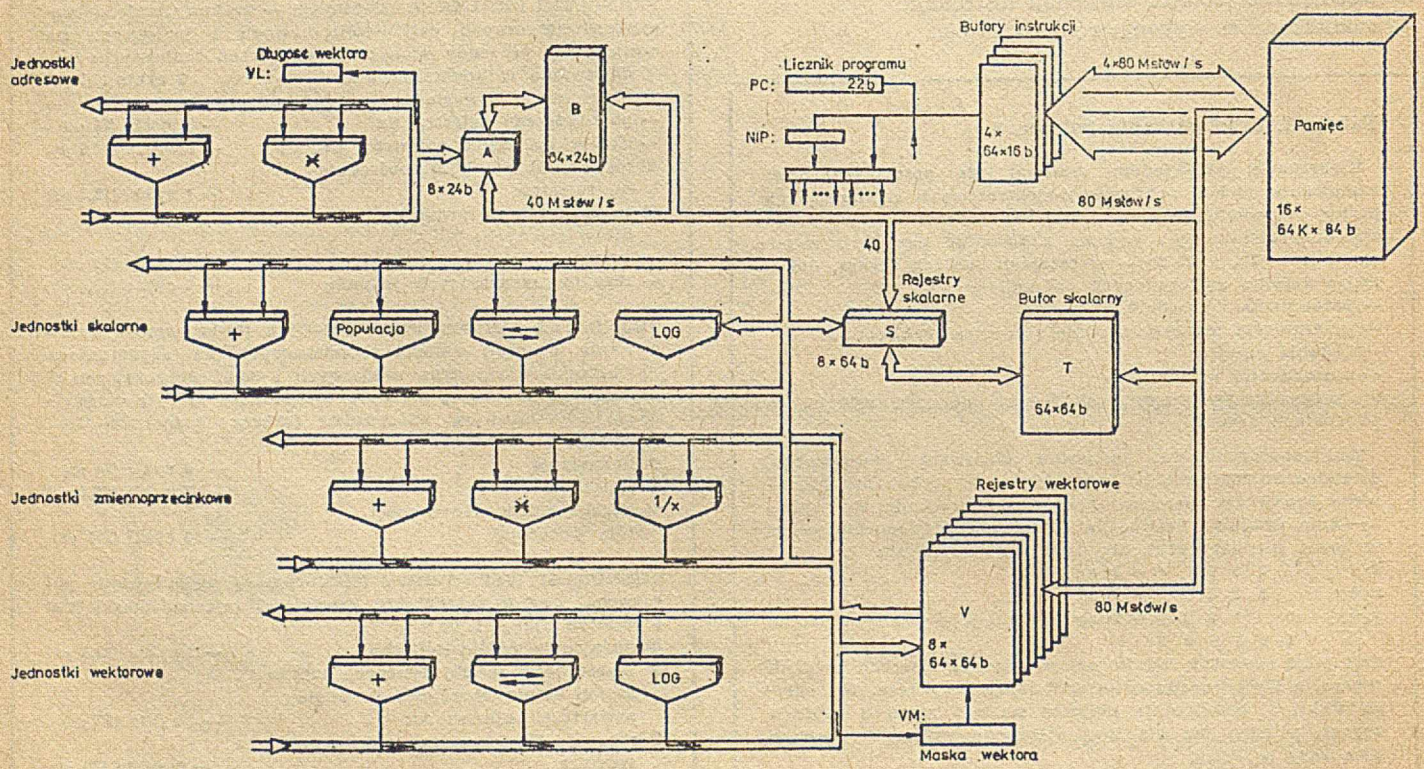
Tabela 3. Realizacja dzielenia zmienno-przecinkowego

Realizacja dzielenia dwóch rejestrów skalarnych S1/S2		
aproksymacja odwrotności	S3=1/S2	14 cykli
iteracja odwrotności	S4=(2—S3×S2)	7 cykli
mnożenie przez numerator	S5=S1×S3	7 cykli
mnożenie przez korekcję	S6=S4×S5	7 cykli
	Razem:	35 cykli

Na przyspieszenie działania komputera ma też wpływ szybkość dostarczania nowych rozkazów wykonawczych. Dla uniknięcia pojedynczych odwołań do pamięci, cztery bufory instrukcji mogą przechowywać cyklicznie po 64 16-bitowych paczek instrukcji (ang. instruktion parcel). Instrukcje mogą być złożone z jednej (16-bitów) lub dwóch (32 bity) paczek, wypełniając dokładnie po cztery paczki kolejne słowa w pamięci operacyjnej. Każdy z czterech buforów instrukcji może być całkowicie wypełniony słowami z kolejnych 16 modułów pamięci w jednym cyklu, wykorzystując pełną szybkość dostępu 320 M słów/s.

Instrukcje są pobierane z bufora instrukcji według 22-bitowego licznika programu (PC — Program Counter) do 16-bitowego rejestru następnej paczki instrukcji (NIP — Next Instruction Parcel). W rejestrze tym instrukcja jest dekodowana oraz sprawdzana jest możliwość jest wykonania w aktualnym stanie działania maszyny. Instrukcja wykonalna jest umieszczona w aktualnym 16-bitowym rejestrze instrukcji (CIP — Current Instruction Parcel); ewentualnie z drugą częścią paczki instrukcji w rejestrze dodatkowym (LIP — Lower Instruction Parcel) — dla instrukcji złożonych z dwóch paczek. Instrukcja z rejestru aktualnego jest w pełni dekodowana i rozpoczyna się jej wykonanie. W przypadku zmiany sekwencji sterowania (wykonanie instrukcji skoku), kolejny bufor instrukcji jest wypełniany z pamięci operacyjnej.

Badanie wykonalności instrukcji polega na sprawdzeniu, czy elementy sprzętu (szyny, rejestry, jednostki funkcjonalne) wykorzystywane w badanej instrukcji nie są zajęte przez wykonywane jeszcze instrukcje. Ograniczenia te odnoszą się jedynie do rejestrów skalarnych i adresowych, blokowanych dla zapamiętania wyniku obliczeń, oraz dla rejestrów wektorowych będących argumentami lub przeznaczonymi na wynik. Zawartości rejestrów skalarnych i adresowych będących argumentami są kopiowane do jednostek funkcjonal-



Schemat architektury superkomputera CRAY-1

nych. Również zawartość rejestru długości wektora jest kopiowana, a więc nie musi być on blokowany. W przypadku rejestrów wektorowych dopuszczono realizację łańcuchowania instrukcji, polegającą na odblokowywaniu dostępu do kolejnych elementów wektorów już wykorzystanych w aktualnie wykonywanych operacjach (patrz przykład w tabeli 4).

Tabela 4. Łańcuchowanie instrukcji

Sekwencja instrukcji:
V1+V2=V3
V2×V3=V4
dla długości wektora równej 10, bez wykorzystania cechy łańcuchowania instrukcji, będzie wykonywana:

1+6+1 [8 cykli na otrzymanie pierwszego wyniku z dodawania] +9×1 [po jednym cyklu na otrzymanie każdego z pozostałych 9 wyników +1+7+1+9×1 [analogicznie dla mnożenia] = 35 c y k l i

a z wykorzystaniem łańcuchowania:

1+6+1+9×1 [analogicznie jak poprzednio] — 9 [gdyż po ósmym cyklu można rozpocząć równoczesne wykonywanie drugiej instalacji] +1+7+1+9×1 [analogicznie dla mnożenia] = 26 c y k l i

Powiązanie superkomputera CRAY-1 z otoczeniem odbywa się poprzez 24 kanały wejścia—wyjścia (12 dla wejścia, 12 dla wyjścia). Maksymalna szybkość przesyłu informacji przez kanały wynosi 10 M słów/s dla każdego kanału. Uwzględniając możliwości dostępu do pamięci, równocześnie może być aktywnych osiem kanałów. Komputerem peryferyjnym zarządzającym urządzeniami peryferyjnymi CRAYA był minikomputer ECLIPSE firmy DATA GENERAL.

TECHNOLOGIA

Dla uzyskania odpowiedniej szybkości działania maszyny, określonej czasem cyklu zegara równym 12,5 μs, zastosowano technikę ECL, w której rejestry pomocnicze B, T i wektorowe V są złożone z układów bipolarnych o cyklu zapisu—odczytu równym 6 ns, a rejestry A i S z układów ECL — o cyklu 1 ns. W centralnym procesorze są wykorzystywane układy logiczne NAND jednego typu o opóźnieniu 0,5—1 ns. Główna pamięć komputera została złożona z 1K 1-bitowych bipolarnych układów LSI (Fairchild). Stosując odpowiednie upakowanie i cylindrycznie układając moduły, maksymalne długości połączeń zostały skrócone do 4 stóp (122 cm). Całość maszyny została umieszczona w trzech 90-stopniowych segmentach cylindra, wokół centralnego chłodzenia z cyrkulującym freonem. Zasilacze oraz sterowanie chłodzeniem znajdują się wokół cylindra — tworząc wygodne siedzenia (stąd wzięło się powiedzenie, że jest to najdroższa kanapa świata). Poza pomieszczeniem maszyny znajdują się dwa 25-tonowe kompresory systemu chłodzenia i generator zasilający o mocy 150 KW.

MOC OBLICZENIOWA

Jaką szybkość działania uzyskano w tym superkomputerze? Odpowiedź jest dość trudna, bo już przy analizie podstawowych zasad działania widać, że efektywne wykorzystanie maszyny jest zależne od realizowanej sekwencji instrukcji, a więc od rodzaju obliczeń, a także odpowiedniego zapisu algorytmu.

CRAY-1 jest programowany w FORTRANIE, którego kompilator generuje optymalny kod wynikowy w asemblerze CAL. Jednak o ostatecznej efektywności obliczeń decydują umiejętności programisty i złożoność algorytmu. W tabeli 5 przykładowo pokazano realizację fragmentu programu. Analizując działanie maszyny dla różnych programów, uzyskano maksymalną szybkość równą 153 M flop/s operacji zmiennoprzecinkowych (ang. flop — floating point operations/s), natomiast 138 M takich operacji na sekundę — dla mnożenia macierzy kwadratowych 64×64. Z kolei, np. dla programu odwracania macierzy kwadratowych 64×64 uzyskano szybkość równą 16 M flop/s a dla programu szybkiej transformaty Fouriera (FFT) tylko 4,7 M flop/s. Inaczej rzecz ujmując — właściwości superkomputera przypominają

Tabela 5. Potokowe wykonywanie programu

Programowi w języku FORTRAN:

```
   DO 10 I=1,50
   A(I)=B(I) × C(I)
10 CONTINUE
```

odpowiada następujący zapis symboliczny, który może być odwzorowany w asemblerze CRAY-CAL:

```
VL <= 50                ; długość wykorzystywanego wektora
V1 <= czytaj (B(1)..B(50))
V2 <= czytaj (C(1)..C(50))
V3 <= V1×V2
zapisz (A(1)..A(50)) <= V3
```

właściwości samochodu wyścigowego, którego pełne walory można wykorzystać tylko na torze, natomiast w ruchu miejskim jest on gorszy od naszego malucha.

Tym niemniej superkomputer CRAY-1 pozwala obecnie na praktyczne rozwiązanie wielu problemów, które znajdowały się dotychczas poza zasięgiem opłacalnych obliczeń (np. otrzymanie prognozy pogody 10-dniowej wymaga wykonania ok. 500 mld operacji, co komputerowi CRAY-1 zabiera ok. 105 minut [4].

RODZINA MASZYN CRAY

Superkomputery CRAY są konstruowane i wytwarzane przez firmę CRAY RESEARCH INC. (Wisconsin, USA), założonej przez Seymoura Cray'a, byłego konstruktora firmy CONTROL DATA CORPORATION. Pierwsza maszyna CRAY-1 została zainstalowana na początku 1976 roku w laboratorium w Los Alamos. Obecnie zainstalowanych jest już ponad 50 różnych mutacji tych maszyn.

Pierwszym krokiem w rozwoju superkomputera CRAY było powiększenie w 1979 r. pojemności pamięci operacyjnej do 2 lub 4 M słów (stosując układy 4 K × 1 bit) oraz wymianę układów wejścia—wyjścia na cztery procesory WE-WY umożliwiające dołączenie 48 kaanłów we-wy. Zmiana ta pozwoliła na wprowadzenie możliwości bezpośredniej komunikacji z systemami IBM oraz połączenie urządzeń peryferyjnych z główną pamięcią operacyjną przy szybkości przesyłania 850 M bitów/s. Rozbudowany system WE-WY umieszczono w czwartym segmencie walca. Nowa konfiguracja została nazwana superkomputerem CRAY-1S.

W kwietniu 1982 wprowadzono na rynek następny superkomputer — CRAY X-MP, będący złożeniem dwóch procesorów centralnych w jednej obudowie, dzielących wspólną pamięć operacyjną o pojemności od 2 M do 4 M słów. W wyniku dalszego skrócenia połączeń uzyskano czas cyklu równy 9,5 μs.

Kolejnym krokiem było wprowadzenie układów trójwymiarowych całkowicie zanurzonych w chłodzącej cieczy fluorokarbonowej. Maksymalna długość przewodów nie przekraczająca 16 cali zredukowała czas cyklu do 4 μs. W ten sposób powstał CRAY-2 złożony z czterech procesorów i pamięci operacyjnej o pojemności 32 M słów. Wykonuje on operacje skalarne 6 razy, a wektorowe 12 razy szybciej niż CRAY-1.

A jaka będzie dalsza przyszłość? Dowcip w czasopiśmie COMUNICATION of the ACM z kwietnia 1984 mówi, że pętlę nieskończoną nowy CRAY-3 wykona w czasie krótszym niż dwie minuty.

LITERATURA

[1] Enslow P. H. jr.: Systemy cyfrowe wieloprocesorowe. WNT, 1978
[2] Gołdys B., Iszkowski W.: Świat superkomputerów CRAY, Informatyka, nr 1/85, s. 26
[3] Hochney R. W., Jesshope C. R.: Parallel Computers. Adam Huger Ltd, Bristol, 1981
[4] Informatyka, nr 4/84, s. 33 („W skrócie")
[5] Killmon P., Asco J.: Processor Technology Computer Design. December 1982
[6] Koppe P. M.: The Architecture of Pipelined Computers. Mc Graw-Hill Book Company, 1981
[7] Thomspon P. D.: Matematyka w meteorologii. w: Matematyka Współczesna, red. Steen L. A., WNT, 1983.

ADA nie jest jeszcze językiem rozpowszechnionym w Polsce — trwają prace nad jego kompilatorami. Jednakże w krajach wysokouprzemysłowionych, głównie w USA i w Europie Zachodniej, w użyciu przemysłowym znajduje się już kilkanaście zatwierdzonych translatorów tego języka, działających zarówno na dużych komputerach, jak i na mini- i mikrokomputerach. A warto pamiętać, że na ostateczny kształt ADY miała wpływ niemal cała współczesna wiedza w zakresie języków programowania. Komentarz do obecnej normy języka przyczynia się do wyjaśnienia wielu istotnych szczegółów implementacyjnych i do jej jednoznacznej interpretacji.

Osoby bardziej zainteresowane ADĄ mogą sięgnąć do szczegółowego jej omówienia w numerze 11—12/1981 i 1—6/1982 (J. Zalewski: ADA nowy język programowania) lub — sprawdzić swoją znajomość języka przy użyciu testów zamieszczonych w nr. 11/1983 (S. Błaszczak, W. Iszkowski: Język ADA w testach). Warto też wspomnieć, że od trzech lat przy Polskim Towarzystwie Informatycznym działa Sekcja Języka ADA, z którą można skontaktować się za naszym pośrednictwem. (Red.)

JAN BIELECKI
Instytut Informatyki
Politechnika Warszawska

ADA — standard operacji wejścia-wyjścia

W artykule tym komentuję standard operacji wejścia—wyjścia w ADZIE [1], a także — opis metody implementacji pakietu TEXT_IO.

KRYTYKA STANDARDU OPERACJI NA PLIKACH TEKSTOWYCH

Ze względu na przyjęty w normie nieformalny opis zasad wykonywania operacji wejścia—wyjścia, pojawiają się wątpliwości co do sposobu interpretowania niektórych podanych tam sformułowań. Przedstawiam je poniżej odnosząc omówienie do poszczególnych punktów i akapitów rozdziałów 14 tego dokumentu.

Pkt. 14.2/3, 14.2.2/3, 14.2.2/6

Plik sekwencyjny zdefiniowano jako liniowo uporządkowany zbiór **wartości**. Natomiast plik bezpośredni zdefiniowano jako liniowo uporządkowany zbiór **elementów**, do których i z których przesyłane są wartości. Z opisów procedur READ i WRITE wynika natomiast, że z pliku wprowadzane są elementy, a do pliku wyprowadzane są wartości. Wydaje się, że najtrafniej byłoby mówić o wprowadzaniu i wyprowadzaniu **danych** o określonych typach i wartościach — i zrezygnować z pojęcia element.

Pkt. 14.2.1/4

Tak w tym miejscu, jak i w wielu innych, zrezygnowano z wyszczególnienia wszystkich dopuszczalnych sytuacji wyjątkowych. Nie wiadomo — na przykład — czy dla wywołania procedury CREATE z jednym argumentem może powstać sytuacja wyjątkowa USE_ERROR.

Pkt. 14.2.4

W opisie przetwarzania plików bezpośrednich nie wyjaśniono, czy otwarcie pliku w trybie OUT_FILE powoduje zmianę wartości (danych) już znajdujących się w pliku (tak jak to występuje np. w języku PL/I).

Pkt. 14.2.4/4

Zwrot **exceeds the size** (przekracza rozmiar) powinien zostać zastąpiony zwrotem **exceeds the current size** (przekracza bieżący rozmiar).

Pkt. 14.2.4/7

Sformułowanie, że podczas wykonywania procedury WRITE może powstać sytuacja wyjątkowa USE_ERROR — jeśli miałaby zostać przekroczona pojemność pliku zewnętrznego — powinno być tak zmienione, aby nie dotyczyło enigmatycznej **pojemności** (ang. capacity), lecz relacji między użytym w operacji wyprowadzania indeksem a bieżącym rozmiarem pliku zewnętrznego.

Pkt. 14.3/6, 14.3.7/14

Norma nie wypowiada się jawnie, czy ogranicznik pliku (który nie musi być realizowany jako stan „koniec pliku") jest ogranicznikiem bezpośrednio poprzedzającego go literału. Jeśli nie jest, to w opisach wyprowadzania literałów ze zmiennych typu STRING powołanie się na ogranicznik pliku powinno zostać zastąpione powołaniem się na ogranicznik wiersza.

Pkt. 14.3.2/2, 14.3.2/5

Opis procedur SET_INPUT i SET_OUTPUT w połączeniu z opisami procedur OPEN i CLOSE sugeruje, że po zamknię-

Dr inż. JAN BIELECKI jest adiunktem Instytutu Informatyki Politechniki Warszawskiej. W roku 1966 ukończył studia informatyczne na Wydziale Elektroniki PW, a w 1970 obronił pracę doktorską z tej dziedziny. Był twórcą koncepcji i współautorem wdrożenia pierwszego w kraju kompilatora języka FORTRAN 66 dla m.c. K-202 (obecnie MERA 400) oraz projektantem systemu operacyjnego dla minikomputera GEO-20, wykonanego w Instytucie Informatyki PW. W latach 1978—1980 był zatrudniony w USA jako konsultant firmy UNITED COMPUTING SYSTEMS (CRAY-1). Jest autorem kilkudziesięciu opracowań z zakresu informatyki, w tym kilkunastu książek i skryptów na temat języków programowania i systemów operacyjnych. Zajmuje się problematyką programowania operacji wejścia—wyjścia w językach wysokiego opziomu i sprzężeń programów z systemami operacyjnymi.

Pierwszą wersję pracy przedstawiono na seminarium Sekcji Języka Ada Polskiego Towarzystwa Informatycznego (7 grudnia 1983).

ciu bieżącego pliku domniemanego (ang. current default file) pozostaje w mocy przyjęte domniemanie pliku. Ponieważ stanowi to znaczne ograniczenie swobody implementacji wynikającej z założenia, że obiekty plikowe są typu prywatnego, uważam, że zamknięcie bieżącego pliku domniemanego powinno czynić bieżącym plikiem domniemanym odpowiedni (wejściowy albo wyjściowy) plik standardowy.

W szczególności, w implementacji typu plikowego za pomocą typu wskaźnikowego, takiej jak np. opisana w [2], program przytoczony na wydruku 1 byłby niepoprawny mimo że nie wynika to z normy.

```
WITH TEXT_IO;
USE TEXT_IO;
PROCEDURE EWA IS
  INP: FILE_TYPE;
  CHR: CHARACTER;
BEGIN
  OPEN(INP,IN_FILE,"KAJA");
  SET_INPUT(INP);
  CLOSE(INP);
  OPEN(INP,IN_FILE,"KAJA");
  GET(CHR);
  PUT(CHR);
END EWA;
```

Wydruk 1. Zachowanie domniemania pliku

Pkt. 14.3.8/16

Norma nie precyzuje, czy wartość 0,0 może być wyprowadzona ze znakiem „-" w wykładniku.

Pkt 14.3.8/21

Opis domniemanej wartości FORE dotyczy wyprowadzania do zmiennych o „dostatecznie dużym" rozmiarze. Należy przypuszczać, że w przypadku, gdy dobór FORE jest niemożliwy, intencją normy jest powstanie sytuacji wyjątkowej LAYOUT_ERROR.

OMÓWIENIE METODY IMPLEMENTACJI PAKIETU TEXT_IO

Chociaż w zamyśle norma opisuje wszystkie istotne skutki wykonania procedur wejścia—wyjścia, dopiero pojawienie się konkretnej implementacji może stanowić podstawę dyskusji na temat jednoznaczności opisu. Implementacja taka została przedstawiona w 2, a choć zawiera szereg niezgodności z normą — zostanie tu przedstawiona jako egzemplifikacja metody wdrożenia pakietu.

Podstawę prezentowanej implementatcji stanowi założenie o realizacji typu plikowego FILE_TYPE jako typu wskaźnikowego:

```
access FILE_CONTROL_BLOCK
```

związanego z typem rekordowym:

```
type FILE_CONTROL_BLOCK is
   record
      SEQFILE:          STREAM.FILE_TYPE;
      COLNO, LINENO, PAGENO: POSITIVE;
      MAXCOLNO, MAXLINENO:
COUNT;
      CH:
CHARACTER;
      CH_OK:
BOOLEAN;
end record;
```

Komponentami obiektów przytoczonego typu rekordowego są: plik sekwencyjny zawierający znaki pliku tekstowego, skalarne zmienne liczbowe określające bieżący numer kolumny, bieżący numer wiersza i bieżący numer strony, skalarne zmienne liczbowe określające bieżący maksymalny rozmiar wiersza i bieżący maksymalny rozmiar strony, skalarna zmienna znakowa, której wartością jest ostatni jeszcze nie wprowadzony albo jeszcze nie wyprowadzony znak oraz — skalarna zmienna boolowska o wartości TRUE, jeśli wymieniona zmienna znakowa ma interpretację (o wartości FALSE — w przeciwnym przypadku).

Ponieważ pakiet TEXT_IO jest dość obszerny, omówienie jego struktury zostanie ograniczone do rozpatrzenia wydruku 2, na którym przedstawiono: procedurę KAJA odwołującą się do tego pakietu, istotne części pakietów IO_EXCEPTIONS i SEQUENTIAL_IO oraz ważniejsze fragmenty pakietu TEXT_IO, który jest realizowany w oparciu o konkretyzację pakietu SEQUENTIAL_IO dla argumentu rodzajowego CHARACTER.

Przed omówieniem wykonania programu należy zwrócić uwagę na wybrane fragmenty wykorzystywanych przez ten program pakietów.

Pakiet SEQUENTIAL_IO jest pakietem rodzajowym z parametrem rodzajowym ELEMENT_TYPE, określającym typ komponentów pliku sekwencyjnego. Specyfikacja pakietu zawiera m.in. deklarację typu plikowego przewidzianego dla plików sekwencyjnych, realizowanego za pomocą ograniczonego typu prywatnego oraz deklarację typu FILE_MODE, który jest typem wyliczeniowym określającym tryb przetwarzania. Specyfikacje procedur OPEN i READ wyznaczają zestaw realizowanych przez pakiet operacji na plikach. Część prywatna pakietu oraz jego ciało są w całości określone przez implementację.

Pakiet TEXT_IO jest pakietem zwykłym (nie rodzajowym). Jego specyfikacja zawiera deklarację unikalnego typu FILE_TYPE przewidzianego dla plików tekstowych, specyfikacje typów pomocniczych COUNT i FIELD oraz specyfikacje podprogramów OPEN i MODE dotyczących plików tekstowych. W części widocznej pakietu występuje ponadto specyfikacja pakietu rodzajowego INTEGER_IO, przewidzianego dla wprowadzenia ciągów znaków o postaci literałów całkowitych. Część prywatna specyfikacji pakietu TEXT_IO zawiera powołanie pakietu STREAM, który jest wcieleniem pakietu SEQUENTIAL_IO dla argumentu rodzajowego CHARACTER, oraz deklarację typu implementującego prywatny typ plikowy FILE_TYPE.

Ciało pakietu TEXT_IO zawiera deklarację standardowego pliku tekstowego INPUT, ciała podprogramów OPEN i MODE, ciała pomocniczego pakietu rodzajowego READ_FILE, konkretyzowanego w procedurze GET zawartej w

```
WITH TEXT_IO;
  USE TEXT_IO;
PROCEDURE KAJA IS
  EWA: INTEGER;
  PACKAGE INTEGRAL IS NEW INTEGER_IO(NUM=)INTEGER);
  USE INTEGRAL;
BEGIN
  GET(INPUT,EWA,WIDTH=)2);
END KAJA;

-------------------------

PACKAGE IO_EXCEPTIONS IS
  STATUS_ERROR: EXCEPTION;
  MODE_ERROR:   EXCEPTION;
  -- DEKLARACJE POZOSTALYCH SYTUACJI WYJATKOWYCH
END IO_EXCEPTIONS;

WITH IO_EXCEPTIONS;
GENERIC
  TYPE ELEMENT_TYPE IS PRIVATE;
PACKAGE SEQUENTIAL_IO IS
  TYPE FILE_TYPE IS LIMITED PRIVATE;
  TYPE FILE_MODE IS (IN_FILE,OUT_FILE);
  PROCEDURE OPEN(FILE: IN OUT FILE_TYPE;
                 MODE: IN FILE_MODE;
                 NAME: IN STRING;
                 FORM IN STRING:="");
  PROCEDURE READ(FILE: IN FILE_TYPE;
                 ITEM: OUT ELEMENT_TYPE);
PRIVATE
  -- CZESC PRYWATNA PAKIETU OKRESLONA PRZEZ IMPLEMENTACJE
END SEQUENTIAL_IO;

PACKAGE BODY SEQUENTIAL_IO IS
  -- CIALO PAKIETU OKRESLONE PRZEZ IMPLEMENTACJE
END SEQUENTIAL_IO;

WITH IO_EXCEPTIONS,SEQUENTIAL_IO;
PACKAGE TEXT_IO IS
  TYPE FILE_TYPE IS LIMITED PRIVATE;
  TYPE FILE_MODE IS (IN_FILE,OUT_FILE);
  TYPE COUNT IS RANGE 0 .. COUNT_LAST;      -- IMPLEMENTACJA
  SUBTYPE FIELD IS INTEGER RANGE 0 .. FIELD_LAST;     -- JW.
  PROCEDURE OPEN(FILE: IN OUT FILE_TYPE;
                 MODE: IN FILE_MODE;
                 NAME: IN STRING;
                 FORM: IN STRING:="");
  FUNCTION MODE(FILE: IN FILE_TYPE) RETURN FILE_MODE;

-- SPECYFIKACJA PAKIETU RODZAJOWEGO DO WPROWADZANIA
--               DANYCH CALKOWITYCH

  GENERIC
    TYPE NUM IS RANGE <>;
  PACKAGE INTEGER_IO IS
    PROCEDURE GET(FILE:  IN FILE_TYPE;
                  ITEM:  OUT NUM;
                  WIDTH: IN FIELD:=0);
  END INTEGER_IO;

  STATUS_ERROR: EXCEPTION RENAMES IO_EXCEPTIONS.STATUS_ERROR;
  MODE_ERROR:   EXCEPTION RENAMES IO_EXCEPTIONS.MODE_ERROR;
  -- POZOSTALE PRZEMIANOWANIA
```

```
PRIVATE
  PACKAGE STREAM IS NEW SEQUENTIAL_IO(CHARACTER);

  TYPE FILE_CONTROL_BLOCK IS

    RECORD
      SEQFILE:                STREAM.FILE_TYPE;
      COLNO,LINENO,PAGENO: POSITIVE;
      MAXCOLNO,MAXLINENO:  COUNT;
      CH:                     CHARACTER;
      CH_OK:                  BOOLEAN;
    END RECORD;
  TYPE FILE_TYPE IS ACCESS FILE_CONTROL_BLOCK;
END TEXT_IO;

PACKAGE BODY TEXT_IO IS
  INPUT: FILE_TYPE;

  PROCEDURE OPEN(FILE: IN OUT FILE_TYPE;
                 MODE: IN FILE_MODE;
                 NAME: IN STRING;
                 FORM: IN STRING:="") IS
  BEGIN
    IF FILE /= NULL THEN RAISE STATUS_ERROR; END IF;
    FILE:=NEW FILE_CONTROL_BLOCK;
    STREAM.OPEN(FILE.SEQFILE,
                STREAM.FILE_MODE'VAL(FILE_MODE'POS(MODE)),
                NAME,FORM);
    -- INNE INSTRUKCJE WNETRZA PROCEDURY
  END OPEN;

  FUNCTION MODE(FILE: IN FILE_TYPE) RETURN FILE_MODE IS
  BEGIN
    IF FILE=NULL THEN RAISE STATUS_ERROR; END IF;
    RETURN FILE_MODE'VAL(
          STREAM.FILE_MODE'POS(STREAM.MODE(FILE.SEQFILE)));
  END MODE;

-- POMOCNICZY PAKIET RODZAJOWY
-- KONKRETYZOWANY W PROCEDURZE GET

  GENERIC
    FILE:  IN OUT FILE_TYPE;
    WIDTH: IN FIELD;
  PACKAGE READ_FILE IS
    FUNCTION NEXT RETURN CHARACTER;
  END READ_FILE;

  PACKAGE BODY READ_FILE IS
    TALLY: NATURAL:=0;
    FIRST: BOOLEAN:=TRUE;

    FUNCTION NEXT RETURN CHARACTER IS
    BEGIN
      IF NOT FIRST THEN
        STREAM.READ(FILE.SEQFILE,FILE.CH);
        -- POZOSTALE INSTRUKCJE
      END IF;
      TALLY:=TALLY+1;
      FIRST:=FALSE;
      RETURN FILE.CH;
    END NEXT;

  BEGIN
    IF MODE(FILE) /= IN_FILE THEN RAISE MODE_ERROR; END IF;
    IF NOT FILE.CH_OK THEN
      STREAM.READ(FILE.SEQFILE,FILE.CH);
      FILE.CH_OK:=TRUE;
    END IF;
    -- POZOSTALE INSTRUKCJE
  END READ_FILE;

-- CIALO PAKIETU RODZAJOWEGO
-- DO WPROWADZANIA/WYPROWADZANIA

-- DANYCH CALKOWITYCH

  PACKAGE BODY INTEGER_IO IS

  -- POMOCNICZA FUNKCJA RODZAJOWA
  -- KONKRETYZOWANA W PROCEDURZE GET

    GENERIC
      WITH FUNCTION NEXT RETURN CHARACTER;

    FUNCTION GET_GENERIC RETURN NUM IS
    C:   CHARACTER:=NEXT;
    VAL: NUM;
    BEGIN
      -- POZOSTALE INSTRUKCJE
    END GET_GENERIC;

    PROCEDURE GET(FILE:  IN FILE_TYPE;
                  ITEM:  OUT NUM;
                  WIDTH: IN FIELD:=0) IS
      PACKAGE READ IS NEW READ_FILE(FILE.WIDTH);
      FUNCTION GET_FROM_FILE IS GET_GENERIC(READ.NEXT);
    BEGIN
      ITEM:=GET_FROM_FILE;
    END GET;
  END INTEGER_IO;

BEGIN
  OPEN(INPUT,IN_FILE,"INPUT");
END TEXT_IO;
```

Wydruk 2. Struktura pakietu TEXT_IO

ciele pakietu INTEGER_IO, a także — sam pakiet INTEGER_IO oraz część czynną pakietu, zawierającą instrukcję wywołania procedury OPEN dla otwarcia pliku standardowego i skojarzenia go z plikiem zewnętrznym o nazwie „INPUT".

Zgodnie z wymaganiami normy, wykonanie procedury KAJA jest poprzedzone opracowaniem pozostałych segmentów programu, np. w kolejności: pakiet IO_EXCEPTIONS, pakiet SEQUENTIAL_IO, pakiet TEXT_IO.

Opracowanie pakietu IO_EXCEPTIONS sprowadza się do opracowania deklaracji wyjątków wyszczególnionych w jego specyfikacji (przyjęto, że w danej implementacji pakiet ten nie zawiera ciała).

Opracowanie deklaracji pakietu rodzajowego SEQUENTIAL_IO polega na przedstawieniu nazwy tego pakietu, a następnie opracowaniu parametru rodzajowego ELEMENT_TYPE, którego identyfikator reprezentuje w pakiecie typ komponentów pliku sekwencyjnego i występuje m.in. w specyfikacjach procedur OPEN i READ.

Opracowanie ciała pakietu SEQUENTIAL_IO powoduje jedynie ustanowienie go jako modelu spodziewanych konkretyzacji.

Opracowanie specyfikacji pakietu TEXT_IO składa się z opracowania: deklaracji typów FILE_TYPE, FILE_MODE i COUNT, deklaracji podtypu FIELD, specyfikacji podprogramów OPEN i MODE, deklaracji pakietu rodzajowego INTEGER_IO oraz deklaracji wyjątków STATUS_ERROR i MODE_ERROR, przemianowujących odpowiednie wyjątki zadeklarowane w pakiecie IO_EXCEPTIONS.

W ramach opracowania prywatnej części specyfikacji pakietu TEXT_IO; opracowuje się powołanie pakietu STREAM stanowiącego wcielenie pakietu SEQUENTIAL_IO dla argumentu rodzajowego CHARACTER, a następnie — specyfikację i ciało powołanego pakietu, na co składa się opracowanie: deklaracji typów FILE_TYPE i FILE_MODE, deklaracji procedur OPEN i READ oraz części pakietu określonych przez implementację. Powoduje to utworzenie w pakiecie STREAM procedury READ:

```
procedure READ (FILE:  in FILE_TYPE;
                ITEM:  out CHARACTER);
```

służącej do wprowadzania z plików typu STREAM. FILE_TYPE pojedynczych znaków. W ramach dalszego opracowywania specyfikacji pakietu TEXT_IO, następuje opracowanie deklaracji typu rekordowego FILE_CONTROL_BLOCK i typu wskaźnikowego FILE_TYPE, związanego z tym typem rekordowym.

Na opracowanie ciała pakietu TEXT_IO składa się opracowanie: deklaracji pliku INPUT, ciał podprogramów OPEN i MODE, deklaracji i ciała pomocniczego pakietu rodzajowego READ_FILE, ciała pakietu INTEGER_IO oraz — wykonanie instrukcji zawartych w części czynnej ciała pakietu, w danym wypadku instrukcji:

```
OPEN (INPUT, IN_FILE "INPUT");
```

równoważnej, na podstawie reguł widoczności instrukcji:

```
TEXT_IO.OPEN/TEXT_IO.INPUT, TEXT_IO.IN_FILE, "INPUT"/;
```

której wykonanie powoduje otwarcie wejściowego pliku standardowego.

Po wykonaniu tych czynności wstępnych, rozpoczyna się opracowywanie procedury KAJA, w której ciele — dzięki klauzuli **USE** — staje się bezpośrednio widoczny identyfikator pakietu INTEGER_IO.

Po opracowaniu deklaracji skalarnej zmiennej całkowitej EWA, następuje opracowanie powołania pakietu INTEGRAL, będącego wcieleniem pakietu INTEGER_IO dla argumentu rodzajowego INTEGER. Powołany pakiet zawiera m.in. specyfikację:

```
procedure GET (FILE:   in FILE_TYPE;
               ITEM:   out INTEGER;
               WIDTH:  in FIELD:=0);
```

uzyskaną przez zastąpienie parametru rodzajowego NUM argumentem INTEGER. Podobna zmiana dotyczy ciała tej procedury.

Bezpośrednio po utworzeniu pakietu INTEGRAL następuje opracowanie jego specyfikacji, a następnie jego ciała. Opracowanie specyfikacji sprowadza się do opracowania specyfikacji procedury GET, zaś ciała — do opracowania deklaracji pomocniczej funkcji rodzajowej GET_GENERIC

(opracowywanie ciała procedury GET zostaje zaniechane, ponieważ utworzono już specyfikację tej procedury).

Po opracowaniu zwrotu,

```
use INTEGRAL;
```

kończącego część bierną procedury KAJA, uzyskuje się widoczność identyfikatorów zawartych w pakiecie INTEGRAL, co powoduje m.in. że do procedury, której pełną nazwą przed opracowaniem wymienionego zwrotu było — KAJA.INTEGRAL.GET, można odwoływać się po prostu przez nazwę GET.

Po utworzeniu części biernej procedury KAJA, wykonywana jest jej część czynna, na co składa się wykonanie instrukcji:

```
GET (INPUT, EWA, WIDTH=>2);
```

zlecającej wprowadzenie z dwuznakowego pola pliku INPUT liczby zapisanej zgodnie ze składnią literałów typu INTEGER ze znakiem, a następnie nadanie wartości tej liczby zmiennej EWA.

Wykonanie procedury GET rozpoczyna się od skojarzenia parametrów FILE, ITEM i WIDTH odpowiednio z argumentami INPUT, EWA i 2. Po dokonaniu tego skojarzenia, opracowywana jest część bierna ciała procedury, a następnie wykonywana jego część czynna.

Opracowanie części biernej ciała składa się z opracowania powołania pakietu READ, który dla każdego wywołania procedury GET jest wcieleniem pakietu rodzajowego READ_FILE dla argumentów rodzajowych FILE i WIDTH, będących jednocześnie parametrami procedury GET.

Opracowanie ciała pakietu READ — to utworzenie deklaracji zmiennych TALLY i FIRST oraz wykonanie instrukcji części czynnej ciała pakietu (opracowanie ciała funkcji NEXT zostaje zaniechane, ponieważ określono już jej specyfikację). Powoduje to w szczególności wywołanie procedury:

```
STREAM.READ(FILE.SEQFILE.FILE.CH);
```

równoważne w danym kontekście wywołaniu:

```
TEXT_IO.STREAM.READ(INPUT.SEQFILE.INPUT.CH);
```

a w konsekwencji przygotowanie w zmiennej INPUT.CH kolejnego znaku pliku sekwencyjnego, realizującego plik tekstowy INPUT.

Opracowanie powołania funkcji GET_FROM_FILE, będącej wcieleniem funkcji rodzajowej GET_GENERIC dla argumentu rodzajowego READ.NEXT, powoduje utworzenie funkcji, której zadaniem jest kompletowanie ze znaków pliku sekwencyjnego literałów całkowitych ze znakiem i udostępnianie ich wartości. Ponieważ w funkcji rodzajowej GET_GENERIC kolejne znaki kompletowanego literału są uzyskiwane za pomocą bezparametrowej funkcji NEXT, która jest parametrem rodzajowym pakietu, opracowanie powołania funkcji GET_FROM_FILE jako wcielenia pakietu GET_GENERIC dla argumentu (funkcji) READ_NEXT powoduje, że funkcja GET_FROM_FILE uzyskuje kolejne znaki za pomocą funkcji READ.NEXT. Ponadto, ponieważ funkcja rodzajowa GET_GENERIC jest zawarta w pakiecie INTEGRAL stanowiącym wcielenie pakietu rodzajowego INTEGER_IO z argumentem rodzajowym INTEGER, funkcja GET_GENERIC udostępnia wartości typu INTEGER.

W omówionym programie nie przewidziano obsługi wyjątków, dlatego też powstanie sytuacji wyjątkowej, będącej następstwem np. niewłaściwej składni wprowadzanych znaków, spowoduje zaniechanie dalszego wykonywania programu.

LITERATURA

[1] Ada Programming Language. ANSI/MIL-STD-1815A, 1985

[2] Madsen J.: An Ada Implementation of TEXT_IO. Christian Rovsing A/s. 1983

[3] Bielecki J.: Ada — interpretacja standardu operacji wejścia—wyjścia. WPW, Warszawa 1985 (w druku).

JULIAN WINIEWSKI

Instytut Podstaw Informatyki PAN
Warszawa

Język programowania C (2)

W bieżącym numerze omówione zostały pozostałe konstrukcje języka C, tj. instrukcje, funkcje, wskaźniki i struktury, a także — preprocesor języka.

INSTRUKCJE

Instrukcje wykonuje się w kolejności ich występowania, chyba że wykonanie określonej instrukcji powoduje jej zmianę. Instrukcją jest każde wyrażenie zakończone średnikiem, a także sam średnik oznaczający instrukcję pustą. Jedna z instrukcji — **return** zostanie omówiona w punkcie poświęconym funkcjom.

Instrukcja złożona — blok

Instrukcja złożona ma postać:

```
{deklaracje
 instrukcje}
```

i stanowi ciąg instrukcji, ewentualnie poprzedzony deklaracjami, ujęty w nawiasy klamrowe: {}

Deklaracje identyfikatorów zadeklarowanych już w blokach obejmujących dany blok powodują przesłonięcie tych uprzednich deklaracji na czas wykonywania bloku. Wszystkie inicjowania występujące w deklaracjach **auto** i **register** są wykonywane przy każdym otwarciu bloku, natomiast inicjowania zmiennych statycznych dokonuje się tylko raz przy rozpoczęciu programu.

Instrukcja warunkowa

Ma ona ogólną postać:

```
if (wyrażenie) instrukcja1
```

lub

```
if (wyrażenie) instrukcja1 else instrukcja2
```

Po obliczeniu wartości wyrażenia, jeśli jest ona niezerową, wykonuje się instrukcja1, a w przeciwnym wypadku — instrukcja2 (dla drugiej postaci **if**) lub następna po instrukcji **if** (dla pierwszej postaci). Słowo zastrzeżone **else** jest zawsze wiązane z ostatnim dotąd nie związanym **if**, np.:

```
if(a) if(b) s1; else s2;
```

jest rozumiane jako

```
if(a) {if(b) s1; else s2}
```

Przełącznik

Instrukcja ta pozwala na wybór jednej z wielu gałęzi programu i ma postać:

```
switch (wyrażenie)
instrukcja1
```

Instrukcja1 powinna być w zasadzie instrukcją złożoną, w której dowolnie wiele instrukcji może mieć postać:

```
case wyrażenie—stałe : instrukcja
```

Wykonanie instrukcji1 w przełączniku rozpoczyna się od tej instrukcji case, której **wyrażenie—stałe** ma wartość równą obliczonemu uprzednio **wyrażeniu.** Jeśli wartość **wyrażenia** nie zgadza się z żadną z wartości **case**, to wykonanie rozpoczyna się od instrukcji poprzedzonej słowem **default**, o ile ono występuje. W przeciwnym wypadku nie wykonuje się żadnej z instrukcji wnętrznych przełącznika.

```
switch (i)
{case 1 : ins1;
 case 2 : ins2;
 ...
 case 10 : ins10;
 default : ins11;}
```

W powyższym przykładzie, jeśli wartość zmiennej i jest równa np. 5, to wykonuje się kolejno instrukcje ins5, ins6,..., ins10, ins11, chyba że któraś z nich zmienia kolejność wykonania. Chcąc uzyskać konstrukcję wybierającą tylko jedną z tych instrukcji do wykonania, należy po każdej instrukcji **case** umieścić instrukcję **break**, powodującą przerwanie wykonywania przełącznika.

Pętle

Do zapisania iteracji służą w języku C trzy rodzaje instrukcji: **while, do** i **for.**
Instrukcja **do** postaci:

```
do instrukcja1
while (wyrażenie)
```

oraz **while:**

```
while (wyrażenie)
instrukcja1
```

działają bardzo podobnie. Instrukcja1 jest wykonywana w pętli dopóty, dopóki wartość wyrażenia jest niezerowa. Obie powyższe instrukcje różnią się tylko miejscem sprawdzenia warunku — w pętli **while** przed, a w pętli **do** po wykonaniu instrukcji wewnętrznej. Przykładem użycia pętli **while** jest ciało funkcji bibliotecznej **strlen**, obliczającej długość tekstu:

```
i=0; while (s[i] != '/0') i++ ;
```

Bardziej złożoną formą pętli jest instrukcja **for** postaci:

```
for (wyr1 ; wyr2 ; wyr3) instrukcja1
```

Jest ona równoważna programowi:

```
wyr1 ;
while (wyr2)
{instrukcja 1 ; wyr3 ;}
```

Każde z trzech wyrażeń w pętli **for** można pominąć, ale średniki należy zachować. Drugie z tych wyrażeń jest zastępowane w takiej sytuacji stałą 1. Przykładem pętli **for** jest ciało funkcji obliczającej n-tą potęgę zmiennej x:

```
for (p=1 ; n>0 ; ——n) p *= x;
```

Skoki

Jak już wspomniano, instrukcja **break** powoduje przerwanie wykonywania przełącznika. Ogólnie, instrukcja o postaci:

```
break;
```

powoduje przerwanie wykonywania najmniejszej z obejmujących ją instrukcji pętli lub przełącznika.

Podobnie instrukcja:

```
continue;
```

powoduje przejście do końca ciała wykonywanej pętli; np. w pętli:

```
while (war)
{...
 cont: ;
}
brek: ;
```

instrukcja **continue** jest równoważna goto cont, a **break** — goto brek.

Obie te instrukcje są zatem specjalną formą instrukcji skoku:

```
goto etykieta ;
```

Instrukcja skoku jest w języku C raczej nietypowa. Ogólnie, powoduje ona przejście do wykonywania instrukcji poprzedzonej **etykietą.** W odróżnieniu od postaci skoku w innych językach, jedynym jego ograniczeniem jest to, że nie może on wyprowadzać sterowania poza funkcję (etykieta musi być zdefiniowana w tej samej funkcji co skok). Możliwe jest więc wykonanie skoku do bloku, przełącznika lub pętli. Skok taki powoduje, że nie zostanie wykonane inicjowanie zmiennych lokalnych bloku lub obliczenie wartości wyrażeń sterujących pętlą i przełącznikiem.

FUNKCJE

Funkcja jest to nazwany blok wykonywany w chwili napotkania jego nazwy w wyrażeniu lub w instrukcji programu. Deklaracja funkcji jest jednakże potrzebna tylko wtedy, gdy funkcja jest zewnętrzna względem programu ją wywołującego lub gdy wywołanie funkcji występuje w programie przed jej definicją. W innych sytuacjach wystarczy tylko definicja funkcji.

Definicja funkcji

Postać definicji funkcji jest następująca:

```
typ identyfikator (lista_nazw_parametrów)
  deklaracja_parametrów blok
```

Specyfikacja typu dotyczy wartości obliczanej przez funkcję. Brak specyfikacji typu, podobnie jak w deklaracjach, powoduje domyślne przyjęcie typu **int.** Funkcje nie zadeklarowane lub nie zdefiniowane przed wywołaniem też są tak traktowane.

Funkcja udostępnia wartość, jeśli w jej ciele zostanie wykonana instrukcja:

```
return wyrażenie ;
```

co spowoduje powrót do miejsca wywołania funkcji z wartością **wyrażenia.** Powrót bez wartości można uzyskać przez użycie instrukcji **return** bez wyrażenia. Ten sam efekt daje zakończenie wykonania ciała funkcji.

Jeśli w definicji funkcji wystąpi specyfikator **static**, oznacza to, że funkcja ta nie może być wywoływana w innych programach.

Wywołanie funkcji

Wywołanie funkcji, o postaci:

identyfikator (lista_wyrażeń)

może być instrukcją (wtedy ewentualna wartość funkcji jest pomijana) albo wystąpić w wyrażeniu (wtedy funkcja powinna udostępniać wartość).

Lista wyrażeń w wywołaniu służy nadawaniu wartości parametrom funkcji. Wartości kolejnych wyrażeń są podstawiane na odpowiadające im parametry, ale kolejność ich obliczania jest zależna od kompilatora. Parametry są zawsze wywoływane przez wartość. Chcąc przesłać do funkcji adres

zmiennej, należy na liście argumentótw wywołania umieścić wyrażenie adresowe (np. &x). Wyrażenie to zostanie obliczone, a jego wartość — czyli adres zmiennej x — zostanie przesłany do funkcji, gdzie odpowiedni parametr powinien być zadeklarowany jako wskaźnik. Liczba wyrażeń w wywołaniu nie musi być równa liczbie parametrów — jeśli jest większa, to ostatnie zostaną pominięte, a jeśli jest mniejsza, to niektóre parametry będą miały nieokreślone wartości. Należy pamiętać, że nie dokonuje się tu żadnych konwersji typów, bowiem typy parametrów funkcji nie są porównywane z typami wyrażeń.

Funkcje mogą być wywoływane rekurencyjnie w dowolny sposób, chociaż ich definicje nie mogą się w sobie zawierać.

Przykładem kompletnej definicji funkcji jest poniższa funkcja dokonująca odwrócenia tekstu, korzystająca przy tym z funkcji **strlen.**

```
reverse (s)
char s [ ];
{int temp, i, j;
 for (i = 0, j = strlen(s)—1 ; i < j ; i++, j— —)
     {temp = s[i];
      s[i] = s[j];
      s[j] = temp;}
}
```

TABLICE I WSKAŹNIKI

Wskaźnik jest to zmienna zawierająca adres innej zmiennej. Tablica zaś — to zespół zmiennych tego samego rodzaju. Jednakże w języku C oba te rodzaje zmiennych są związane tak dalece, że każdy program, w którym korzysta się z indeksowania tablic, da się zapisać przy użyciu wskaźników.

Tablice

Deklaracja:

```
int tabi [5];
```

definiuje tablicę jednowymiarową, czyli spójny obszar pamięci dla pięciu zmiennych całkowitych o nazwach **tabi [0], tabi [1],..., tabi [4].** Wielowymiarową tablicę deklaruje się przez wielokrotne powtórzenie nawiasów kwadratowych, np.:

```
char tabc [50] [5];
```

Powyższa deklaracja definiuje zespół zmiennych znakowych **tabc [0] [0], ..., tabc [49] [4].**

Zakresy wymiarów są stałymi i muszą być znane w czasie kompilacji. W deklaracjach tablic można pominąć zakres pierwszego wymiaru, np.:

```
int tab [];
int tabi [] [10];
```

wtedy gdy tablice **tab** i **tabi** są zdefiniowane gdzie indziej w programie. Dzieje się tak, gdy tablica jest parametrem funkcji lub gdy jest zmienną zewnętrzną. Inna sytuacja, w której wolno opuścić zakres pierwszego wymiaru, ma miejsce, gdy w definicji tablicy następuje inicjowanie; bowiem wyznacza ono jednocześnie zakres tego wymiaru. Pozostałe zakresy muszą wystąpić, gdyż operacje na indeksach, np.:

```
tabi [i++] [3]
```

byłyby nie zdefiniowane w czasie kompilacji. W przykładzie tym wyrażenie i++ jest zwiększeniem pierwszego indeksu o 1, ale oznacza zwiększenie adresu **i** o długość elementu, w tym przypadku — o dziesięć długości liczby całkowitej, czyli najczęściej — o 20 lub 40 bajtów, zależnie od maszyny.

Zmienne indeksowane **(tabi [i] [j] lub tabc [a*2+4])** są adresami i mogą wystąpić w programie wszędzie tam, gdzie i każda inna zmienna. Natomiast nazwa tablicy **(tabi** lub **tabc)** oznacza adres początku tablicy, tak więc wyrażenia **tabi** i **&tabi [0]** są równoważne. Ponieważ tablica może być parametrem funkcji, to z powyższego wynika, że w wywołaniu takiej funkcji przesyłany jest adres tablicy będącej argumentem wywołania, a nie jej zawartość.

Przykładem zastosowania tablic jest funkcja standardowa **strcpy**, kopiująca teksty.

```
strcpy (s, t)
char s [ ], t [ ];
{int i;
for (i=0 ; s[i]=t[i] ; i++);}
```

Wywołując tę funkcję, należy napisać np.:

```
char TC [8], TX[]=„abcdefgh";
strcpy (TC, TX);
```

lub

```
strcpy (TC, „ijklmno");
```

Wskaźniki

Poniższa deklaracja:

```
int *pn;
```

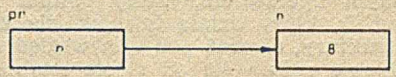
oznacza, że od tego momentu identyfikator **pn** jest nazwą zmiennej, której wartością może być tylko adres zmiennej całkowitej. Jeśli teraz nastąpi przypisanie:

```
pn = &n;
```

czyli pobranie adresu zmiennej **n** i wpisanie go jako wartości **pn**, to przypisania:

```
*pn = 8; i n = 8;
```

będą równoważne, co wyjaśniono na rysunku, gdzie widać wyraźnie różnicę między adresem a wskaźnikiem. Każda z ramek symbolizujących zmienne jest oznaczona przez adres (l-value) tej zmiennej. Wewnątrz ramki znajduje się wartość tej zmiennej.



Różnica między adresem a wskaźnikiem

Porównując tablice ze wskaźnikami, tablicę można określić jako „stałą adresową", a wskaźnik jako „zmienną adresową". Przykładowo — po deklaracji:

```
int buf [50], *ip;
```

instrukcje:

```
ip = & buf [20];
ip = buf+20;
```

są równoważne, gdyż przypisują wskaźnikowi **ip** adres dwudziestego elementu w tablicy **buf.**

Wskaźnikowi można przypisywać stałe tekstowe (jeśli oczywiście jest on typu **char**), np.:

```
char *CP = „abcdefgh";
```

Jedynymi operatorami działającymi na wskaźnikach, poza przypisaniem i pobraniem zawartości (operacja ***ptr**), są — dodawanie i odejmowanie. Dokładniej, do wskaźnika można dodać (lub od niego odjąć) liczbę całkowitą, a dwa wskaźniki, ale tylko tego samego typu, można od siebie odjąć. Dodanie liczby do wskaźnika (odejmowanie od wskaźnika), np.:

```
ip + 10
```

oznacza pobranie adresu większego (mniejszego) od zawartości **ip** o dziesięć długości typu wskazywanego przez ten wskaźnik (czyli o 20 lub 40 bajtów). Natomiast odejmowanie od siebie dwóch wskaźników, np.:

```
ip — buf
```

daje w wyniku liczbę elementów wskazywanego typu mieszczących się między tymi adresami (wynikiem odejmowania w tym przypadku jest 30). Jak wynika z powyższego przykładu, operacje te można stosować również do nazw tablic.

Funkcja standardowa **strcpy** jest dobrym przykładem zastępowania tablic wskaźnikami.

```
strcpy (s,t)
char *s,*t;
{while (*s++ = *t ++);}
```

Takie przedstawienie funkcji **strcpy** jest nieco prostsze od wersji tablicowej. Ogólnie przyjmuje się, że operacje na wskaźnikach są szybsze od operacji na zmiennych indeksowanych, jednakże w niektórych implementacjach zależność ta jest odwrotna.

STRUKTURY I UNIE

Struktura jest to zespół zmiennych zgrupowanych razem i występujących pod wspólną nazwą, które — inaczej niż w tablicy — nie muszą być tego samego typu. Unie omówione są łącznie ze strukturami, ponieważ tak samo deklaruje się je i identycznie odwołuje się do nich w programie, mimo że służą innemu celowi.

Struktury

W przykładzie:

```
struct {char nazwisko [NDLUG];
        char adres [ADLUG];
        int wiek;} x,y,z ;
```

zdefiniowano trzy struktury — **x, y, z.** Każdej z nich przydzielono obszar pamięci składający się z trzech **składowych.** Dwie pierwsze — o nazwach **nazwisko** i **adres,** to teksty o długościach zdefiniowanych przez uprzednio zdefiniowane stałe NDLUG i ADLUG, a trzecią jest **wiek** typu całkowitoliczbowego. Aby wykonać dowolną operację na składowych struktury, można napisać, np.:

```
x. wiek = 54;
strcpy (y. nazwisko, „Kowalski");
```

Kropka (.) jest więc operatorem działającym na strukturze, oznaczającym pobranie adresu składowej o podanej nazwie. Jedynym innym operatorem, który wolno stosować do struktury jest &, który działa tak jak dla zmiennych, tj. daje adres struktury, chociaż pod tym względem poszczególne implementacje różnią się od siebie istotnie. W niektórych wersjach języka C struktury są traktowane jak pełnoprawne zmienne, tj. mogą być parametrami wywołania funkcji i przypisania, co oznacza możliwość tzw. „agregatowego" przypisywania wartości strukturze. W innych przypadkach dopuszczana jest tylko możliwość przesyłania wartości struktury przy wywołaniu funkcji. Klasyczna wersja języka nie dopuszcza jednak operowania na strukturze jako całości w sposób bezpośredni, a jedynie — podobnie jak w przypadku tablic — przez wskaźniki.

W celu skrócenia zapisu powtarzających się deklaracji struktur można zadeklarować nazewnik struktury (ang. structure tag), np.:

```
struct personalia {char nazwisko [30];
                   int wiek; }
```

po czym można już definiować struktury:

```
struct personalia x, y, *sptr;
```

Do składowej struktury można odwołać się w inny jeszcze sposób, a mianowicie przez wskaźnik w postaci:

```
wskaźnik —> składowa
```

Przykładowo — jeśli uprzednio wystąpiło przypisanie:

```
sptr = &x;
```

to poniższe wyrażenia są równoznaczne

```
sptr —> nazwisko
*(sptr). nazwisko
x. nazwisko
```

Składową struktury może być dowolny obiekt z wyjątkiem funkcji, ale nie — wskaźnika do funkcji. Struktury mogą być wielopoziomowe, pod warunkiem, że żadna ze struktur podrzędnych nie jest tego samego typu co któraś względem niej nadrzędna. Tego rodzaju rekursywną deklarację struktury można jednak utworzyć przez zastosowanie wskaźników; np.
struktura opisująca drzewo ma postać:

```
struct węzeł {char *opis;
              struct węzeł l—gałąź;
              struct węzeł p_gałąź; }
```

Definicja **węzła** korzysta z siebie samej, ale nie zawiera składowych typu węzeł, lecz dwa wskaźniki **l_gałąź** i **p_gałąź** do tego typu.

Istnieje jeszcze jeden specyficzny rodzaj składowych zwany **polem.** Jest to konstrukcja umożliwiająca umieszczanie kilku obiektów w jednym słowie maszynowym. Polem jest kilka sąsiednich bitów w obszarze typu **int,** np.:

```
struct {unsigned extern...var : 1;
        unsigned static_var : 1;
        unsigned auto_var : 1; }
```

Liczba po dwukropku oznacza długość pola w bitach.

Unie

Deklaracja unii różni się od deklaracji struktury tylko użyciem słowa zastrzeżonego **union** w miejsce **struct,** np.:

```
union wart {short swart;
            int iwart;
            long lwart;
            float fwart; } xval;
```

Znaczenie tej definicji jest już jednak inne. Zmiennej **xval** przydzielony zostaje obszar pamięci o długości nie będącej sumą długości składowych, jak w przypadku struktury, lecz — o długości najdłuższej składowej. Odwołanie do dowolnej składowej unii oznacza pobranie zawsze tego samego (lub prawie tego samego) adresu, ale rozumianego jako zmienna innego typu. Unia jest więc zmienną, która w trakcie wykonywania programu może przechowywać wartość różnych typów. Kompilator nie zapamiętuje typu wartości ostatnio przechowywanej w tym obszarze, tak więc programista powinien wiedzieć jakiego typu wartość chce odczytać. Adresy składowych krótszych niż długość obszaru unii są zależne od implementacji — w niektórych kompilatorach składowe te są nałożone na początek obszaru, w innych na jego koniec.

PREPROCESOR

Kompilator języka C automatycznie wywołuje preprocesor podczas translacji programu. Preprocesor umożliwia kompilację warunkową, włączanie innych programów, makrodefinicje itp. Linie programu zaczynające się od znaku # są interpretowane przez preprocesor.

Definicje stałych i makrodefinicje

Linia postaci:

```
# define identyfikator ciąg_symboli
```

oznacza, że preprocesor zastąpi każde wystąpienie **identyfikatora** (oczywiście poza stałymi tekstowymi) w dalszej części programu przez podany ciąg symboli. Symbolem jest tu dowolny identyfikator, słowo zastrzeżone, stała, operator lub znak przestankowy. Przykładowo — dyrektywa:

```
# define BUFSIZE 512
```

powoduje zastąpienie każdego wystąpienia BUFSIZE w programie przez stałą 512.

Makrodefinicja jest podobna, np. dla linii;

```
# define max (a, b) ((a>b? (a) : (b))
                     ( (a) > (b)? (a) : (b) )
```

preprocesor zastąpi każde wystąpienie **max (n1,n2)** przez podany ciąg z zastąpionymi parametrami. Przykładowo — użycie:

```
max (i++, j++)
```

da w wyniku podstawienia:

```
( (i++) > (j++) ? (i++) : (j++) )
```

Inaczej niż w przypadku funkcji, preprocesor sprawdza, czy liczba argumentów wywołania jest równa liczbie parametrów.

Linia postaci:

```
# undef identyfikator
```

oznacza że od tego miejsca uprzednia definicja identyfikatora traci swą ważność. Wiele kompilatorów używa takich definiowanych identyfikatorów, dlatego zawsze należy upewnić się, że nowo definiowane identyfikatory są inne od istniejących.

Włączanie plików

Preprocesor może włączyć zawartość innego pliku do programu. Plik włączany może zawierać dowolne linie sterujące preprocesorem. Dzieje się tak wskutek użycia dyrektywy:

```
# include „nazwa_pliku"
```

Preprocesor w miejsce tej linii wpisuje całą zawartość nazwanego pliku. **Nazwa pliku** jest szukana najpierw w macierzystym skorowidzu, z którego pochodzi program, a następnie — w standardowych skorowidzach systemowych. Użycie dyrektywy postaci:

```
# include <nazwa_pliku>
```

ma skutek podobny, z tym że plik jest szukany tylko w standardowych skorowidzach systemowych.

Kompilacja warunkowa

Konstrukcja ta umożliwia kompilację części programu zależnie od podanego warunku, np. zależnie od rodzaju komputera docelowego lub od stanu zaawansowania programu. Zapisuje się to w formie:

```
# if stałe_wyrażenie
    część1
# else
    część2
# endif
```

lub — bez części # **else**. Preprocesor oblicza **stałe wyrażenie** i jeśli jego wartość jest niezerowa, to włącza do programu **część** — w przeciwnym wypadku włącza **część2** lub nie robi nic.

Innymi formami warunku są:

```
# ifdef identyfikator
```

lub

```
# ifndef identyfikator
```

W pierwszej formie warunek jest spełniony, gdy **identyfikator** jest na liście definiowanych w preprocesorze identyfikatorów, a w drugiej — odwrotnie.

* * *

Program w języku C powstaje wskutek połączenia (ang. link) skompilowanych plików. Plik zawiera ciąg definicji funkcji i (lub) zewnętrznych deklaracji zmiennych. Zmienne te, podobnie jak funkcje, mogą być tylko klasy **static** lub **extern**. Jeśli ich deklaracje występują przed funkcjami tego pliku, zmienne te mogą być w nich używane już bez deklaracji wewnętrznych. Brak specyfikatora klasy pamięci oznacza, że zmienna jest definiowana, tj. pamięć będzie jej przydzielana w tym pliku. Specyfikator **extern** oznacza natomiast, że pochodzi ona z innego pliku, a specyfikator **static** — że ze zmiennej tej nie można korzystać w innych plikach. We wszystkich plikach stanowiących jeden program zewnętrzna deklaracja danego identyfikatora może wystąpić tylko raz bez specyfikatora.

W kolejnym odcinku na temat języka C, przedstawimy bibliotekę systemową oraz przykład programu.

Ekspansja mikrokomputerów zrodziła dążenie do wyodrębnienia z informatyki wiedzy dotyczącej problematyki mikro — zagadnień tworzących logiczną całość: od projektu do efektu. Realizację takiej koncepcji, zasygnalizowanej już w poprzednim numerze w sprawozdaniu z Seminarium Metodycznego, prezentuje artykuł Wojciecha Cellarego i Jana Węglarza. Zachęcamy do uważnego przestudiowania tej propozycji. (Red.)

WOJCIECH CELLARY
JAN WĘGLARZ
Instytut Automatyki
Politechnika Poznańska

Koncepcja nauczania mikroinformatyki

Jesteśmy obecnie świadkami, czy raczej — obserwatorami nowego etapu rewolucji naukowo-technicznej, spowodowanej masowym zastosowaniem techniki mikroprocesorowej. Rewolucja ta przeradza się — zdaniem wielu — w rewolucję cywilizacyjną. Masowe zastosowanie mikroprocesorów powoduje istotne zmiany w życiu społecznym, zaś informatyka zostaje udostępniona — dzięki tej technice — wielkiej rzeszy nieprofesjonalistów, od uczniów szkół podstawowych począwszy. I choć określenia typu r e w o l u c j a mogą razić przesadną egzaltacją, faktem jest zjawisko gospodarczo-społeczne o skali i randze światowej.

Również Polska stoi przed koniecznością udziału w tej rewolucji, pomimo swych niewątpliwych zaniedbań i opóźnień. Technika mikroprocesorowa jest — co oczywiste — jednym z warunków koniecznych rozwoju gospodarki. Podstawowym zaś problemem jest dzisiaj wykształcenie kadry specjalistów w tej dziedzinie. Myślimy tu o wykształceniu gruntownym, a nie o kursach czy nawet studiach podyplomowych, które mogą stanowić jedynie rozwiązanie częściowe i doraźne. Spotyka się bowiem dość często pogląd, że wystarczy dokonać zakupu gotowych urządzeń mikrokomputerowych i przyuczyć do ich obsługi dotychczasową kadrę

Artykuł ten jest syntetyczną wersją referatu plenarnego wygłoszonego na konferencji „Mikrokomputery w Automatyce i Technice Systemów", Wrocław, wrzesień 1984.

w danym zakładzie. Nie negujemy konieczności uaktualniania wiedzy, uważamy jednak, że pogląd ten stanowi niebezpieczne uproszczenie. Niezbędne jest bowiem wszechstronne kształcenie ludzi konstruujących systemy mikromputerowe dla konkretnych zastosowań i twórczo posługujących się nimi.

W strukturze nauczania mikroinformatyki wyróżniamy trzy poziomy.

Na poziomie pierwszym umieścimy **kształcenie ogólne**, którego celem jest informatyzacja szerokich kręgów społeczeństwa. Odbywałoby się ono już w szkole średniej, a w dalszej perspektywie — podstawowej.

Drugi poziom obejmowałby **profesjonalne zastosowania użytkowe**: obliczenia inżynierskie, wspomaganie projektowania, automatyzację prac administracyjnych czy rozliczeń, sterowanie urządzeniami i procesami przemysłowymi itp. Celem jest tu przygotowanie kadry umiejącej posługiwać się mikrokomputerami w swojej działalności zawodowej. Kształcenie to obejmowałoby przede wszystkim studentów szkół wyższych, zwłaszcza kierunków technicznych, ekonomicznych, przyrodniczych i medycznych.

Najwyższy poziom stanowi **kształcenie inżynierów mikroinformatyków**, którzy po ukończeniu studiów byliby w stanie skonstruować dowolną konfigurację sprzętu i oprogramowania systemów mikrokomputerowych. Kształcenie to byłoby realizowane w postaci specjalności na kierunkach „Informatyka", „Elektronika" i „Automatyka".

W tym artykule zajmiemy się problemem kształcenia inżynierów mikroinformatyków, czyli kadry o najwyższych kwalifikacjach. Wykształcenie tej kadry ma bowiem znaczenie zasadnicze — dla przyspieszenia rozwoju systemów mikrokomputerowych w Polsce, a także dla przygotowania kadry realizującej kształcenie na niższych poziomach.

ZAKRES PRZEKAZYWANEJ WIEDZY

Zakładamy na wstępie, że studenci mikroinformatyki odbyli podstawowy kurs matematyki, fizyki, elektroniki, informatyki i automatyki.

Ważne jest, by mikroinformatycy byli wykształceni wszechstronnie — zarówno w dziedzinie sprzętu, jak i oprogramowania. Przy układaniu programu studiów szczególny nacisk należy zatem położyć na specyfikę projektowania systemów mikrokomputerowych na to, co różni je od innych systemów komputerowych, czy szerzej — systemów cyfrowych. Dlatego też trzeba pokazywać możliwości elastycznego podziału funkcji systemu mikrokomputerowego pomiędzy jego sprzęt i oprogramowanie. Innymi słowy — należy przedstawiać możliwości realizacji określonych funkcji systemu w różnych wariantach: na drodze bardziej sprzętowej lub bardziej programowej. Wybierając odpowiednie warianty, można konstruować systemy najlepiej dostosowane do danego zastosowania, z punktu widzenia kosztów i efektywności działania.

Szczególną rolę w takim nauczaniu projektowania systemów mikrokomputerowych pełni przedmiot kształcący umiejętności modelowania i oceny działania tych systemów. Studenci zapoznają się najpierw z głównymi metodami wykorzystywanymi przy analizie i syntezie systemów mikrokomputerowych, z punktu widzenia oceny i poprawy efektywności ich działania. Następnie omawia się zastosowanie badań ewaluacyjnych do syntezy systemów mikrokomputerowych. Przedstawia się modele funkcjonalne i strukturalne oraz analityczne i symulacyjne, a także różne modele obciążenia. Podaje się techniki testowania, kalibracji i weryfikacji omawianych modeli. Dalej omawia się zastosowanie badań ewaluacyjnych do analizy systemów mikrokomputerowych, w celu szeroko rozumianej poprawy ich działania, również w aspekcie rozwojowym. Omawia się tu przede wszystkim metody pomiaru kryteriów oceny działania, środki pomiarowe: sprzętowe, programowe i hybrydowe, a także metodykę eksperymentu pomiarowego.

Podział na sprzęt oraz oprogramowanie systemów mikrokomputerowych dokonany jest wyłącznie dla ułatwienia prezentacji tematu. W programie studiów kształcenie w obu dziedzinach przenika się wzajemnie, a wiadomości z jednej są konieczne przy nauczaniu drugiej.

Zakładamy, że przed przystąpieniem do zajęć dotyczących sprzętu systemów mikrokomputerowych studenci zdobyli podstawową wiedzę z dziedziny projektowania systemów złożonych z systemów cyfrowych małej i średniej skali integracji. Grupa przedmiotów „sprzętowych" obejmuje: architekturę i organizację systemów mikrokomputerowych (w tym wielomikroprocesorowych), charakterystykę wybranych układów reprezentujących poszczególne rodziny mikroprocesorowe wraz z przykładami ich zastosowań, budowę i zasady pracy urządzeń zewnętrznych systemów mikrokomputerowych, metodykę projektowania technicznego sprzętu systemów mikrokomputerowych oraz metodykę uruchamiania tych systemów.

Proponujemy, aby wykłady na temat sprzętu ilustrować przede wszystkim przykładami z rodzin INTEL 8080/8085 w grupie mikroprocesorów 8-bitowych, INTEL 8086 w grupie mikroprocesorów 16-bitowych oraz INTEL 8048 w grupie mikrokomputerów monolitycznych. Układy pochodzące z tych rodzin są w Polsce najbardziej rozpowszechnione bądź przewidywane do rozpowszechnienia w najbliższej perspektywie. Wymienione rodziny powinny być uznane za reprezentatywne podczas omawiania organizacji i architektury systemów mikrokomputerowych. Przedstawienie ich powinno być ściśle powiązane z wykładem na temat organizacji samych mikroprocesorów, ich architektury logicznej, sygnałów itd. — tak, aby można było wyjaśnić cechy ogólne systemów mikrokomputerowych oraz cechy wynikające ze szczególnych rozwiązań omawianych mikroprocesorów. Trzeba też zwrócić uwagę na architekturę systemów wielomikroprocesorowych, jako że w najbliższych latach będzie ona zapewne dominować.

W tej grupie przedmiotów należy również omówić języki asemblerowe mikroprocesorów. Pojawienie się tego tematu w przedmiotach dotyczących sprzętu ma na celu uwypuklenie związków pomiędzy wykonywaniem rozkazu przez mikroprocesor a przebiegiem sygnałów na jego magistrali. Ignorowanie tych związków mogłoby prowadzić do niekorzystnego, a niestety dość rozpowszechnionego traktowania poziomu języka asemblerowego jako najniższego poziomu systemu, a sprzętu systemu — jako „czarnej skrzynki".

Omawiając układy rodzin mikroprocesorowych, należy podać oryginalne przeznaczenie danego układu, klasę jego zastosowań (ze względu na charakterystykę efektywności działania), strukturę logiczną układu, szczegółowy opis realizowanych funkcji, diagramy czasowe przebiegu sygnałów, opis programowania układu oraz przykłady jego zastosowania do realizacji określonych funkcji w konkretnym systemie mikrokomputerowym.

W ramach opisu urządzeń zewnętrznych powinny być ujęte urządzenia wprowadzania i wyprowadzania danych (monitory ekranowe, klawiatury, czytniki i dziurkarki taśmy papierowej itd.), pamięci zewnętrzne (dyskowe — w szczególności pamięci na dyskach elastycznych oraz dyski typu Winchester, pamięci taśmowe na kasetach), kanały automatyki (przetworniki analogowo-cyfrowe, cyfrowo-analogowe, cyfrowo-cyfrowe), monitory graficzne oraz urządzenia adaptowane, np. telewizor w roli monitora ekranowego czy magnetofon w roli pamięci. W opisie każdego urządzenia winny być zawarte następujące treści: funkcje i przeznaczenie urządzenia, zasada pracy, wymagania sterowania, porównanie z innymi urządzeniami różnych firm, opis alternatywnych rozwiązań sterownika urządzenia, opis wymagań oprogramowania urządzenia oraz ocena efektywności jego działania.

Przedmiot dotyczący projektowania sprzętu powinien omawiać wszystkie czynności od momentu ukończenia projektu logicznego — do zakończenia montażu mikrokomputera. Należy przedstawić: technologie wykonywania obwodów drukowanych i metodykę ich projektowania, metody przeciwdziałania zakłóceniom, zasady montażu pakietów, technologie wykonywania połączeń między pakietami w kasecie oraz standardy i wymagania mechaniczne systemów.

W ramach przedmiotu dotyczącego uruchamiania sprzętu powinny być omówione: urządzenia wykorzystywane w trakcie uruchamiania (w szczególności emulatory układowe) oraz sposób posługiwania się nimi, oprogramowanie wykorzysty-

dokończenie na str. 21

Wielu czytelników zarzuca nam zbyt małe zainteresowanie mikrokomputerem COMMODORE C64. Faktycznie, liczba jego użytkowników gwałtownie wzrasta, a wolnorynkowa cena spadła poniżej 150 tys. złotych (luty 1985)...

Zniesienie ograniczeń celnych zwiększyło możliwości zakupu mikrokomputera i wiele osób stoi przed dylematem: co wybrać. Zamieszczamy więc tym razem porównanie niektórych cech C64 z ZX SPECTRUM — z pozycji tzw. przeciętnego użytkownika, którego mniej interesuje, co siedzi w skrzynce, a bardziej — co z tego wynika. Nieco przewrotnie pominęliśmy możliwość wykorzystywania obu komputerów jako uniwersalnych „maszynek" do gier telewizyjnych, choć jest to niestety bardzo częsta motywacja zakupu. Może jednak uda nam się przekonać czytelników, że komputery powinny przede wszystkim p r a c o w a ć, czyli — ułatwiać nam życie. A to, że równocześnie mogą nam dostarczać rozrywki, jest jedynie ich cechą dodatkową, drugorzędną.

Rzetelnie zaznaczamy, że zarówno ZX SPECTRUM jak i C64 są już w wieku przedemerytalnym. Niemniej o użyteczności komputera w znacznie większym stopniu decyduje ilość dostępnego oprogramowania, niż wewnętrzne cechy konstrukcyjne. Gotowe programy dla tych komputerów (wymieniane przez krajowych użytkowników) liczy się w setkach. Inne komputery, często o znacznie lepszych cechach funkcjonalnych, nie mają niestety w naszym kraju takiego wsparcia.

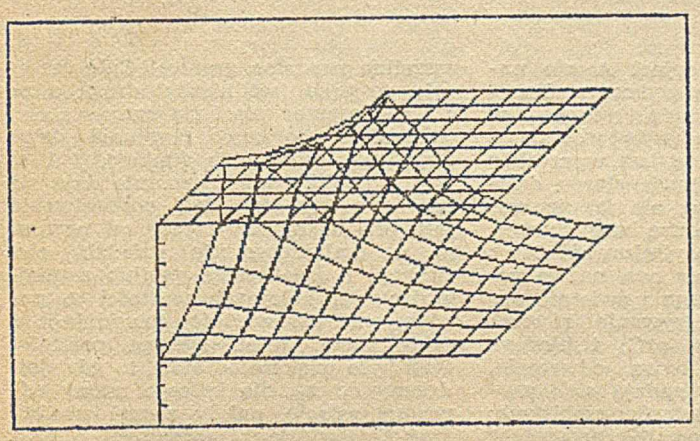
Autor poniższego tekstu wykorzystuje mikrokomputer do obliczeń naukowych. Przez pewien czas użytkował ZX SPECTRUM, a potem „przesiadł" się na C64. Oto jego refleksje.

COMMODORE C 64 czy ZX SPECTRUM?

ZX SPECTRUM, z jego możliwościami graficznymi, pozwala na niemal całkowite odejście od wyników liczbowych i prezentowanie rezultatów w formie graficznej. Problem pojawia się, gdy trzeba je jakoś utrwalić. Standardowa drukarka — to niska jakość wykresów; zdjęcia z ekranu — to kłopot.

Firma COMMODORE oferuje w stosunkowo niskiej cenie system mikrokomputerowy zawierający mikrokomputer C64, stację dysków elastycznych (!) i drukarkę lub kolorowy ploter. Jest to propozycja bez wątpienia atrakcyjna dla tych, którzy mają za mało czasu (lub wiedzy), by kompletować urządzenia różnych producentów.

Pierwsze zetknięcie z C64 było dla mnie przykrym przeżyciem. Przyzwyczajony do przemyślanego zestawu instrukcji języka BASIC w ZX SPECTRUM, z jego pełnymi możliwościami graficznymi, łatwością przechowywania na taśmie tablic z danymi lub graficzną zawartością ekranu, z prostotą przesyłania grafiki na drukarkę — byłem C64 bardzo rozczarowany. Reklamowana wysokorozdzielcza grafiki (200×320 punktów) jest trudno dostępna, przekazywanie danych poprzez instrukcję Print mało efektywne (wolne i mała gęstość zapisu), nie mówiąc już o wyprowadzaniu grafiki na drukarkę.



Przykład zobrazowania funkcji przestrzennej z wykorzystaniem C64 i drukarki MPS 801

Eksploatując już ponad rok zestaw składający się z mikrokomputera C64, stacji dyskowej 1541 i drukarki mozaikowej MPS 801, dorobiłem się zbioru kilkuset programów, sterty literatury i odrobiny doświadczeń. Od tamtej strasznej chwili pierwszego zetknięcia się z C64 zmieniłem o nim zdanie. Porównując tylko komputery — wybrałbym ZX SPECTRUM, ale przy porównaniu systemów sytuacja wygląda inaczej. Podstawową zaletą zestawu z C64 jest bowiem jego stacja dyskowa.

BASIC C64 jest bardzo ubogi. Istnieje natomiast dość bogate oprogramowanie dodatkowe, umożliwiające wykorzystanie potencjalnych możliwości C64. Aby w pełni wykorzystać możliwości graficzne, konieczne jest użycie jednego z rozszerzeń języka BASIC, np.: SIMON'S BASIC, SUPERGRAFIKA lub ULTRABASIC (dostępne instrukcje typu „postaw punkt", „wykreśl linię", „wyprowadź obraz z ekranu na drukarkę"). Dopiero teraz możliwości graficzne ZX SPECTRUM i C64 są porównywalne. SPECTRUM oferuje do dyspozycji programisty 40 KB RAM-u, a C64 — 30 KB (C64 ma potencjalnie większą pamięć — 64 KB, ale jest ona dostępna tylko w języku maszynowym).

Szybkość realizacji programów w języku BASIC dla C64 jest prawie dwukrotnie większa niż w przypadku SPECTRUM. Można ją zwiększyć (2—3-krotnie, czasem więcej), stosując kompilatory. Na C64 mogą być stosowane: AUSTRO-COMPILER, PETSPEED, DTL-COMPILER. Niestety, nie dają sobie one rady z grafiką.

Porównując pamięci masowe: magnetofony, microdrive i stację dyskową, należy podkreślić, że ZX SPECTRUM współpracuje z dowolnym typem magnetofonu (czasami są duże kłopoty z zapewnieniem poprawnej współpracy) i robi to szybko, natomiast C64 współpracuje tylko ze specjalnym, robi to wolniej, ale za to — bardzo pewnie. Istnieje program (TURBO TAPE) radykalnie zwiększający szybkość współpracy C64 z magnetofonem (10-krotnie).

Mikrodrive'y są bardzo szybkie, ale wymagają specjalnego interfejsu i okazują się czasem zawodne. Stacja dyskowa 1541, współpracująca z C64, jest bardzo wygodna, cicha, pewna, ale b a r d z o wolna. Program o wielkości 23 KB ładowany jest 60 sekund! Sytuację częściowo ratują programy szybkiego ładowania zmniejszające ten czas 4—10 krotnie.

W ciągu rocznej eksploatacji zestawu C64, stacji 1541 i drukarki MPS 801 wystąpiły pewne niesprawności. Dwukrotnie mikrokomputer po włączeniu nie zgłaszał systemu. Pozostawiony na kilka godzin bez aktywnego programu — „pada", natomiast wykonując program może pracować wiele godzin. Stacja dysków nie wykazywała żadnych niesprawności, poza koniecznością powtórnego formatowania niektórych dysków. Używam dyskietek jednostronnych — dwustronnie. Przy 50 dyskietkach różnych firm nie miałem z tego powodu żadnych kłopotów.

Po jakimś czasie w drukarce wysycha zbiorniczek tuszu i wydruki stają się blade. Nadal jednak są czytelne.

Niemal wszystkie programy pojawiające się w Europie są dostępne w kraju. Na razie nie istnieje w Polsce rynek programów na C64. Są one wymieniane lub przekazywane bezpłatnie. Liczba programów sięga tysiąca. Wśród firmowych programów można znaleźć gry (zręcznościowe, logiczne, strategiczne), programy do obliczeń finansowych (**Multiplan, Omnicalc**), do grafiki (**Supergrafika**), programy wspo-

magające programowanie — monitory i asemblery (**Supermonitor, Macrofire, Develop 64, Profi-mon**), bazy danych (**Datamat, Database, Data Manager**). Programy kopiujące (**Supercopy, F-16**) i języki (**FORTH, LOGO, PILOT, COMAL, PASCAL**, rozszerzenia języka BASIC — **Simon's Basic, UltraBasic, C64Plus, C64 exbasic, Exbasic, levell II**) Są programy służące do tworzenia grafiki i „malarstwa komputerowego" (**KoalaPainter, Moodle Doodle, Paintpic**). Istnieją także programy do redagowania tekstów (**Texter, Heswriter, Wordpro 3 plus**).

Interesującym programem jest syntetyzator mowy (SAM/reciter), umożliwiający zaprogramowanie w języku BASIC tekstu zrozumiale mówionego przez komputer (tekst angielski jest wyraźny; z polskim jest znacznie gorzej).

Kopiowanie programów napisanych przed 1984 rokiem nie przedstawia większych trudności. Najnowsze programy i te sprowadzane z USA mają nowe systemy zabezpieczeń i w zasadzie nie dają się kopiować.

Po kraju krąży trochę książek poświęconych C64, a w zasadzie — jego ukrytym możliwościom. Są to m.in.: **Programer's Reference Guide, More on the Sixty Four, Inside the Commodore 64, How to program your Commodore 64 in 6502/20 machine language, Das Grafikbuch zum Commodore 64, Beherschen Sie Ihren Commodore 64, 64 Tips & Trics, Commodore intern, Der Commodore 64 und der Rest der Welt, Das Gross Floppy Buch.** Pojawiają się też czasopisma poświęcone tylko COMMODOROWI: **Ahoy! Commodore Users, Compute, Compute's! Gazette, Power Play,** Czasopisma te zawierają wiele interesujących materiałów o użytkowaniu C64, a także — sporą gamę programów. Szczególnie interesujące są rubryki z odpowiedziami na pytania czytelników.

C64 ma dwie możliwości rozbudowy. Poprzez złącza użytkownika (ang. user port) można dołączyć płytkę umożliwiającą szybką współpracę z magnetofonem, zrealizować wyjście na drukarkę (typu Centronics), dołączyć konwertery napięć (aby uzyskać kanał szeregowy, zgodny ze standardem RS 232C) lub przetwornik analogowy. Drugą możliwość daje złącze rozszerzające (ang. expansion port), w które wkłada się moduły z ROM-ami.

Interesującą możliwością rozbudowy jest dołączenie przystawki z mikroprocesorem Z80, co umożliwia pracę w systemie CP/M. System ten jest w kraju dość popularny (możliwa jest więc wymiana — przenoszenie oprogramowania, a czasem nawet plików danych). W systemie CP/M dostępne są niezłe kompilatory języków PASCAL i FORTRAN, co pozwala na realizację poważniejszych programów i korzystanie z bogatego oprogramowania w FORTRANIE. Nadal jednak pozostaje nie rozwiązana sprawa grafiki.

Wydaje się, że C64 jest niezłym komputerem dla zaawansowanych programistów, znających język maszynowy i konstrukcję wewnętrzną komputera. Jego możliwości są w wielu przypadkach niedostępne dla przeciętnego użytkownika. Jest to też zarazem komputer dla użytkowników, którzy potrafią tylko naciskać kilka klawiszy, potrzebnych do załadowania gotowego programu.

Użytkownicy tacy jak ja, nie należący do żadnej z wymienionych grup, mają kłopoty ze zrealizowaniem swoich zamierzeń. Pocieszające jest to, że pojawiają się wciąż nowe programy, w których kolejne problemy są choć częściowo rozwiązywane. Przykładowo PASCAL daje możliwość programowania strukturalnego, ale nie ma grafiki. SIMON'S BASIC ma grafikę, ale jest wolny. Natomiast najnowsza wersja języka COMAL dla C64, zawierająca elementy PASCALA, MODULI-2, PL/1, LOGO i języka C, chyba rzeczywiście grupuje sensowny zestaw instrukcji.

Myślę, że mimo pojawienia się nowych modeli mikrokomputerów, C64 — ze względu na niewysoką cenę, potencjalne (ciągle jeszcze nie w pełni wykorzystane) możliwości a przede wszystkim ogromną ilość dostępnego oprogramowania — jeszcze długo będzie się cieszył powodzeniem.

JANUSZ PIECHNA
Warszawa

COMMODORE C64

Mikroprocesor: 6510 (kompatybilny z 6502)
Częstotliwość zegara: 1 MHz
ROM: 20 KB
RAM: 64 KB
RAM użytkowy: 38 KB (BASIC); 52 KB (kod maszynowy)
Szybkość zapisu na kasecie magnetofonowej: 600 bitów/s
Klawiatura: QWERTY (semiprofesjonalna)
Klawisze funkcyjne: 4 (podwójnie definiowane)
Klawiatura numeryczna: brak
Gnerowanie dźwięków: 3 kanały po 8 oktaw
Generator szumów: jest
Znaków w linii: 40
Liczba linii na ekranie: 25
Rozdzielczość: 320×200
Liczba kolorów: 16
Wyjście wizyjne: FBAS
Wyjście TV: PAL
Interfejs z drukarką: Port równoległy + nakładka programowa dla drukarek z wejściem Centronics
Interfejs szeregowy: funkcjonalnie V24 (elektrycznie wymaga przystawki)
Standardowy język: MICROSOFT BASIC
Rozszerzenia: dyski elastyczne +CP/M (zapis dyskietek nie kompatybilny); manipulatory; pióro świetlne
Cena: 700 marek RFN — 190 funtów ang.

Akademia uczy eleganckiego programowania. Przeznaczona jest dla osób mających podstawowe doświadczenie w programowaniu. Zainteresowanych prosimy o skomentowanie dotychczasowych „wykładów".

Akademia mikroKLANU (6)

Dekada

W tym odcinku odejdziemy nieco od strukturalizacji programów — prezentowana procedura zupełnie nie nadaje się do włączenia w długi program, choć jest procedurą iteracyjną. A nie nadaje się, gdyż chcemy wykorzystać najlepszy komputer — ludzki mózg — w algorytmie heurystycznym. Działanie heurystyczne jest wprawdzie absolutnym przeciwstawieństwem działania algorytmicznego, ale jak inaczej można nazwać procedurę heurystyczną, która daje się zalgorytmizować? Nasz algorytm heurystyczny nosi nazwę **poszukiwania dekadowego** i świetnie nadaje się do rozwiązywania równań przestępnych metodą prób i błędów.

Chociaż z reguły bywa odwrotnie, omawiany algorytm nadaje się wyłącznie dla komputerów osobistych: niechby ktoś spróbował pracować w ten sposób w systemie interakcyjnym. Szybko koszt jego pracy przekroczyłby wszelkie rozsądne granice! Całe szczęście, że prąd jest niezbyt drogi, a za mikrokomputer płaci się raz.

Problem dotyczy równania typu $f(x)=g(x)$, gdzie obie funkcje, f i g, są „skomplikowane", tzn. nie dają się łatwo odwracać. Idea poszukiwania jest prosta: należy zacząć od pewnego x_0 (należącego do dziedzin obu funkcji) i zwiększając go (lub zmniejszając) dla kolejnych wartości x_i porównywać, czy wartości obu funkcji są takie same. Jeżeli np. na początku $f(x_0)$ było większe od $g(x_0)$, to gdy dojdziemy do x_n, dla którego mamy odwrotną relację, należy zacząć zmniejszać (lub zwiększać) argument o krok dziesięciokrotnie mniejszy, przy czym decyzję pozostawiamy tu użytkowniko-

wi. Niewielka liczba kroków wystarczy by osiągnąć rozwiązanie równania z dużą dokładnością, a program, którego używamy, liczy tylko kilkanaście linijek. Porównajmy to z bibliotecznymi programami służącymi do rozwiązywania równań przestępnych. Prawda, jaka potęga tkwi w naszym komputerze osobistym?

Proponujemy Czytelnikom rozwiązanie przykładu przy $x_0=1$ oraz $dx=1$. Mamy nadzieję, że wszystkim uczestnikom Akademii udało się osiągnąć rozwiązanie (1,5132...). Proszę zwrócić uwagę, że nasz program jest stabilny, tzn. trudno go wyprowadzić z równowagi, a jednocześnie pozwala znajdować wiele różnych rozwiązań. Wszystko w rękach użytkownika.

W ten sposób nasza Akademia doszła do najpoważniejszego problemu eleganckiego programowania — algorytmów. Tak naprawdę dobre, efektywne programowanie to właśnie znajdowanie dobrych algorytmów, cała reszta jest już tylko kodowaniem. Naszym Czytelnikom polecamy liczne książki, wydane przez WNT w serii INFORMATYKA, gdzie można znaleźć algorytmy wielu interesujących problemów. W szczególności — książkę „Zbiór zdań z informatyki" F. L. Bauera, R. Gnatza i U. Hilla (WNT, 1981).

JAKUB TATARKIEWICZ

Oto nasza procedura:

```
LIST
5 REM PROCEDURA POSZUKIWANIA DEKADOWEGO,
6 REM WSTAW FUNKCJE W LINIACH 10 ORAZ 20.
10 DEF FN F(X)=X*COS(X)
20 DEF FN G(X)=SIN(X)/(X+10)
30 INPUT "PUNKT STARTOWY X0=";X0
40 INPUT "KROK POCZATKOWY DX=";DX
50 X=X0-DX
60 X=X+DX
70 Y1=FN F(X)
80 Y2=FN G(X)
90 PRINT "X=";X,"F=";Y1,"G=";Y2
100 INPUT "CZY ZMNIEJSZYC KROK (1-TAK,0-NIE)";P
110 IF P=0 THEN GOTO 60
120 DX=-DX/10
130 PRINT "NOWY KROK DX=";ABS(DX)
140 INPUT "MOZE JUZ WYSTARCZY (1-TAK,0-NIE)";P
150 IF P=0 THEN  GOTO 60
160 PRINT "ROZWIAZANIE X=";X
170 END
READY.
```

Po drugim odcinku Akademii pt. „Silnia" otrzymałem sygnały, że program nie działa. Rzeczywiście, pewne komputery nie dają tego samego wyniku gdy mnożymy przez 0,1 w porównaniu z dzieleniem przez 10 (linie 100 i 140 programu).

Wszystkim kursantom polecam sprawdzenie, jak zachowuje się ich komputer — ostatecznie trzeba znać swoje narzędzie pracy.

Otwieramy nową rubrykę. Publikować będziemy w niej drobne pomysły, które — zdaniem autorów — nie są godne całego artykułu. Ale czasem na drobnych pomysłach opiera się rewelacyjna całość. Pomysły mogą dotyczyć spraw programowych, sprzętowych lub łączyć elementy obydwu. Powinny być sprawdzone (o ile nie są oczywiste) — jest to bardzo ważne, gdyż redakcja nie ma na razie takich możliwości. Autor pomysłu firmuje go więc swoim nazwiskiem. Każdy opublikowany pomysł premiowany będzie egzemplarzem INFORMATYKI i nagrodą w wysokości od 500 do 1000 złotych. O wysokości nagrody decyduje, zgodnie ze swoim sumieniem i poglądami, redaktor mikroKLANU. Decyzja jest ostateczna, choć podobno można ją zaskarżyć do sądu.

Ponieważ mikroKLAN przygotowywany jest z paromiesięcznym wyprzedzeniem, w kilku najbliższych numerach — oczekując na odzew czytelników — publikować będziemy pomysły zaczerpnięte z różnych pism zachodnich.

I jeszcze jedna uwaga. Nadsyłane pomysły mogą być inspirowane przez materiały ukazujące się w innych pismach. Powinny jednak zawierać przynajmniej pewien wkład wniesiony przez nadsyłającego. Prosimy w takich przypadkach o podawanie źródła inspiracji.

POMYSŁ!

Jak odzyskać kontrolę nad programem realizowanym z pamięci, której nie ma?

W nowszych konstrukcjach mikrokomputerów 8-bitowych zaczyna brakować przestrzeni adresowej i stosuje się różne sztuczki, aby dołączyć trochę dodatkowego RAM-u lub ROM-u oprócz przepisowych 64 KB. Nadal jednak istnieją systemy z „dziurami adresowymi", w których nie ma żadnej pamięci (np. ZX SPECTRUM — wersja 16K). Dziury zdarzają się zresztą nawet w systemach z przełączanymi blokami pamięci.

Skoro jest dziura, to zgodnie z prawami Murphy'ego prędzej czy później w nią wpadniemy. Może się to stać za sprawą błędnie wyliczonego adresu (dla 8080 istnieje niebezpieczny rozkaz PCHL), błędnej operacji na stosie lub... kolegi, który włączył młynek do kawy i wygenerował w ten sposób falę impulsów zakłócających.

Istnieje jednak pewien sposób na wydobycie się z takiej opresji. Otóż odczytując informację spod adresu, do którego nie jest przypisana pamięć, zazwyczaj otrzymujemy kod FF_H. Dla mikroprocesorów takich jak 8080, Z80 czy NSC800 jest to kod rozkazu RST7, powodującego wywołanie podprogramu rozpoczynającego się od adresu 0038_H. Jeśli pod adresem 0038_H umieścimy odwołanie do procedury obsługi zaistniałego błędu, to odzyskamy kontrolę nad pracą systemu. Dodatkowo, gdy wskaźnik stosu ciągle jeszcze odwołuje się do pamięci RAM, to na stosie mamy zapamiętany błędny adres, który próbował odczytać procesor. Informacja ta może ułatwić odnalezienie błędu w programie, który był wcześniej realizowany.

Rozszerzeniem opisanego pomysłu jest wypełnianie pamięci RAM kodem FF_H przed wpisaniem programu przeznaczonego do realizacji. W ten sposób, jeśli program „wyskoczy" poza wykorzystywany obszar, otrzymamy reakcję analogiczną do wcześniej opisanej.

Warto jeszcze wspomnieć o przypadkach kiedy odwołanie do nieistniejącej pamięci nie musi oznaczać odczytania kodu FF_H. Jeżeli mamy do czynienia z systemem zbudowanym na kilku płytkach, to na pojemnościach pasożytniczych — związanych z rozbudowaną szyną danych — przez pewien czas utrzymuje się informacja związana z poprzednią operacją. Gdy szyna nie zostanie wysterowana nową informacją, to wprowadzony będzie kod odpowiadający „resztkom" poprzedniej operacji. Może się więc tak niefortunnie złożyć, że odczytany kod spowoduje ponowne wejście w obszar istniejącej pamięci i realizację różnych dziwnych rzeczy. W tej sytuacji pozostaje jednak nadzieja (poparta stosunkowo dużym prawdopodobieństwem), że odczytany kod będzie inny od kodów odpowiadających rozkazom skoku (wywołanie podprogramu) i po kilku krokach pojemności rozładują się.

Niektóre standardy magistrali systemowej (np. MULTIBUS) przewidują stosowanie buforów odwracających (są one szybsze). W takim przypadku mikroprocesor odczytywałby kod 00 odpowiadający rozkazowi NOP (nic nie rób), co oczywiście eliminuje możliwość wykorzystania opisanego pomysłu. W standardach tych przewidziano jednak zabezpieczenia realizowane sprzętowo, które blokują dostęp do pustych obszarów.

AJP

Inspiracją był tekst „Software scheme restarts your stalled µP", zamieszczony w EDN, 27 grudnia 1984

Dla wielu osób w naszym kraju możliwość generowania dźwięków przez mikrokomputer to cecha mało istotna. Faktycznie gdyby wykorzystywać komputery osobiste wyłącznie do wykonywania obliczeń, to jedynym sensownym dźwiękiem mogłoby być sapanie. Stworzono je jednak do zupełnie innych celów i dlatego praktycznie każdy oferowany na Zachodzie model posiada rozbudowane możliwości generowania różnych dźwięków. Niestety, cecha ta pobudza jedynie wyobraźnię autorów gier, mimo że mogłaby chyba z równym powodzeniem urozmaicać programy użytkowe. Wydaje się, że np. grający procesor tekstu zrobiłby furorę. Zresztą pomysł odciążenia wzroku i przerzucenia niektórych zadań na inne zmysły (np. słuch) nie jest nowy. Na razie jednak nie doczekał się żadnej spektakularnej realizacji. Osoby z inwencją nadal mają szansę.

Układ firmy GENERAL INSTRUMENTS stał się niemal standardem i można go spotkać w wielu bardzo różnych mikrokomputerach (np. komputery w standardzie MSX czy AMSTRAD CPC 464).

Poniższy opis powinien okazać się użyteczny zarówno dla tych, którzy chcą rozbudować posiadany komputer, jak i tych, którzy korzystają z komputera już wyposażonego we wspomniany układ, lecz mają skąpe informacje o jego sposobie wykorzystania.

Programowany generator dźwięków AY-3-8910/12

Produkowany przez firmę GENRAL INSTRUMENTS układ programowanego generatora sygnałów akustycznych umożliwia realizację różnorodnych efektów dźwiękowych. Rola mikroprocesora sprowadza się do wpisania pewnej sekwencji programującej, po czym układ w sposób autonomiczny realizuje otrzymane polecenia.

Producent przewidział trzy niezależne od siebie wyjścia analogowe. Sprzyja to uzyskiwaniu bardziej realistycznych efektów. Pasmo możliwych do uzyskania częstotliwości rozpoczyna się na kilkunastu Hz, a kończy się powyżej 60 kHz. Częstotliwości mogą być generowane z rozdzielczością od 15 do 30 Hz (zależy to od częstotliwości zegara doprowadzonego do wejścia CLOCK). Napięcie na wyjściach analogowych może zmieniać się z dynamiką 60 dB. Poziom sygnału wyjściowego sterowany jest niezależnie dla każdego kanału przez 4-bitowy przetwornik cyfrowo-analogowy. Pozwala to uzyskać 16 poziomów głośności, które — aby lepiej dopasować się do możliwości percepcyjnych człowieka — rozłożone zostały w skali logarytmicznej. Napięcie na wyjściach analogowych zmienia się w zakresie 0—1 V.

Oprócz wyjść analogowych układ ma również uniwersalne we-wy cyfrowe. W zależności od wersji układu, przewidziany jest jeden (AY-3-8912) lub dwa (AY-3-8910) uniwersalne porty 8-bitowe. Są to jednak wyjścia o niewielkiej obciążalności (1 bramka TTL).

Opis wyprowadzeń

DA7...DA0 — 8-bitowa szyna danych wykorzystywana do komunikacji z systemem mikroprocesorowym.

A8, A9/ — linie wyboru, które mogą być wykorzystane przy zapisie adresu rejestru. Jeżeli linie nie są podłączone, układ zachowuje się tak, jakby podane były na nie stany aktywne. Jednak dla uniknięcia skutków ewentualnych zakłóceń wskazane jest połączenie A8 i A9/ odpowiednio z +5 V i masą.

RESET/ — wejście umożliwiające wyzerowanie wszystkich rejestrów przez podanie poziomu niskiego.

CLOCK — wejście zegara (1—2 MHz) wykorzystywanego do generacji przebiegów okresowych.

BDIR, BC1, BC2 — linie sterujące kierunkiem komunikacji z mikroprocesorem i rodzajem wykonywanej operacji:

BDIR	BC2	BC1	Realizowana funkcja
0	0	0	układ nie komunikuje się z mikroprocesorem
0	0	1	zapis adresu rejestru
0	1	0	układ nie komunikuje się z mikroprocesorem
0	1	1	odczyt z zaadresowanego wcześniej rejestru
1	0	0	zapis adresu rejestru
1	0	1	układ nie komunikuje się z mikroprocesorem
1	1	0	zapis do zaadresowanego wcześniej rejestru
1	1	1	zapis adresu rejestru

Niektóre funkcje można zrealizować kilkoma sposobami, co pozwala uprościć układ adaptujący sygnały generowane przez mikroprocesor.

Kanał analogowy A, B, C — wyjścia sygnałów akustycznych.

IOA7...IOA0, IOB7...IOB0 — uniwersalne porty we-wy. Jeżeli zostały zaprogramowane jako wejście, to w przypadku braku zewnętrznego sterowania odczytywane są logiczne jedynki (zastosowano wewnętrzne rezystory podciągające).

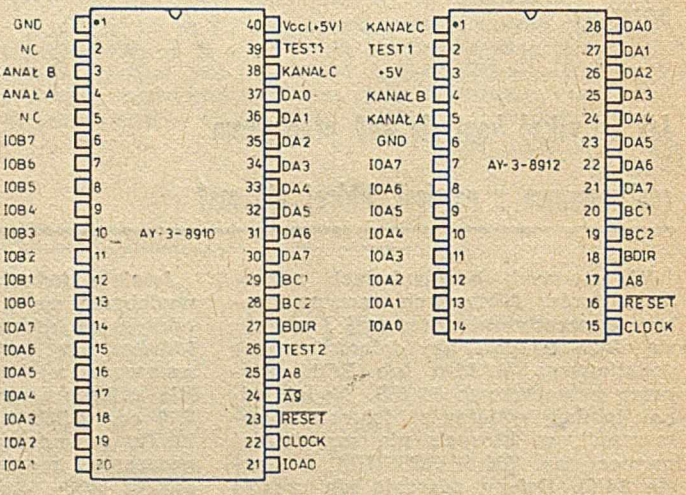
TEST 1, TEST 2 — linie wykorzystywane w procesie produkcyjnym; powinny pozostawać nie podłączone.

V_{cc} — napięcie zasilające +5 V,

V_{ss} — masa układu.

Architektura układu

Układ zawiera następujące bloki funkcjonalne:
— generator tonu podstawowego
— generator szumu
— mikser
— sterownik poziomu wyjściowego
— generator obwiedni
— przetwornik cyfrowo-analogowy
— matryca rejestrów
— porty we-wy

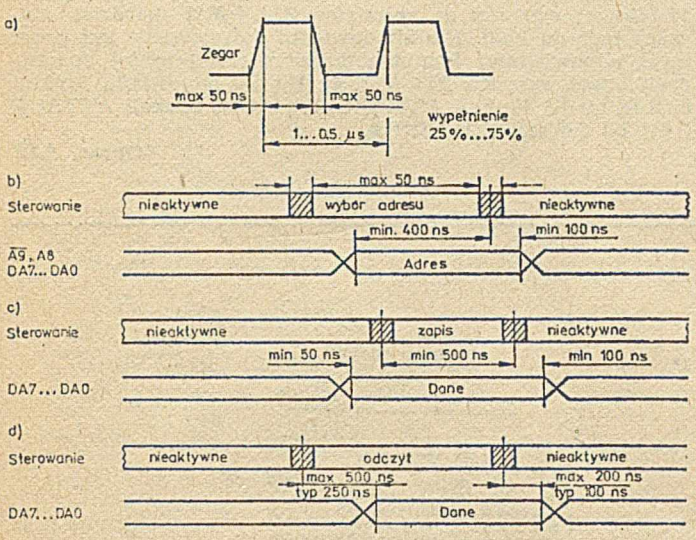


Rys. 1. Rozmieszczenie końcówek programowanego generatora dźwięków

Generator tonu podstawowego zawiera trzy niezależnie programowane bloki produkujące falę prostokątną dla każdego z kanałów. Generator szumu moduluje (częstotliwościowo) w sposób pseudolosowy przebieg prostokątny o zaprogramowanej częstotliwości bazowej.

Przebieg generatora tonu podstawowego uzyskiwany jest w wyniku podziału przez 16 sygnału zegarowego, doprowa-

dzonego do wejścia CLOCK i dodatkowego podziału przez zaprogramowaną wartość 12-bitową. Przebieg bazowy generatora szumów uzyskiwany jest w wyniku podziału sygnału zegarowego przez 16 a następnie — przez zaprogramowaną wartość 5-bitową.



Rys. 2. Przebiegi czasowe; a) zegar, b) wpisywanie adresu rejestru (na wejścia BDIR, BC2, BC1 powinna być podana jedna z sekwencji: 001, 100 lub 111), c) wpisywanie danej do zaadresowanego rejestru (na wejściach BDIR, BC2, BC1 — wymuszona sekwencja 110), d) odczytywanie danej z zaadresowanego rejestru (na wejściach BDIR, BC2, BC1 — wymuszona sekwencja 001)

Mikser umożliwia — osobno dla każdego kanału — zmieszanie szumu z przebiegiem z generatora tonu podstawowego lub wybranie jednego z tych źródeł.

Sterownik poziomu wyjściowego ustawia stały (zaprogramowany przez mikroprocesor) poziom sygnałów wychodzących z miksera lub zmienia je zależnie od chwilowych „dyspozycji" generatora obwiedni. Przetworniki C/A umożliwiają konwersję wpisanej przez mikroprocesor binarnej informacji o poziomie wyjściowym na napięcie odniesienia dla sterownika poziomu wyjściowego.

Okres, z jakim powtarzany jest kształt obwiedni, otrzymywany jest w wyniku podziału sygnału zegarowego przez 256, a następnie — przez zaprogramowaną wartość 16-bitową.

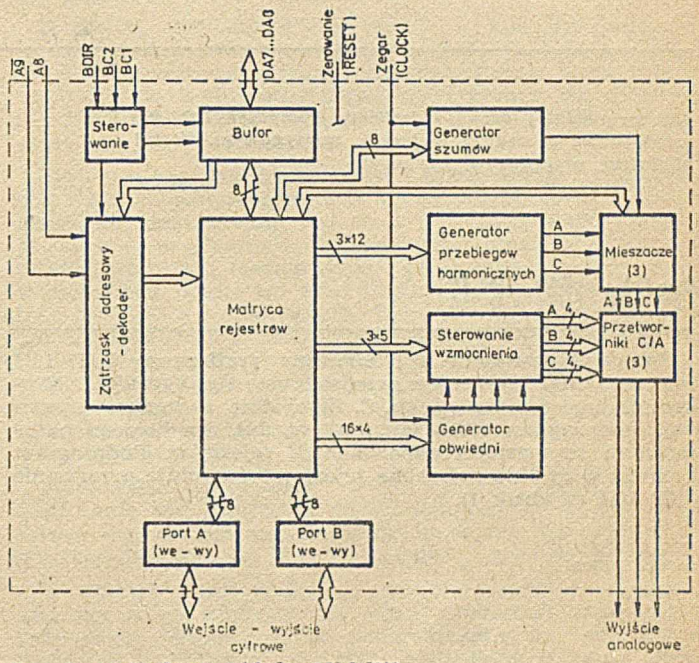
Matryca rejestrów służy do zapamiętania wpisywanych przez mikroprocesor kodów sterujących poszczególnymi blokami funkcjonalnymi układu. Dla uproszczenia programu obsługi, informacja wpisana do każdego z 16 rejestrów sterujących (R0...R15) może być w dowolnym momencie odczytana.

Rejestry R14 i R15 związane są z przesyłaniem informacji przez porty we-wy (odpowiednio: IOA i IOB). Jeżeli port został zaprogramowany jako wejście, to wprowadzaną informację można odczytać z odpowiadającego mu rejestru. Podobnie dla portu zaprogramowanego jako wyjście — informację wpisuje się do związanego z nim rejestru. Odczyt i zapis portów nie ma wpływu na przebiegi akustyczne generowane przez układ. Porty muszą być definiowane w całości (8 bitów) jako wejście lub wyjście.

Programowanie układu

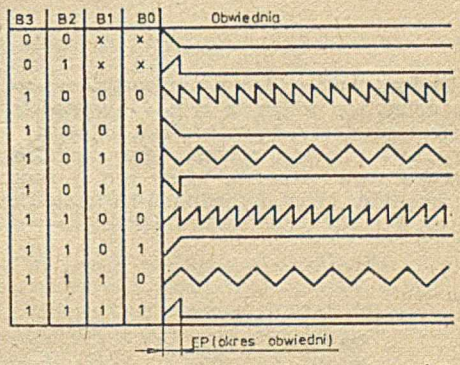
Dostęp do określonego rejestru możliwy jest po uprzednim wpisaniu adresu do układu. Adres pamiętany jest do momentu, kiedy wykonany zostanie zapis innego adresu. Umożliwia to wielokrotny dostęp do rejestru.

Wpis adresu może odbywać się dwoma sposobami, w zależności od rodzaju wykorzystywanego mikroprocesora. Dla mikroprocesorów z multipleksowaną szyną danych i adresową (na tych samych liniach podawane są na przemian dane i adresy), można zrezygnować z dodatkowego układu dekodera przestrzeni adresowej. Jeżeli na wejście BDIR, BC1, BC2, A9/, A8 podana zostanie odpowiednia kombinacja stanów logicznych, to informacja wymuszona na szynie w cyklu adresowym zostanie wprowadzona do układu. Jako ważny adres rejestru zostanie ona potraktowana tylko wtedy, jeżeli cztery bardziej znaczące bity będą zerami [1]. Cztery mniej znaczące bity odpowiadają numerowi rejestru.



Rys. 3. Architektura układu AY-3-8910

W przypadku mikroprocesorów z rozdzieloną szyną danych i adresową (np. Z80, 8080), niezbędne jest wykonanie dwóch operacji. W pierwszej zapisywana jest dana określająca rejestr, natomiast druga jest już właściwym zapisem lub odczytem żądanego rejestru.



Rys. 4. Różne typy obwiedni generowane w zależności od sekwencji wpisanej do rejestru R15

[1] Przy zamawianiu większej liczby układów u producenta można wybrać dowolną kombinację binarną

Częstotliwość generatora tonu podstawowego programowana jest wpisem 12-bitowej wartości, przy czym cztery bardziej znaczące bity wpisywane są do jednego z rejestrów R1, R3 lub R5, a pozostałych osiem bitów — do jednego z rejestrów R0, R2 lub R4, odpowiednio dla kanałów A, B lub C.

Programowanie częstotliwości przebiegu bazowego generatora szumów odbywa się przez zapisanie pięciu mniej znaczących bitów rejestru R6.

Mikser programuje się wykorzystując rejestr R7. Wpisanie zera na bicie B0, B1 lub B2 otwiera przejście dla tonu podstawowego — odpowiednio dla kanału A, B lub C. Wpisanie zera na bicie B3, B4 lub B5 analogiczne otwiera przejście szumu.

Wpisanie zera na bicie B6 rejestru R6 definiuje port IOA jako wejście i wreszcie zero wpisane na najbardziej znaczącym bicie (B7) definiuje port IOB jako wejście.

Określony poziom wyjściowy dla poszczególnych kanałów programuje się wpisując odpowiednią wartość na czterech mniej znaczących bitach rejestrów R8, R9 lub R10.

Okres powtarzania obwiedni programowany jest wpisem 16-bitowej wartości do rejestrów R11 i R12 (bardziej znaczący bajt do R12). Kształt obwiedni definiowany jest przez zapis odpowiedniej kombinacji na czterech mniej znaczących bitach rejestru R13 (rys. 4). Bit B0 odpowiada funkcji HOLD, bit B1 funkcji ALTERNATE, bit B2 funkcji ATTACK i bit B3 funkcji CONTINUE.

Oprac. AJP

W poprzedniej części opisaliśmy komputer IBM-PC i przykładowe procedury w języku maszynowym. W tej — bibliotekę procedur w języku C.

Grafika IBM-PC (2)

Zasady korzystania z przerwania graficznego INT 10H można poznać analizując przedstawione na wydruku 3 procedury zapisane w języku C. Wszystkie procedury wywołują podprogram **video (ax, bx, cx, dx)**, dla którego parametrami są wartości odpowiednich rejestrów. Podprogram ustawia wszystkie potrzebne rejestry i wywołuje przerwanie INT 10H (wydruk 1).

```
public video
video:
        push    bp
        mov     bp, sp
        push    di
        mov     ax, (bp+4)      ; ax <— 1 parametr
        mov     bx, (bp+6)      ; bx <— 2 parametr
        mov     cx, (bp+8)      ; cx <— 3 parametr
        mov     dx, (bp+10)     ; dx <— 4 parametr
        int     10h             ; przerwanie graficzne
        pop     di
        pop     bp
        ret
```

Wydruk 1. Procedura video (ax, bx, cx, dx)

UWAGA: Podprogram wideo z przyrostkiem **-ax, -bx, -cx, -dx** jest podobny do video, z tym że wartością powrotną procedury jest wartość odpowiedniego rejestru.

Procedury wykorzystują również inny podprogram napisany w języku wewnętrznym, który mniej znaczące bajty dwóch argumentów łączy w jedną 2-bajtową liczbę całkowitą (wydruk 2).

```
        public con
con:
        push    bp
        mov     bp, sp
        mov     ah, (bp+4)      ; ah <—1.parametr
        mov     al, (bp+6)      ; al<—2.parametr
        pop     bp
        ret
```

Wydruk 2. Procedura con (rh, rl)

```
/*                      BIBLIOTEKA PROCEDUR
                          W  JEZYKU "C"

                    OBSLUGA PRZERWANIA:  INT 10H
                    ============================                    */

/* WYBOR TRYBU PRACY                                    (A/N,GR)
        mode = 0 -  40 x 25  monochromatyczny A/N
             = 1 -  40 x 25  kolorowy A/N
             = 2 -  80 x 25  monochromatyczny A/N
             = 3 -  80 x 25  kolorowy A/N
             = 4 - 320 x 200 kolorowy GR
             = 5 - 320 x 200 monochromatyczny GR
             = 6 - 640 x 200 monochromatyczny GR                    */
select_mode (mode)
  int mode;
{ video (con (0,mode),0,0,0); }

/* WYBOR TYPU KURSORA                                   (A/N)
        upp = 0,...,7 - linia poczatkowa
        low = 0,...,7 - linia koncowa                           */
cursor_type (upp,low)
  int upp,low;
{ video (con (1,0),0,con (upp,low),0); }

/* POZYCJONOWANIE KURSORA                               (A/N)
        page = 0,...,7 - numer strony (zobacz : select_page)
        row  = 0,...,24 - numer wiersza
        col  = 0,...,79 - numer kolumny                         */
position (page,row,col)
  int page,row,col;
{ video (con (2,0),con (page,0),0,con (row,col)); }

/* ODCZYT POZYCJI KURSORA                               (A/N)
        page = 0,...,7 - numer strony (zobacz : select_page)
WYJSCIE ah = 0,...,24 - numer wiersza
        al = 0,...,79 - numer kolumny                           */
int read_position (page)
  int page;
{ return (video_dx (con (3,0),con (page,0),0,0)); }

/* WYBOR AKTYWNEJ STRONY OBRAZU                         (A/N,GR)
        page = 0,...,7 - numer strony dla A/N 40x25
             = 0,...,3 - numer strony dla A/N 80x25
             = 0       - numer strony dla GR                    */
select_page (page)
  int page;
{ video (con (5,0),con (page,0),0,0); }

/* OBROT OBSZARU W GORE                                 (A/N)
        lines = 0,...,24 - ilosc linii obrotu
        attr  = 0,...,255 - atrybut koloru dopisywanych linii
        row_u = 0,...,24 - wiersz lewego gornego rogu obszaru
        col_l = 0,...,79 - kolumna lewego gornego rogu obszaru
        row_d = 0,...,24 - wiersz prawego dolnego rogu obszaru
        col_r = 0,...,79 - kolumna prawego dolnego rogu obszaru */
roll_up (lines,attr,row_u,col_l,row_d,col_r)
  int lines,attr,row_u,col_l,row_d,col_r;
{ video (con (6,lines),con (attr,0),con (row_u,col_l),con (row_d,col_r))
; }

/* OBROT OBSZARU W DOL                                  (A/N)
        lines = 0,...,24 - ilosc linii obrotu
        attr  = 0,...,255 - atrybut koloru dopisywanych linii
        row_u = 0,...,24 - wiersz lewego gornego rogu obszaru
        col_l = 0,...,79 - kolumna lewego gornego rogu obszaru
        row_d = 0,...,24 - wiersz prawego dolnego rogu obszaru
        col_r = 0,...,79 - kolumna prawego dolnego rogu obszaru */
roll_dn (lines,attr,row_u,col_l,row_d,col_r)
  int lines,attr,row_u,col_l,row_d,col_r;
{ video (con (7,lines),con (attr,0),con (row_u,col_l),con (row_d,col_r))
; }

/* ODCZYT ZNAKU I ATRYBUTU                              (A/N)
        page = 0,...,7 - numer strony (zobacz : select_page)
WYJSCIE ah = 0,...,255 - atrybut znaku
        al = 0,...,255 - znak                                   */
read_ac (page)
  int page;
{ return (video_ax (con (8,0),con (page,0),0,0)); }

/* ZAPIS ZNAKU I ATRYBUTU                               (A/N)
        chr  = 0,...,255 - znak
        page = 0,...,7   - numer strony (zobacz : select_page)
        attr = 0,...,255 - atrybut znaku
        time = 0,...     - ilosc powtorzen                      */
write_ac (chr,page,attr,time)
  int chr,page,attr,time;
{ video (con (9,chr),con (page,attr),time,0); }
```

```
/* ZAPIS ZNAKU                                    (A/N)
        chr  = 0,...,255 - znak
        page = 0,...,7   - numer strony (zobacz : select_page)
        time = 0,...     - ilosc powtorzen                    */
write (chr,page,time)
  int chr,page,time;
{ video (con (10,chr),con (page,0),time,0); }

/* WYBOR PALETY                                   (GR)
        nr = 0,1 - numer palety                              */
pallette (nr)
  int nr;
{ video (con (11,0),con (1,nr),0,0); }

/* WYBOR KOLORU TLA                               (GR)
        color = 0,...,15 - kolor tla                         */
background (color)
  int color;
{ video (con (11,0),con (0,color),0,0); }

/* RYSOWANIE PUNKTU                               (GR)
        color = 0,...,3 - kolor punktu

        u = 0,...,319 - 1.wspolrzedna
        v = 0,...,199 - 2.wsplorzedna                        */
w_dot (color,u,v)
  int color,u,v;
{ video (con (12,color),0,u,v); }

/* RYSOWANIE PUNKTU Z UWZGLEDNIENIEM POPRZEDNIEGO KOLORU (EXCLUSIVE OR)
                                                  (GR)
        color = 0,...,3 - kolor punktu
        u = 0,...,319 - 1.wspolrzedna
        v = 0,...,199 - 2.wspolrzedna                        */
x_dot (color,u,v)
  int color,u,v;
{ video (con (12,color ! 0x80),0,u,v); }

/* ODCZYT KOLORU PUNKTU                           (GR)
        u = 0,...,319 - 1.wspolrzedna
        v = 0,...,199 - 2.wspolrzedna
WYJSCIE a1 = 0,...,3 - kolor                                 */
r_dot (u,v)
  int u,v;
{ return (video_ax (con (13,0),0,u,v)); }
```

Wydruk 3.

LITERATURA

[1] Kernigham B. W., Ritchie D. M.: The C Programming Language. Prentice-Hall, London, 1983

[2] Lafore R. W.: Assembly Language Primer for the IBM PC & XT. The Wait Book, New York, 1984

[3] Morgan Ch. L.: Bluebook of Assembly Routines for the IBM PC & XT. The Wait Book, New York, 1984

[4] Norton P.: Inside the IBM PC Access to Advanced Feature and Programming. Prentice-Hall, Maryland, 1983

[5] Technical Reference. IBM Library, 1983.

ANTONI URBAN

Computer Studio Kajkowscy
Gdynia

Urząd Pocztowy pewnie jeszcze nie prędko dopuści posiadaczy prywatnych komputerów do linii telefonicznych. Poza „względami na interes państwa", przemawia za tym fatalny stan sieci telefonicznej, która po dołączeniu żarłocznych modemów zapewne zakorkowałaby się na amen. Rozważanie standardowego połączenia komputera z modemem wydaje się być zatem pozbawione głębszego sensu. Jednak ten sam standard może być użytecznie wykorzystany do wielu innych także celów, w naszym zaścianku. Przykładem niech będzie tu możliwość przekazywania plików danych między komputerami, które nie mają ze sobą nic wspólnego; poza — łączem RS 232C.

Standard RS 232C

Istnieje kilka standardów określających rozwiązanie interfejsów szeregowych. Jednym z najbardziej popularnych jest RS 232C i bardzo do niego zbliżony V24.

Standard RS 232C określony został przez EIA (Electronic Industries Association) jako opis interfejsu umożliwiającego współpracę terminala nadawczo-odbiorczego z modemem.

Terminal to urządzenie pozwalające użytkownikowi na wprowadzenie informacji (komendy, dane itp.) i odbieranie informacji (np. na ekranie monitora lub wydrukowanej na papierze). Modem (modulator — demodulator) to urządzenie pośredniczące, umożliwiające dołączenie się do sieci telekomunikacyjnej.

Dla określenia źródeł generowanych sygnałów w przypadku współpracy terminal-modem przyjęto następujące oznaczenia:
DCE (ang. Data Communication Equipment) — modem
DTE (ang. Data Terminal Equipment) — terminal nadawczo-odbiorczy.

Z chwilą upowszechnienia mini- i mikrokomputerów, transmisja szeregowa okazała się wygodną metodą realizacji współpracy różnych urządzeń. Chociaż istnieje wiele różnorodnych rozwiązań, nadal często wykorzystuje się standard RS 232C (V24), co ułatwia łączenie sprzętu dostarczonego przez różnych producentów.

Eksplozja komputerów osobistych spowodowała dalszy rozwój sieci telekomunikacyjnych. Bardzo często do komputera domowego dokupywany jest modem i użytkownik uzyskuje — przez przyłączenie do linii telefonicznej — dostęp do plików informacji obsługiwanych przez duże systemy. Upowszechnieniu uległy też sieci lokalne łączące mikrokomputery znajdujące się na terenie jednej instytucji. Dołączenie do adaptera sieci lokalnej często wykonywane bywa przez interfejs szeregowy w standardzie RS 232C. Stosunkowo rzadkie (choć ciągle jeszcze popularne w sprzęcie profesjonalnym krajowej produkcji) są rozwiązania, w których mikrokomputer łączy się przez RS 232C z terminalem nadawczo-odbiorczym.

Przykładem „domowego" wykorzystania standardu RS 232C może być połączenie dwóch komputerów w celu wymiany danych lub nawet oprogramowania (gdy przechowywane jest ono na różnych nośnikach pamięci masowej).

W dalszej części omówiony zostanie standard RS 232C tak, jak został zdefiniowany, czyli do komunikacji terminal-modem. Warto jednak podkreślić, że mikrokomputer może być traktowany jako tzw. inteligentny terminal (czyli DTE) i źródła sygnałów pozostają bez zmian w stosunku do określonych w standardzie.

W standardzie określono cztery typy linii połączeniowych:

— linie transmisji danych
— linie sterująco-kontrolne
— linie synchronizacji czasowej
— linie masy sygnałowej i ekranu.

Linie sterowane są sygnałami napięciowymi zmieniającymi się w zakresie od —12 do +12 V. Na linii danych stan określany jest jako logiczne „1", gdy sterowana jest ona napięciem ujemnym (ang. Marking State). Gdy linia danych sterowana jest napięciem dodatnim określane jest to jako stan logiczny „0" (ang. Spacing State).

Dla linii kontrolno-sterujących stan aktywny (ang. On) oznacza sterowanie napięciem dodatnim.

Stosowanie napięć sterujących z zakresu od —12 do +12 jest dość niewygodne we współczesnych systemach mikrokomputerowych, gdzie z reguły wykorzystywane bywa tylko jedno napięcie zasilające: +5 V. Dlatego też czasami producenci popularnego sprzętu (np. firma COMMODORE w mikrokomputerze C64) nie realizują standardu „do końca", lecz zostawiają użytkownika z sygnałami o poziomach TTL. Jeżeli łączy się dwa komputery leżące obok siebie, to faktycznie konwertery napięć można pominąć. Jeśli jednak chcemy przyłączyć mikrokomputer C64 do modemu, trzeba dokupić dodatkową płytkę, która „dokańcza" realizację standardu.

W tabeli zestawiono sygnały przewidziane w standardzie RS 232C. Zaznaczono również sygnały spoza standardu, wprowadzone w sprzęcie krajowej produkcji.

Sygnały interfejsu RC 232C

EIA RS 232C	CCITT V24	Wyprowadzenia	Oznaczenie mnemoniczne	Źródło	Nazwa linii interfejsu
AA	101	2	—	—	ekran
AB	102	7	GND	—	masa sygnałowa
AB	103	2	TxD	DTE	dane nadawane (ang. Transmitted Data)
BB	104	3	RxD	DCE	dane odbierane (ang. Recived Data)
CA	105	4	RTS	DTE	żądanie nadawania (ang. Request to Send)
CB	106	5	CTS	DCE	gotowość nadawania (ang. Clear to Send)
CC	107	6	DSR	DCE	gotowość modemu (ang. Data Set Ready)
CD	108.2	20	DTR	DTE	żądanie dołączenia modemu do linii (ang. Data Terminal Ready)
CE	125	22	—	DCE	dzwonek (ang. Ring Indicator)
CF	109	8	DCD	DCE	nośna odbierana (ang. Data Carrier Detect)
CG	110	21	SQD	DCE	poziom sygnału odbieranego (ang. Signal Quality Detector)
—	—	21	—	DTE*)	zgłoszenie zapełnienia bufora DZM 180 KSR
CH	111	23	—	DTE	wybór prędkości transmisji (ang. Data Signal Rate Selector)
CI	112	23	—	DCE	
DA	113	24	—	DTE	zegar transmisji nadajnika (ang. Transmiter Signal Element Timing)
DB	114	15	—	DCE	zegar transmisji nadajnika (ang. Transmiter Signal Element Timing)
DD	115	17	—	DCE	zegar transmisji odbiornika (ang. Reciver Signal Element Timing)
SBA	118	15	—	DTE	dane nadawane — kanał 2 (ang. Secondary Transmitted Data)
—	—	14		DTE*)	zgłoszenie zajętości monitora MERA 7952
SBB	119	14	—	DCE	dane odbierane — kanał 2 (ang. Secondary Recived Data)
SCA	120	19	—	DTE	żądanie nadawania — kanał 2 (ang. Secondary Request to Send)
SCB	121	13	—	DCE	gotowość nadawania — kanał 2 (ang. Secondary Clear to Send)
SCF	122	12	—	DCE	nośna odbierana — kanał 2 (ang. Secondary Data Carrier Detect)

*) Sygnały niestandardowe

Standard dopuszcza możliwość zredukowania niektórych linii kontrolnych i linii synchronizacji czasowej. W minimalnych zestawach wykorzystywane są tylko linie transmisji danych **TxD** i **RxD** (i oczywiście linia masy). W wersjach uproszczonych najczęściej wykorzystywane są linie **RTS, CTS, DSR** i **DTR.**

W zasadzie, aby mogła odbywać się transmisja po liniach danych, wszystkie wymienione linie sterujące powinny być w stanie aktywnym. Sesja współpracy zaczyna się od wymuszenia przez terminal (komputer) stanu aktywnego na linii **DTR.** Urządzenie komunikacyjne (modem) powinno potwierdzić gotowość do współpracy sygnałem aktywności na linii DSR [1]).

Komunikacja inicjowana jest przez terminal podaniem stanu aktywnego na linię RTS. Jeżeli modem przyłączony jest do linii, to odpowiada sygnałem CTS. Od tego momentu terminal może wysyłać dane po linii TxD. Dane wprowadzane do terminala (komputera) po linii RxD są ważne, gdy aktywny jest sygnał DCD i SQD.

Gdy łączymy mikrokomputer z terminalem, powstaje nieco zamieszania. Przyjmuje się wtedy, że w układzie terminal (DTE) — modem (DCE) komputer przyłącza się w miejsce modemu. Ponieważ łącze w komputerze przystosowane jest do współpracy z modemem, niezbędne jest zastosowanie specjalnego kabla przejściowego. W kablu tym należy przede wszystkim zamienić (na jednej z łączówek) linie TxD i RxD. Należy również zamienić miejscami linie DSR i DTR. Natomiast linie RTS i CTS najlepiej zapętlić, osobno dla każdego z urządzeń. Ponadto sygnały z linii RTS należy podłączyć do wejść DCD i SQD współpracującego urządzenia. Dodatkowy problem może wyniknąć, gdy któreś z urządzeń nie generuje sygnału DTR. Należy wtedy w drugim urządzeniu zapętlić DTR i DSR.

Sytuacja — gdy łączymy ze sobą dwa komputery — jest bardzo zbliżona do sytuacji łączenia komputera i terminala. Podobnie jak poprzednio, jeden z komputerów musi „wczuć się" w rolę modemu, podczas gdy drugi spełnia funkcję inteligentnego terminala. Uwagi dotyczące przełączania lub zapętlania niektórych linii można odnieść i do tej sytuacji.

Standard nie określa postaci przesyłanych danych. Nie narzuca też, czy transmisja ma być synchroniczna czy asynchroniczna. Najczęściej stosowana jest jednak transmisja asynchroniczna. Przesyłana dana rozpoczyna się od tzw. bitu startowego. Inicjuje on próbkowanie przez odbiornik stanu linii przesyłowej. Jeśli transmisji nie towarzyszy opisany wcześniej protokół wymiany sygnałów sterujących, może się zdarzyć, że transmisja rozpocznie się przed włączeniem odbiornika. Oprócz utraty części informacji, mogą wtedy nastąpić dalsze przekłamania — gdy któryś z bitów informacyjnych zostanie potraktowany jako bit startowy. Jeżeli więc — na przykład — realizujemy przesyłanie między dwoma komputerami, należy zadbać, aby program obsługujący odbiornik rozpoczął działanie przed programem obsługującym nadajnik.

Po bicie startu przesyłanych jest siedem lub osiem bitów informacyjnych, przy czym jako pierwszy wysyłany jest najmniej znaczący bit. Po bitach informacyjnych przesyłany jest bit parzystości, za którym następują bity stopu, rozdzielające informacje. Liczba bitów informacyjnych, rodzaj kontroli (parzystość, nieparzystość lub brak bitu kontrolnego) oraz liczba bitów stopu (1, $1^1/_2$ lub 2) — to parametry transmisji, które należy „uzgodnić" między nadajnikiem i odbiornikiem. Uzgodnić należy też szybkość transmisji. Mierzona jest ona w bodach (bitach na sekundę) i przyjmuje jedną z następujących wartości: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200.

Im dłuższe jest połączenie między odbiornikiem a nadajnikiem, tym mniejszą szybkość transmisji należy stosować. Wskazane też jest stosowanie mniejszych szybkości, gdy nadajniki są realizowane na elementach dyskretnych.

AJP

*) Nazwa DSR (ang. Data Set Ready) jest nieco myląca, gdyż modem wcale nie musi w momencie zgłoszenia dysponować plikiem danych, które przeznaczone są do wyprowadzenia na terminal

dokończenie ze str. 12

wane w trakcie uruchamiania systemów, zasady opracowywania testów, metody uruchamiania sprzętu i jego części składowych.

Osobne zajęcia w grupie przedmiotów dotyczących sprzętu powinny dotyczyć mikroprocesorów segmentowych (na przykładzie serii Am 2900) oraz zastosowań tych mikroprocesorów i techniki mikroprogramowania do konstrukcji specjalizowanych procesorów.

Drugim podstawowym zadaniem jest sprecyzowanie zakresu kształcenia w dziedzinie oprogramowania systemów mikrokomputerowych. Wspomnieliśmy już o koncepcji nauczania języków asemblerowych mikroprocesorów. Konieczne jest ponadto nauczanie studentów języków ogólnego przeznaczenia szeroko stosowanych w mikroinformatyce, takich jak BASIC czy PASCAL, a także języków specyficznych dla systemów mikrokomputerowych, jak PL/M czy FORTH. Omawiając języki wysokiego poziomu, należy przedstawić związki programu napisanego w takim języku ze sprzętem systemów mikrokomputerowych, w szczególności — sposoby programowania układów sprzęgających, łączenia programów napisanych w różnych językach (np. wstawki w asemblerze) itp. Niezbędny jest również wykład obejmujący: zasady organizacji systemów oprogramowania użytkowego, techniki programowania, metodologię uruchamiania oprogramowania (w tym — oprogramowania uzależnionego czasowo) oraz metodologię testowania systemów mikrokomputerowych wraz z oprogramowaniem użytkowym.

Dwa dalsze tematy związane z oprogramowaniem, które koniecznie powinny znaleźć się w programie kształcenia inżynierów mikroinformatyków, to systemy operacyjne oraz systemy programowania (translatory, programy łączące, ładujące, redagujące itd.). Zakres kształcenia zasadniczo nie odbiega tu od edukacji informatyków innych specjalności i obejmuje zarówno teorię (zasady konstrukcji systemów), jak i praktykę (sposoby adaptacji, eksploatacji, oceny efektywności itd.). W związku z silnie zaznaczającą się tendencją rozwoju systemów wielomikroprocesorowych, szczególnie ważne jest jednak zwrócenie uwagi na aspekt systemów współbieżnych.

Wspomnijmy teraz o grupie przedmiotów dotyczących zastosowań systemów mikrokomputerowych. Ich celem jest przedstawienie głównych klas zastosowań. Naturalne jest przy tym skupienie się na zastosowaniach najbliższych danemu kierunkowi studiów; zastosowania te mogą stanowić przedmiot pełnych projektów, np.: sterowanie w czasie rzeczywistym wybranym procesem przemysłowym, konstrukcja mikrokomputerowej sieci lokalnej czy systemu biurowego. Niemniej zastosowania systemów mikrokomputerowych trzeba przedstawiać najszerzej. Omawianie każdej klasy zastosowań powinno obejmować: klasyfikację problemów w tej klasie, wymagania sprzętowe dla systemu mikrokomputerowego, wymagania dla systemów operacyjnych oraz oprogramowania podstawowego, wymagania dla oprogramowania specjalistycznego oraz analizę istniejących systemów mikrokomputerowych wykorzystywnych w danej klasie zastosowań.

KONCEPCJA ĆWICZEŃ LABORATORYJNYCH I PROJEKTOWYCH

Prowadzenie odpowiednich ćwiczeń laboratoryjnych i projektowych jest podstawowym warunkiem poprawnego kształcenia inżynierów mikroinformatyków. Wyłącznie werbalne nauczanie jest tu całkowicie niewystarczające i nie może prowadzić do pożądanych efektów dydaktycznych. Proponujemy zorganizowanie ćwiczeń laboratoryjnych i projektowych, w ramach których problematyka systemów mikrokomputerowych podzielona byłaby na cztery grupy:

- modelowanie i ocena działania
- programowanie
- projektowanie sprzętu
- zastosowania.

Ćwiczenia laboratoryjne z modelowania i oceny działania systemów mikrokomputerowych powinny dotyczyć przede wszystkim metod i technik pomiarowych wybranych fragmentów sprzętu (np. układów transmisji), oprogramowania (np. funkcji systemów operacyjnych) oraz systemów mikrokomputerowych wraz z oprogramowaniem użytkowym jako całości. Ćwiczenia projektowe z tego przedmiotu powinny koncentrować się na konstrukcji modeli systemów mikrokomputerowych i ich fragmentów oraz obciążenia, kalibracji i weryfikacji tych modeli.

Ćwiczenia laboratoryjne z programowania powinny obejmować wszystkie języki wymienione w rozdziale poprzednim — zarówno asemblerowe, jak i wyższego poziomu. Ćwiczenia powinny być opracowane w ten sposób, aby ilustrowały podstawowe techniki programowania. Nacisk należy położyć na metodologię uruchamiania oprogramowania (dzieląc programy na napisane w językach asemblerowych oraz w językach wyższego poziomu), a szczególnie — na posługiwanie się narzędziami uruchamiania programów.

Ćwiczenia projektowe z programowania powinny dotyczyć projektowania fragmentów systemów operacyjnych oraz systemów programowania. Powinny one obejmować podstawowe algorytmy tych systemów.

Zajęcia związane ze sprzętem powinny obejmować: projektowanie logiczne, projektowanie techniczne oraz metodologię uruchamiania. Szczególnie ważna jest właśnie metodologia uruchamiania sprzętu — ze względu na jej znaczenie nie tylko na etapie konstruowania systemów mikrokomputerowych, lecz również na etapie ich eksploatacji (konserwacji, naprawiania, adaptacji).

Zajęcia dotyczące zastosowań powinny dotyczyć projektowania specjalizowanych systemów mikrokomputerowych dla określonych celów oraz ocenę ich działania. W każdym ćwiczeniu projektowym należy uwzględnić następujące elementy:

— ocenę rozpatrywanego zastosowania, z punktu widzenia wymagań stawianych systemowi mikrokomputerowemu
— optymalny podział funkcji systemu między sprzęt i oprogramowanie
— projekt architektury sprzętu
— projekt oprogramowania
— uruchomienie systemu
— połączenie systemu z rzeczywistym procesem lub jego modelem
— ocenę działania systemu i jego strojenie.

* * *

Natychmiastowe rozpoczęcie kształcenia gruntownie przygotowanej kadry inżynierów mikroinformatyków ma decydujące znaczenie dla przyszłości mikroinformatyki. Kształcenia na tym poziomie nie mogą zastąpić rozmaite kursy ani poszczególne przedmioty mikroinformatyczne w ramach innego kierunku studiów. Projektowanie systemów mikrokomputerowych ma bowiem swoją istotną specyfikę, wymagającą wszechstronnego i całościowego naświetlenia. O twórczym rozwoju zastosowań tych systemów, w tym o włączeniu się z czymś atrakcyjnym do konkurencji choćby na rynku RWPG, mogą zadecydować jedynie ludzie dogłębnie rozumiejący tę specyfikę.

Podkreślamy istotną rolę problematyki modelowania i oceny działania systemów komputerowych. Problematyka ta, reprezentowana słabo lub wcale w dotychczasowych programach nauczania, musi uzyskać należną rangę [2].

Realizacja naszkicowanej tutaj koncepcji nauczania mikroinformatyki wymaga wyposażenia szkół wyższych w sprzęt mikrokomputerowy i odpowiednie urządzenia towarzyszące. Musi to być sprzęt nowoczesny o zdecydowanie wyższym standardzie niż spotykany aktualnie w naszym przemyśle, a więc musi objąć między innymi mikroprocesory 16-bitowe i systemy wielomikroprocesorowe [1], a w bliskiej perspektywie — mikroprocesory i systemy 32-bitowe.

LITERATURA

[1] Cellary W.: Systemy wielomikroprocesorowe. Materiały Jesiennej Szkoły PTI (w przygotowaniu)
[2] Węglarz J.: Ocena działania systemów komputerowych — przedmiot, metody, kierunki rozwoju. Ibidem.

CSK—Computer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

Komputer osobisty może być przydatny niemal na każdym stanowisku pracy. Wymaga jednak odpowiedniego oprogramowania użytkowego. W ramach tego oprogramowania oferujemy zainteresowanym dostawę uniwersalnych pakietów programowych:

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

To doskonałe narzędzia pracy dla każdego. Aby z nich korzystać, nie trzeba być informatykiem! Zupełnie samodzielnie można tworzyć złożone systemy zarządzania przedsiębiorstwem, k a ż d y m p r z e d s i ę b i o r s t w e m; nawet najbardziej specyficzne uwarunkowania nie są przeszkodą.
To jednak jeszcze nie wszystko... Kiedy dotychczasowe problemy łatwo i szybko zostały rozwiązane — pojawiają się zupełnie nowe. Można wtedy bez kłopotów samemu udoskonalić dotychczasowy system!

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

składają się w zakładowe systemy płacowe, osobowe, finansowo-księgowe lub magazynowe. Korzystając z nich, z łatwością można prowadzić planowanie, kalkulacje i sprawozdawczość. Można też sporządzać kosztorysy i oferty, a nawet prowadzić „automatyczną" korespondencję czy redagować dowolne teksty. Można wreszcie skorzystać z już zgromadzonych zasobów na komputerze ODRA (pod nadzorem systemu GEORGE-3), wykorzystując komputer osobisty jako inteligentny terminal — stację lub emulator TTY.

Nowość:

Oferujemy system operacyjny kompatybilny z CP/M 2.2. dla mikrokomputerów ROBOTRON 5120/5130 oraz systemy finansowo-księgowe FK dla dowolnych mikrokomputerów

---

Szczegółowych informacji udziela:

CSK—Computer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

EO/129/K/85

57 MTP

Pogoń za mikro

Czy po Międzynarodowych Targach Poznańskich '85, których mottem mogło by być „komputer w każdym kącie", można optymistycznie stwierdzić, że fenomen mikrokomputera został wreszcie dostrzeżony? A może raczej stwierdzić, że mikrokomputerowa moda stała się jeszcze jednym elementem zasłony dymnej kryjącej szamotaninę ambicji i niemocy? Mimo wszelkich jednak wątpliwości trzeba przyznać, że zjawisko pt. mikrokomputer zostało niewątpliwie dostrzeżone i to nawet na odpowiednio wysokim szczeblu, chociaż sposób postrzegania, a już tym bardziej dotychczasowe efekty, mogą rodzić pewne wątpliwości.

Na MPT '85 komputery atakowały niemal z każdego stoiska. Gwizdy i pomruki komputerowych gier ściągały tłumy nastolatków i... kilkudziesięciolatków. Wiele firm, nie oferujących samych komputerów, próbowało wykorzystywać tę zdobycz techniki do prezentacji oferty własnych wyrobów. Jednak komputer osobisty, jak wskazuje sama nazwa, niezbyt dobrze nadaje się do obsłużenia kilkudziesięciu tłoczących się wokół osób, z których większość oczekuje, że „drętwe" napisy zaraz się skończą i wreszcie rozpocznie się pogoń za kosmitami. Pod „społecznym" naporem znakomita część wystawionych komputerów faktycznie służyła do strzelania i gonitw. Mikrokomputery stały się narzędziem prestiżu wystawcy i przyciągania uwagi publiczności. Albo też... odwracania uwagi publiczności od realiów. Przykładowo POLKOLOR ustawiając urzekającą szklaną ścianę z monitorów TV, na których rozgrywały się komputerowe przygody, zapewne chciał zasugerować, że w PRL nie będzie już problemu z monitorem do komputera. Może użytkownicy ZX SPECTRUM poczują się w jakimś zakresie usatysfakcjonowani — nie przypadkiem jednak COMPAN 8 demonstrowany był w połączeniu z monitorem produkcji zachodniej. I choć niektóre firmy polonijne proponują stosować w zestawie z komputerem á la IBM PC monitor monochromatyczny NEPTUN, to żaden z wystawców w obliczu konkurencji nie zdecydował się zademonstrować takiego zestawu.

Jeszcze rok temu nie sposób było dopatrzeć się na Targach jakiegokolwiek sprzętu 16-bitowego. W tym roku furorę robił IBM-PC. I to nie tylko ten oryginalny. Komputery z nim kompatybilne oferowały za złotówki firmy: COMPUTEX, IMPOL, ITM, KOMPLEX EFC, POL-ROL. Natomiast za dewizy można nabywać takie komputery od firm: BATAX, ENSCH, OLIVETTI, SCAN-PROD, WANG i MERA STER. Wszystkie te firmy z naciskiem twierdzą, że oferują komputery w pełni kompatybilne z IBM PC. Faktycznie obnoszone po Targach dyskietki z programami Flight Simulator, Lotus 1,2,3, a nawet Symphony (programy „wyłapujące" niepełną kompatybilność) zdawały się to potwierdzać.

Przemysł państwowy postanowił nie ustępować pola. Do komputera COMPAN 8 przygotowywana jest karta procesora 8088, która umożliwi pracę pod kontrolą systemu operacyjnego MS-DOS. Przyrzeczenia przyrzeczeniami, ale oferta firm polonijnych ma już konkretny wyraz na rynku, podczas gdy karta 8088 zapowiadana jest dopiero na lipiec 1986. Należy lojalnie przyznać, że w warunkach, jakie obecnie ma przemysł państwowy, dotrzymanie takiego terminu i tak było by rewelacją.

W tym miejscu pora na drobną dygresję. Konstruktor MERITUM — obecnie dyrektor zakładów, które je produkują — skarżył się, że ze wszystkich stron odczuwa jakby bezinteresowną wrogość. Faktem jest, że MERITUM zewsząd poddawane jest bezlitosnej krytyce. Inną racją jest natomiast to, że użytkownika nie obchodzi „kuchnia", czyli np. fakt, że „państwowy" konstruktor m u s i opierać się wyłącznie na układach produkcji RWPG.

Na razie nie zanosi się, aby te nierówne reguły gry (nonsensownie blokujące środki o potężnym w końcu potencjale) uległy zmianie. Dopóki to jeszcze nie nastąpiło, można zaobserwować ciekawe zjawisko. Zapewne jest to rozwiązanie przejściowe, ale przywracające wiarę w inwencję naszego narodu — państwowi potentaci przystępują do spółek z firmami prywatnymi (rzemieślniczymi lub polonijnymi). Tak na przykład powstaje płytka grafiki o rozdzielczości 512×256 punktów do w wersji MERITUM III. Co ciekawsze, współpracującym rzemieślnikiem jest p. Podsiadło, do niedawna jeszcze pracownik... MERY.

Skoro już mowa o MERITUM, to warto wspomnieć o bliskich zamierzeniach jego producenta. Uzupełniona wersja MERITUM, z rozszerzoną pamięcią RAM i dyskami elastycznymi, wyposażona zostanie w znaki polskiego alfabetu. Poszerzony też zostanie zestaw instrukcji języka BASIC, a firmowe oprogramowanie uzupełnione tak, aby przez interfejs RS 232C można było łączyć komputery tworząc pewnego rodzaju sieć... lokalną. Będzie można nią przesyłać programy (BASIC i ASSEMBLER), obrazy oraz komunikaty o rozmiarze do 200 znaków. Wobec powszechnie zgłaszanych kłopotów ze współpracą z magnetofonem, zdecydowano się przystosować (czytaj: przerobić) mikrokomputer do współpracy z magnetofonem MK 232P. Dyr. Korga zadeklarował „odtajnienie" schematu ideowego komputera — kto chce, może go otrzymać. Utworzono też „Biuletyn użytkowników komputerów osobistych MERITUM" — pierwszy jego numer już się ukazał, a następny spodziewany jest w sierpniu i będzie kosztować 200 zł. Rozmiary przyszłorocznej produkcji MERITUM będą zależały od liczby zamówień, jakie zostaną zarejestrowane do września br.

Z propozycją konkurencyjną w stosunku do MERITUM wystąpiło ELWRO pokazując mikrokomputer SOLUM (ELWRO 700). Nie jest to jeszcze pełna oferta — komputer poddawany jest ciągłym udoskonaleniom. W założeniach miał przypominać ZX SPECTRUM, jednak już teraz wiadomo, że nie będą to komputery kompatybilne. Założono również, że komputer ten będzie szybko wdrożony do produkcji — stąd jego obudowa przypomina obecnie bardziej organy elektroniczne niż komputer. Konstrukcja opiera się na Z80, a jako sterownik CRT wykorzystany będzie układ 6895. Producent twierdzi, że cena nie powinna przekroczyć 100 tys. zł. Mimo że komputer nie jest jeszcze gotowy, są już pierwsi klienci. Otóż z inicjatywy red. Nowaka z „Wieczoru Wrocławia" powstała Fundacja Zakładów Dolnego Śląska, zamierzająca wspomóc szkoły zakupem mikrokomputerów.

Trudno na razie ocenić parametry nie ukończonej jeszcze konstrukcji — producent zbiera głosy krytyczne (a jest ich już trochę) i twierdzi, że postara się je uwzględnić. Choć trudno uwierzyć w 100 tys. złotych jako ostateczną cenę, to nie ulega wątpliwości, że ciągle nie jest to oferta konkurencyjna do prywatnego importu ZX SPECTRUM. Może jednak okazać się konkurencyjna do ZX SPECTRUM rozprowadzanego przez firmy polonijne w cenie 280 tys. zł, nawet biorąc pod uwagę to, że dyrektor firmy APINA zadeklarował Ministerstwu Oświaty zniżkę ceny w przypadku uzyskania większego zamówienia. Firma APINA postanowiła też uwolnić klientów od niefortunnego pomysłu Sinclaira z ZX MICRODRIVE oferując pamięci na dyskach elastycznych w cenie 640 tys. zł (z interfejsem).

Na Targach pokazano też sławetnego QL. Chociaż zdaniem fachowców jest to rozwiązanie chybione (szczególnie w polskich warunkach), zainteresowanie było spore i znaleźli się chętni do zapłacenia 995 tys. zł za egzemplarz. Ciekawostką jest oferowany za 45 tys. zł. VIDEO COLOR INTERFACE, który przy minimalnych przeróbkach telewizora (wlutowanie we wskazane miejsca przewodu) pozwala uzyskać kolor nawet na RUBINACH. VIDEO COLOR INTERFACE dostarczany jest firmie APINA przez rzemieślnika i do niego odsyłani są klienci zaskoczeni nieco za wysoką ceną.

Coraz powszechniej zaczyna się mówić o nastającym zmierzchu ZX SPECTRUM. Jednak w ofercie mikrokomputerów 8-bitowych niewiele było nowych propozycji. Firma POL-ROL proponuje komputery AMSTRAD/ /SCHNEIDER i APPLE II. Spółka rzemieślnik — ZETO ZOWAR proponuje komputer do zastosowań biurowych AX-100 oraz jego wersję modułową AX100L. O ile pierwszą wersję charakteryzują dość przeciętne parametry, o tyle AX-100L może być rozszerzony np. o moduł grafiki wysokiej rozdzielczości na układzie NEC 7220, pamięć do 1 MB, interfejsa HP1B i in.

Zaletą tego komputera jest dosyć ciekawe oprogramowanie oferowane przez ZETO. Np. program KOMPRIM pozwala zautomatyzować żmudne zmia-

ny kosztorysów robót budowlano-montażowych po każdej zmianie cen materiałów. Program SAGO to generator systemów użytkowych. Napisany w języku COBOL podobno znakomicie przydaje się w zastosowaniach ekonomicznych do zakładania i aktualizacji zbiorów, tworzenia raportów itp. Ciekawostwą jest fakt, że pierwowzór programu powstał... na zamówienie holenderskiej firmy INFOSYSTEMS. Wadą AX-100 jest małoseryjna produkcja (kilkanaście systemów rocznie).

Najbardziej zaskakującą propozycją wśród komputerów 8-bitowych jest DPC-8 oferowany przez firmę polonijną „Dialog". Otóż jest to komputer... kompatybilny ze standardem MSX! Ale na tym nie koniec: komputer ten można rozbudować o dodatkowe moduły pozwalające uzyskać znacznie lepsze parametry niż przewiduje standard. Podłączenie dodatkowych pakietów odbywa się przez szynę BUSMAT, stosowaną przez firmę IMPOL I w modułowym sterowniku przemysłowym. Tak więc już w chwili startu jest to komputer o bardzo szerokich możliwościach rozbudowy i co za tym idzie — dużej klasie potencjalnych zastosowań.

Wykorzystanie standardu MSX zaowocowało znakomitym interpreterem języka BASIC (32 K ROM), pozwalającym np. na pracę z przerwaniami i stosowanie makrojęzyka graficznego. Wśród wyposażenia dodatkowego oferowana jest m.in. pamięć CMOS (128 KB) z podtrzymaniem bateryjnym — służąca jako... szybki dysk elektroniczny! W wersji podstawowej DPC-8 kosztuje 800 tys. zł, a z dyskami (+ oprogramowanie) — 1,5 mln zł. Już obecnie producent dysponuje obszerną biblioteką programów użytkowych. Wobec propozycji współpracy z użytkownikami należy oczekiwać, że będzie ona szybko rosła.

Wśród firm o ugruntowanej pozycji na rynku, nie wystawiających nowych propozycji 8-bitowych (np. AMEPROD), rysuje się bardzo pozytywna tendencja do sprzedawania systemów realizujących rozwiązania określonego procesu, a nie „gołych" komputerów.

Znamienne jest pojawienie się kilku nowych firm oferujących głównie oprogramowanie. Znana już wcześniej CSK Computer Studio Kajkowscy wzbogaciła ofertę o implementację standardu GKS na IBM PC. Wprawdzie konkurencja twierdzi, że do pełnej implementacji jeszcze sporo brakuje, to powszechnie przyznaje się, że zrobiono solidny kawałek roboty.

Prawdziwą sensacją jest jednak oferta Zakładu Informatyki Przemysłu Okrętowego (ZIPO), który do niedawna produkował oprogramowanie wyłącznie na potrzeby przemysłu okrętowego. Obecnie firma oferuje adaptacje większości popularnych pakietów programowych dla komputerów 16- i 8-bitowych. Ewenementem jest katalog, w którym poszczególnym programom powystawiano... stopnie za jakość dokumentacji i niezawodność. Dla sporej grupy programów opracowano dokumentację w języku polskim. Najistotniejsze jest jednak to, że dla przyszłych użytkowników organizuje się szkolenia. Utworzono również skład konsygnacyjny, w którym za dewizy można kupić płytki do IBM PC, dyski typu WINCHESTER, kolorowe monitory, drukarki itp. Czas realizacji zamówienia... 6 tygodni.

Kończąc ten spisany na gorąco przegląd oferty mikroinformatycznej na MPT '85 trzeba jeszcze wspomnieć o prezentowanych przez firmy państwowe propozycjach 16-bitowych systemów do zastosowań przemysłowych.

W ofercie MERY-ZAP, dotyczącej systemu INTERDIGIT-PROWAY znalazł się pakiet jednostki centralnej 8086, wyposażony w 8 K pamięci RAM, 8 K ROM, interfejs równoległy i szeregowy (V-24), zegar 8253 oraz 16-poziomowy system przerwań. Z kolei ELWRO zaproponowało system mikroprocesorowy ELWRO 800, w którym mogą być równocześnie wykorzystywane pakiety procesorów 8- i 16-bitowych, wyposażone w lokalną pamięć i interfejs szeregowy. Najpoważniejszą wadą jest tu znów termin rozpoczęcia produkcji — 1986 rok. Jak każdemu rozwiązaniu pionierskiemu można by również i tej konstrukcji wytknąć wiele niedociągnięć, np. brak szyny lokalnej.

Na podstawie tego skrótowego przeglądu Czytelnik może odnieść wrażenie, że coś drgnęło w krajowej mikroinformatyce. Nie będę tu jednak przestrzegał przed ułudą targowych ofert, szczególnie przemysłu państwowego. Bez wątpienia poszerza się oferta firm prywatnych, jednak ich niewielkie moce produkcyjne nie mogą w istotny sposób wpłynąć na szybkie zaspokojenie istniejącego popytu.

Dla przeciętnego odbiorcy, który nie miał szczęścia odpowiednio wcześnie zapewnić sobie miejsca w kolejce po komputer, nic nie wskazuje na szybki kres pogoni za mikro...

ANDRZEJ J. PIOTROWSKI

RATFOR na ODRZE 1305

RATFOR [1]) jest mobilnym preprocesorem tekstowym umożliwiającym strukturalne programowanie na bazie języka FORTRAN. RATFOR zawiera takie typowe instrukcje ułatwiające programowanie strukturalne, jak: IF...ELSE, SWITCH (odmiana CASE), WHILLE, REPEAT, REPEAT...UNTIL, FOR, DO i inne. Posiada również ułatwienia w zakresie manipulowania plikami (instrukcja INCLUDE) oraz mianowania obiektów (instrukcja DEFINE) i określania napisów (instrukcja STRING), RATFOR zezwala na swobodny format zapisywania instrukcji.

[1]) RATFOR = Rational FORTRAN

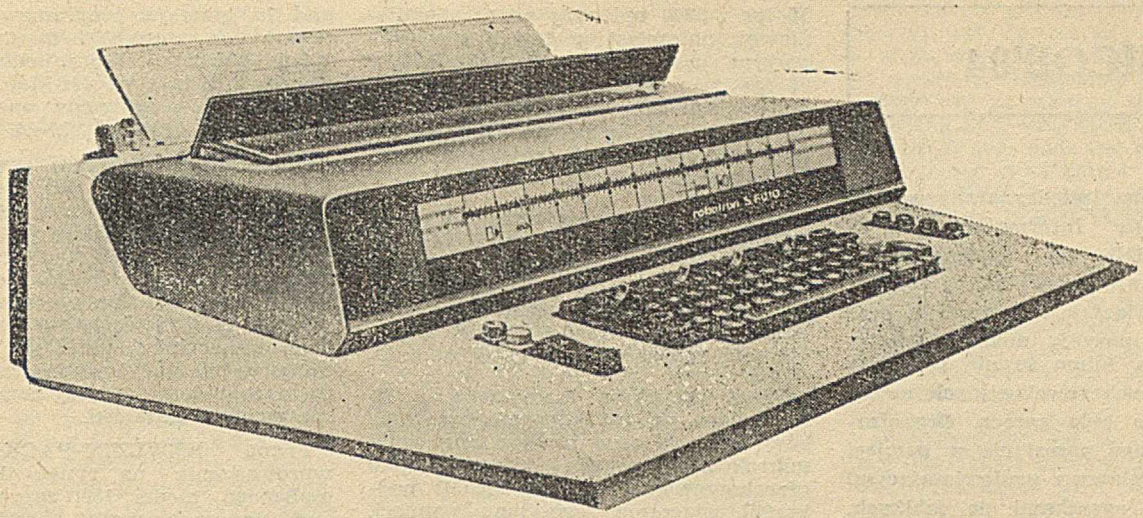
Pisząc program w RATFORZE można używać zupełnie dowolnych wstawek w czystym FORTRANIE, co w połączeniu ze strukturalną nadbudową stwarzaną przez RATFOR daje narzędzie programowania mogące zainteresować programistów używających FORTRANU (chociażby ze względu na jego powszechność), a ceniących zarazem te możliwości programowania strukturalnego, które daje — na przykład — język PASCAL; aczkolwiek RATFOR nie ingeruje w sferę struktur danych.

Wskutek swojej mobilności, RATFOR doczekał się zainstalowania na wielu typach komputerów, także wykonywanych w najnowszych technologiach. Niektóre firmy zalecają go do użytku zaawansowanym programistom — obok takich języków jak C lub LISP.

Mobilność RATFORU sprawia, że napisanie w nim oprogramowania samo w sobie staje się przenośne. Tak jest w szczególności z językiem ICON. Ten pasjonujący poznawczo język — wymagający chyba osobnej prezentacji na łamach INFORMATYKI — został również implementowany (w swojej wcześniejszej wersji 2) w RATFORZE. W RATFORZE zaprogramowany jest również znany pod nazwą „Software Tools" pakiet procedur w ramach inżynierii programowania.

Dzięki uprzejmości Departamentu Informatyki Uniwersytetu stanu Arizona, w Instytucie Informatyki Uniwersytetu Wrocławskiego opracowuje się wersję RATFORA dla maszyny ODRA 1305. Tak zwana „zerowa" wersja pracuje już pod nadzorem systemu operacyjnego GEORGE-3. Obecnie prowadzone są prace nad wersją optymalizującą szybkość działania preprocesora. Przewiduje się też zainstalowanie RATFORA na minikomputerze SM-4.

Osoby zainteresowane pracami nad RATFOREM dla wymienionych komputerów proszone są o kontakt z mgr. Zdzisławem Płoskiem; Instytut Informatyki Uniwersytetu Wrocławskiego, Przemyckiego 20, 51-151 Wrocław.



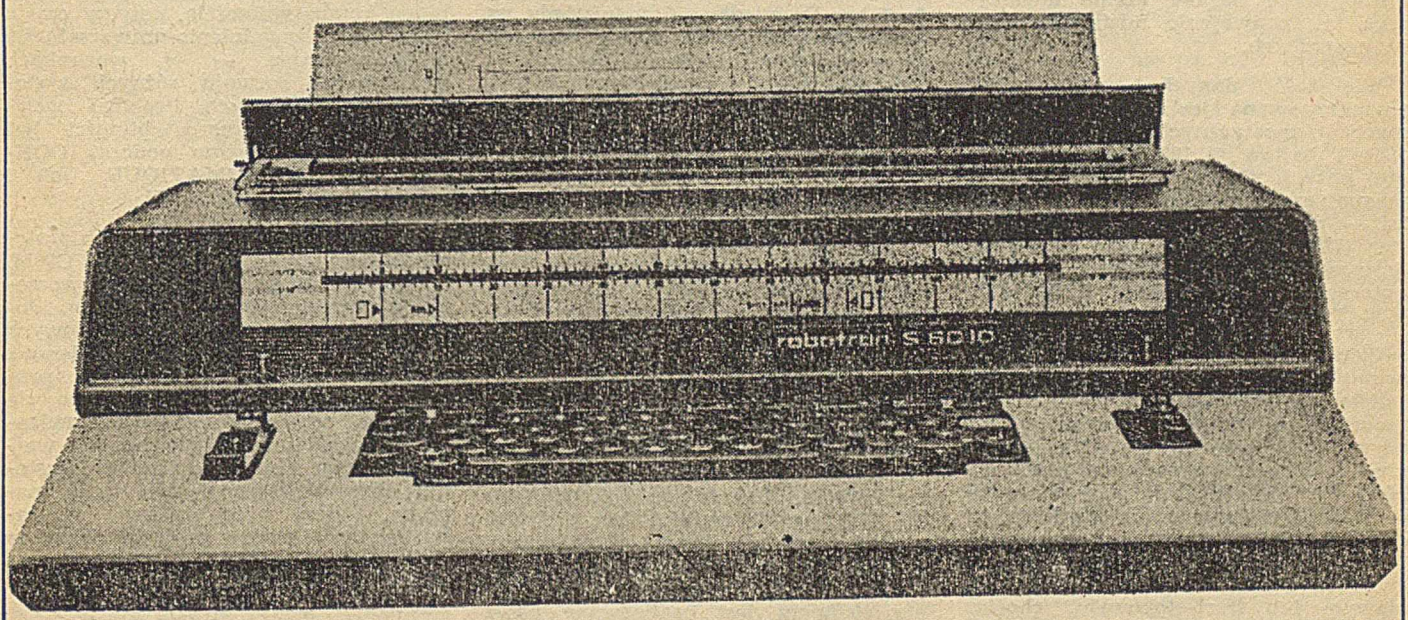
robotron S 6010

wykazuje wiele zalet

Elektroniczne klawisze funkcyjne upraszczają wykonywanie operacji powtarzalnych.
Wysoki komfort obsługi maszyny ułatwia pracę osoby piszącej
Wymienne głowice rozetkowe umożliwiają stosowanie różnych rodzajów czcionek.
robotron S 6010 — to wzrost efektywności maszynopisania

Robotron Export-Import

Państwowe Przedsiębiorstwo Handlu Zagranicznego NRD NRD 1140 Berlin, Allee der Kosmonauten 24



robotron

EO/245/K/85

SICOB'84

Ubiegłoroczna międzynarodowa wystawa sprzętu informatycznego — XXXV SICOB — zakończyła się 28 września, z bilansem blisko pół miliona zwiedzających. Potwierdziła oraz ugruntowała pozycję mikroinformatyki w świecie. Mimo że nie pokazano na niej żadnych rewelacyjnych nowości, atmosfera była gorąca. Organizatorom na długo pozostanie w pamięci krajobraz wystawy z poniewierającymi się wszędzie ogryzkami po jabłkach, które firma APPLE (potentat rynku francuskiego, ma w nim obecnie aż 25-procentowy udział i wyprzedza nawet IBM) rozdawała swoim gościom. Szkoda, że nie były to MACINTOSHE...

SICOB, wraz z poprzedzającą go konferencją CONVENTION INFORMATIQUE podkreślił kilka aktualnych tendencji rozwojowych.

Podczas CONVENTION INFORMATIQUE poruszono m.in. **zagadnienie przechowywania danych.** Szacuje się, że w 1982 roku objętość danych przechowywanych osiągnęła na nośnikach maszynowych w skali światowej wartość 60 000 T bajtów (1 T bajt = 1012 M bajtów), z czego lwią część (97%) stanowiły dane nienumeryczne. Na rok 1988 przewiduje się wzrost do poziomu 90 000 T bajtów, czemu towarzyszyć ma podwojenie udziału danych numerycznych.

Większość danych numerycznych zmagazynowana jest obecnie na taśmach magnetycznych (61% spośród 1800 T bajtów w 1982 roku), na drugiej pozycji plasują się sztywne dyski magnetyczne wymienialne (22%), dalej — dyski elastyczne typu Winchester (7%). W perspektywie 1988 roku dostrzega się już dyskowe pamięci optyczne, które przejmą 24%, czyli 6500 T bajtów danych numerycznych.

Wobec notowanego ostatnio gwałtownego rozwoju zastosowań dysków, spadło zainteresowanie pamięciami taśmowymi. Są one jednak nadal szeroko używane ze względu na dużą pojemność oraz malejące rozmiary przewijaków. Na SICOBIE nie zaprezentowano żadnej nowości z tej dziedziny. W dalszym ciągu dominują dwa rodzaje przewijaków — dla taśmy pół- i ćwierćcalowych, z odczytem strumieniowym lub start-stopowym, chociaż w niektórych urządzeniach korzysta się już z obydwu rodzajów pracy.

W grupie przewijaków półcalowych, w stoisku firmy CIPHER wystawiono model 1880 IPI MICROSTREAMER. Ma on prędkość przesuwu taśmy 100 ips (cali na sekundę) i gęstość zapisu 1600 bpi (bitów na cal) lub 50 ips i 3200 bpi przy pracy strumieniowej, natomiast w trybie start-stopowym gęstość 1600 bpi i przesuw 25 ips oraz prędkość przesyłania 6 M bitów/s. Dostosowany jest do norm ANSI/IBM. W trybie podwójnego zapisu pozwala zapisać na jednym krążku taśmy 92 M bajty informacji.

Wśród przewijaków ćwierćcalowych DOMEL zaprezentował model WANGTEK seria 5000E w wersji ciągłego przewijania, z interfejsem QicII i automatycznym ustawieniem głowic. Przesuw taśmy wynosi 90 ips, a pojemność zapisu 60 M bajtów.

Minął już okres, kiedy sądzono, że nie opłaca się już kupować przewijaków taśm za cenę równoważną cenie mikrokomputera dla samego tylko przechowywania danych. SICOB podkreślił znaczenie tego typu pamięci.

Co do przyszłości, to jej wizję zaprezentowała firma CONTROL DATA, która zapowiedziała wprowadzenie na rynek urządzenia LASER DRIVE 1200, opracowanego w kooperacji z PHILIPSEM. Zastosowano tu cyfrowy dysk optyczny z zapisem i odczytem laserowym. Cechą charakterystyczną tego rozwiązania jest równoczesność odczytu informacji z jej zapisem.

W dyskach optycznych osiąga się gęstość zapisu 220 M bitów na cal kwadratowy, wobec osiąganych dotąd 20 M bitów w dyskach magnetycznych. Maksymalna wartość stopy błędu w dziesięć lat po zapisie wynosi tu 10^{-12}, a średni czas dostępu do informacji — 200 ms. Metoda dostępu jest podobna do stosowanej w dyskach magnetycznych.

W ekspozycji wyraźnie zaznaczył się szybki rozwój **mikrokomputerów przenośnych,** tj. ważących mniej niż 6—7 kg. Do niedawna, ze względu na brak oprogramowania, niewiele mówiło się o sprzęcie tego typu. Bariera ta została już jednak pokonana. Pierwszym etapem było powstanie maszyn, których oprogramowanie zawarto w pamięci ROM. Lansowane przez firmy GRID COMPASS i SHARP pamięci pęcherzykowe wyznaczyły kierunek rozwoju. Potwierdził go HEWLETT-PACKARD modelem HP 110. Następnym krokiem jest zaprezentowany na SICOBIE, kompatybilny z wyrobami IBM, prawdziwie przenośny mikrokomputer ONE firmy DATA GENERAL. Konfiguracja obejmuje jednostkę centralną opartą na układzie 80C88 (wersja CMOS procesora IBM-PC), 512 K bajtów pamięci operacyjnej, dwie dyskietki 3,5 cala (737 K bajtów), klasyczną 79-klawiszową klawiaturę, alfanumeryczny i graficzny ekran LCD (25 linii po 80 znaków lub 256×640 punktów adresowalnych). Wyposażony jest w baterię pozwalającą na 10-godzinną nieprzerwaną pracę. Sterowanie ekranem jest realizowane przez dwa układy matrycowe wykonane w technologii CMOS, które emulują układy wyświetlania zastosowane w IBM PC.

Użytkownik tego mikrokomputera ma swobodę wyboru między językami COBOL, PASCAL MT+, FORTRAN 77, C i PL/I oraz dysponuje zbiorem ponad 70 pakietów programowych zrealizowanych przez 30 firm (m.in. Lotus 1-2-3, Wordstar, Multiplan, dBase II, Condor 3, Kman, Access Manager). Może korzystać z systemów operacyjnych VENIX, MS-DOS, CP/M-86.

ONE dobrze wytrzymuje współzawodnictwo z PC5000 SHARPA i HP110, które zresztą też nie są pozbawione zalet. Przewagą PC5000, jest przechowywanie przez niego w pamięci systemu operacyjnego MS-DOS oraz możliwość realizacji funkcji maszyny do pisania. W tym względzie jest on bardziej uniwersalny niż ONE, który — mimo że ma służyć jako narzędzie pracy także podczas podróży służbowych — nie zawiera drukarki.

Firma HEWLETT-PACKARD po wprowadzeniu na rynek komputera osobistego, model 150, zaprezentowała swój przenośny model HP110, nie ważący nawet 5 kg. Zbudowany w oparciu o mikroprocesor INTEL 8086 zawiera 384 K bajty pamięci ROM i 272 K bajty RAM. W wersji standardowej dysponuje takimi składnikami oprogramowania, jak MS-DOS, Memomaker, Lotus 1—2—3, emulator terminali, a także pamięcią CMOS, ekranem na ciekłych kryształach (16 linii × 80 znaków) oraz baterią pozwalającą na 16 godzin pracy.

Spotkać można było również konstrukcje innych firm: MAGNUM DE GOUPIL skonstruowany przez firmę australijską, OLIVETTI M-10, SORD IS 11. Mikrokomputery przenośne stały się ostatnio modne i nawet IBM zapowiedział zza Atlantyku, że niedługo zaprezentuje w tej klasie własną propozycję.

Popularna dziś cecha sprzętu, jaką jest jego przenośność, jest dyskutowana także w innym wymiarze. Uzupełniona kompatybilnością, dotyczy często omawianego zagadnienia **uniwersalności oprogramowania** — w warunkach gwałtownego rozwoju różnych systemów komputerowych. Problem poziomu i zakresu kompatybilności był dyskutowany zarówno podczas CONVENTION INFORMATIQUE, jak i między stoiskami SICOBU.

Dużo mówiło się o dyskowych systemach operacyjnych MS-DOS i CP/M, które zapewniają możliwość stworzenai oprogramowania przenośnego, dostarczając programistom standardowych punktów wejścia. Ich wadą jest jednak to, że przetworzone przez nie programy są zbyt długie. Powinno się więc, za pomocą programu zarządzającego, poddawać ocenie punkty wejścia tworzące interfejs z częścią systemu charakterystyczną dla danej maszyny. Należy więc stworzyć programistom możliwość ominięcia dekodera funkcji i łatwiejszej współpracy z komputerem oraz szansę dobrej prezentacji wyników na ekranie. Niestety, przy aktualnym stanie rozwoju sprzętu nie istnieje jeszcze unifikacja ekranów i sposobów ich zarządzania.

Stwierdzenie, że oprogramowanie jest przenośne implikuje jego kompatybilność, począwszy od poziomu kodu źródłowego, oraz możliwość wykonania na różnych maszynach. Żadne opro-

gramowanie nie spełnia na razie tego warunku. Wymagałoby to modyfikowania składni języka, wprowadzania adaptacji maszyny lub — co jest najtrudniejsze — dokonywania przeróbek jej niektórych modułów.

Obecnie tendencje akcentują konieczność upowszechnienia języka C. Wiąże się to jednak z pewnymi problemami. Nie można, na przykład, mówić o oprogramowaniu przenośnym między maszynami, z których jedna pracuje pod nadzorem systemu UNIX, a druga pod CP/M. Oceniając sytuację trzeźwo, stwierdza się, że możliwości języka C ograniczają się jedynie do dużej łatwości tworzenia funkcji i dostępu do systemu. Co do rzeczywistej przenośności i kompatybilności, to na dzisiejszym etapie bardzo zróżnicowanego rozwoju konstrukcji komputerów jest ona nadal nie rozwiązana. Dla przykładu — zegar stymulujący jednostkę centralną 80C88 komputera ONE ma częstotliwość 4 MHz, podczas gdy zegar w BIG BLUE pracuje z częstotliwością 4,77 MHz. Podobne różnice występują między wszystkimi składowymi urządzeń. Praktyka potwierdza, że ciągle jeszcze daleko jest do pełnej kompatybilności.

Na czoło systemów operacyjnych wysunął się zdecydowanie system UNIX. Dostrzegało się go na SICOBIE w stoiskach wszystkich najważniejszych firm. UNIX — wysoko oceniany przez fachowców — zyskał miano systemu roku 1984. Z powodzeniem funkcjonuje on w licznych komputerach uniwersalnych, minikomputerach oraz w ponad stu mikrokomputerach. Liczy ponad sto tysięcy instalacji.

Nie można mówić o jakimkolwiek systemie operacyjnym bez odwoływania się do UNIXA. Wydawać by się mogło, że stanie się on wkrótce systemem uniwersalnym. Dla wielu maszyn ma on jednak zbyt ograniczone możliwości, mimo że podkreśla się takie jego zalety, jak interakcyjność, wielozadaniowość i wielodostępność. I nie należy zapominać, że to oprogramowanie, napisane w języku C, jest przenośne.

Po okresie, który dostarczył kilku produktów przebojowych, tendencje dalszego rozwoju oprogramowania stanowią wielką niewiadomą. Typowe oprogramowanie mikrokomputera obejmuje funkcje: przetwarzania tekstu, zarządzania bazą danych, tablicowania, graficzne i telekomunikacyjne. Jako przykład może posłużyć LOTUS 1—2—3, który skupia trzy funkcje: zarządzanie kartotekami, prezentację graficzną i tabelaryczną. FRAMEWORK, ukierunkowany głównie na zarządzenie kartotekami — oferuje nieco inny zestaw funkcji: tabelaryczną, przetwarzania tekstu i rozbudowanych możliwości graficznych, spełnia też warunki do samodzielnego pisania procedur w stworzonym specjalnie w tym celu języku programowania.

Innymi produktami programowymi, które rozwiązują blisko 80% typowych zagadnień są: KNOWLEDGE MAN, VISI-ON, SYMPHONY, JANE itp.

O ile tylko dysponuje się odpowiednim oprogramowaniem graficznym, i monitorem ekranowym, to wszystkie komputery osobiste dają możliwość **przetwarzania graficznego.** W najszerszym zakresie dotyczy to prezentowania danych w postaci wykresów kołowych, histogramów, krzywych itp., które znacznie lepiej przemawiają do wyobraźni człowieka niż ciągi liczb.

Szczególnie dużo spośród zaprezentowanych na SICOBIE terminali graficznych dotyczyło zastosowań typu CAD/CAE. Ostatnio na rynek wchodzi także szeroka oferta związana z przetwarzaniem obrazów. Jej rozwój wyznaczony jest w znacznym stopniu pojawieniem się laserowych wideodysków, które umożliwiają tworzenie obszernych banków obrazów.

Terminale graficzne ewoluują powoli w kierunku inteligentnych stacji — superkomputerów osobistych, zdolnych do przetwarzania lokalnego, bez korzystania z komputera centralnego. Dobrym przykładem tej tendencji jest zaanonsowana przez firmę TEKTRONIX 32-bitowa seria 6000, uwzględniająca podstawowe standardy w dziedzinie języków, systemów operacyjnych (UNIX, CP/M 86, MS/DOS) i systemów łączności (ETHERNET, IEEE). Firma GIXI zaprezentowała RADIANCE 2000A, wprowadzony na rynek w maju 1984. Model ten, w wersji rozszerzonej, oferuje — dzięki omiataniu rastra — 2048×1598 punktów adresowanych. Zawiera dwa procesory, dwa poziomy pamięci, ma możliwość wyświetlania obrazów w czterech kolorach. Jest kompatybilny z TEKTRONIXEM 4010 i może przesyłać informacje za pomocą interfejsu RS232C z szybkością 38 400 bodów.

Interesującą nowością było stanowisko do przetwarzania obrazów firmy GIXI. Obejmuje ono terminal graficzny RADIANCE 320 (256 kolorów, 640×488 punktów), magnetoskop sterowany mikroprocesorem, laserowy czytnik wideodysku, zawierającego banki obrazów nieruchomych i obrazów animowanych, drukarkę dysponującą bardzo szeroką gamą kolorów i możliwością zapisu na przezroczystej folii, odtwarzacz wideo oraz projektor z dużym ekranem.

W dziedzinie tradycyjnych zastosowań CAD/CAE zwracał uwagę kolorowy terminal graficzny firmy BENSON, dobrze znanej w dziedzinie grafiki komputerowej. Terminal ten pozwala rozróżniać 768×712 punktów adresowalnych. Jest kompatybilny z urządzeniami TEKTRONIX 4010, PLOT 10 i DEC VT100/32. Wyposażony jest w dwa procesory, co umożliwia przetwarzanie lokalne i zwiększoną szybkość transmisji do komputera centralnego.

Obok terminali graficznych ugruntowaną pozycję nadal zajmują **drukarki,** wśród których prym wiodą dwa typy: mozaikowa i z głowicą rozetkową. Kilka firm przestawia się na nowe techniki: laserową i natryskową, których główną zaletą jest cicha praca. W drukarkach laserowych największe problemy konstrukcyjne stawia nadal element ruchomy — lustro. Jak na razie trudności te udało się przezwyciężyć japońskiemu CANONOWI, który zaprezentował drukarkę o szybkości 6 stron/min., wprowadzoną następnie na rynek w listopadzie 1984. Modele o podobnej konstrukcji wystawili także HEWLETT-PACKARD i CORONA. SIEMENS ma nadzieję zaprezentować swoje nowe rozwiązanie w tym roku. Także APPLE pracuje nad drukarką laserową do MACINTOSHA.

Technika natryskowa stwarza nadal problemy natury nie tyle elektronicznej, co chemicznej. W precyzyjnie kalibrowanych otworach, przez które wytryskiwany jest atrament, bardzo łatwo dochodzi do jego zasychania, co oczywiście unieruchamia drukarkę. Na rynku dostępne są już modele PT 88 i 89 SIEMENSA (55 do 120 znaków/s, przy szerokości papieru 250 i 400 mm) oraz THINKJET HP (150 znaków/s). Konkurencja japońska — TOSHIBA, FUJITSU i HITACHI — nie tai swoich ambicji. TOSHIBA — na przykład — prezentowała drukarkę z nanoszeniem termicznym, siedmioma kolorami, rozdzielczością 8 linii/mm i szybkością 1 strony/min. Inna propozycja to model firmy ADVANCED COLOR TECHNOLOGY, oparty na technologii SIEMENSA z rozdzielczością pionową 85 punktów/cal i poziomą od 40 do 140 punktów/cal. Szybkość drukowania sięga 14 cm/min. przy możliwości uzyskania 125 odcieni barw.

Dalszy rozwój informatyki wymaga doskonalenia **dialogu człowieka z maszyną.** Na CONVENTION INFORMATIQUE stwierdzono, że jeżeli przeanalizować kształtowanie się kosztu minuty pracy wielostanowiskowego sprzętu informatycznego i kosztu minuty pracy obsługującego go personelu, okazuje się, że sprzęt systematycznie tanieje, natomiast ludzie są coraz drożsi. Optymalizacja w dziedzinie rozwoju sprzętu musi implikować szybki wzrost wydajności pracy jego użytkowników, przez co rozumie się m.in. zwiększenie liczby operacji wykonywanych przez operatora w jednostce czasu.

M. Lechaczinski z firmy IBM zaprezentował rezultaty swoich badań, z których wynika, że wydajność operatora wzrasta liniowo wraz z maleniem czasu odpowiedzi komputera w zakresie powyżej 1 s, natomiast niewspółmiernie szybciej w zakresie czasów poniżej sekundy. Wydaje się, że gdy czas odpowiedzi sięga kilka sekund operator po prostu oddala się myślami od wykonywanej pracy i powrót do tematu (odnalezienie kontekstu) zajmuje dłuższą chwilę. Jeżeli natomiast na odpowiedź komputera nie trzeba czekać, myśli operatora są ściśle związane z wykonywanymi czynnościami, co oczywiście daje efekt w postaci zwiększonej wydajności.

Należy więc zaoferować użytkownikowi bardzo krótkie czasy odpowiedzi, zwiększając zdolność obliczeniową procesora, pojemność pamięci i szybkość urządzeń telekomunikacyjnych oraz wyposażając stanowisko pracy w „inteligentne" oprogramowanie.

Część tych warunków spełnia zaprezentowany na wystawie system SM90. Jest rodzajem super-mikrokomputera —

maszyną wieloprocesorową o modularnej architekturze z systemem operacyjnym UNIX.

Architekturę SM90 określono mianem rewolucyjnej — tak ze względu na poziom rozwiązań zdecentralizowanych funkcji i łączności, jak i na możliwości oferowane przez UNIX. Ulrich Finger, z Centrum Telekomunikacji — twórca koncepcji SM90 — umieszcza swoją maszynę w erze informatyki rozproszonej i komputerów indywidualnych. Treścią jego filozofii jest danie użytkownikowi do dyspozycji maszyny, której moduły są dostosowane do realizacji jego indywidualnych potrzeb, a jednocześnie zapewniają znaczną moc obliczeniową.

SM90 gwarantuje zwiększenie możliwości obliczeniowej, prostą integrację rozproszonych systemów informatycznych, wysoki stopień dyspozycyjności oraz zakładaną — możliwość adaptacji do potrzeb dowolnego użytkownika. To ostatnie uzyskano przez zastosowanie heterogenicznych mikroprocesorów definiujących interfejs standardowej łączności między modułami, a także decentralizację spełnianych przez nie funkcji.

W strukturze SM90 rozróżnia się trzy rodzaje modułów:

— moduły przetwarzające wyposażone we własne pamięci, do których nie mają dostępu inne jednostki przetwarzające (mikroprocesor 16- lub 32-bitowy z układem sterowania dostępem do szyny),

— moduły wymiany informacji zarządzające wejściami i wyjściami modułów przetwarzania (mikroprocesor 8- lub 16-bitowy),

— moduły pamięci wspólnych, dostępne zarówno dla jednostek przetwarzających, jak i pamięci lokalnych, przyłączone do szyny komunikacyjnej.

Całkowita przestrzeń adresowa ma pojemność 16 M bajtów. Moduły są zrealizowane na kartach podwójnego formatu europejskiego 233,4×220 mm, wyposażonych w dwa łącza pozwalające przyłączyć kartę albo do szyny komunikacyjnej, albo do szyny lokalnej, lub do obu jednocześnie. Konstrukcja szyny komunikacyjnej SM BUS pozwala na użycie do siedmiu procesorów 16- lub 32-bitowych.

* * *

Na koniec najbardziej znaczący eksponat SICOBU — TELECOM 1. Zaskakująca obecność satelity na wystawie poświęconej informatyce oznacza uruchomienie we Francji — na szczeblu tak krajowym, jak i międzynarodowym — zintegrowanych sieci usług informatycznych. Przełom lat 1984—1985 przyniósł zapowiedziane przez SICOB oddanie do dyspozycji całej gamy usług sieci informatycznych stałych (TRANSFIX) lub komutowanych, z sygnalizacją typu telefonicznego (TRANSCOM) lub wzbogaconą (TRANSDYN) — w trzech klasach, pozwalających na transmisję z szybkością od 2400 do 2 M bitów/s.

Opracowała HANNA KONTKIEWICZ-CHACHULSKA

na podstawie: MINIS ET MICROS 218, 219, 220/84 oraz BUREAUX ET SYSTEMES 11/84

PICK tajemnica informatyki

Każdy numer miesięcznika BYTE ma około 500 stron. Kilkadziesiąt innych pism opisujących produkty informatyki ma zbliżoną objętość. Kilkadziesiąt pism naukowych prezentuje też co miesiąc nowe idee i rozwiązania w zakresie organizacji sprzętu i oprogramowania. Ponadto ukazują się setki raportów, sprawozdań i materiałów z konferencji czy seminariów oraz dokumentacje, skrypty, książki... Istna lawina informacji, w której aż trudno się czasem poruszać. Nie od rzeczy będzie jednak pytanie czy informacje te faktycznie oddają istniejącą rzeczywistość.

Przed wielu laty, w jakimś przypływie szczerości, ujawniono, że w czasopismach naukowych publikowane są też prace, które autorzy — ze względu na brak oczekiwanych rezultatów wyrzucili do kosza. Opracowania te, prowadzące do nikąd, są następnie uważnie studiowane przez czytelników, gdy tymczasem sami autorzy zajmują się już czym innym. I dużo czasu mija, zanim pozostali zorientują się, że ta droga rozwiązania nie rokuje żadnych sukcesów. Takie postępowanie, niezgodne z zasadami etyki, ułatwia uzyskanie przewagi nad konkurencją.

Do tego trzeba też dodać ochronę metod i wyników prowadzonych badań. Zasada ta jest z powodzeniem realizowana w informatyce przez wiele instytucji i firm. Często w ogłoszeniu o konferencji można spotkać uwagę, że o ostatecznym uczestnictwie w niej decyduje tajemnicza agencja X, która — jak łatwo się domyślić — ma ścisłe powiązanie z departamentem obrony USA. Na ogólnodostępnych konferencjach niektóre referaty są wycofywane przed prezentacją i to bynajmniej nie dlatego, że są na niskim poziomie.

Ilustracją takiego stanu rzeczy może być historia powstania i rozwoju systemu operacyjnego PICK. Większość informatyków — i to nie tylko w Polsce — powie, że pierwszy raz słyszy tę nazwę i że pewnie jest to najnowszy produkt jakiejś firmy. Otóż nie, system PICK istnieje i działa już od dwudziestu lat na ponad 32 typach komputerów i w 20 tys. instalacji. Nawet bezpośredni użytkownicy nie mieli pojęcia o skali jego rozpowszechnienia. Dla niewtajemniczonych PICK krył się pod wieloma nazwami: REALITY (Rzeczywistość), ULTIMATE (Podstawowy) czy CRYSTAL (Kryształ) zrealizowany bezpośrednio w technice VLSI. Pierwsze wzmianki o rzeczywistej historii tego systemu pojawiły się dopiero w 1984 roku i to w formie reklamowej.

Twórcą systemu PICK jest Dick Pick, który od 1965 roku do chwili

Właściwości systemu PICK

Dostęp — rozpoznawanie poleceń w języku angielskim dla kilku plików w oparciu o kryteria selekcji i operatorów relacji, włączając w to sortowanie, sumowanie, analizę statystyczną i formatowanie wydruków.

Edytor — dostęp do dowolnego elementu danych oraz tworzenie, modyfikowanie, usuwanie rekordów w plikach danych, słowniku danych, programach źródłowych i procedurach systemowych.

Ewidencja systemu — automatyczne monitorowanie systemu z uwzględnieniem czasów dołączenia nowych użytkowników, numerów portów, wykorzystania centralnego procesora oraz liczby drukowanych stron.

Przechowywanie wydruków (ang. spooler) — szeregowanie do 600 plików z wydrukami, zarządzanie maks. 16 drukarkami, uwzględnianie dyrektyw operatora kontrolujących wydruki oraz bezpośrednich do określonej drukarki, określających liczbę kopii, czas wydruku oraz ewentualne przeniesienie na taśmę magnetyczną.

Język obsługi terminala — podstawowy kontakt z użytkownikiem z dostępem do ponad 200 różnych procedur wspomagających oraz do innych użytecznych procedur systemowych.

Procedury systemowe — formatowanie ekranu, przekazywanie parametrów, weryfikacja danych wprowadzanych oraz dostęp do wszystkich programów systemowych.

Data BASIC — język programowania z kompilatorem i generatorem kodu relokowalnego, zawierający wspomaganie uruchamiania programów; w języku zawarto też powiązanie z administratorem bazy danych oraz z dostępem do funkcji „dostępu" do informacji oraz programu przechwytującego.

Ochrona — opcjonalna kontrola akceptacji nowego użytkownika, zezwolenia na dostęp do pliku, dostępu do rachunkowości systemu, dyrektyw oraz wykorzystywania urządzeń.

Redagowanie tekstów — na ekranie, z pełnymi możliwościami adiustowania, literowania, formatowania strony, numeracji stron i automatycznej indeksacji.

obecnej zajmuje się swoim dziełem. Pierwsze prace były opłacane z funduszów publicznych przeznaczonych dla projektu helikoptera CHEYENNE, zamówionego przez armię USA. Według prawa amerykańskiego, wykorzystywanie dotacji publicznych obliguje do przekazania rezultatów pracy do ogólnego wykorzystania. W tym wypadku trwało to wyjątkowo długo.

Po raz pierwszy system został zaimplementowany na IBM 360, a po czterech latach na MICRODATA 1600. Następnie autor systemu rozpoczął proces sądowy z firmą MICRODATA o to, kto jest autorem, a kto właścicielem i czy autor może zabrać ze sobą logikę projektu, implementując go następnie na innym sprzęcie i dla innej firmy. Sprawę wygrał autor i po dziewięciu miesiącach pracy — system został zaimplementowany na komputerze HONEYWELL Level 6. Dalsze losy systemu potoczyły się już lawinowo. Do 1981 roku było pięć implementacji, w 1982 powstało dalszych pięć, w 1983 osiem następnych, a w 1984 dalszych czternaście. A więc po dwudziestu latach pracy — sukces. Można już odsłonić tajemnicę.

Przedstawione zestawienie zawiera główne elementy systemu PICK spełniające prawie wszystkie cechy funkcjonalne dobrego, nowoczesnego systemu. Jest to system sprzęgający w jedną całość system operacyjny z systemem zarządzającym bazą danych, w którym zaimplementowana pamięć wirtualna udostępnia do 8000 M bajtów informacji (jeżeli istnieje odpowiednia wielkość powierzchni nośnika na dyskach). Zbiór danych w systemie jest zapisywany w plikach o dość rozbudowanych metodach formatowania rekordów. Dostęp do informacji w plikach odbywa się poprzez hierarchicznie uporządkowane słowniki, o wspólnym korzeniu w słowniku systemowym. Szczegóły rozwiązania nie zostały jeszcze ujawnione, ale jego możliwości ilustruje fakt, że użytkownik może zwrócić się do systemu z takim oto dość swobodnie sformułowanym poleceniem: *List customers with overdue balance greater than 100 £ showing company name, address and overdue balance.*

Po przeszukaniu odpowiednich słowników, system rozpozna treść zdania i na podstawie kluczy zapisanych w rekordach plików wypisze na drukarce lub wyświetli na ekranie żądane informacje w odpowiednim formacie.

Intryguje pytanie, co jeszcze ukryto w głębinach pamięci komputerowych. Nie ma się co łudzić, że system PICK był jedyny. Prawie całkowity brak możliwości ochrony oprogramowania przed skopiowaniem, finansowanie w ponad 60% rozwoju informatyki przez agencje powiązane z wojskiem — prowadzą do znacznego ograniczenia prezentacji uzyskanych wyników prac i rozwiązań praktycznych. To paradoks, bo przecież dążeniem informatyki jest coraz pełniejsze i nowocześniejsze zorganizowanie, wyszukiwanie i udostępnianie informacji. Natomiast informacje o informatyce są chowane, a my możemy się tylko głowić, jaka rzeczywista różnica dzieli naszą informatykę od tej najnowocześniejszej.

Opracował WACŁAW ISZKOWSKI

na podst. MICROPROCESSORS AND MICROSYSTEMS, Vol. 8, No. 7, september 1984

Uwagi o terminologii języka ADA

We wrześniu 1983 Polskie Towarzystwo Informatyczne opublikowało propozycję polskiego słownictwa języka ADA, opracowaną przez członków sekcji ADY i obejmującą ponad 170 terminów. W czasie, który minął od momentu ogłoszenia słowniczka drukiem (BIULETYN PTI, nr 9—10, 1983), większość tych terminów została przyjęta i utrwalona, lecz niektóre jego fragmenty należy poddać pewnej weryfikacji i krytyce. Wyraz „krytyka" powinien być odebrany jako „analiza i ocena z określonego punktu widzenia" (por. Słownik języka polskiego), a nie jako „ocena ujemna", gdyż większość proponowanych terminów uważam za trafne, chociażby dlatego, że miałem swój udział w ich opracowywaniu. Inaczej rzecz ma się jednak z doborem polskiego odpowiednika obcego terminu, a inaczej z jego użyciem w określonym kontekście. Z tego drugiego punktu widzenia mam do słownika trochę zastrzeżeń i propozycji, które można podzielić na dwie grupy.

Po pierwsze — nie uwzględniono w nim wielu istotnych terminów występujących w podręcznikach języka. Przedstawiam tu ważniejsze z nich wraz z propozycjami polskich odpowiedników:

abandon — zaniechać
access rights — prawa dostępu
actual parameter — argument
assignment — przypisanie
association — skojarzenie
body stub — korpus
context specification — specyfikacja kontekstu
declarative part — część bierna
erroneous — niepoprawny
establish — ustanowić
exception handling — obsługa wyjątku
form — postać
fixed point — stałopozycyjny
floating point — zmiennopozycyjny
formal parameter — parametr
hide — przesłaniać
introduce — przedstawić
raise an exception — zgłosić wyjątek
sequence of statements — część czynna
unit — segment.

Po drugie — polskie odpowiedniki części terminów angielskich nie są dobrane zbyt dobrze. Przedstawiam propozycje zmian wraz z uzasadnieniem:
box — kaseta, zamiast romb
użycie nazwy figury geometrycznej nie oddaje sensu zbliżonego do zwrotu **black box** (czarna skrzynka).
default — domniemany, zamiast domyślny

(Termin „domniemany" jako przestarzały nie będzie wchodzić w kolizję, ze słowem „domyślny").
encapsulated data type (okapsułkowany typ danych)

(Termin ten jest zbędny, gdyż w normie użyto go tylko raz i to w znaczeniu przenośnym.)
entity — wielkość, zamiast rzecz

(„Wielkość programowa" jest bardziej zrozumiałe niż rzecz programowa".)
entry — ingresja, zamiast wejście

(Proponuję ten neologizm, aby zachować termin „wejście" dla angielskiego **input**.)
generic — rodzajowy, zamiast generyczny

(W rosyjskim tłumaczeniu książki Petera Wegnera (Programming with Ada, Prentice-Hall, 1980) użyto przymiotnika „rodowy".)

instance — wcielenie, zamiast egzemplarz
(Pakiet rodzajowy może mieć wiele wcieleń.)

instantiation — powołanie, zamiast egzemplifikacja
(Egzemplifikacja oznacza tylko przedstawienie przykładu.)

interface — sprzężenie, zamiast sprzęg
(Takie tłumaczenie nasuwa mniej skojarzeń ze sprzętem.)

limited — ograniczony, zamiast zastrzeżony
(Ograniczenie dotyczy użycia operatora równości (=).)

numeric — liczbowy, zamiast numeryczny
(Takie jest właściwe tłumaczenie przymiotnika angielskiego.)

range — przedział, zamiast zakres
(Należałoby zachować zwyczajowe tłumaczenie terminu **scope**.)

scope — zakres, zamiast zasięg
(„Zakres" deklaracji jest szerszy od „zasięgu".)

string — tekst, zamiast napis
(Napis „UB" jest literałem tekstowym (tekstem), a napis 'A' jest literałem znakowym.)

subprogram — subprogram, zamiast podprogram
(Funkcja też jest podprogramem, co może prowadzić do nieporozumień.)

template — model, zamiast wzorzec
(Należałoby uniknąć skojarzeń z terminem „liczba wzorcowa" — ang. model number).

Z przytoczonego zestawienia wynika, że nie należy spieszyć się z domykaniem słownika, zwłaszcza że przytoczone braki zostały wykryte na podstawie niezbyt wnikliwego porównania z podręcznikiem języka. Jestem ponadto zdania, że potwierdzenie trafności terminu można uzyskać tylko przez użycie go w szerszym kontekście oraz że każdy termin powinien mieć popartą odpowiednio dobranym przykładem definicję.

JAN BIELECKI

Listy

Mówić o sobie

Gwałtowny rozwój polskiej informatyki w latach siedemdziesiątych zwodniczo sugerował, że zaczęło powszechnie funkcjonować myślenie informatyczne, niezbędne do trwałego wykorzystywania komputerów w życiu społecznym i gospodarczym. Potrafiliśmy wówczas wykorzystać koniunkturę i zainteresować decydentów komputeryzacją. Niestety, tylko decydentów, i to niedostatecznie. W obliczu kryzysu dość szybko zrezygnowali oni z tej „drogiej imprezy", krótkowzrocznie dostrzegając tylko jej wysokie koszty, my zaś — choć w poszerzonym gronie — pozostaliśmy nadal w swoim kręgu.

Winimy za ten stan decydentów, i słusznie. Ich postanowienia mają ogromny wpływ na zmianę sytuacji. Nie możemy więc przestać dobijać się do kolejnych drzwi, by uzyskać zgodę na realizację naszych pomysłów, nie możemy przestać mówić o tym i pisać. Jest to jedna płaszczyzna „sporu o informatykę". Ale czy jedyna? Czy najważniejsza?

Małe są szanse na uzyskanie zadowalającego stanu przez pertraktacje z decydentami. Nie zapominając o nich, zwróćmy się więc ku właściwemu adresatowi naszych działań — potencjalnemu użytkownikowi.

Lata siedemdziesiąte, oprócz rozwoju bazy zawodowej, przyniosły również izolację środowiska informatyków. Godzono się na nasze pomysły, ale niechętnie. Nie rozumiano do końca zasadności komputeryzacji. Kto nie rozumiał? Nie tylko decydenci. Przede wszystkim przeciętny, potencjalny użytkownik naszych pomysłów. Staliśmy się środowiskiem hermetycznym. Podnosiliśmy swoje zawodowe kwalifikacje, doskonaliliśmy oprogramowanie podstawowe i użytkowe, zwiększaliśmy naukowy dorobek polskiej informatyki. Sami rozumiejąc potrzebę takich działań, nie potrafiliśmy jednak tego zrozumienia przekazać innym, niefachowcom.

Jakże często spotykałam studentów, nawet na kierunkach technicznych, którzy na dźwięk słowa ETO bądź z lekceważeniem machali ręką, bądź żywo reagowali... złością. Zawsze z konkluzją: po co nam ten przedmiot!? A ileż koniecznych zastosowań przerwano jedynie z powodu braku kompetencji odpowiedzialnego personelu.

Komputer już wtedy (rok 1980) funkcjonował w ludzkich umysłach jako monstrum, może budzące nieco szacunku, ale przede wszystkim sprawiające bezustanne kłopoty. Powstał mit informatyka, człowieka mądrego (czytaj: rozumiejącego komputer), ale — uciążliwego. Ten mit w dużym stopniu stworzyliśmy sami, ograniczając się do swoich zawodowych działań.

Dzisiaj mity funkcjonują nadal, mimo pojawienia się na polskim rynku licznych mikrokomputerów, przybliżających laikom technologię informatyczną. Nadal mówimy językiem zbyt trudnym albo nie mówimy wcale. Grono osób rozumiejących konieczność obcowania z komputerem, osób spoza środowiska zawodowych informatyków poszerzyło się, ale i tak jest to tylko garstka. Ich oddziaływanie nie wystarcza, a dopóki nie przekonamy niemal każdego członka naszej polskiej społeczności do korzystania z tego narzędzia, dopóty — jako całość — pozostaniem zaściankiem.

Podstawą w „sporze o informatykę" powinny stać się działania zmierzające do wykształcenia w skali społeczeństwa powszechnej potrzeby komputeryzacji. Wiem, brzmi to fantastycznie. Brak sprzętu, brak oprogramowania dającego możliwość łatwego — bez konieczności zaznajomienia się z arkanami informatyki — wykorzystania sprzętu istniejącego, kosztowność każdego zastosowania... Ale przecież gdyby to wszystko było, komputery broniłyby się same. Właśnie dlatego, że nie ma, koniecznością staje się wykształcenie takiej społecznej potrzeby innymi sposobami.

Z punktu widzenia prywatnego użytkownika mikrokomputer jest drogi, ale dla wielu jego koszt stanowi sumę przynajmniej wyobrażalną, co w połączeniu z przekonaniem o przydatności, a także kilkuletnią praktyką życia w kryzysie — może dać nadspodziewane efekty. Prywatni użytkownicy przecież już są. Dla instytucji cena mikrokomputerów nie powinna stanowić większego problemu i gdy ludzie w niej pracujący zechcą z tego sprzętu korzystać, zdołają pokonać wiele finansowych barier. Prędzej czy później i decydenci znajdą się w gronie użytkowników. Społeczna potrzeba obcowania z informatyką musi w końcu wpłynąć na ich decyzje.

Nie można już dłużej zamykać się w zawodowym getcie. Tylko wykraczając poza nie zdołamy zniszczyć pokutujące mity. Mówmy przystępnie o możliwościach informatyki, szczególnie — o zastosowaniach mikrokomputerów. Przestańmy skupiać się na drobnych innowacjach konstruktorskich czy programistycznych!

Trzeba wreszcie zacząć szerzej mówić i pisać o już istniejących w Polsce zastosowaniach mikrokomputerów. Nie tylko o tych ambitnych, nie pozostawiających cienia wątpliwości co do wysokich kwalifikacji ich projektantów. Nie czas na wyłączne przekonywanie innych i siebie o wysokim poziomie zawodowej myśli informatycznej. Piszmy również o drobnych rozwiązaniach wymagających użycia mikrokomputera.

Opisując oprogramowanie użytkowe, pozbądźmy się pretensji do fachowości opisu tam, gdzie nie jest ona potrzebna, bo może prowadzić do rozwlekłości, a nawet zagmatwania tekstu. Jednocześnie w opisach uwzględniajmy ewentualnych przyszłych użytkowników, dla których już nawet stwierdzenie typu: „systemy biurowe" bez odpowiedniego rozwinięcia — może być niewiele znaczącym hasłem. Nie zapominajmy, że ludzie często, by podjąć wysiłek korzysta-

nia z mikrokomputera, który nie jest jeszcze ani tani, ani (dla ignoranta) łatwy w obsłudze, chcą dokładnie znać cel takiego działania. Mówmy do nich w kategoriach ich potrzeb — prywatnych, towarzyskich, zawodowych. Zostawmy na razie z boku metody informatyczne.

Mówmy i piszmy również o przeobrażeniach społecznych, jakie informatyka powoduje; na czym one miałyby polegać. W jaki sposób, dzięki niej, może pojawić się — na szerszą niż kiedykolwiek skalę — człowiek kreatywny.

Zacznijmy mówić zrozumiale. Nie tylko do decydentów i do siebie nawzajem.

BARBARA FABIAŃSKA
Poznań

Konferencje

INFOPROD'85
Techniki komputerowe w zarządzaniu produkcją

To temat konferencji, która odbędzie się w dniach 3—5 października 1985 r. w Ciechocinku. Jej organizatorami są: Instytut Badań Systemowych PAN w Warszawie, Instytut Organizacji i Projektowania Systemów Produkcyjnych Politechniki Gdańskiej, Zakład Informatyki Stosowanej Akademii Techniczno-Rolniczej w Bydgoszczy oraz Polskie Towarzystwo Ekonomiczne, Oddział Wojewódzki w Bydgoszczy.

Sekretarzem naukowym konferencji jest doc. dr hab. Ludosław Drelichowski z Akademii Techniczno-Rolniczej w Bydgoszczy.

Zadaniem konferencji jest ocena dotychczasowych zastosowań komputerów, mini- i mikrokomputerów w zarządzaniu produkcją oraz wytyczenie metod dalszych ich wdrożeń. Jej program przewiduje zaprezentowanie

- przykładów praktycznych zastosowań minikomputerów MERA 9150 we współpracy z dużymi komputerami
- możliwości wykorzystania mikrokomputerów IMP-85 (MK-4501), MERITUM itp.
- sprzętu i oprogramowania

oraz przedyskutowanie koncepcji zmian w metodach sterowania produkcją z wykorzystaniem sprzętu komputerowego, mini- i mikrokomputerów.

Konferencja jest adresowana głównie do dyrektorów produkcji i dyrektorów ekonomicznych przedsiębiorstw, szefów produkcji oraz kierowników wydziałów produkcyjnych i służb (kompletacyjnych i planowania produkcji, gospodarki materiałowej, ekonomicznych i informatycznych).

W czasie jej trwania przewiduje się wygłoszenie 15 referatów, które znajdą się we wcześniej wydanych materiałach konferencyjnych.

Koszt uczestnictwa jednej osoby wynosi 8500 zł i obejmuje: zakwaterowanie, wyżywienie, komplet referatów oraz materiały pokonferencyjne. Zgłoszenia przyjmuje:

PTE Oddział Wojewódzki
ul. Długa 34, 85-034 Bydgoszcz
Konto: PKO I Oddział Bydgoszcz,
nr 9511-619-132.

Iszkowski W.: Superkomputer CRAY

INFORMATYKA 1985, nr 5, s. 1

Charakterystyka pojęcia superkomputera na przykładzie maszyny CRAY-1. Omówiono architekturę, technologię oraz moc obliczeniową tego komputera, a także dalszy rozwój rodziny maszyn CRAY.

Bielecki J.: ADA — standard operacji wejścia—wyjścia

INFORMATYKA 1985, nr 5, s. 4

Krytyczny komentarz na temat standardu operacji wejścia—wyjścia w języku ADA oraz omówienie metody implementacji pakietu TEXT_IO.

Winiewski J.: Język programowania C (2)

INFORMATYKA 1985, nr 5, s. 7

Druga część charakterystyki języka C. Omówiono instrukcje, funkcje, tablice, wskaźniki i struktury, a także preprocesor języka.

Ишковски В.: Суперкомпьютер CRAY

INFORMATYKA 1985, № 5, стр. 1

Характеристика понятия суперкомпьютера на примере машины CRAY-1 Обсуждено архитектуру, технологию, возможности машины и перспективы развития.

Белецки Й.: ADA — стандарт операции ввода-вывода

INFORMATYKA 1985, № 5, стр. 4

Критическое обсуждение стандарта операции ввода-вывода на языке АДА и методы внедрения программного обеспечения TEXT—IO.

Вишевски И.: Язык программирования С (2)

INFORMATYKA 1985, № 5, стр. 7.

Вторая часть характеристики языка С. Обсуждено функции, таблицы, индексы, агрегаты данных и препроцессор языка.

Iszkowski W.: The CRAY supercomputer

INFORMATYKA 1985, No. 5, p. 1

Characteristics of the supercomputer concept on example of the CRAY-1 computer. Architecture, technology and performance of this computer, as well as development of CRAY computer family, are presented.

Bielecki J.: ADA input/output operations standard

INFORMATYKA 1985, No. 5, p. 4

Critical remarks on ADA language input/output operations standard and implementation method of the TEXT_IO package are presented.

Winiewski J.: C programming language (2)

INFORMATYKA 1985, No. 5, p. 7

Second part of the C language characteristics. Statements, functions, tables, indicators and structures, as well as preprocessor of the language, are discussed.

Iszkowski W.: Supercomputer CRAY

INFORMATYKA 1985, Nr. 5, S. 1

Eine Charakteristik von Supercomputerkonzept auf Beispiel des CRAY-1-Computers. Es wurden Architektur, Technologie und Leistung dieses Computers, sowie Weiterentwicklung der CRAY-Rechnerfamilie, besprochen.

Bielecki J.: ADA-Standard im Bereich der Ein- und Ausgabeoperationen

INFORMATYKA 1985, Nr. 5, S. 4

Ein kritischer Kommentar zum ADA-Standard für Ein- und Ausgabeoperationen und eine Besprechung der Implementierungsmethode von TEXT_IO-Paket.

Winiewski J.: C-Programmiersprache (2)

INFORMATYKA 1985, Nr. 5, S. 7

Zweiter Teil einer Charakteristik von C-Sprache. Es wurden Instruktionen, Funktionen, Tafel, Indikatoren und Strukturen, sowie Preprozessor der Sprache, vorgestellt.

WARUNKI PRENUMERATY NA 1986 R.

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat na blankiecie „polecenie przelewu" rozszerzonym dla potrzeb Wydawnictwa o część dotyczącą zamówienia. Blankiety te będą dostarczane przez Zakład Kolportażu.

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie Wydawnictwa lub blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty.
Wpłacać należy na konto NBP III O/M Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły.

Sposób zamawiania prenumeraty taki sam jak dla prenumeraty indywidualnej.

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

Przedpłaty na prenumeratę przyjmowane są w terminach:
— do 10 listopada na I kwartał, I półrocze i cały rok następny,
— do 28 lutego na II, III, IV kwartał i II półrocze,
— do 31 maja na III, IV kwartał i II półrocze,
— do 31 sierpnia na IV kwartał.

U w a g a!

Wpłaty na dwumiesięczniki przyjmowane są na okresy półroczne lub roczne.

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

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej w Warszawie ul. Mazowiecka 12, tel. 27-43-65 oraz w Dziale Handlowym Wydawnictwa ul. Bartycka 20 skr. poczt. 1004, 00-950 Warszawa, na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena miesięcznika INFORMATYKA została ustalona na 120 zł za numer (35 zł — cena ulgowa).

Cena prenumeraty wg cennika					
kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
360	105	720	210	1440	420

Ceny ogłoszeń

Od 1 stycznia br. obowiązują następujące ceny ogłoszeń publikowanych na naszych łamach:
ogłoszenia duże (zależnie od objętości):
cała strona — 35 tys. zł, 3/4 — 30 tys., 1/2 — 25 tys., 1/4 — 20 tys., 1/8 — 15 tys.
ogłoszenia drobne (zależnie od liczby słów):
jedno słowo — 30 zł

Dodatki do ceny podstawowej:
— za dodatkowy kolor (na okładce) +30%
— za zamieszczenie ogłoszenia na czwartej stronie okładki +100%
— za zamieszczenie ogłoszenia na trzeciej stronie okładki +50%

Zniżki:
— za ogłoszenie 3—5-krotne —5%
— za ogłoszenia 6—10-krotne —10%
— za ogłoszenia 11-krotne i powyżej —20%
— za artykuły reklamowe i wkładki wykonane przez zleceniodawcę —40%
— za bloki i biuletyny wykonane przez zleceniodawę — maks. —60%

W przypadku dostarczenia przez zleceniodawę materiału ilustracyjnego nie odpowiadającego warunkom technicznym druku lub tekstu wymagającego redakcyjnego opracowania, do powyższych cen doliczane będą koszty odpowiednich usług fotograficznych, graficznych lub przygotowania tekstów.

Nożyce norymberskie

No to nie trzeba już pozwolenia na przywożenie mikrokomputerów (p. INFORMATYKA 11/84, s. 13). Firmy polonijne dosłownie koszami dostarczają po umownych cenach komputer SPECTRUM (sam to widziałem w Pałacu Kultury i Nauki na wystawie Interpolcomu). Młodzież garnie się do każdego komputera, gdzie tylko się da — niektórzy piszą nawet własne programy (p. POLITYKA z 9 lutego br.). Spekulanci sprzedają fabryczne oprogramowanie jak przysłowiowe chrupiące bułeczki (p. ogłoszenia w Życiu Warszawy). Nawet rodzimy przemysł dostarcza nam radości — w tym roku wyprodukuje się aż trzy tysiące sztuk MERITUM-I. Jednym słowem — raj mikrokomputerowy w Polsce.

A ja pozrzędzę. I oczywiście nie dlatego, iżbym był przeciwny mikroinformatyce.

Proszę się tylko zastanowić, jaki to model informatyki masowej lansujemy. Ano właśnie taki, jaki sprawdził się już w przypadku masowej motoryzacji, produkcji urządzeń gospodarstwa domowego i wielu, wielu innych. Model oparty na imporcie dla mających pieniądze (którzy i tak mogą sobie kupić mikrokomputer za granicą) oraz dla instytucji zasobnych w „kwadratowe" złotówki. A w całym tym szumie informatycznym ginie gdzieś owa grupa ludzi, dla których mikrokomputer mógłby być nie tylko zabawką, ale urządzeniem do pracy. Znikają z oczu potrzeby dziennikarzy, lekarzy, pisarzy, ekonomistów, architektów, kolekcjonerów i tysiące innych zajęć, które traktowane serio wymagają użycia komputera osobistego. Dla tych ludzi SPECTRUM jest tylko zabawką; nawet IBM PC może w swej podstawowej konfiguracji okazać się niezbyt dostosowany. Bo czy można w Polsce przesłać list elektroniczny? Nie! Czy można przesłać skomplikowane obliczenia do wykonania na dużej i szybkiej maszynie? Nie! Czy można dostarczyć jakiejkolwiek redakcji zamówiony maszynopis na dysku? Nie! Co gorzej — wielu wymienionych „zainteresowanych" wcale nie zdaje sobie sprawy z możliwości komputera osobistego. Ostatnio różni znajomi pytają mnie, jaki mikrokomputer warto kupić. Kiedy odpowiadam pytaniem: — „A po co Ci komputer?", na ogół reagują głupawym śmiechem — „Jak to, przecież namawiacie w mikroKLANIE!...". Jeżeli jednak takiego delikwenta dobrze przycisnąć, to okazuje się, że jego potrzeby doskonale spełni albo prosty kalkulator programowany z funkcjami, albo dobra gra telewizyjna z wymiennymi pakietami (żeby się za szybko nie znudziła). O zastosowaniach, które wymieniłem poprzednio, a które umownie nazwać można osobistym przetwarzaniem informacji, na ogół wcale ci ludzie nie słyszeli. A nawet jeśli obił im się o uszy termin „procesor tekstu" czy też „baza danych", to na tym kończy się ich wiedza.

Zresztą, wielu zawodowych informatyków też zapewne nigdy nie zetknęło się z dobrym procesorem tekstu. No bo niby gdzie! Co innego napisać system operacyjny. O, tu mamy wspaniałych specjalistów (p. MERA 400). Ale żeby komuś chciało się napisać polski procesor tekstu, z polskim alfabetem i polskim słownikiem...

Oczywiście, są chlubne wyjątki: widziałem w zeszłym roku prywatną (to właściwie wszystko tłumaczy!) firmę polecającą usługi urlopowe. Wszystkie operacje wybierania miejsca, ustalania opłat i wreszcie rezerwowania terminu dokonywano na komputerze. Był to wprawdzie ZX81, ale — widząc moje zdziwienie — poinformowano mnie, że jest już zamówiony COMMODORE C64 z dyskiem i drukarką. W ten prosty sposób użytkownik dorósł do wyższego poziomu, ale... byt kształtuje świadomość! Mam zresztą nadzieję, że firma ta dorobi się w końcu profesjonalnego komputera. Chociaż chyba nie od jednej z firm polonijnych, która sprzedaje IBM PC za 11 mln zł, a trudno jej się dziwić, nie jest to wszak instytucja charytatywna i na podatki musi zarobić.

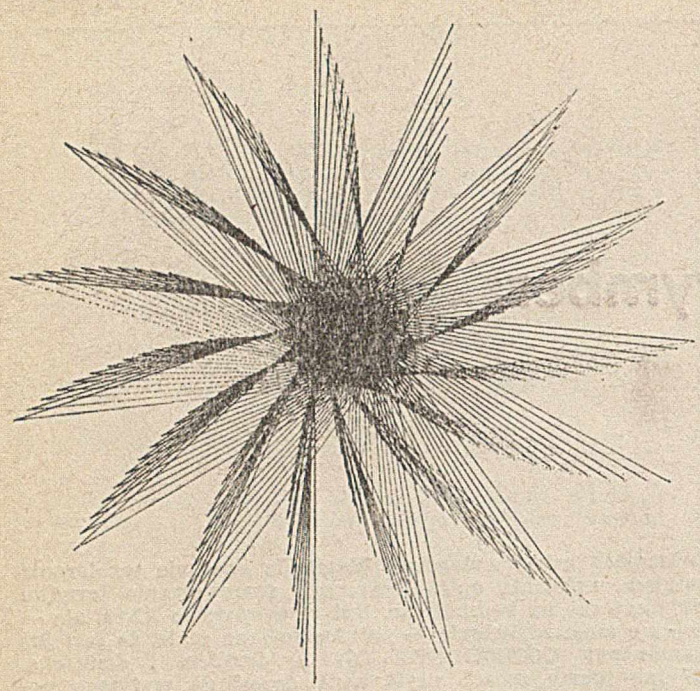
Tu dochodzimy do zjawiska, które nazywam nożycami norymberskimi. Wielu pamięta pewnie z młodości taką zabawkę: parę kawałków listewek, które rozwierane powodują wyskoczenie do przodu ukrytej za nimi pięści. Potem, zwierając listewki, wciąga się pięść z powrotem i wszystko wraca do normy. Otóż odnoszę wrażenie, że wspomniane już akcje: motoryzacyjna, domowa, a teraz komputerowa — poruszane są takim mechanizmem norymberskim. Jak się dobrze sprężyć, to można wyskoczyć wysoko i nawet mocno uderzyć. Tyle tylko, że potem rusztowanie na nożycach jeszcze szybciej opadnie. Brakuje bowiem punktu podparcia — otoczenia.

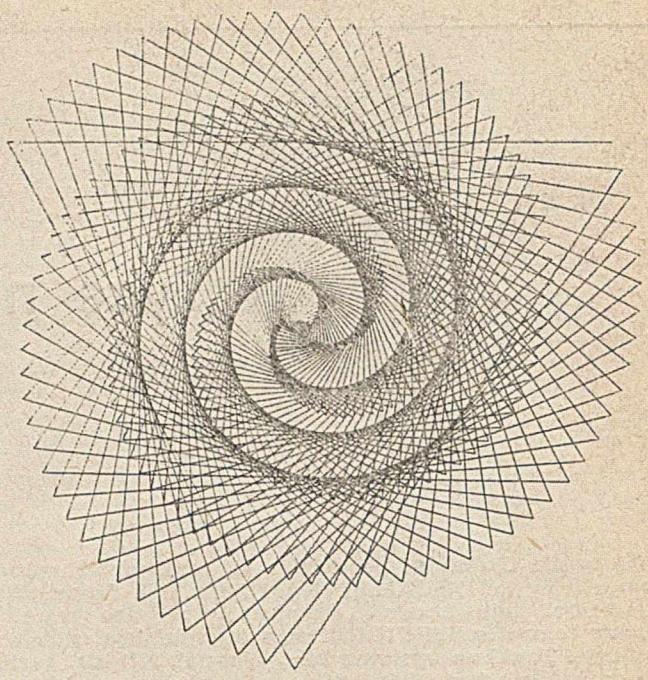
Dla komputerów otoczeniem jest nie tylko przemysł elektroniczny, ale też telekomunikacja (łącza modemowe) oraz usługi. Tak wygląda rozwinięta informatyka. A u nas? Owszem, znam paru ludzi, którzy kupili sobie prywatne komputery profesjonalne. Dali jednak zarobić komuś innemu, bo nawet PEWEX nie prowadzi tego towaru. Łatwo też sobie wyobrazić, że gdyby nawet prowadził, to ceny byłyby równie absurdalne, jak ceny samochodów, a i cło szybko by zaczęto — motywując to zdrowym rozsądkiem — liczyć od sztuki. Myślę, że wynosiłoby np. 1000 zł od sztuki pamięci 1 KB... Więc cieszmy się, że na razie płacimy od kilograma; na szczęście mikrokomputery są lekkie.

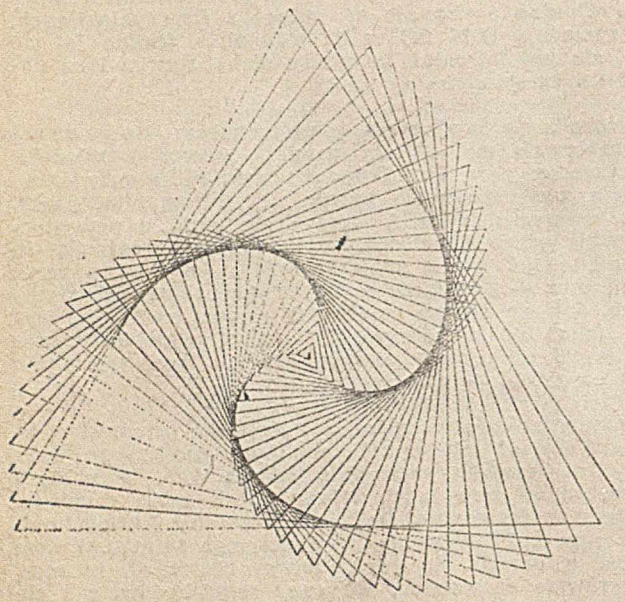
Cóż jednak z potrzebami profesjonalistów? Ano, będzie po staremu. Ostatecznie Stradivariusów też się w Polsce nie sprzedaje (podobno nawet ze zwykłymi skrzypcami są kłopoty), a mamy tylu wspaniałych skrzypków. Więc można zakazywać użycia modemów, można nie tworzyć baz danych; nie usłyszymy o skomputeryzowanej informacji naukowej ani też nie będziemy załatwiać spraw bankowych przez komputer. Nożyce i tak same opadną. Wtedy okaże się, że komputer to artykuł luksusowy i społeczeństwo nie może do posiadacza takowego dopłacać. Można też wymyślić kartki na energię elektryczną oraz wprowadzić obowiązkowe ubezpieczenie komputerów. Na koniec (polecam ten sposób decydentom leniwym) wystarczy odciąć dopływ części zamiennych i po herbacie.

A tak naprawdę to fajnie jest. SPECTRUM leży w koszach od bielizny, jeden facet w Warszawie ma już ponad 500 gier programów, a w przyszłym roku zaspokoimy potrzeby profesjonalistów komputerami COMPAN. O ile oba będą działały.

JAKUB TATARKIEWICZ







Znajdujemy punkt o współrzędnych biegunowych (R, α). Z tego punktu rysujemy odcinek do punktu (2R, 2α), następnie z punktu (2R, 2α) do (3R, 3α) itd., n razy. Kształt figury zależy od parametrów α, R oraz n. Przybliżony zakres parametrów:

$0{,}5 \text{ rad} < \alpha < 3 \text{ rad}$

$50 < n < 350$

$R = \frac{d}{2n}$, gdzie d jest szerokością rysunku, który chcemy otrzymać.

Pomysł został zaczerpnięty z książki R. Millera i in.: Graphic Display System GD3 — User's Guide.

PIOTR ZIELCZYŃSKI