mikroKLAN 5

5 1984

informatyka

FORTH
Robotyka
Zagrożenia informatyki

WYDAWNICTWO NOT SIGMA

informatyka

5 1984

Nr 5
Miesięcznik Rok XIX
Maj 1984

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

Materiałów nie zamówionych redakcja nie zwraca

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. 2084. Obj. 4,0 ark. druk. Nakład 4000 egz. T-43.

INDEKS 36124

Cena egzemplarza zł 75,—
Prenumerata roczna zł 900,—

NACZELNA ORGANIZACJA TECHNICZNA WYDAWNICTWO CZASOPISM I KSIĄŻEK TECHNICZNYCH SIGMA

W NUMERZE:

W NAJBLIŻSZYCH NUMERACH:

- Roman Żelazny o narzędziach inżynierii oprogramowania
- Marek Rakowski i Andrzej T. Rosiński o jednostrukturowych mikrokomputerach 8-bitowych
- Roman Trechciński o systemie MULTIBUS-II
- Krzysztof Rzymkowski o systemie modularnym VME
- Marek Pawłowski i Andrzej Woźniak o programatorze PROG-2
- Jerzy Dworzecki o bibliotece oprogramowania CP/M
- Ryszard Rybus o oprogramowaniu systemu mikroprocesorowego bez pamięci masowej
- Zbigniew Poznański i Jerzy Dańda o języku PL/M

WOJCIECH TROJNIAR
Zakład Systemów Automatyki Komputerowej
Polska Akademia Nauk
Gliwice

FORTH — język i system programowania (1)

FORTH został opracowany przez Charlesa Moore'a podczas prac programowych w dziedzinie astronomii. Moore stwierdził, że programowanie w takich językach jak ALGOL czy FORTRAN zabiera zbyt wiele czasu w stosunku do tego jaki uważałby za niezbędny. FORTH rozwijał się w ciągu kilkunastu lat praktyki programowej. Obecnie łączy on w sobie język wysokiego poziomu, język asemblera, system operacyjny z edytorem i zbiór procedur uruchamiania programów. Całość tworzy jednolite środowisko komputerowe, którego elementami użytkownik operuje według jednolitych reguł syntaktycznych.

Język FORTH ma cechy strukturalnego języka wysokiego poziomu. Postać języka jest kompromisem między elegancją języka opisu algorytmów, jak np. PASCAL, a prostotą implementacji oraz wygodą przy operowaniu jego elementami. W implementacji języka wykorzystano technikę interpretacji, kompilacji i języka pośredniego. Dzięki temu, możliwe jest programowanie na dowolnym poziomie oddalenia obiektów języka od języka maszynowego. Użytkownik może pisać programy w języku FORTH-asemblera lub w języku obiektów utworzonych według pewnych reguł syntaktycznych.

Obiekty języka, nazywane słowami, pełnią rolę analogiczną do instrukcji w innych językach programowania. Każdemu słowu odpowiada nazwa i przyporządkowany jej program. Użytkownik może wywołać wykonanie programu słowa lub zdefiniować nowe słowo przy użyciu słów już istniejących w systemie. W pierwszym przypadku mówimy o interpretacji, a w drugim o kompilacji słowa. Połączenie techniki interpretacji i kompilacji daje dużą elastyczność przy posługiwaniu się komputerem w trybie interakcyjnym. Programowanie polega na definiowaniu kolejnych słów aż do momentu, kiedy program przyporządkowany kolejnemu słowu jest programem pożądanym.

Wykonanie programu następuje po przesłaniu nazwy odpowiedniego słowa z klawiatury lub wywołaniu — za pomocą innych słów — interpretacji danego słowa z dysku. Czas wykonywania programu w języku FORTH dla pewnych typów mikroprocesorów jest o 20—75% dłuższy, niż czas wykonywania analogicznego programu napisanego w języku asemblera. Słowa krytyczne pod względem czasowym mogą być zdefiniowane w języku asemblera. Pamięć operacyjna potrzebna do przechowywania programu w języku FORTH jest na ogół mniejsza od pamięci potrzebnej dla analogicznego programu asemblerowego, ponieważ program wynikowy jest w postaci kodu nizanego (ang. threaded code) [1]).

Zbiór słów, nazywany słownikiem, odpowiada zbiorowi instrukcji w innych językach programowania. Repertuar słów w słowniku pozwala tworzyć strukturalne programy za pomocą wyspecjalizowanych słów-kompilatorów oraz — programować własne kompilatory za pomocą słów-metakompilatorów. W charakterze tworzonych słów można używać własnych nazw, zbliżając w ten sposób tekst programu do terminologii danej dziedziny zastosowań. FORTH może zatem stanowić bazę do projektowania języków problemowych.

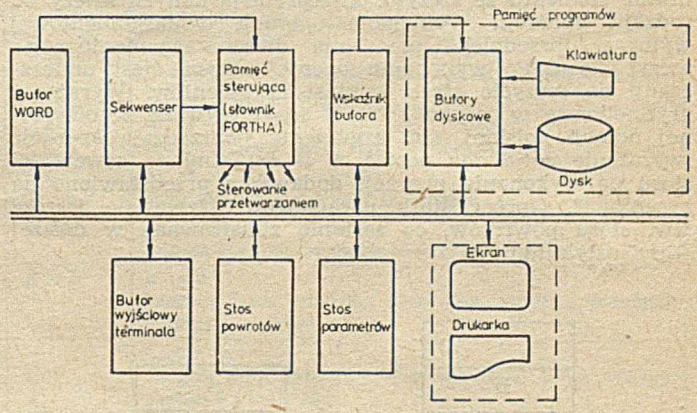
System operacyjny FORTHA zarządza terminalem, dyskiem i drukarką [2]). Pamięć dyskowa wraz z buforami dyskowymi w pamięci operacyjnej tworzy pamięć wirtualną. Programy operacji wejścia-wyjścia umożliwiają wczytanie znaku lub linii z klawiatury oraz wyprowadzenie tekstu na ekran. Do przygotowywania programu i danych służy edytor ekranowy.

W celu zapewnienia przenośności oprogramowania dla różnych implementacji, zdefiniowano standardowe zbiory słów. Obecnie istnieją dwa takie zbiory: fig-FORTH i FORTH-79. Pierwszy z nich został opracowany przez FORTH Interest Group. Grupa ta rozpowszechnia podręcznik „fig-FORTH Installation Manual" oraz tekst programu w językach asemblera dziewięciu mikroprocesorów. Standard FORTH-79 jest opracowany przez FORTH Standard Team.

Obecnie FORTH jest zaimplementowany na większości systemów mikrokomputerowych. Znalazł on zastosowanie w takich dziedzinach, jak: sterowanie procesami technologicznymi, medycyna, systemy zbierania danych itp. Z mojej praktyki wynika, że FORTH jest doskonałym narzędziem do pisania programów testujących aplikacyjne systemy mikrokomputerowe, ponieważ umożliwia łatwą ich modyfikowalność i rozszerzalność.

MASZYNA FORTH

Słowa i reguły ich wyboru tworzą środowisko, w którym są wykonywane programy. Środowisko to będziemy dalej nazywać maszyną FORTH. Funkcjonowanie procesora wirtualnego, jakim jest maszyna FORTH, można przedstawić w podobny sposób, jak przedstawia się funkcjonowanie procesorów maszyn cyfrowych (rys. 1). Elementy wirtualne są połączone magistralą, którą przepływa informacja. Większość przedstawionych elementów ma swoje odpowiedniki w typowym procesorze mikroprogramowanej maszyny cyfrowej (tab. 1). Pamięć sterująca zawiera słownik, tj. zbiór słów wraz z programami, wykonywanymi po wywołaniu słowa.



Rys. 1. Konfiguracja maszyny FORTH

[1]) Por. Kott R., Magdzik D.: Techniki interpretacji dla mikrokomputerów. INFORMATYKA, nr 4, 1984

[2]) Tradycyjny podział na język, system operacyjny itd., ma w przypadku FORTHA tylko porządkujące znaczenie. Wynika on z dotychczasowych sposobów implementacji systemów komputerowych. Zwykle kompilatory języków wysokiego poziomu oraz systemy operacyjne są rozróżnianymi częściami oprogramowania. W zasadzie FORTH nie wprowadza takiego podziału, dlatego mówiąc o języku lub systemie operacyjnym odwołujemy się do znaczeń tych terminów, jakie występują w tradycyjnych systemach komputerowych.

Tabela 1. Analogie między procesorem mikroprogramowanej maszyny cyfrowej a maszyną FORTH

Maszyna cyfrowa	Maszyna FORTH
Rozkaz	Słowo
Dekoder rozkazów	Interpreter zewnętrzny
Mikroprogram	Program słowa
Lista rozkazów	Słownik
Pamięć programu	Bufor wejściowy terminala dysku
Program w pamięci operacyjnej	Tekst umieszczony w buforze
Sekwenser mikroprogramu	Interpreter wewnętrzny
Rejestry	Stos parametrów
Stos śladów mikrorozkazów	Stos powrotów
Rozszerzanie listy rozkazów	Kompilacja nowego słowa
Licznik operacji	Wskaźnik bufora terminala lub dysku

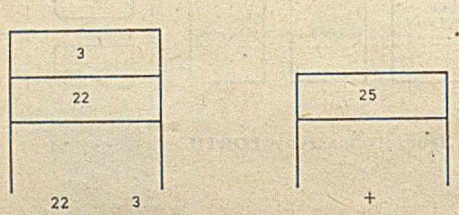
Program w języku FORTH jest ciągiem znaków ASCII, które mogą być oddzielone spacjami lub znakiem kończącym linię. Program rezyduje w pamięci dyskowej lub pamięci operatora korzystającego z terminala. Po zainicjowaniu systemu, maszyna FORTH oczekuje na linię tekstu z terminala, zakończoną znakiem CR. Wprowadzony tekst jest programem poddawanym interpretacji. W szczególności, ciąg znaków będący słowem jest interpretowany przez procesor wirtualny. W tym celu przesyła się go do oddzielnego bufora, zwanego WORD.

Załóżmy, że w buforze terminala znajduje się tekst:

VLIST FORTH

oznaczający wyprowadzenie zawartości słownika (wskaźnik bufora wskazuje literę V). Po przesłaniu ciągu znaków z bufora terminala do bufora WORD, wskaźnik przesuwa się za słowo VLIST, a w buforze WORD znajduje się tekst VLIST. Następnie program zwany interpreterem zewnętrznym pobiera słowo z bufora WORD (w tym przypadku VLIST) sprawdza czy w pamięci sterującej istnieje słowo identyczne i wywołuje wykonanie programu tego słowa. Po zakończeniu wykonania programu pojedynczego słowa, maszyna FORTH pobiera kolejny ciąg znaków z bufora, Gdy programy wszystkich słów bieżącej linii zostaną wykonane, następuje wczytanie kolejnej linii tekstu z terminala. Proces ten jest wykonywany w nieskończonej pętli, tak jak wykonuje się cykl rozkazowy procesora maszyny cyfrowej. Rola sekwensera sterującego wywoływaniem programów kolejnych słów jest analogiczna do sekwensera maszyny mikroprogramowanej, którego zadaniem jest wybór następnego mikrorozkazu.

Operacje arytmetyczno-logiczne są wykonywane w maszynie FORTH na stosie. Stos jest strukturą danych, dla której określone jest przesłanie elementu na szczyt i pobranie elementu ze szczytu. Interpretacja danych zależy od rodzaju operacji. Wykonanie operacji dwuargumentowej wymaga uprzedniego przesłania dwóch argumentów na szczyt stosu. Po wykonaniu operacji, wynik jest umieszczany na szczycie stosu zamiast argumentów. Wyrażenia arytmetyczne w języku FORTH są zapisywane w odwrotnej notacji polskiej, tzn. argumenty poprzedzają operacje (wyrażenie **A+B** ma postać **A B +**). Zmianę zawartości stosu po wykonaniu operacji dodawania przedstawiono na rysunku 2. Oprócz stosu parametrów używa się jeszcze tzw. stosu powrotów, co zostanie zilustrowane w dalszej części artykułu.



Rys. 2. Zmiana zawartości stosu parametrów podczas interpretacji tekstu "22 3 +"

Interpreter zewnętrzny może być w jednym z dwóch trybów pracy: „interpretacja" (co opisano powyżej) lub „kompilacja". W tym drugim trybie następuje kompilowanie programu nowego słowa wzbogacającego repertuar słów znajdujących się w słowniku. Poniżej przedstawiamy przykładową definicję słowa. Zakładamy, że tekst w buforze terminala ma postać:

: KWADRAT-PRZECIWPROSTOKĄTNEJ DUP *-SWAP DUP * + ;

W trybie „interpretacja" wykonywany jest program słowa ":", który tworzy w słowniku nową nazwę KWADRAT-PRZECIWPROSTOKATNEJ i zmienia stan interpretera zewnętrznego na stan „kompilacja". Kolejne słowo jest kompilowane, tzn. adres pośredni początku programu kompilowanego słowa jest wstawiany do pierwszego wolnego miejsca pamięci sterującej. Proces ten jest kontynuowany aż do napotkania słowa ";", które powoduje powrót interpretera zewnętrznego do stanu „interpretacja". Wynikiem tego procesu jest utworzenie w pamięci sterującej słowa KWADRAT-PRZECIWPROSTOKATNEJ o programie składającym się z ciągu programów odpowiadających słowom: **DUP * SWAP DUP * +**. Poszczególne operacje oznaczają (przy założeniu, że na szczycie stosu znajdują się kolejno dwie przyprostokątne):

DUP, skopiowanie liczby na szczyt stosu (przyprostokątna)

*, pomnożenie dwóch liczb ze szczytu stosu (kwadrat przyprostokątnej)

SWAP, zamiana dwóch liczb na stosie (druga przyprostokątna na szczycie)

DUP, skopiowanie drugiej przyprostokątnej na szczyt stosu

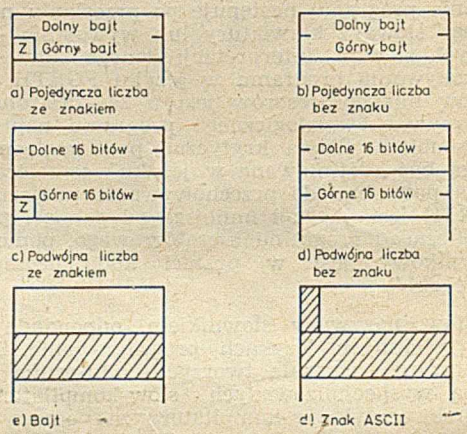
*, kwadrat drugiej przyprostokątnej na szczyt stosu

+, suma kwadratów na szczyt stosu.

Proces kompilacji można porównać z rozszerzeniem listy rozkazów maszyny mikroprogramowanej. Dokonuje się tego przez wpisanie do pamięci sterującej tej maszyny mikroprogramu, który będzie wykonywany po przesłaniu do rejestru rozkazów kodu nowego rozkazu.

TYPY DANYCH

W języku FORTH jawnie występują proste typy danych, jak znaki ASCII, liczby pojedynczej i podwójnej precyzji ze znakiem lub bez znaku oraz typy danych bez narzuconej interpretacji, takie jak bajt lub 16-bitowe słowo maszynowe. Format liczb zależy od implementacji. W systemach 8-bitowych liczba pojedyncza [3]) zajmuje dwa kolejne bajty w pamięci. W niektórych typach mikroprocesorów bajt zawierający bardziej znaczącą część liczby poprzedza bajt mniej znaczący. W innych typach mikroprocesorów kolejność jest odwrotna. Ułożenie bajtów na stosie dla poszczególnych typów danych w implementacji fig-FORTH INTEL-8080 przedstawiono na rysunku 3.



Rys. 3. Formaty danych na stosie (nieistotna część 16-bitowego słowa jest zakreskowana)

Zakresy liczb są następujące:

- liczba pojedyncza ze znakiem: —32768 .. + 32767
- liczba pojedyncza bez znaku: 0 .. 65535

[3]) Dla uproszczenia — zamiast „liczby pojedynczej i podwójnej precyzji", będziemy mówić — odpowiednio — „liczba pojedyncza i podwójna"

• liczba podwójna ze znakiem: —2 147 483 648 .. +2 147 483 648
• liczba podwójna bez znaku: 0 .. 4 294 967 295

Do oznaczania typów danych używa się następujących skrótów:

n — liczba pojedyncza ze znakiem
u — liczba pojedyncza bez znaku
d — liczba podwójna ze znakiem
ud — liczba podwójna bez znaku
c — znak ASCII
b — bajt
w — 16-bitowe słowo maszynowe
addr — adres komórki pamięci.

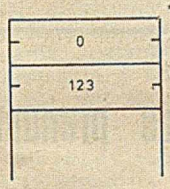
W języku FORTH nie przewidziano zmiennoprzecinkowej reprezentacji liczb. W tym formacie liczba jest przechowywana wraz z pozycją kropki (np. dziesiętnej), co umożliwia operowanie liczbami o dużym zakresie. W formacie stałoprzecinkowym pozycja kropki dziesiętnej nie jest przechowywana wraz z liczbą. Programy operujące tymi liczbami zakładają a priori położenie kropki, co wymaga od programisty kontrolowania formatu liczby po wykonaniu operacji. W zależności od zastosowań, konieczne jest skalowanie. W formacie zmiennoprzecinkowym natomiast skalowanie jest wykonywane automatycznie (tzw. normalizacja liczby). Dlaczego więc autorzy FORTHA nie przewidzieli reprezentacji zmiennoprzecinkowej? Argumentem, który się podaje, jest czas wykonania procedur zmiennoprzecinkowych. Mnożenie zmiennoprzecinkowe jest półtora raza, a dodawanie 3—10 razy wolniejsze od analogicznych operacji stałoprzecinkowych. Zakres zmienności liczb stałoprzecinkowych (±2 mld) jest wystarczający dla większości programów zbierania danych, sterowania procesami, transformacji sygnałów itp. Niektóre wersje FORTHA mają jednak w swoim słowniku operacje zmiennoprzecinkowe.

SŁOWNIK

Maszyna FORTH po zainicjowaniu wykonuje wyłącznie programy odpowiadające słowom języka. Aby w pełni przedstawić jej funkcjonowanie, omówimy semantykę tych słów oraz ich zastosowanie do tworzenia programów.

Operacje podstawowe

Liczby wprowadzane z terminala są dekodowane na kod dwójkowy i umieszczane na szczycie stosu. Jeżeli w ciągu znaków tworzących liczbę znajduje się kropka „.", to ciąg ten jest tłumaczony na liczbę podwójną. Przykładowo — po wprowadzeniu ciągu **+123** na szczycie stosu zostanie umieszczona liczba pojedyncza ze znakiem, natomiast po wprowadzeniu liczby **12.3** (a także **.123** lub **1.23** lub **123.**) szczyt stosu będzie miał postać:



Liczby można również przesyłać na szczyt stosu z pamięci operacyjnej (w terminologii maszyny FORTH to pamięć sterująca).

Jeżeli na szczycie stosu znajduje się adres komórki pamięci, to w miejsce tego adresu można przesłać zawartość dwóch kolejnych komórek pamięci o identycznym adresie, wykonując program słowa "*v*". Zmianę stanu stosu oznacza się przez podanie jego stanu przed wykonaniem i po wykonaniu operacji. Zmianę stanu stosu po wykonaniu programu słowa „*v*" zapisujemy w postaci:

addr → w

Nową wartość oznaczamy przez "w", ponieważ nie wiadomo jak ją interpretować.

Przesłanie dwóch bajtów ze szczytu stosu do pamięci wykonuje się operacją "!" i zapisuje [4]) jako:

w addr →
Instrukcja przypisania w języku PASCAL:
B: = A;
w języku FORTH ma postać:
A *v* B !
gdzie **A** i **B** są nazwami zmiennych.

Tabela 2. Operacje arytmetyczno-logiczne

Oznaczenie	Opis działania
+	n1 n2 → n3 Dodanie dwóch liczb całkowitych ze znakiem
D+	d1 d2 → d3 Dodanie dwóch liczb całkowitych podwójnych długości ze znakiem
*	n1 n2 → n3 Mnożenie dwóch liczb całkowitych
U*	u1u2 → ud Mnożenie dwóch liczb całkowitych bez znaku; wynik jest w postaci podwójnej liczby bez znaku
M*	n1 n2 → d Mnożenie dwóch liczb całkowitych ze znakiem; wynik jest w postaci liczby całkowitej podwójnej długości ze znakiem
/	n1n2 → n3 Dzielenie dwóch liczb całkowitych bez znaku, n1 przez n2
U/	ud u1 → u2 u3 Dzielenie liczby o podwójnej długości bez znaku przez liczbę o pojedynczej długości bez znaku; wynikiem jest reszta z dzielenia u2 i iloraz u3
M/	d n1 → n2 n3 Dzielenie liczby o podwójnej długości ze znakiem przez liczbę o pojedynczej długości ze znakiem; wynikiem jest reszta z dzielenia n2 i iloraz n3
MOD	n1 n2 → n3 Reszta z dzielenia dwóch liczb pojedynczej długości ze znakiem, ma ten sam znak co liczba n1
MINUS	n1 → n2 Zmiana znaku liczby całkowitej
DMINUS	d1 → d2 Zmiana znaku liczby całkowitej o podwójnej długości
ABS	n → n Zmiana liczby pojedynczej długości ze znakiem na jej wartość bezwzględną
DABS	d → d Zmiana liczby podwójnej długości ze znakiem na jej wartość bezwzględną
MIN	n1 n2 → n3 Pozostawienie na szczycie stosu mniejszej z dwóch liczb pojedynczej długości ze znakiem
—	n1 n2 → n3 Pozostawienie na szczycie stosu różnicy n1—n2
MAX	n1 n2 → n3 Pozostawienie na szczycie stosu większej z dwóch liczb pojedynczej długości ze znakiem
AND	n1 n2 → n3 Iloczyn logiczny dwóch słów 16-bitowych
OR	n1 n2 → n3 Suma logiczna dwóch słów 16-bitowych
XOR	n1 n2 → n3 Różnica symetryczna dwóch słów 16-bitowych
S →D	n → d Zamiana liczby całkowitej ze znakiem na liczbę podwójnej długości ze znakiem
+!	n addr → Dodanie liczby całkowitej n do zawartości dwóch komórek pamięci o adresie addr
1+	n1 → n2 Zwiększenie o 1 wartości na szczycie stosu
+—	n1 n2 → n3 Przypisanie liczbie n1 znaku liczby n2
D+—	d1 d2 → d3 Przypisanie liczbie d1 znaku liczby d2
*/	n1 n2 n3 → n4 Operacja n4 = n1*n2/n3

[4]) Oba elementy (w, addr) zostają usunięte ze stosu, dlatego miejsce po prawej stronie strzałki jest puste — wartość wskaźnika stosu zmniejsza się o 2

W języku FORTH można deklarować zmienne, gdyż każda zmienna jest słowem, którego program przesyła na szczyt stosu jej adres. Powyższy program przesyła na szczyt stosu adres zmiennej **A**, a następnie w miejsce adresu na szczycie stosu umieszcza wartość tej zmiennej, po czym przesyła na szczyt stosu adres zmiennej **B**. Z kolei program słowa "!" przepisuje wartość zmiennej **A** pod adres zmiennej **B** ze szczytu stosu.

Wyrażenia arytmetyczne w języku FORTH muszą być napisane w odwrotnej notacji polskiej. Przykładowo — wyrażenie:

A : = (B+C) * D;

ma postać

B ▫ C ▫ + D ▫ * A !

Liczby znaków potrzebnych do zapisu wyrażeń w obu notacjach mogą się różnić. Zapis wyrażeń w odwrotnej notacji polskiej jest wygodny dla maszyny cyfrowej, ponieważ uporządkowania argumentów i operacji dokonuje użytkownik. Kolejność operacji jest zgodna z kolejnością ich wykonywania przez interpreter.

Operacje arytmetyczno-logiczne

Dla liczb pojedynczych ze znakiem zdefiniowano podstawowe działania arytmetyczne oraz szereg innych operacji, jak: zmiana znaku, obliczanie wartości bezwzględnej, obliczanie wartości minimalnej lub maksymalnej dwóch liczb i obliczanie reszty z dzielenia. Na liczbach podwójnych wykonuje się operacje dodawania, zmiany znaku i obliczanie wartości bezwzględnej.

W języku FORTH istnieje szereg operacji złożonych, które można zastąpić ciągiem operacji elementarnych. Przykładowo — operację "*/" można zastąpić ciągiem operacji "* /". Zwykle operacji złożonych używa się do częściej stosowanych obliczeń, np. operacji "*/" można używać do obliczania procentów — definiując słowo "%":

: % 100 */ ;

Po wprowadzeniu z terminala ciągu:

500 5 %

otrzymamy wynik równy 5% liczby 500 czyli 25. Spis podstawowych operacji arytmetyczno-logicznych przedstawiono w tabeli 2.

Operacje stosowe

Stos parametrów jest przeznaczony do tymczasowego przechowywania danych. W celu efektywnego operowania elementami stosu, w języku FORTH przewidziano zestaw słów przeznaczonych do manipulacji na stosach. Rozważmy instrukcję podnoszenia do sześcianu:
B := A * A * A ;

Początkowo przesyłamy na szczyt stosu wartość zmiennej **A**, powielamy dwukrotnie jej wartość przy użyciu słowa "DUP", wykonujemy dwukrotnie mnożenie i przesyłamy otrzymaną wartość do zmiennej **B**. Program w języku FORTH będzie miał zatem postać:

Tabela 3. Operacje stosowe

Oznaczenie	Opis działania
OVER	n1 n2 → n1 n2 n1
DROP	n →
SWAP	n1 n2 → n2 n1
DUP	n → n n
2DUP	n1 n2 → n1 n2 n1 n2
ROT	n1 n2 n3 → n2 n3 n1
—DUP	n → n n? Powtórzenie następuje tylko wtedy, gdy n jest liczbą ujemną
SPa	→addr Załadowanie wskaźnika stosu na szczyt stosu
SP!	Inicjowanie wskaźnika stosu wartością zmiennej Sø
!CSP	Zapamiętanie wskaźnika stosu w komórce CSP
RP!	Inicjowanie wskaźnika stosu powrotów wartością zmiennej Rø
RPa	→addr Załadowanie wskaźnika stosu powrotów na szczyt stosu danych
>R	stos powrotów → n stos danych n → Załadowanie zawartości szczytu stosu danych na szczyt stosu powrotów
R>	stos powrotów n → stos danych → n Załadowanie zawartości szczytu stosu powrotów na szczyt stosu danych
R	stos powrotów n→n stos danych → n Załadowanie zawartości szczytu stosu powrotów na szczyt stosu danych; zawartość szczytu stosu powrotów pozostaje bez zmian

A ▫ DUP DUP * * B !

Pełny zbiór słów służących do przeprowadzania manipulacji na stosach przedstawiono w tabeli 3.

Niekiedy wygodnie jest przechowywać dane przez pewien czas na stosie powrotnym. Instrukcja obliczania wartości trójmianu kwadratowego:

D = A * X ↑ 2 + B * X + C;

przy wykorzystaniu stosu powrotnego w języku FORTH ma postać:

X ▫ > R A ▫ R * B ▫ + R > * C ▫ + D !

*

W następnej części artykułu zostaną przedstawione pozostałe słowa FORTHA i sposób ich użycia w programach.

CEZARY ZIELIŃSKI
Instytut Automatyki
Politechnika Warszawska

Roboty (1)

Konstrukcja i możliwości zastosowań

Jeszcze względnie niedawno słowo robot kojarzyło się z fantastyczną wizją społeczeństwa przyszłości. Dzisiaj natomiast weszło już na trwałe do słownictwa inżynierów i techników. Pod pojęciem tym rozumiemy łatwo dające się przeprogramować urządzenie techniczne mające zdolności manipulacyjne oraz charakteryzujące się pewną autonomią działania w danym środowisku. Powszechne zainteresowanie robotyką spowodowane jest możliwościami jej zastosowań. Manipulatory [1]) nadają się szczególnie do usprawnienia mało- i średnioseryjnej produkcji, a uniwersalność oraz łatwość programowania czyni je doskonałym środkiem zwiększenia wydajności pracy w wytwórczości tego typu.

Liczne przykłady świadczą o coraz silniejszym związku robotyki i informatyki. Chcąc przybliżyć Czytelnikom tę tematykę, rozpoczynamy druk cyklu artykułów obejmujących obie dziedziny. Na wstępie zajmiemy się wybranymi metodami programowania robotów oraz przedstawimy przykładowe zastosowania tych urządzeń. Kolejny artykuł zaznajomi Czytelników z zagadnieniami sterowania maszynami kroczącymi [2]). Ponadto znajdzie miejsce przegląd tekstowych języków programowania robotów — poczynając od najprostszych, a kończąc na językach wykorzystujących metody sztucznej inteligencji.

Kiedy w 1962 roku dwie amerykańskie firmy UNIMATION i AMF VERSATRAN wypuściły na rynek swoje roboty przemysłowe, zainteresowanie nimi było nikłe. Niemniej w 1967 roku Japończycy zakupili kilka sztuk Versatran'ów. Już w rok później firma KAWASAKI HEAVY INDUSTRIES wyprodukowała na licencji UNIMATION pierwsze manipulatory. Nie trzeba było długo czekać, aby Japonia stała się czołowym producentem robotów. Największy popyt na narzędzia tego typu wykazuje przemysł samochodowy. Przemysł elektromaszynowy zajmuje drugie miejsce. Roboty stosuje się najczęściej do czynności transportowych, zgrzewania, spalania, malowania natryskowego, rzadziej do polerowania i zdejmowania naddatków z odlewów. Oczywiście nie jest to pełna lista zastosowań. Stosuje się je nawet do tak wyrafinowanych przedsięwzięć jak organoleptyczna kontrola wnętrz rur zasilających reaktory atomowe w wodę.

Niektóre zastosowania wymagają bardzo prostych manipulatorów. Inne potrzebują zarówno złożonej części manipulacyjnej, jak i specyficznego systemu sterowania. Złożoność robota, a w szczególności jego układu sterowania, wpływa bezpośrednio na stopień autonomii działania w otaczającym środowisku. W początkowej fazie ewolucji układy sterujące były czysto sprzętowe. Obecnie, szczególnie w bardziej złożonych robotach, większość funkcji sterowania została przejęta przez oprogramowanie. Spróbujemy prześledzić rozwój tej tendencji na tle kolejnych generacji robotów.

GENERACJA 1

Generacja 1 jest zarówno najstarsza, jak i najliczniejsza. Do niej zaliczane są wszystkie manipulatory nie posiadające sprzężenia zwrotnego z otoczeniem. Oznacza to, że roboty nie mają czujników dostarczających układowi sterowania informacji o zmianach zachodzących w środowisku. Brak sprzężenia zwrotnego powoduje konieczność precyzyjnego określenia wszystkich ruchów części manipulacyjnej robota. Jakakolwiek zmiana stanu środowiska, nie przewidziana w programie, uniemożliwi dalsze wykonywanie zadanych czynności.

Sposób programowania robotów tej generacji jest różny, w zależności od stopnia złożoności układów sterowania oraz napędów. Układy sterujące najprostszych robotów wyznaczają jedynie kolejność włączeń napędów. Uruchomiony napęd powoduje przemieszczenie ramienia do chwili natrafienia na ogranicznik krańcowy lub wyłącznik drogowy. Oznacza to, że fragmenty ramion mogą zatrzymać się tylko w punktach krańcowych danej osi ruchu. Każda oś ma dwa takie punkty, a więc dla robota o n stopniach swobody jego chwytak można zatrzymać w 2^n punktach przestrzeni roboczej. Ze względu na prostotę realizacji układu sterowania, w każdym kroku programu włączony jest tylko jeden napęd. Czas trwania każdego ruchu zależny jest od jego zakresu oraz prędkości wymuszonej przez napęd. Przejście do kolejnego kroku programu następuje w momencie uzyskania sygnału z wyłącznika drogowego, od urządzenia zewnętrznego lub w wyniku upływu określonego czasu.

Sterowanie tego typu najwygodniej zrealizować za pomocą matrycy diodowej. Zadanie programisty ogranicza się tu do umieszczenia kołków na przecięciu odpowiednich wierszy, wyznaczających kolejne kroki programu, oraz kolumn określających uruchamianie urządzenia (np. napęd lub czasomierz), a także na ustawieniu położeń ograniczników krańcowych lub wyłączników drogowych. Taki sposób programowania zastosowano w polskim robocie pneumatycznym RIMP-401.

W przypadku robotów, w których zakres ruchu wyznaczany jest przez nastawienie potencjometrów wartości zadanej, stosuje się układy sterowania z programatorami bębnowymi. W tym przypadku bęben z umieszczonymi w nim kołkami obraca się ze stałą prędkością. Kolejne szeregi kołków, natrafiając na mikrowyłączniki, powodują dołączanie napięcia z potencjometrów wartości zadanej do wejść układów regulacji położenia poszczególnych stopni swobody ramienia lub przekazanie sygnałów do urządzeń zewnętrznych. Ponieważ prędkość obrotowa bębna jest stała, czas realizacji każdego kroku programu jest taki sam. Zastosowanie potencjometrów zamiast ograniczników krańcowych lub wyłączników drogowych ułatwia ustalenie zakresu

[1]) Słowa robot i manipulator będą tu używane zamiennie, aczkolwiek niektórzy autorzy odróżniają te pojęcia
[2]) Uwaga o zamienności odnosi się także do terminów maszyna krocząca, robot kroczący i pedipulator

Mgr inż. CEZARY ZIELIŃSKI ukończył studia w 1982 roku na Wydziale Elektroniki Politechniki Warszawskiej. W tym samym roku rozpoczął studia doktoranckie również w Politechnice Warszawskiej na kierunku Automatyka i Informatyka. Zajmuje się zagadnieniami związanymi z robotyką, a w szczególności tekstowymi językami programowania robotów.

ruchu w danej osi oraz umożliwia zatrzymanie jej w tylu położeniach, ile jest potencjometrów wartości zadanej.

Podobnie jak poprzednio, programowanie polega tu na umieszczeniu kołków we właściwych otworach bębna oraz ustawieniu potencjometrów. Tego typu system sterowania zastosowano w robocie VERSATRAN 500P.

W obu powyżej opisanych przypadkach program jest liniową sekwencją czynności. Nie ma możliwości wykonywania skoków programowych lub instrukcji warunkowych. Poza tym dyskretna przestrzeń położeń uniemożliwia zastosowanie takich robotów do wielu zadań produkcyjnych.

Istnieje jednak bardzo wiele sytuacji, w których dyskretna przestrzeń położeń jest wystarczająca. Przykładem może być zastosowanie wspomnianego dwuramiennego robota RIMP-401 w zakładach ZELMOT do obsługi prasy w linii montażu cewek zapłonowych [6]. Manipulator ten pobiera cewkę z magazynku na taśmie wejściowej, wkłada ją pod prasę, wysyła sygnał powodujący uruchomienie prasy zagniatającej kołnierz cewki, a następnie odstawia ją na taśmę odbiorczą. Ponieważ RIMP-401 ma dwa ramiona, czynności pobierania cewek z magazynku oraz rozładowywania prasy mogą odbywać się równolegle, co wydatnie przyspiesza wykonanie zadania.

W ramach Generacji 1 mieszczą się również roboty odtwarzające (ang. playback robots). Są to manipulatory, które programuje się metodą uczenia. Uczenie polega na wprowadzeniu do pamięci układu sterowania współrzędnych kolejnych punktów trajektorii powstałej na skutek przemieszczania ramienia przez operatora. Implikuje to konieczność zastosowania pamięci typu zapis-odczyt. Początkowo wykorzystywano pamięć ferrytową, obecnie zaś — półprzewodnikową. Najczęściej stosuje się także dodatkową pamięć zewnętrzną (np. taśmę magnetyczną), co zwiększa możliwości zapamiętywania programów różnych ruchów.

Zmiana położenia ramienia w trakcie uczenia może być spowodowana jego ręcznym wodzeniem przez operatora (ang. walk through) lub w efekcie uruchomienia kolejnych napędów ramienia za pomocą pulpitu operatorskiego (ang. lead through). Obie możliwości stosowane są w punktowych systemach sterowania (PTP — Point To Point). W tym przypadku programowanie polega na ustawieniu, jedną z powyższych metod, ramienia w pożądanym położeniu, a następnie naciśnięciu przycisku wczytującego to położenie do pamięci. Natomiast odtworzenie zapamiętanego programu polega na przetworzeniu cyfrowo-analogowym położeń zapamiętanych w postaci cyfrowej i przekazanie ich do serwomechanizmów, jako wartości zada-

Istnieją również roboty o ciągłych systemach sterowania (CPC — Continous Path Control). Ich uczenie odbywa się metodą ręcznego wodzenia ramienia przez operatora, ale w tym przypadku punkty do zapamiętania nie są określone przez operatora, lecz przez układ czasowy. Układ ten wprowadza do pamięci punkty trajektorii ruchu po każdym ściśle określonym kwancie czasu. W ten sposób odtworzenie trajektorii jest o wiele wierniejsze. Dodać należy, że szybkość odtwarzania nie musi być równa prędkości uczenia. W ten sposób przejście po zadanej trajektorii może być zrealizowane ze zmniejszoną lub zwiększoną prędkością.

System sterowania punktowego zastosowano w szwedzkich robotach IRb-6 i IRb-60 firmy ASEA [3]. Są to manipulatory o pięciu lub sześciu stopniach swobody i udźwigu odpowiednio 6 i 60 kg. Dokładność pozycjonowania, wynosząca ± 0,2 mm, pozwala stosować je do bardzo precyzyjnych czynności. System sterowania oparto na mikroprocesorze INTEL 8008.

Programista, uczący robota IRb przemieszczania, ma do dyspozycji dodatkowe możliwości. Jedną z nich jest zdeterminowanie sposobu ruchu chwytaka lub innej głowicy specjalistycznej, pomiędzy kolejno zapamiętanymi punktami. Istnieje tu kilka możliwości, omówionych poniżej.

- **Pozycjonowanie dokładne.** Ruch wszystkich stopni swobody jest tu rozpoczynany jednocześnie. Natomiast czas przestawienia każdej osi jest różny. Wynika to z różnych zakresów ruchu poszczególnych stopni swobody. Realizacja tego rozkazu zostanie zakończona w momencie dokładnego osiągnięcia punktu docelowego.
- **Pozycjonowanie zgrubne.** W tym przypadku ruch rozpoczyna się jak poprzednio, ale wykonanie tego rozkazu zostaje przerwane w momencie gdy ostatni ze stopni swobody rozpoczyna hamowanie. W rezultacie, zanim cel zostanie osiągnięty rozpocznie się realizacja kolejnego rozkazu.
- **Pozycjonowanie prostoliniowe.** Programista specyfikuje tu czas przejścia między kolejno zapamiętanymi punktami. Czas przemieszczenia każdej z osi jest wówczas równy wyspecyfikowanemu. Ponadto na ruch prostoliniowy można nałożyć oscylacje. Jest to bardzo przydatne przy wkładaniu przedmiotów w otwory, zapobiega bowiem zakleszczaniu.
- Ponadto programista ma do dyspozycji **instrukcje oczekiwania przez określony czas lub czekania na określone zdarzenie, włączania i wyłączania wyjść binarnych oraz badania wejść binarnych.** Istnieją również **instrukcje skoków warunkowych i bezwarunkowych.** Wszystkie te instrukcje wraz z ich parametrami wprowadzane są do pamięci układu sterującego za pomocą programatora klawiszowego. Jedynym wyjątkiem są parametry instrukcji pozycjonujących ramię, odczytywane z przetworników położenia umieszczonych w ramieniu robota.

System sterowania ciągłego ruchem ramienia zastosowano w robotach norweskiej firmy TRALLFA. Urządzenie to ma napęd hydrauliczny, sześć stopni swobody oraz dokładność pozycjonowania rzędu ± 1 mm. Przeznaczone jest głównie do malowania natryskowego, aczkolwiek można je również stosować wszędzie tam, gdzie wymagane jest przejście po ściśle określonej trajektorii, np. przy spawaniu. Aby zaprogramować tego robota, należy ręcznie przesuwać jego ramię wzdłuż pożądanej trajektorii. System sterujący zapewnia pomiar położenia ramienia w odstępach o ściśle określonym kwancie czasu. Zmierzona wartość zapamiętywana jest na taśmie magnetycznej i przy odtwarzaniu stanowi wartość zadaną dla serwonapędów.

W Polsce również produkuje się roboty należące do klasy „playback". Przykładem może być RIMP-1000 produkowany przez Instytut Mechaniki Precyzyjnej w Warszawie. Robot ten ma punktowy system sterowania. Umożliwia on przestawienie ramienia do dowolnego punktu strefy roboczej, wysterowanie oraz sprawdzenie stanu dziewięciu dwustanowych urządzeń zewnętrznych, realizację oczekiwania przez czas od 0 do 99,9 sekund oraz rozejścia warunkowe w programie. Program może być nauczony przez zapamiętanie w pamięci ferrytowej kolejnych położeń ramienia oraz instrukcji pomocniczych. Pojemność pamięci wystarcza na 8128 kroków programowych. Możliwe jest również utworzenie biblioteki programów dla robota przez przesłanie ich do pamięci kasetowej.

Przykładem zastosowania tego manipulatora jest stanowisko produkcyjno-doświadczalne zgrzewania punktowego pasa tylnego nadwozia do samochodu Polonez [2]. RIMP 1000 wykonuje tu 36 zgrzein na elemencie zamocowanym w przyrządzie prepozycjonującym. Załadowanie i rozładowanie przyrządu odbywa się jednak ręcznie.

GENERACJA 1.5

W robotach opisanych dotychczas informacje układu sterowania dostarczane były jedynie przez wejścia binarne. Umożliwiało to uzyskanie sygnałów od urządzeń współpracujących, co z kolei zapewniło zsynchronizowanie ruchów z procesem technologicznym. Aby robot mógł funkcjonować w środowisku zmieniającym swój stan w sposób niedeterministyczny, konieczne są czujniki. Spełniają one rolę receptorów, umożliwiających zbieranie informacji o otoczeniu. Manipulatory wyposażone w czujniki zaliczane są do Generacji 1.5 i wyższych, w zależności od rodzaju zastosowanych czujników.

Do Generacji 1.5 zalicza się wszystkie roboty wyposażone w przetworniki sił i momentów, czujniki dotyku czy przetworniki zbliżeniowe. W urządzeniach tej generacji nie stosuje się jednak bardziej złożonych metod uzyskiwania informacji z otoczenia, takich jak np. metody rozpoznawania i przetwarzania obrazów. Roboty, te stosuje się wszędzie tam, gdzie precyzyjne prepozycjonowanie przedmiotów obrabianych jest niemożliwe lub nieekonomiczne.

Przykładem może być spawanie dwóch płyt metalowych o niezbyt precyzyjnie określonym łączu [5]. Aby wykryć położenie złącza, zastosowano detektor prądów wirowych. Detektor składa się z dwóch cewek nawiniętych na rdzeniach ferrytowych i umieszczonych tak, aby ich osie były równoległe do siebie i w przybliżeniu prostopadłe do powierzchni płyt. Obie cewki zasilane są z tego samego generatora wysokiej częstotliwości. W układzie sterującym następuje porównanie faz sygnału z generatora i różnicowego sygnału uzyskanego z cewek. Wartość różnicy faz jest proporcjonalna do impedancji cewek, która z kolei zależy od położenia cewek względem szczeliny między pły-

tami. Poruszając ramieniem manipulatora i kontrolując sygnał z detektora, można tak umiejscowić cewki, aby szczelina znajdowała się dokładnie między ich osiami. Oczywiście ruch aparatu spawalniczego jest sprzężony z ruchem cewek. W momencie ustawienia aparatu nad szczeliną można rozpocząć spawanie. Wyżej opisany czujnik umożliwi śledzenie szczeliny w trakcie wykonywania zadania, a więc wytyczenie trajektorii spawu.

Przykładem może być też zastosowanie robota, z przetwornikami sił i naprężeń umieszczonymi w „nadgarstku", do wkładania trzpieni w otwory. Ruch manipulatora jest tu zależny od sygnałów otrzymywanych z przetworników.

Jak pokazują przytoczone przykłady, układy sterowania Generacji 1.5 mają dodatkowe, w porównaniu z Generacją 1, układy przetwarzania sygnałów oraz układy decyzyjne określające akcje podejmowane przez robota w zależności od uzyskanych informacji. Układy decyzyjne najczęściej realizowane są na mikrokomputerach, a więc na drodze programowej.

GENERACJA 2

Kolejną generacją robotów jest Generacja 2. Główną cechą tych robotów jest ograniczona zdolność rozpoznawania przedmiotów w ich bezpośrednim otoczeniu oraz możliwość klasyfikacji niezbyt złożonych sytuacji. Ponieważ w urządzeniach tych zaistniała konieczność zastosowania metod rozpoznawania i przetwarzania obrazów, są one zazwyczaj sprzężone z komputerem.

Przykładem zastosowania robotów tej generacji jest wyjmowanie z kuwety bezładnie wrzuconych elementów [1]. Zadanie to zostało zdekomponowane na trzy części. Pobranie elementu z kuwety, wyliczenie jego orientacji w „dłoni" oraz przeniesienie go do celu. Manipulator jest wyposażony w system wizyjny zastępujący mu wzrok. Dostarcza on informacji na temat zawartości kuwety. Układ sterowania wylicza, który z elementów jest najłatwiejszy do pobrania, po czym następuje przyssanie tego elementu przez chwytak pneumatyczny. Układ sterujący jest zrealizowany na minikomputerze.

Roboty obu powyższych generacji zaczynają stopniowo wkraczać do hal fabrycznych. Najwięcej z nich znajduje się jednak w laboratoriach badawczych. Dla większości przeznaczone są specjalne sposoby programowania. Coraz częściej stosuje się tu tekstowe języki programowania, ponieważ tylko one umożliwiają przejrzystą dokumentację programu, ułatwiają opracowanie planu dla manipulatora działającego w niedeterministycznym środowisku (zaprogramowanie wielowariantowego działania) oraz — w przypadku zastosowania w przemyśle — skrócenie do minimum przerw w produkcji spowodowanych przeprogramowaniem warunkującym rozpoczęcie wytwarzania innego asortymentu elementów.

GENERACJA 2.5

Najbardziej zaawansowana technicznie jest Generacja 2.5. Obecnie roboty tej generacji znajdują się w fazie prac badawczych. Charakteryzują się one zdolnością rozpoznawania złożonych kształtów i umiejętnością klasyfikacji złożonych sytuacji.

Przykładem urządzenia tej generacji może być HIVIP Mk1 firmy HITACHI [7]. Robot ten ma ramię o siedmiu stopniach swobody oraz system wizyjny składający się z dwóch kamer. Przeznaczony jest do montażu pewnej klasy konstrukcji mechanicznych. Zaprogramowanie jego czynności polega na pokazaniu rysunku montażowego [4]. Jedna z kamer przeznaczona jest do identyfikacji elementów rozrzuconych na stole montażowym. Niemniej jednak system ten ma pewne ograniczenia. Chociaż elementy mogą mieć dowolne położenie, to jednak muszą znajdować się na kontrastującym tle oraz być obiektami o wielokątnych rzutach prostopadłych. Całość zadania rozbita jest na wiele faz o różnych czasach realizacji. Przykładowo — rozpoznanie każdego obiektu trwa 50 sekund, rozpoznanie rysunku — 20, przetwarzanie obrazów — 240, podjęcie decyzji — 10, a montaż konstrukcji o średniej złożoności — 180.

Całość oprogramowania jest wykonana w języku FORTRAN i wymaga ponad 40 tys. słów pamięci. System ten został zaimplementowany na komputerze HITAC 7250. Jak widać z podanych czasów, montaż niezbyt złożonych konstrukcji trwa tu o wiele dłużej niż potrzeba człowiekowi na realizację takiego samego zadania. Niewątpliwie zastosowanie szybszych maszyn cyfrowych oraz bardziej wyrafinowanych algorytmów rozpoznawania obrazów i sterowania ruchem może znacznie przyspieszyć montaż, a więc uzasadnić ekonomiczność zastosowania takich manipulatorów w przemyśle.

* * *

Przegląd różnych generacji robotów, wraz z ich przykładowymi reprezentantami, pokazuje, że oprogramowanie zaczyna odgrywać coraz większą rolę w rozwoju robotyki. Mam tu na myśli zarówno algorytmy sterowania ruchem, rozpoznawania postaci, przetwarzania informacji uzyskanej z czujników, jak i sposoby komunikacji użytkownika z robotem. Spośród przedstawionych metod określania zadań coraz więcej zwolenników zaczynają zdobywać języki programowania tekstowego.

LITERATURA

[1] Birk J. R., Kelly R., Martins H.: An orienting robot for feeding workpieces stored in bins. IEEE Transaction on Systems, Man and Cybernetics, Vol. SMC-11, No. 2, 1981

[2] Czajka T., Kubicki A.: Zastosowanie robota złożonego RIMP-1000 w procesie zgrzewania w FSO. Referaty XXIII Seminarium IMP, Warszawa, październik 1983

[3] Industrial Robot System. ASEA Information YB 110—301E, YFB, June 1977

[4] Lec M.: Research into flexible task performance in manipulator systems. Proceedings of the 8th International Symposium on Industrial Robots, Vol. 2, Stuttgart, 1978

[5] Marchal P., Cornu J., Detriche J.: Self Adaptive Arc-Welding Operation by Means of an Automatic Joint Following System. 4th Symposium on Theory and Practice of Robots and Manipulators, PWN, 1983

[6] Rudowski P.: Zastosowanie robota RIMP-401 w linii montażu cewek zapłonowych. Referaty XXIII Seminarium Instytutu Mechaniki Precyzyjnej, Warszawa, październik 1983

[7] Young J.: Robotics. Butterworths, 1973.

TADEUSZ SZUBA
Instytut Informatyki
Akademia Górniczo-Hutnicza
Kraków

PROLOG-PASCAL: automatyczna synteza programu dla obrabiarek

Ostatnio zaczynają się pojawiać systemy oprogramowania przeznaczone do zastosowań praktycznych, a zbudowane metodami sztucznej inteligencji. Są to głównie systemy projektowania wspomaganego komputerem (CAD), przeznaczone np. do automatycznej syntezy związków chemicznych czy automatycznych prac typu projektowego i konstrukcyjnego. Opierają się one na następującej zasadzie: **System komputerowy dysponuje ograniczoną zdolnością rozwiązywania problemów danego typu — poprzez budowę drzewa wariantów i zdefiniowanie strategii doboru ścieżki w drzewie, zorientowanej na dany proces technologiczny, i rokującej największe szanse rozwiązania problemu. Operator systemu obserwuje rozrost drzewa wariantów rozwiązań oraz stan zasobów systemu (zwłaszcza pamięci) i podejmuje decyzje o charakterze strategicznym. Owemu technologowi czy inżynierowi na ogół dość łatwo jest zablokować rozrost pewnych gałęzi w drzewie wariantów, a inne gałęzie skierować do natychmiastowej analizy przez system.**

Należy dodać, że operator pełni tu rolę ściśle nadzorczą, tj. działa w sposób pomocniczy i optymalizujący. Zakłada się jednak, że system komputerowy jest w zasadzie zdolny do samodzielnego rozwiązania problemu. Oczywiście, rozwiązanie takie może być gorsze, nieefektywne — np. trwające bardzo długo. W skrajnych przypadkach, nawet system może nie znaleźć rozwiązania.

Systemy tego rodzaju mają ogromne szanse — obok prac projektowych w zastosowaniach przemysłowych; wszędzie tam, gdzie wymagana jest produkcja jednostkowa lub małoseryjna. Problemy technologiczne prezentowane w tym artykule (montaż, demontaż, obróbka manipulatorami czy obróbka skrawaniem) — są stosunkowo łatwe do opisania i formalizacji dla konkretnego procesu technologicznego.

Przykładowo — dla montażu nadwozi samochodowych różnych typów można opracować wspólną strategię rozwiązywania problemów montażu karoserii. W takiej sytuacji system komputerowy na bieżąco syntetyzuje program, np. montażu elementów przesuwających się na taśmie produkcyjnej. Nadzór technologiczny spełnia zadania optymalizujące działanie systemu, lub interwencyjne — w momencie „logicznego zacięcia się systemu".

Na podobnej zasadzie mogą być konstruowane systemy czasu rzeczywistego, służące do sterowania procesami przemysłowymi o dużym stopniu nieoznaczoności. System komputerowy może być tak zbudowany, aby dysponował pewnymi „standardami sterowania". Natomiast w momencie pojawienia się sytuacji nieoczekiwanej, dokonywana jest automatyczna synteza sterowania procesem.

Przyjmuje się następujący schemat postępowania:
System analizuje sytuację za pomocą modelu symulującego proces technologiczny. Wychodząc z zaistniałej sytuacji, bada on różne postępowania, generując tym samym drzewo wariantów sterowań. Celem strategii „nałożonej" na to drzewo wariantów jest przeprowadzenie procesu technologicznego ze stanu zaistniałego (awaryjnego) do stanu docelowego (np. uspokojenie procesu). Operator procesu jest wzywany tylko wtedy, gdy system nie może sobie sam poradzić.

W niniejszym artykule przedstawione zostaną w zarysie rezultaty badań nad systemem służącym do automatycznej syntezy programu obróbki skrawaniem. Problem ten można postawić w sposób następujący:
Obrabiarka sterowana numerycznie dostaje zadany obiekt początkowy (surową bryłę metalu) i docelowy (kształt finalny przedmiotu). Przy użyciu znanego zespołu operacji przesuwania narzędzi oraz ruchów skrawających, system generuje drzewo wariantów procesu obróbki. Węzłami są kształty pośrednie zrealizowane na danej ścieżce. Strategia nałożona na drzewo ma tak ukierunkować obróbkę, aby osiągnąć kształt docelowy możliwie najszybciej.

Nie będzie tu wprowadzony element interwencji operatora. Automatyczna synteza następuje więc całkowicie samodzielnie.

Należy jeszcze omówić kwestie sprzętowe. Systemy, takie jak omawiany, tworzone są z myślą o małych komputerach dających się zainstalować np. w hali produkcyjnej lub w pokoju konstruktora. Pojemność pamięci powinna wynosić ponad 1 MB (co wynika z dotychczasowych doświadczeń symulacyjnych) — bowiem system operuje na drzewie wariantów, w którym węzły i gałęzie są etykietowane złożonymi strukturami danych. Maszyna powinna być wieloprocesorowa i umożliwiać obliczenie współbieżne.

OPIS OBRABIARKI

Obrabiarka użyta do badań symulacyjnych umożliwia:
— przesuwanie i obracanie obrabianego przedmiotu
— podnoszenie i przesuwanie zespołu narzędziowego
— obracanie imaka narzędziowego.
Imak narzędziowy może wybierać z magazynu dowolne narzędzia: frezy, wiertła, gwintowniki. Obrabiarka może więc dokonywać stosunkowo skomplikowanej obróbki obiektów.

Bardzo ważny problem stanowi opis obrabianego przedmiotu oraz narzędzi. Rzutuje to bowiem na struktury danych, a tym samym na efektywność całego procesu przetwarzania informacji. W badaniach przyjęto, że obrabiane obiekty są opisywane przy użyciu: prostopadłościanów (ze współrzędnymi trzech narożników), walców (z daną średnicą, długością, ustawieniem kątowym osi oraz współrzędnymi wyróżnionego punktu walca) oraz gwintów (opisanych tak jak walec, z podanym dodatkowo skokiem gwintu).

W pierwszym, prezentowanym tutaj etapie badań przyjęto, że obróbka zawsze rozpoczyna się od bloku metalu opisanego wysokością, szerokością i długością. W trakcie obróbki opis ten jest rozszerzany o wycięte prostopadłościany oraz o wywiercone i nagwintowane otwory.

Przedstawiona tu metoda jest tylko pozornie prymitywna, bowiem możliwość dowolnego trójwymiarowego usytuowania walców czy też prostopadłościanów pozwala na opisywanie nawet bardzo wyrafinowanych kształtów. Przykładowo — opis dosyć złożonego kółka zębatego zmieścił się na osiemnastu kartach perforowanych.

Jak już wspomniano, w opisie obrabiarki występują operacje przesuwania imaka narzędzi, pobierania odpowiedniego narzędzia oraz operacje obróbcze. Każda z nich jest opisywana w kategoriach:

<warunki stosowalności> → <rezultaty zastosowania>

Operacja wycinania prostopadłościanu frezem składa się — na przykład — z następujących elementów:
— policzenie rozmiaru freza potrzebnego do skrawania
— sprawdzenie czy taki frez jest w magazynie narzędzi
— obliczenie pozycji startowej do skrawania
— sprawdzenie gdzie jest aktualnie imak narzędzi

— ustawienie imaka w położenie startowe, co ze względu na kolizję często nie jest wykonywalne
— zamocowanie potrzebnego narzędzia w imaku
— redefinicja opisu obrabianego przedmiotu po obróbce
— redefinicja opisu położenia imaka narzędzi po obróbce.

Zasadniczy wpływ na spełnienie podstawowych warunków stosowalności ma dobór właściwego narzędzia, obliczenie pozycji startowej, a także bezkolizyjne ustawienie imaka w pozycji umożliwiającej rozpoczęcie skrawania.

Jak pokazały prace przy konstrukcji systemu — szczególne znaczenie ma problem kolizji. Bowiem najtrudniej systemowi jest tak manewrować obrabianym obiektem i imakiem narzędzi, aby przesunąć imak bezkolizyjnie z aktualnego położenia do położenia umożliwiającego skrawanie. Należy pamiętać, że imak musi być ustawiony także pod odpowiednim kątem. Ponadto w trakcie obróbki pewne fragmenty bloku metalu zostają wycięte i ruchy do tej pory kolizyjne — są już do wykonania.

Problem kolizji jest w istocie złożonym problemem obliczeniowym, nie dającym się realizować w klasycznym PROLOGU. Tak więc konieczne było utworzenie połączenia PROLOGU z PASCALEM, aby umożliwić obliczanie kolizji (a także innych problemów typu numerycznego w PASCALU) i transmitowanie wyników do PROLOGU.

Problem kolizji liczony był — z grubsza biorąc — w następujący sposób:

- Każda bryła ma swoją geometryczną reprezentację (w PASCALU, przy użyciu układów nierówności), pozwalającą stwierdzić, czy dany punkt leży wewnątrz tej bryły.
- Wzdłuż symulowanej trajektorii, z krokiem co 1 mm, sprawdzono czy punkty leżą wewnątrz obrabianej bryły, aż do momentu uzyskania kolizji albo dotarcia do końca trajektorii.
- Ponieważ obróbka zawsze prowadzi do zmniejszania rozmiarów bryły względem stanu początkowego, test był realizowany w sposób następujący:

— czy dany punkt leży wewnątrz bryły stanu początkowego (był to zawsze prostopadłościan — surowy blok metalu): jeśli nie — weź następny punkt na trajektorii; jeśli jest to koniec trajektorii — to STOP
— jeśli dany punkt leży wewnątrz bryły początkowej, to sprawdź, czy ten punkt leży wewnątrz któregoś z obszarów wyciętych z bloku; jeśli tak — kolizji nie ma i można posuwać się dalej po trajektorii; jeśli nie — kolizja zaistniała, czyli STOP
— procedura kończy się z chwilą pojawienia się kolizji albo końca trajektorii.

W praktyce okazało się, że procedura kolizji pochłaniała ogromną część czasu obliczeń.

POŁĄCZENIE PROLOG-PASCAL

Podczas konstruowania prezentowanego systemu okazało się, że brak jest narzędzia programowego do realizacji wszystkich wymaganych funkcji. PROLOG umożliwiał realizację tylko funkcji logicznych, tj. budowę drzewa wariantów i strategii jego przeszukiwania (też w nie wystarczającym — jak się potem okazało — stopniu). Nie można było natomiast zrealizować:
— obliczania czy zachodzi kolizja
— obliczania rozmiarów niezbędnych narzędzi
— obliczania położeń startowych dla obróbki
— obliczania kierunku ruchu.

Obliczenia takie wymagały bowiem aparatu ściśle numerycznego, takiego jakim dysponuje FORTRAN czy PASCAL, a nie logiczno-symbolicznego, którym dysponuje PROLOG. Fakt, że pewne wersje PROLOGU (w tym także ta używana do symulacji opisywanego systemu) są napisane w PASCALU, nie był specjalnym ułatwieniem, bowiem struktury programu PROLOGOWEGO są przechowywane w skomplikowany sposób w tablicach interpretera i są trudno dostępne z poziomu PASCALA.

Jedynym rozwiązaniem było zatem „spięcie" programu napisanego w PROLOGU, realizującego funkcje logiczne, i programu w PASCALU, realizującego funkcje numeryczne. W rezultacie powstał system hierarchiczny — z poziomem nadrzędnym w PROLOGU i PASCALOWYM poziomem podrzędnym, realizującym funkcje usługowe (kalkulator pracujący w trybie pytanie-odpowiedź).

Patrząc pod trochę innym kątem, zauważymy dość ciekawą sytuację:
— główna część systemu „myśli logicznie" w PROLOGU, natomiast na uboczu — w razie potrzeby — uruchamia podrzędne elementy do specjalnych obliczeń
— podrzędne fragmenty systemu symulują zadane podproblemy, np. w jakimś języku symulacyjnym
— podrzędne elementy przeprowadzają obliczenia numeryczne, np. w PASCALU.

Można sobie, oczywiście, wyobrazić inny dobór języków dla każdego z tych poziomów.

Idea połączenia PROLOG-PASCAL przedstawia się w sposób następujący: Jak już wspomniano — mimo że PROLOG w wersji dostępnej w systemie CYBER 72 był napisany w PASCALU, nie można było wpisywać wprost dodatkowych procedur numerycznych do wnętrza PROLOGU. Każda taka „wstawka" byłaby zbyt skomplikowana, poza tym wiadomo było z góry, że uruchomienie systemu wymagać będzie wielokrotnych zmian, modyfikacji, kasacji oraz dodawania nowych procedur numerycznych o trudnej do przewidzenia postaci. Każda procedura realizująca obliczenia numeryczne w PASCALU była w programie głównym (napisanym w PROLOGU) deklarowana jako:

+LIBRARY (<nazwa procedury>)

co powodowało odnotowanie tej nazwy w odpowiedniej tablicy. Tak więc obliczenia w ramach programu głównego, napisanego w PROLOGU, szły w sposób klasyczny — do momentu natknięcia się na wywołanie procedury zadeklarowanej wcześniej jako LIBRARY. W tym momencie następował STOP programu głównego, w PROLOGU (niestety obliczenia były jednoprocesorowe) oraz następowało wpisanie ciągu tworzącego nazwę procedury i listę parametrów aktualnych do odpowiedniego bufora. Następnie aktywizowana była część biblioteczna, realizująca (już w PASCALU) odczyt zawartości bufora i konwersję do odpowiedniej postaci. Kolejnym etapem było zidentyfikowanie za pomocą prostej struktury decyzyjnej, co należy w PASCALU policzyć.

Wyniki obliczeń były przekształcane do PROLOGOWEJ postaci standardowej oraz umieszczane w buforze powrotnym. Odpowiedź taka była czymś w rodzaju odzwierciedlenia pytania (tj. treści buforu wejściowego do PASCALA). Nazwa procedury i parametry aktualne były przepisywane bez zmian, natomiast wyniki obliczeń były podstawiane za symbole zmiennych.

Przykładowo — jeśli pytanie miało postać:

KOLIZJA(PROSTOPADŁOŚCIAN(10, 15, 20, 15.3, 18), ŁUK (30, 15, 15.3), *X, *Y, *Z)

i interesowało nas, gdzie ta bryła przetnie się z trajektorią opisaną łukiem, to odpowiedź miała np. postać:

+KOLIZJA(PROSTOPADŁOŚCIAN(10, 15, 20, 15.3, 18), ŁUK (30, 15, 15.3),10.3, 20.7, 18.3).

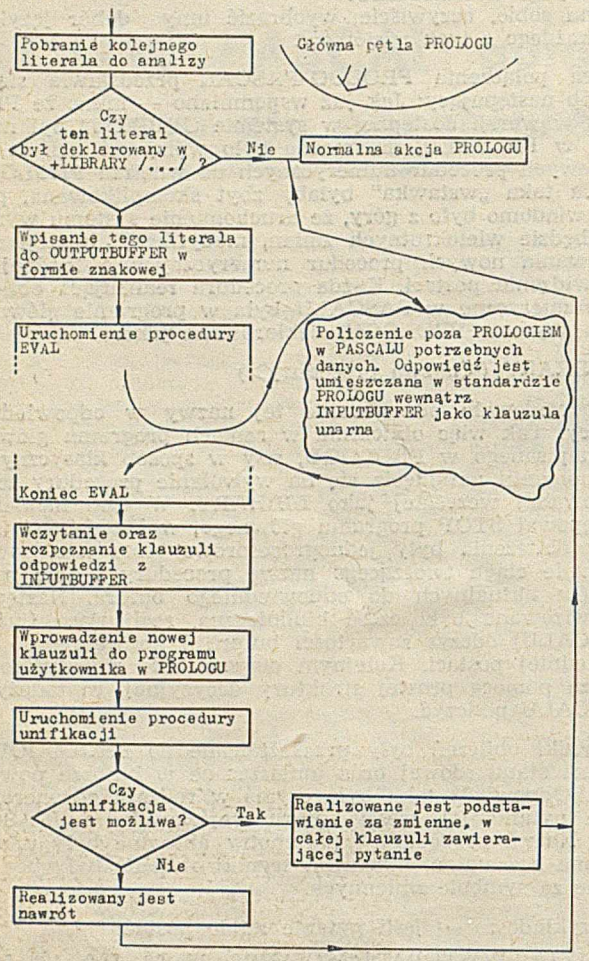
Po uzyskaniu odpowiedzi PROLOG „ruszał" w sposób następujący: odpowiedź była odczytywana z buforu i dokonywane było jej rozpoznanie. Następnie klauzula odpowiedzi była wprowadzana do tablic PROLOGU przy użyciu opcji AJSWITCH. Przekazanie parametrów obliczonych przez PASCAL (np. punktów kolizji) do wnętrza PROLOGU odbywało się poprzez wywołanie procedury unifikacji pytania z odpowiedzią. W rezultacie za zmienne *X, *Y, *Z podstawione były odpowiednio 10.3, 20.7, 18.3 i przekazane tym samym do wnętrza programu PROLOGOWEGO.

Unifikacja pytania z odpowiedzią, oprócz czysto użytkowego aspektu przekazania wyników, ma także inny ciekawy aspekt:
— odpowiedź wcale nie musi dotyczyć tego pytania (czyli jest to test)
— stawiając pytanie znamy odpowiedź, ale oczekujemy potwierdzenia.

Te dwa problemy były zatem „od ręki" załatwiane i często dawały dodatkowe korzyści typu programowego. Schemat połączenia PROLOGU z PASCALEM podaje rysunek 1.

Już w tym miejscu warto się pokusić o pewne wnioski. Obliczenia dla PROLOGU były realizowane zupełnie odrębnie, poza jego strukturą. Poza tym proces przeszukiwania drzewa wariantów ma to do siebie, że można na pewien czas poniechać daną gałąź, analizować inną, a potem

powrócić (gdy np. dla poprzedniej skończono już wspomagające obliczenia symulacyjne lub/i numeryczne). Tak więc przy prezentowanym podejściu aż prosi się o rozbicie obliczeń na system wieloprocesorowy, ale o wyspecjalizowanych — tj. logicznych, numerycznych i symulacyjnych — procesorach.

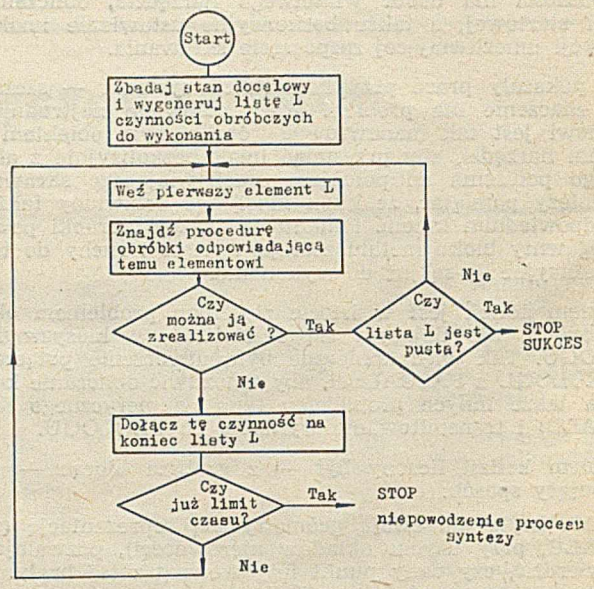


Rys. 1. Schemat połączenia PROLOG-PASCAL

ORGANIZACJA I STRATEGIA PROGRAMU OBRÓBKI

Główny program został zatem napisany w PROLOGU i był stosunkowo prosty, ponieważ bardzo pracochłonne procedury kolizji, kierunku itp. zostały zrealizowane poza programem głównym. Podstawowym celem programu głównego było tworzenie drzewa wariantów dla procesu obróbki. Należy zwrócić uwagę, że obróbka skrawaniem jest to stosunkowo prosty proces, bowiem jeśli pominiemy kwestie optymalizacji (co na tym etapie jest dozwolone), to wolno wycinać wszystko, byle tylko nie naruszyć konturu przedmiotu docelowego, który tkwi wewnątrz obrabianej bryły metalu. Można więc wiercić otwory, gwintować je, dokonywać frezowań naddatku materiału lub wycinać frezem wpusty, jeśli tylko zdołamy ustawić narzędzie tak, aby można było daną czynność wykonać. Jeśli — przykładowo — chcemy gwintować nie nawiercony jeszcze otwór, to wystąpi kolizja gwintownika z naddatkiem materiału jeszcze na etapie dosuwania gwintownika. Zresztą procedura kolizji odgrywa niemal decydującą rolę w strategii obróbki.

Jak już powiedziano, proces obróbki zaczyna się w omawianym systemie od prostopadłościennej bryły metalu. Bryła docelowa określona jest za pomocą listy, na której występują dodatkowe elementy opisujące wywiercone otwory, sfrezowane wpusty itp. Elementy te są pobierane i stanowią wskazówkę dla głównych czynności, które należy wykonać. Oczywiście generuje to poddrzewo podproblemów, typu: jak dosunąć narzędzie, czy jest np. odpowiedni frez w magazynie itd. Jeśli dana czynność prowadzi do impasu, jest ona umieszczana na końcu listy i próbowana dopiero wówczas, gdy inne wiercenia, frezowania itp. zostały wykonane. Mimo pozorów jest to strategia całkiem efektywna. Ilustruje ją schemat z rysunku 2.



Rys. 2. Główna pętla strategii obróbki

Parę słów trzeba także powiedzieć na temat strategii ruchu przy próbach dosuwania narzędzia do pozycji warunkującej rozpoczęcie obróbki. Budowa drzewa wiariantów ruchu (jest to poddrzewo drzewa wariantów procesu obróbki), w którym łuki etykietowane są w sposób przypadkowy, prowadziłaby do nadmiernego rozrostu drzewa, a tym samym do upadku systemu poprzez przekroczenie dostępnej pamięci lub czasu przeznaczonego na obliczenia. Drzewo to zostało z dobrym skutkiem zredukowane poprzez zastosowanie połączenia PROLOG-PASCAL:

— dla każdego obrabianego detalu liczone jest w PASCALU położenie startowe obróbki

— dla każdego ruchu obliczany jest tzw. punkt najlepszy, który można osiągnąć w tym ruchu, a następnie robione jest oszacowanie czy ten ruch ma sens, tj. czy zbliża do punktu docelowego; w przypadkach negatywnych — odpowiedź jest generowana tak, aby wywołać impas (oczywiście robi to procedura zapisana w PASCALU)

— dodatkowym filtrem jest procedura kolizji, która nie dopuszcza do dopisania do drzewa wariantów ruchów kolizyjnych.

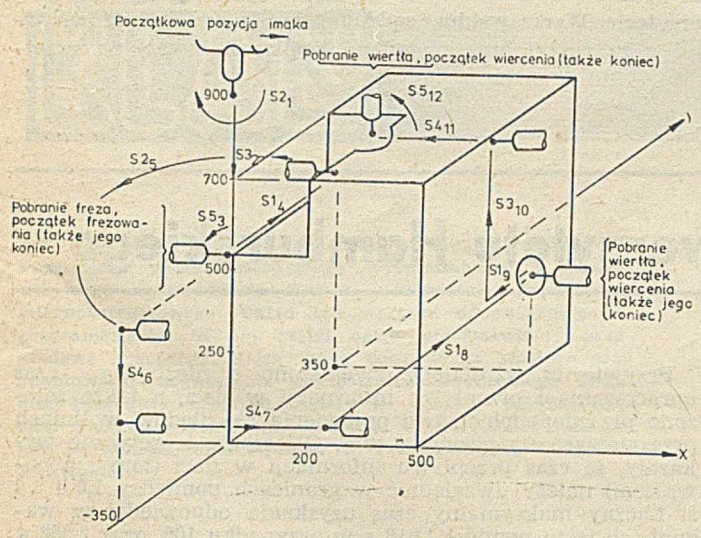
Jak zatem widać, zasadnicza strategia zrealizowana w PROLOGU jest wspomagana funkcjami oceniającymi — napisanymi w PASCALU. W rezultacie otrzymany program okazał się skuteczny i efektywny.

* * *

Prezentowany system przeznaczony jest — jak zakładałem — dla małego komputera o wieloprocesorowej konstrukcji, która byłaby możliwa do zainstalowania w hali produkcyjnej lub biurze konstrukcyjnym. Z powodu braku podobnego sprzętu, symulacje były realizowane w systemie CYBER 72.

Program w PROLOGU miał rozmiary ok. 200 kart perforowanych, biblioteka procedur numerycznych w PASCALU — ok. 700 kart. Całość systemu po kompilacji zajmowa $120\,000_8$—$130\,000_8$ (w zależności od rozwiązywanego problemu obróbki) słów 60-bitowych. Efektywność obliczeń zależała od komplikacji geometrycznej obrabianego przedmiotu, a także od stopnia „złośliwości" opisu przedmiotu docelowego. Przykładowo — jeśli na liście opisującej najpierw występowały opisy gwintów, które robi się na końcu, to upływało sporo czasu do momentu poniechania tej czynności i spróbowania innej. System po prostu bezskutecznie próbował na różne sposoby ustawić gwintownik,

aby rozpocząć gwintowanie, za każdym razem napotykając na kolizję (nie wywiercony otwór). Brakowało „inteligencji" nieco wyższego rzędu.



Rys. 3. Przykład zsyntetyzowanego procesu obróbki

Rozpatrzmy przykład procesu obróbki całkowicie zsyntetyzowanego przez system (rys. 3). Początkowa bryła metalu miała kształt bloku o wymiarach 500 × 500 × 700 mm. Opis nie był złośliwy. Kolejność operacji jest oznaczona na rysunku najniższymi indeksami, tj. $S2_1$, $S3_2$, $S5_3$ itp. Czas obliczeń wynosił 29,78 s. Program wynikowy zawierał 3 czynności skrawania, 12 ruchów dosuwania narzędzi oraz 3 czynności pobierania narzędzi. Podczas generacji drzewa wariantów 50 ruchów zostało odrzuconych przez procedurę kolizji.

Przy złożonych kształtach i złośliwych opisach czasy obliczeń dochodziły do 150 s. Ale bardzo ważne jest to, że wraz ze wzrostem komplikacji geometrycznej nie następował wykładniczy wzrost czasu obliczeń.

Można powiedzieć, że zbudowany system jest w pełni udanym, efektywnym prototypem systemu automatycznej syntezy programu obróbki. Czasy obliczeń nawet rzędu 2—3 minut są dopuszczalne w systemach typu CAD, natomiast przy linii produkcyjnej z produkcją jednostkową czy małoseryjną nie stanowią żadnego problemu.

LITERATURA

[1] Dunin-Kęplicz B., Szpakowicz S.: Język programowania PROLOG. Prace IPI PAN, nr 374, Warszawa, 1979

[2] Frydrychewicz S., Jurewicz W.: Zasady programowania obrabiarek sterowanych numerycznie. Warszawa, 1976

[3] International Conference on Programming Languages for Machine Tools. PROLAMAT'73 Budapest, 1973

[4] International Conference on Programming Languages for Machine Tools. PROLAMAT'76 Stirling, Scotland, 1976

[5] International Conference on Programming Research and Operations Logistics in Advanced Manufacturing Technology. PROLAMAT'79 Ann Arbor, Michigan, 1979

[6] Poplestone R. J.: Specifying Manipulations in Terms of Spatial Relationships. DAI Research Paper, No. 117, Department of Artificial Intelligence, University of Edinburgh

[7] SCOPE REFERENCE MANUAL MODELS 72, 73, 74 Version 3.4, Control Data, CYBER 70 Computer Series

[8] Warren D. H.: Higher Order Extensions to PROLOG — Are They Needed? DAI Research Paper, No. 154, Department of Artificial Intelligence.

W ubiegłorocznym grudniowym numerze czasopisma DATA REPORT ukazał się artykuł omawiający uruchomiony w latach 1982/1983 w Uniwersytecie Hamburskim system informatyczny obsługi procesu dydaktycznego i prac badawczych tej uczelni [1]). Ze względu na podobieństwo działań podjętych mniej więcej w tym samym okresie na Uniwersytecie Warszawskim, sądzimy, że dla celów porównawczych warto się zapoznać z zakresem oraz metodami realizacji inwestycji uczelni hamburskiej. (Red.)

System informatyczny Uniwersytetu Hamburskiego

W 1979 r. władze Uniwersytetu Hamburskiego przedłożyły dostawcom sprzętu komputerowego wymagania przyszłego systemu komputerowego uczelni. Sformułowano następujące warunki:

- zainstalowanie w poszczególnych instytutach dostatecznie dużej liczby inteligentnych terminali, zapewniających każdemu studentowi lokalne zarządzanie jego danymi i programami na dysku elastycznym, a równocześnie stały zdalny dostęp do dużego komputera centralnego.
- zainstalowanie w salach wykładowych alfanumerycznych i graficznych terminali ekranowych wraz z projektorami umożliwiającymi wyświetlanie obrazu z terminala na dużym ekranie ściennym
- zapewnienie dla poszczególnych instytutów możliwości zdalnego przetwarzania wsadowego, pracy konwersacyjnej i przetwarzania graficznego, a także przyłączania do komputera centralnego komputerów specjalizowanych, np. do sterowania.

Wymagania te określały podstawowe kryteria wyboru dostawcy sprzętu i oprogramowania. Wyboru tego dokonano w wyniku przetestowania zaproponowanych przez poszczególnych dostawców rozwiązań. Wyniki testów wykazały zdecydowaną przewagę firmy SIEMENS nad pozostałymi oferentami. Test firmy SIEMENS obejmował sprawdzenie efektywności działania systemu w warunkach równoczesnego obciążenia komputera typu 7.882 pracą w trybach wsadowym i konwersacyjnym.

Test pracy w trybie wsadowym składał się z następujących trzech części:
— standardowego syntetycznego testu wydajności SSB (Synthetic Standard Benchmark) opartego na programach napisanych w języku FORTRAN, ALGOL I COBOL
— czterech programów typowych dla intensywnych obliczeń w językach FORTRAN i ALGOL
— grupy programów błędnych celem wypróbowania kompilatorów pod kątem ich zdolności wykrywania błędów.

Test trybu konwersacyjnego polegał na symulacji równoczesnego działania 200 terminali. Rodzaj i częstość występowania podczas konwersacji typowych rozkazów ustalono na podstawie zestawu najczęściej używanego w praktyce. Założono następujące czasy podstawowych części dialogu: 5 s — zapis pytania na ekranie i jego przesłanie, 1 s — wyprowadzenie odpowiedzi na ekran, 10 s — przemyślenie odpowiedzi i podjęcie następnego działania. Pomiar czasu dotyczył ustalenia czasu oczekiwania odpowiedzi na pojedyncze zapytanie oraz wydajności pracy wsadowej w warunkach prowadzenia 100 oraz 200 dialogów (czynnych terminali). Wyniki symulacji zostały przeanalizowane za pomocą zawartych w systemie operacyjnym BS 3000 programów pomiaru i oceny czasów. Na podstawie analizy uzyskanych wyników pomiarów poprawiono efektywność działania niektórych funkcji, m.in. drogą ograniczenia liczby dostępów do dysku (dzięki przeniesieniu do pamięci wirtualnej dodatkowych zbiorów i programów, optymalizacji rozmieszczenia zbiorów na dyskach), a także takiego rozmieszczenia programów wykonywanych w trybie wsadowym, aby uzyskać optymalną „mieszankę" programów intensywnych pod względem obliczeń oraz operacji wejścia-wyjścia.

[1]) Harri Sprenger, Jacques Rozée, Gerd Zeitler: Rechnernetz für Lehre und Forschung. DATA REPORT 6/33, s. 8—11

Przyjęte czasy dialogu zwiększono o niezbędny czas opracowania i przepływu informacji w sieci, a także założono prawdopodobieństwo pojawienia się błędów w liniach przesyłowych na poziomie 10^{-6}. Obliczenia teoretyczne wykazały, że czas przepływu informacji w sieci (tam i z powrotem) należy uwzględnić w granicach pomiędzy 1,0 i 1,8 s. Łączny maksymalny czas uzyskania odpowiedzi w warunkach testu wyniósł 1,918 s w przypadku 100, oraz 2,352 s w przypadku 200 terminali działających równocześnie. Na podstawie tych wyników przyjęto, że uwzględniając „kalkulowane" ryzyko należy przyjąć jako przeciętny czas uzyskania odpowiedzi 3 s, w przypadku działania 150 terminali przewidzianych w pierwszej fazie rozbudowy systemu. Wszystkie założone parametry testu umieszczono w warunkach umownych odbioru systemu przez użytkownika.

KONFIGURACJA SPRZĘTU I OPROGRAMOWANIE

Jak już wspomniano, system oparto na wieloprocesorowym komputerze SIEMENS 7.882 z pamięcią operacyjną o pojemności 16 M bajtów, kanałową szybkością przesyłania 40 M bajtów/s oraz bezpośrednim dostępem do pamięci dyskowej o pojemności 12 G bajtów. Wszystkie istotne dla użytkownika sprzęgi tego sytemu są kompatybilne z rozwiązaniami sprzętowymi i programowymi firmy IBM. Zastosowany system operacyjny BS 3000 zawiera następujące podsystemy: JES (Job Entry System), TSS (Time Sharing System), VTAM (Virtual Telecommunications Access Method oraz NCP (Network Control Program).

Rozwiązaniem istotnym z punktu widzenia swobodnego dostępu do zasobów systemu (podstawowy warunek obsługi wielu tysięcy studentów Uniwersytetu) jest system administrowania dostępem użytkowników i ich zbiorami, który gwarantuje, że każdy użytkownik będzie poruszał się wyłącznie w obszarze przydzielonych mu zasobów i priorytetów.

Do komputera 7.882 można przyłączyć 8 komputerów SIEMENS 6.660—70, pełniających funkcje komputerów obsługi węzłów komunikacyjnych. Do każdego z nich można z kolei przyłączyć po 31 komputerów końcowych SIEMENS 6.611, wyposażonych w monitory ekranowe.

Komputery obsługi węzłów wyposażone są w system oprogramowania podstawowego AMBOSS 3. Przyłączenie do komputera centralnego realizowane jest zgodnie z ustaleniami SNA (Systems Network Architecture), a jako protokół przesyłania przyjęto procedurę SDLC (Synchronous Data Link Control). Komputer typu 6.660—70 (w nomenklaturze katalogowej komputer biurowy — niem. Bürocomputer) działa tu jako jednostka wielofunkcyjna (SNA — Cluster — Controller) sterując przyłączonymi do niego komputerami końcowymi typu 6.611 (w nomenklaturze katalogowej komputer ekranowy — niem. Bildschirmcomputer), pełniącymi funkcje terminali konwersacyjnych lub zdalnych stacji wsadowych.

Komputer obsługi węzła, dysponujący pamięcią operacyjną o pojemności 1 M bajtów służy również jako zdalna stacja wsadowa systemu dzięki wyposażeniu w odpowiednie urządzenia peryferyjne (czytniki kart, drukarki i plotery). Komputer ten może równocześnie realizować obsługę pracy wsadowej i konwersacyjnej). Dla ułatwienia studentom dostępu do zasobów systemu, czytniki kart i dru-

Dokończenie na str. 21

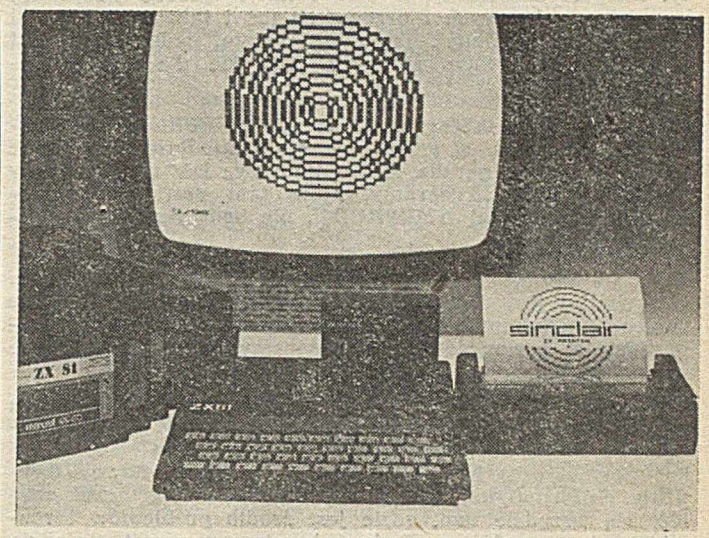
W czwartym mikroKLANIE przedstawiony został opis konstrukcji układu sterującego wyświetlaczami i umożliwiającego wprowadzenie danych z klawiatury do systemu mikroprocesorowego. Układ ten wymaga odpowiedniego oprogramowania. W tej części opisu przedstawiamy przykładowe procedury, które mogą posłużyć za wzorzec przy adaptowaniu rozwiązania na potrzeby konkretnego systemu. Procedury napisane zostały w języku ASSEMBLER 8080. Opisywane rozwiązanie zostało wykorzystane w systemie MSA80.1000, opracowanym w Przemysłowym Instytucie Elektroniki.

Klawiatura i wyświetlacze w 8-bitowym systemie mikroprocesorowym (2)

Do obsługi współpracy klawiatury i wyświetlaczy z systemem wykorzystane zostały procedury **GETKEY** i **DSPDGT.** Zmienne VARBS należy umieścić w pamięci RAM, natomiast pozostałą część programu można przechowywać w pamięci EPROM. Portom: wejściowemu, wyjściowemu i przełączającemu (symbole PINP, DISPLE, POUT) — należy przyporządkować rzeczywiste adresy.

Zastosowany w systemie układ zegara generuje co 2,5 ms przerwanie, w wyniku którego następuje wywołanie

```
; Procedura przetwarza cyfrę heksadecymalną podawaną w rejestrze A
; na kod sterujący wyświetlaczem 7 segmentowym.
; Rezultat zapisywany jest w buforze pod adresem
; wskazanym przez parę rejestrów DE.
; Zmienia zawartość rejestrów: H, L, D, E, A
DSPDGT: ANI     0FH     ; Wyzeruj nieistotne 4 starsze bity liczby re-
                        ; prezentującej cyfrę heksadecymalną
        LXI     H,CODGT ; Oblicz adres, pod którym umieszczono kod
        ADD     L       ; cyfry heksadecymalnej
        MOV     L,A     ;
        MOV     A,M     ; Pobierz kod cyfry
        STAX    D       ; Zapisz kody cyfry do odpowiedniego bufora
                        ; wyświetlacza
        INX     D       ; Zwiększ o 1 wskaźnik bufora wyświetlacza
        RET             ; Wróć
; Tablica kodów wyświetlacza cyfr heksadecymalnych
CODGT:  DB      3FH     ; 0
        DB      06H     ; 1
        DB      5BH     ; 2
        DB      4FH     ; 3
        DB      66H     ; 4
        DB      6DH     ; 5
        DB      7DH     ; 6
        DB      07H     ; 7
        DB      7FH     ; 8
        DB      6FH     ; 9
        DB      77H     ; A
        DB      7CH     ; B
        DB      39H     ; C
        DB      5EH     ; D
        DB      79H     ; E
        DB      71H     ; F
; Procedura wprowadzająca do akumulatora kod ASCII wciśniętego klawisza
; Jeżeli po poprzednim wywołaniu GETKEY nie wciśnięto żadnego klawisza, kod
; w akumulatorze=80H.
; Procedura zmienia zawartość rejestrów: H, L, A
GETKEY: LXI     H,KEYBUF ; Załaduj adres bufora
        MOV     A,M     ; Pobierz kod znaku z bufora
        MVI     M,80H   ; Zapisz kod 80H do bufora
        RET             ; Wróć
```



Przezorni na rynek

Rozmowa z przedstawicielami Przedsiębiorstwa Zagranicznego AMEPROD — WOJCIECHEM LIPKO, dyrektorem ds. systemów komputerowych, i GRZEGORZEM KORYTOWSKIM, z-cą dyrektora ds. elektroniki

µK: Bez wielkiej chyba przesady można powiedzieć, że AMEPROD wprowadził mikroinformatykę na polski rynek. Od kiedy ZX81 można oficjalnie kupić za złotówki, przestała być ona tylko wspaniałą wizją dalekiej przyszłości, staje się faktem. Czy zdawaliście sobie Panowie sprawę z istoty swego przedsięwzięcia, czy też była to po prostu kolejna transakcja?

Grzegorz Korytowski: — Zaczęło się od czysto handlowej analizy. Mieliśmy dewizy z eksportu innych wyrobów, chcieliśmy więc rozpocząć taką produkcję — w ramach elektroniki — w której bylibyśmy na polskim rynku pionierami, a która zapewniała powodzenie. Stąd mikrokomputery. Choć oczywiście, w tej głębi kryzysu, przy ogromnych brakach na każdym kroku, można się było zająć czymkolwiek, np. długopisami z zegarkiem...

µK: — Kiedy to było?

G.K.: — Na Międzynarodowych Targach Poznańskich w 1982 roku. Zorientowaliśmy się wówczas, jakie są nasze możliwości eksportowe. Poza tym musieliśmy wspomóc dewizami produkcję na krajowy rynek — tak, aby były złotówki na produkcję eksportową... Wybór mikrokomputerów był następstwem moich zainteresowań hobbystycznych, może nawet jakiegoś fanatyzmu. Ale gdy w październiku 1982 sprowadziliśmy pierwsze egzemplarze ZX81, nie bardzo wierzyłem w sukces. To co się stało, było zresztą trudne do przewidzenia. Nie była potrzebna żadna reklama, żaden marketing. W ciągu kilku miesięcy teka zamówień przerosła możliwości produkcyjne. I w efekcie sens społeczny naszego przedsięwzięcia stał się oczywisty.

µK: — Wróćmy na chwilę do początków. Od kiedy AMEPROD istnieje?

G.K.: — Od kwietnia 1981. Uruchomiliśmy najpierw zakład metalowy, który wykonywał meble ogrodowe, sanki, stoliki, a także zakład elektroniki, produkujący dzwonki

Przezorni i rynek

elektryczne, gry telewizyjne... wszystko — proste przedmioty. Eksport zaczęliśmy w 1982 roku — przede wszystkim wyrobów metalowych i drzewnych (bo i taki zakład został wkrótce uruchomiony). Trwa on, zresztą, nadal. Natomiast największym źródłem złotówek stała się dla nas mikroinformatyka.

µK: — Zastanawialiśmy się dlaczego AMEPROD, nie organizacja społeczna, a przedsiębiorstwo handlowe, inwestuje w tak nieopłacalne dla siebie imprezy, jak np. konkursy klubu ABAKUS. Programy, które tam powstaną, nagrodzone komputerami AMEPRODU, nie będą nawet własnością firmy...

G.K.: — To nie jest działalność wyłącznie charytatywna. Oczywiście, jest spory element ryzyka, ale przecież takie społeczne działania są w stanie wylansować świetnych programistów, przynieść dobre programy. A na tym także można zarabiać.

Wojciech Lipko: — Nie można zajmować się informatyką bez zrozumienia głębszego sensu tej dziedziny. Już dotychczasowe nasze doświadczenie pozwala pojąć wielkość „mikroinformatycznych" przeobrażeń, znaczenie nowej dziedziny. Informatyka, dzięki mikrokomputerom, może znaleźć się wszędzie tam, gdzie jest źródło problemów i rozwiązać je na miejscu. Duży i kosztowny sprzęt staje się teraz poręcznym narzędziem, mogącym istotnie zmieniać nasze życie. Jest to też forma politechnizacji społeczeństwa, koniecznej, zważywszy nadchodzące zmiany. Musimy patrzeć perspektywicznie.

ZX81 jest dzisiaj najpopularniejszym komputerem w Polsce, biorąc pod uwagę liczbę konfiguracji. Jest ich ponad 600. Użytkownicy tych komputerów, niezależnie — od nas kupionych czy przywiezionych z zagranicy — są potencjalnymi nabywcami naszego oprogramowania, którym także zaczynamy handlować. Jest to komputer pierwszego kroku, od którego można rozpocząć kontakt z informatyką, by potem przejść do poważniejszych, profesjonalnych zastosowań, do większych systemów. Może być on podstawą edukacji szkolnej i uczelnianej, właśnie na prawach elementarza.

Mikrokomputera o tych rozmiarach i takich parametrach nie można dzisiaj robić w kraju bez znacznego udziału dewiz. Przemysł kluczowy, ze swoim samofinansowaniem, nie jest w stanie tego dokonać. My mamy stosunkowo większe możliwości, więc jest to też jakby nasz obowiązek.

G.K.: — Tak jak w innych przedsiębiorstwach zagranicznych, i u nas dominuje biznes. Ale my, ponadto, jesteśmy mikroinformatycznymi zapaleńcami. To w istocie sprawa satysfakcji.

W.L.: — Plany wielu przedsiębiorstw, w tym naszego, wynikają z kultury technicznej ludzi, którzy je tworzą. Rozstrzyga środowisko.

µK: — Jakie więc plany?

W.L.: — Poza samymi mikrokomputerami, chcemy się zająć oprogramowaniem. Pod koniec ubiegłego roku powstał, pod naszym patronatem, klub użytkowników ZX81. Mamy ponad sto zgłoszeń różnych instytucji i prywatnych właścicieli komputerów. Na pierwszym spotkaniu było ponad 250 osób. Mając rozeznanie w oprogramowaniu powstałym u użytkowników, będziemy nabywać najciekawsze, weryfikować je, uzupełniać opisy i przekazywać innym chętnym. Powołaliśmy też redakcję biuletynu użytkowników — informatora techniczno-handlowego. Przygotowujemy do druku już drugi numer.

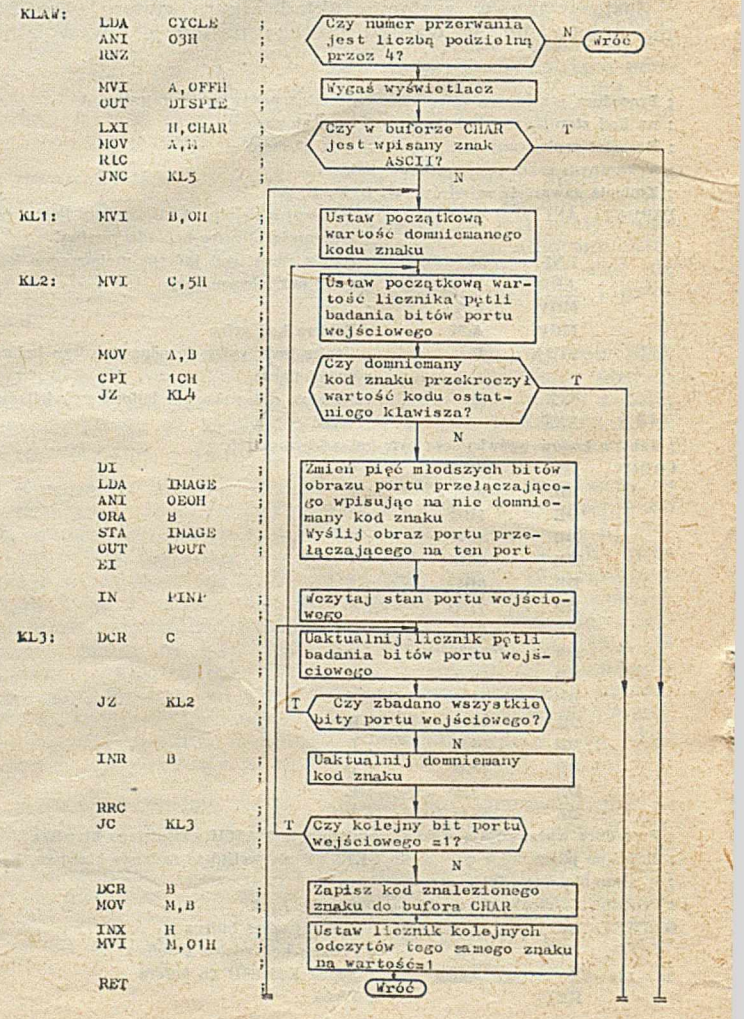
µK: — Wszystko to — za pieniądze?

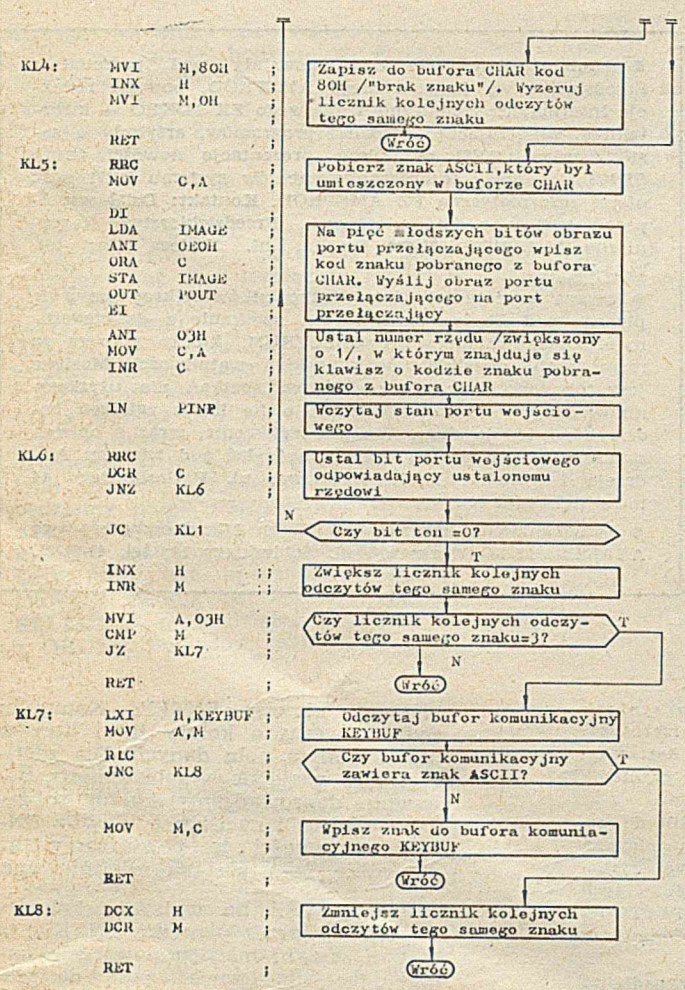
W.L.: — Oczywiście. Nas nikt nie dotuje, musimy się sami utrzymać. Odsprzedając państwu tyle, ile żąda dolarów, płacąc podatki...

```
; Segment zmiennych programu obslugi klawiatury i wyświetlaczy
VARBS:  IMAGE:   DS 1     ; Obraz portu przełączającego
        CHAR:    DS 1     ; Bufor dekodowanego znaku
        COUNT:   DS 1     ; Licznik klawiatury — umożliwia uniezależ-
                          ; nienie się od drgań zestyków
        KEYBUF:  DS 1     ; Bufor komunikacyjny znaku klawiatury
        DISPLE:  DS 8     ; 8 buforów wyświetlaczy
        CYCLE:   DS 2     ; Licznik przerwań (modulo 10000H)

; Procedura obslugi wyświetlaczy
WYSW:   MVI   A,0FFH      ; Wygaś wyświetlacze
        OUT   DISPL
        LDA   CYCLE       ; Pobierz numer (3 bitowy — 0 do 7)
        ANI   07H         ; sterowanego wyświetlacza
        MVI   H,0         ; Przygotuj adres bufora sterowanego wyświe-
        MOV   L,A         ; tlacza
        DAD   D.
        RLC               ; Przygotuj numer wyświetlacza do podania
        RLC               ; na bity 4, 3, 2 portu przełączającego
        MOV   C,A
        DI                ; Bity 7, 6, 5, obrazu portu przełączającego
        LDA   IMAGE       ; pozostaw bez zmiany. Na bitach 4, 3, 2
        ANI   0E0H        ; zestaw numer wyświetlacza. Bity 1, 0 wy-
        ORA   C           ; zeruj
        STA   IMAGE
        OUT   POUT        ; Wyślij obraz portu przełączającego na port
        EI                ; przełączający
        MOV   A,M         ; Pobierz z bufora sterowanego wyświetlacza
        CMA               ; kod znaku
        OUT   DISPL       ; Wylij go na port wyjściowy
        RET               ; Wróć

; Procedura obsługi klawiatury
KLAW:   LDA   CYCLE       ;
        ANI   03H         ;
        RNZ               ;
        MVI   A,0FFH      ;
        OUT   DISPLE      ;
        LXI   H,CHAR      ;
        MOV   A,M         ;
        RLC               ;
        JNC   KL5         ;
KL1:    MVI   B,0H        ;
                          ;
                          ;
KL2:    MVI   C,5H        ;
                          ;
                          ;
                          ;
        MOV   A,B         ;
        CPI   1CH         ;
        JZ    KL4         ;
                          ;
                          ;
        DI                ;
        LDA   IMAGE       ;
        ANI   0E0H        ;
        ORA   B           ;
        STA   IMAGE       ;
        OUT   POUT        ;
        EI                ;
                          ;
        IN    PINP        ;
                          ;
KL3:    DCR   C           ;
                          ;
                          ;
        JZ    KL2         ;
                          ;
        INR   B           ;
                          ;
        RRC               ;
        JC    KL3         ;
                          ;
                          ;
        DCR   B           ;
        MOV   M,B         ;
        INX   H           ;
        MVI   M,01H       ;
                          ;
        RET               ;
```





procedury obsługującej wyświetlacze **WYSW** i klawiaturę **KLAW** oraz zwiększenie o 1 licznika przerwań **CYCLE** (zliczającego modulo 10000H).

Dla każdego z ośmiu wyświetlaczy przewidziano 1-bajtowy bufor. Wywołanie procedury **DSPDGT** powoduje wpisanie kodu do odpowiedniego bufora. Natomiast wywoływana przez przerwanie procedura **WYSW** powoduje wysyłanie przez 2,5 ms na linie 4, 3, 2 portu POUT numeru wyświetlacza (przy każdym przerwaniu innego), a na port DISPLE — kodu wyświetlanej cyfry, pobranego z odpowiedniego bufora.

W systemie zastosowano matrycę złożoną z 26 klawiszy (w tym 16 klawiszy do wprowadzania liczb w kodzie szesnastkowym), rozmieszczonych w 7 kolumnach (numerowane od 0 do 6). W kolumnach 5 i 6 umieszczono po trzy klawisze, a w pozostałych po cztery. Cyfry heksadecymalne zajmują kolumny — 0 do 3.

Procedura **KLAW** realizowana jest raz na cztery przerwania. Sprawdza ona, czy został wciśnięty klawisz — poprzez wygaszenie wyświetlaczy, wybieranie kolejnych kolumn matrycy i poszukiwanie „zera" wśród bitów portu wejściowego PINP. W kodzie każdego klawisza bity 0, 1 są numerem rzędu, zaś 2, 3, 4 numerem kolumny matrycy klawiatury. Kod wciśniętego klawisza wpisywany jest do bufora CHAR. Jeśli trzykrotnie pod rząd zostanie wykryte wciśnięcie tego samego klawisza, kod znaku zostaje umieszczony w drugim buforze (komunikacyjnym) KEYBUF, skąd można go odczytać za pomocą procedury **GETKEY.**

ZBIGNIEW POJMAŃSKI
PIE, Warszawa

Przezorni na rynek

µK: — Wróćmy do planów.

G.K.: — Pakiet zamówień na ZX81 na rok bieżący został już zamknięty, a nie przewidujemy kontynuacji tej produkcji w roku 1985. Najprawdopodobniej baza elementowa będzie już na Zachodzie niedostępna, w efekcie rezygnacji firmy SINCLAIR z tego modelu. Będziemy się natomiast zajmować rozwojem konfiguracji, nowymi jej elementami (np. drukarka SEIKOSHA dla zastosowań profesjonalnych, czy rozbudowa pamięci do 64 K).

W.L.: — Przyjęte zamówienia zrealizujemy. Cały też rozwój konfiguracji, oprogramowania — będzie dokonywany w oparciu o własne rozwiązania. Nie zostawimy klientów samych sobie. Nie zostaną bez opieki — to pragnąłbym podkreślić.

G.K.: — Na tegorocznych Międzynarodowych Targach w Poznaniu zamierzamy zaprezentować nowy mikrokomputer o lepszych parametrach, bardziej uniwersalny od ZX81, częściowo wykorzystujący krajowe elementy. Nie chcemy na razie ujawniać szczegółów. Ważne jednak, że zostajemy przy sprzęcie dla amatorów, początkujących, nie podejmując konkurencji w produkcji systemów profesjonalnych. Chcemy umożliwić użytkownikom kontakt z komputerem przy możliwie najniższych nakładach finansowych, przy użyciu telewizora, magnetofonu kasetowego. Różnimy się więc w założeniach od innych firm, z którymi jesteśmy czasem porównywani — IMPOL, COMPUTEX czy ELWRO. My także chcemy sprzętu efektywnego, lecz — najtańszego!

µK: — Właśnie, pomówmy o cenach. Często się słyszy, że za ZX81 AMEPROD bierze za dużo...

W.L.: — My nie jesteśmy tylko przewoźnikiem i sklepem, ale zakładem produkcyjnym, który z elementów zachodnich tworzy komputer. Koszty wykonania są w kraju wysokie i musimy uwzględniać to w cenie produktu. Poza tym — jak już mówiłem — dolary sprzedawane państwu (50% obrotu) i podatek także robią swoje. Ponadto — utrzymanie grupy serwisowej; nie odsyłamy przecież użytkowników do Anglii.

ZX81 musi też zarobić na rozwój. Nie otrzymujemy przecież kredytów...

G.K.: — Udzielamy gwarancji. To kosztuje, bo ten sprzęt mimo wszystko nie jest pozbawiony wad. Tym bardziej, gdy pracuje intensywniej niż powinien w zastosowaniach amatorskich (np. drukarka), dla których jest stworzony. Ale z drugiej strony — okazuje się, że nawet osoby prywatne często chcą kupować u nas, a nie — co jest znacznie tańsze — sprowadzać z Zachodu.

µK: — Częste są przypadki sprzedaży ZX81 prywatnym osobom?

G.K.: — Bardzo rzadkie, chociaż w zamówieniach jest to pół na pół. Musimy działać strategicznie. Z wielu względów bardziej nam się opłaca sprzedawać komputery instytucjom. Taka działalność, jak nasza, nie jest przez władze doceniana. Często robi nam się różne trudności, a to z pozwoleniem na import, a to na eksport... Bo właściwie po co komu jakieś tam komputery. Wtedy my pokazujemy pisma, czy to z WAT, czy z jakiegoś ministerstwa, szkoły wyższej bądź instytutu, czyli dla władzy — pisma klientów poważnych, i wówczas przeszkodę łatwiej pokonać.

W.L.: — I w efekcie ZX81 jest u nas najczęściej stosowany w różnych instytucjach przez profesjonalistów. A do takich niezbyt się nadaje. Propozycja Sinclaira: taniość i prostota nade wszystko — zaczyna stanowić wadę przy intensywnej eksploatacji. Ustawiczny montaż i demontaż konfiguracji niszczy styki, które nie są dostatecznie trwałe. Musimy więc uwzględniać ten nasz rodzimy sposób funkcjonowania ZX81.

µK: — Wiele mówi się o tym, że niedawne pociągnięcia podatkowe fiskusa sparaliżują działalność przedsiębiorstw zagranicznych...

Przezorni nä rynek

W.L.: — Decyzje podatkowe z sierpnia ub.r. uderzyły w nasz rynek. Praca AMEPRODU jest nadal opłacalna, ale produkcja nie może być rozwijana ilościowo. Tak wielki procent dewiz (50% z obrotu, nie z zysku!), jaki musimy przekazywać państwu, może dla wielu przedsiębiorstw oznaczać koniec pracy. Wiele elementów niestety trzeba kupować za dolary. Jeśli stanowią one — powiedzmy — 60% obrotu a 50% trzeba oddać fiskusowi, to 10% musimy dołożyć...

µK: — Ale wśród Panów raczej optymizm!?

G.K.: — Mamy już sporo doświadczenia, pewien staż — to ułatwia potyczki. W 1983 roku nasz eksport wyniósł — na 500 istniejących w kraju firm zagranicznych — 8%. Mamy nadzieję, że działając przezornie przetrwamy ciężkie czasy. Niemniej wprowadzone przepisy znacznie ograniczą produkcję, to oczywiste.

W.L.: — I ostatecznie — zmniejszą ilość dewiz wpływających do kasy państwa. Najbardziej zaś ucierpi krajowy rynek.

Z. GLUZA
Oprac.

- Klub użytkowników ZX81, powstały pod patronatem Przedsiębiorstwa Zagranicznego AMEPROD, powołał redakcję informatora techniczno-handlowego ZX MIKRO — kwartalnika, zawierającego wydruki programów, artykuły o zastosowaniach ZX81 w Polsce, prezentacje nowości firmy SINCLAIR, opisy techniczne elementów systemu ZX81 oraz ofertę informatyczną PZ AMEPROD. Kontakt: Dział współpracy systemów komputerowych, Przedsiębiorstwo Zagraniczne AMEPROD; 61-632 Poznań, ul. Kmieca 20a; tel. 22-18-79.
- Uwaga nowicjusze! Klub Użytkowników Mikroprocesorów planuje pod koniec września br. spotkanie z doświadczonymi użytkownikami ZX SPECTRUM, którzy pomogą w rozproszeniu wszelkich debiutanckich wątpliwości. Możliwe jest też zorganizowanie podobnych spotkań dla użytkowników innych mikrokomputerów, o ile liczba zgłoszeń będzie odpowiednio duża. Wstępne zgłoszenia, wraz z opisem zasadniczych problemów, należy nadsyłać pod adresem Andrzeja Drożniaka: 00-777 Warszawa, ul. Pytlasińskiego 14 m. 4 (tel. 41-26-01).
- Przypominamy kontakty do Klubu Mikrokomputerowego ABAKUS: 02-930 Warszawa, ul. Sobieskiego 18; tel. 42-91-85.

Po półtorarocznej karierze ZX SPECTRUM (p. mikroKLAN 4), na rynkach zachodnich oferowany jest olbrzymi wybór (kilkaset) programów przeznaczonych dla tego mikrokomputera. Znakomita większość z nich to gry, dostarczające rozrywki, co może utwierdzać w przekonaniu, że mamy do czynienia z bardzo inteligentną, ale jednak — zabawką.

Czarnorynkowy kurs twardych walut zmusza nas byśmy podchodzili do zabawek z uwagą godną profesjonalnego sprzętu komputerowego. Użytkownicy na Zachodzie, płacąc zaledwie kilkaset dolarów więcej, mogą nabyć sprzęt znacznie bardziej efektywny; krajowych użytkowników zazwyczaj nie stać na tych kilkaset dolarów. A niedogodności związane z wykorzystaniem ZX SPECTRUM są mimo wszystko znacznie mniejsze niż trudności, przed jakimi stoimy przy braku mikrokomputerów.

Rozpoczynając serię publikacji o programach użytkowych, nieprzypadkowo wybraliśmy jako pierwszy program VU-File. Oprogramowanie typu baza danych jest u nas realizowane — jak dotąd — głównie na dużych systemach. Natomiast VU File nie jest przeznaczone dla Elektronicznych Ośrodków Obliczeniowych, lecz dla użytkowników ZX SPECTRUM. Ma więc szansę trafić pod strzechy. Program pozwala zaoszczędzić wiele czasu traconego na żmudne sporządzanie rozmaitych zestawień i prowadzenie ewidencji — czynności, które tak kocha nasza rodzima biurokracja.

Program katalogujący VU-File (ZX SPECTRUM)

Program ten jest ulepszoną wersją analogicznego programu dla ZX 81. Wszystkie istotne części programu napisane są w języku ASSEMBLER Z80. Program działa więc bardzo szybko i zajmuje tylko 5 KB pamięci.

Program składa się z dwóch części. W pierwszej projektuje się karty katalogu i zapełnia je danymi. Druga umożliwia różnorodne wykorzystanie przygotowywanych uprzednio kart.

Zakładanie katalogu

Użytkownik ma do dyspozycji czystą kartę, na której w dowolnych miejscach, za pomocą edytora ekranowego umieszcza teksty nagłówków. Edytor ekranowy bardzo ułatwia projektowanie kart. Przykładowo — karta z nagłówkiem wyglądać może następująco:

Nazwisko
Imię
Branża materiałowa
Towary

Adres

Po wpisaniu wszystkich nagłówków przyciskamy równocześnie klawisze SYMBOL SHIFT i STOP. Wtedy na ekranie pojawia się nowy typ kursora i można przejść do projektowania pól danych. W tym celu doprowadzamy kursor do pozycji, od której chcemy, aby zaczynało się pole danych i naciskamy klawisz ENTER. Komputer pyta wówczas o kolory tła i liter w definiowanym polu danych. Dla zdefiniowania następnego pola danych, ponownie doprowadzamy kursor w żądane miejsce i naciskamy klawisz ENTER. Czynności te powtarzamy, aż wszystkie pola zostaną zdefiniowane.

Uwaga: jeśli po wpisaniu kolorów naciśniemy przez pomyłkę ENTER, to w tym samym miejscu powstanie nakładające się „nowe" pole danych. Niestety, nie można go skasować i trzeba zaczynać wszystko od początku. Ten fragment programu nie został odpowiednio dopracowany przez autorów.

Po opisaniu wszystkich pól danych, możemy przejść do wpisania w nie odpowiednich informacji. W tym celu przyciskamy równocześnie klawisze SYMBOL SHIFT i STOP. W odpowiedzi kursor ustawia się w odpowiednim polu i możemy wpisywać dane.

Niestety, jedna linia danych nie może być kontynuowana w następnej linii na ekranie. Jest to kolejna niedogodność. Jeśli więc wiemy, że informacje mogą nie zmieścić się w jednym wierszu, musimy pod nagłówkiem zdefiniować kilka linii pola danych.

Po zapisaniu karty możemy zapisać następne lub nacisnąć SYMBOL SHIFT i STOP, co spowoduje przejście do programu umożliwiającego korzystanie z katalogu.

Korzystanie z katalogu

W tej części programu przewidziano następujące polecenia:

ENTER — wprowadza do części programu, w której wpisujemy nowe karty.

ALTER — służy do wprowadzenia zmian na karcie będącej w danym momencie na ekranie.

COPY — powoduje przepisanie na drukarkę całej karty będącej aktualnie na ekranie.

DELETE — kasuje kartę znajdującą się na ekranie.

PRINT — włącza specjalną opcję oszczędnościową. Użytkownik może zaprojektować rozmieszczenie nagłówków i pól danych niezależnie dla drukarki i ekranu.

ORDER — wskazuje komputerowi, według którego pola mają być uporządkowane karty. Kolejność sprawdzana jest na podstawie kodu ASCII. Gdy wpisujemy liczby, kolejne cyfry dziesiętne powinny znajdować się w tych samych miejscach. Dzięki temu będziemy mogli porządkować również według wielkości liczbowych. Przykładem zastosowania polecenia ORDER może być uporządkowanie dostawców według cen ich wyrobów.

SELECT — może działać równocześnie z ORDER. Powoduje wybieranie przez komputer tylko takich kart, na których występuje określony ciąg znaków w danym polu (bądź na całej karcie). W połączeniu z poprzednim poleceniem można — na przykład — uporządkować według ceny dostawców tylko z Warszawy.

LIST — umożliwia automatyczne przeglądanie kart z szybkością jednej karty na sekundę. Wynik można również skierować na drukarkę.

INFORM — podaje informację o wielkości zajętego obszaru pamięci.

QUIT — umożliwia przejście do programu, w którym mamy następujące opcje:

SAVE file — nagranie całego katalogu na taśmę,

LOAD file — wprowadzenie całego katalogu z taśmy

ERASE — czyli kasowanie katalogu w pamięci komputera.

Program bardzo oszczędnie wykorzystuje pamięć. Na przykład, katalog 515 państw świata, gdzie o każdym są następujące dane: nazwa, stolica, waluta, język, dochód narodowy, liczba mieszkańców, powierzchnia, kontynent — zajmuje ok. 10 KB pamięci.

Wśród przykładów zastosowania programu VU-File, warto wymienić katalogi:

— książek upchniętych na najwyższą półkę
— programów (komputerowych bądź video)
— monet lub znaczków (a jak ktoś ma, to i obrazów Picassa)
— układów scalonych (niestety bez schematów)
— znajomych, których trudno spamiętać (z uwzględnieniem płci bądź innych walorów)
— rzeczy wypożyczonych.

Program ten powinien znaleźć się w bibliotece każdego posiadacza ZX SPECTRUM. Może on posłużyć do walki z tymi siłami domowymi, które twierdzą, że komputer to tylko droga zabawka.

PIOTR PARLEWICZ
Warszawa

Procedury mnożenia i dzielenia dla 8080

Wykonywanie jakichkolwiek obliczeń za pomocą mikrokomputera łączy się zazwyczaj z koniecznością mnożenia i dzielenia. Przedstawione poniżej procedury charakteryzują się stosunkowo krótkim czasem realizacji, co jest szczególnie ważne przy bardziej złożonych obliczeniach. Dla wykonania dzielenia potrzeba w najgorszym przypadku 1745 okresów zegarowych, a dla mnożenia 1023 okresy. Przy najczęściej stosowanej częstotliwości zegara 2,048 MHz daje to dla dzielenia ok. 852 μs, a dla mnożenia — ok. 500 μs.

Dzielenie

Procedura realizuje dzielenie liczby 32-bitowej (rejestry HL i DE — bajt najbardziej znaczący, MSB w rejestrze H) przez liczbę 16-bitową (para rejestrów BC). Powstały w wyniku dzielenia iloraz umieszczany jest w parze rejestrów DE, a reszta z dzielenia w parze rejestrów HL. Jeżeli iloraz przekroczy 16-bitowy zakres, następuje wyjście z procedury z sygnalizacją błędu (wyzerowany bit przeniesienia w rejestrze wskaźników). Oto program źródłowy w języku ASSEMBLER 8080:

```
DIV:    MOV  A,L      ; Czy przekroczony zakres?
        SUB  C
        MOV  A,H
        SBB  B
        RNC           ; Powrót jeśli tak
        MOV  A,B      ; Konwersja zawartości
        CMA           ; pary rejestrów BC
        MOV  B,A      ; na uzupełnienie dwójkowe
        MOV  A,C
        CMA
        MOV  C,A
        INX  B
        CALL LOOP     ; Dzielenie
        RET
; Podprogram 'LOOP' wykonuje dzielenie trzech bardziej znaczących bajtów
; dzielnej przez dwubajtowy dzielnik
LOOP:   MOV  A,D      ; Trzeci dzielony bajt do rej. A
        MOV  D,E      ; Zachowanie najmniej znaczącego bajtu dzielnej
                      ; lub najbardziej znaczącego bajtu ilorazu
        MVI  E,8      ; Licznik pętli 1
LOOP1:  DAD  H        ; Przesunięcie dzielnej w lewo
        JC   OVER     ; Rejestry HL przepełnione?
        ADD  A
        JNC  SUB
        INX  H        ; Dodanie przeniesienia jeśli wystąpiło
SUB:    PUSH H        ; Zachowanie dwóch bajtów dzielnej
        DAD  B        ; Odjęcie dzielnika
        JC   OK       ; Skok gdy wynik > 0
        POP  H        ; Odtworzenie dzielnej
        DCR  E        ; Dekrementacja licznika pętli 1
        JNZ  LOOP1
        MOV  E,A      ; Bajt ilorazu do rej. E
        STC
        RET
OK:     INX  SP       ; Eliminacja niepotrzebnej danej
        INX  SP
        INR  A        ; Wstawienie 1 do ilorazu
        DCR  E        ; Dekrementacja licznika pętli 1
        JNZ  LOOP1
        MOV  E,A      ; Bajt ilorazu do rej. E
        STC
        RET
OVER:   ADC  A        ; Dokończenie przesunięcia dzielnej,
                      ; wstawienie 1 do ilorazu
        JNC  OVRSUB
        INX  H        ; Dodanie przeniesienia jeśli jest
OVRSUB: DAD  B        ; Odjęcie dzielnika
        DCR  E        ; Dekrementacja licznika pętli 1
        JNZ  LOOP1
        MOV  E,A      ; Bajt ilorazu do rej. E
        STC
        RET
```

Mnożenie

Procedura realizuje mnożenie liczby 16-bitowej (para rejestrów BC) przez liczbę 16-bitową (para rejestrów DE). Iloczyn jest umieszczany w rejestrach DE-HL, przy czym najbardziej znaczący bajt wyniku w rejestrze D. W procedurze wykorzystywany jest podprogram wykonujący mnożenie liczby jednobajtowej przez dwubajtową („BMULT"). Liczba 8-bitowa umieszczana jest w akumu-

latorze A, a liczba 16-bitowa w parze rejestrów BC. Wynik mnożenia przekazywany jest przez podprogram w A i parze rejestrów HL (w akumulatorze jest najbardziej znaczący bajt wyniku). Czas realizacji podprogramu wynosi dla najbardziej niekorzystnego przypadku 424 cykle zegarowe.

Program źródłowy procedury mnożenia jest następujący:

```
MULT:   MOV   A,E     ; Mniej znaczący bajt mnożnika do A
        PUSH  D       ; Bardziej znaczący bajt na stos
        CALL  BMULT   ; Mnożenie cząstkowe
        XTHL          ; Wymiana mniej znaczących bajtów
        PUSH  PSW     ; Najbardziej znaczący bajt na stos
        MOV   A,H     ; Bardziej znaczący bajt mnożnika do A
        CALL  BMULT   ; Mnożenie cząstkowe
        MOV   D,A     ; Najbardziej znaczący bajt wyniku do D
        POP   PSW     ; Odtworzenie najbardziej znaczącego bajtu z 1
                      ; mnożenia
        ADD   H       ; Wyliczenie 3 bajtu wyniku
        MOV   E,A     ; i wprowadzenie go do rej. E
        JNC   NOCRR   ; Inkrementacja rej. D tylko gdy występuje prze-
        INR   D       ; niesienie
NOCRR:  MOV   H,L     ; Przesunięcie mniej znaczącego bajtu wyniku z 1
        MVI   L,0     ; mnożenia
        POP   B       ; Odtworzenie mniej znacz. bajtów 1 mnożenia
        DAD   B       ; Wyliczenie 2 mniej znacz. bajtów wyniku
        RNC           ; Koniec gdy nie ma przeniesienia
        INX   D       ; Korekcja 2 bardziej znacz. bajtów wyniku
        RET
; "BMULT" wykonuje mnożenie liczby 1-bajtowej przez 2-bajtową
BMULT:  LXI   H,0     ; Wstępne zerowanie wyniku
        LXI   D,7     ; Zerowanie D; rej. E — licznik
        ADD   A       ; Wydzielenie bitu mnożenia
LOOP1:  JNC   ZERO    ; Skok gdy równy zero
        DAD   B       ; Dodanie mnożnej
        ADC   D       ; Dodanie przeniesienia do 3 bajtu wyniku
ZERO:   DAD   H       ; Przesunięcie wyniku
        ADC   A
        DCR   E       ; Dekrementacja licznika
        JNZ   LOOP1
        RNC           ; Koniec gdy nie ma przeniesienia
        DAD   B       ; Gdy jest, ostatnie dodawanie
        ADC   D
        RET
```

Opracował AJP
na podstawie ELECTRONICS
z 24 lutego 1982

SINCLAIR rezygnuje już z produkcji ZX81. W ślad za nim rezygnację ogłasza rodzimy AMEPROD. Czy zatem wyszukiwanie w przestarzałym rozwiązaniu niuansów, które pozwolą coś tam rozwiązać efektywniej ma sens? Niestety, ma. Sytuacja rynkowa — jak wróżka z bajki — sprawia, że używane w kraju komputery (te mikro i te makro) są wiecznie młode. A przynajmniej muszą w to wierzyć ich użytkownicy, bo inaczej dla zachowania zdrowia psychicznego musieliby przerzucić się na produkcję guzików.

Tak więc życie zmusza nas do maksymalnego wykorzystania posiadanego sprzętu. To prawda, że niektóre rozwiązania ocierają się o granice zdrowego rozsądku; często jednak lepiej rozwiązać jakiś problem przez „postawienie na głowie", niż nie rozwiązać w ogóle. Z drugiej strony — problemy, które na pierwszy rzut oka wymagają znacznie bardziej wyrafinowanego sprzętu — po zastosowaniu kilku trików można często zrealizować nawet za pomocą poczciwego ZX81.

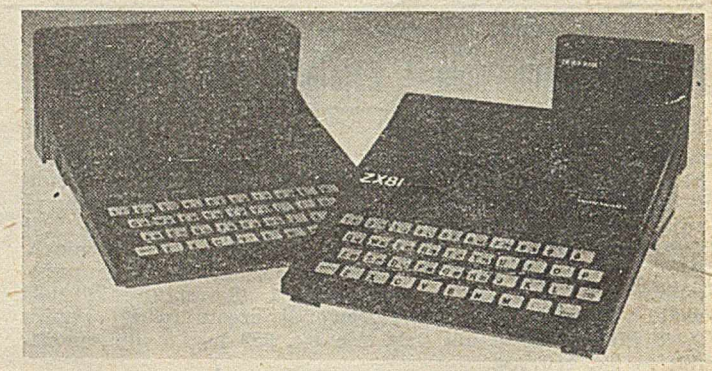
Połączenie dwóch ZX81

ZX81 może być traktowany jako moduł centralny, który został zbudowany w oparciu o najbardziej rozpowszechniony 8-bitowy mikroprocesor Z80. Względnie łatwe jest jego rozbudowywanie, programowanie w języku wewnętrznym, dopasowanie systemu do konkretnego zastosowania itp.

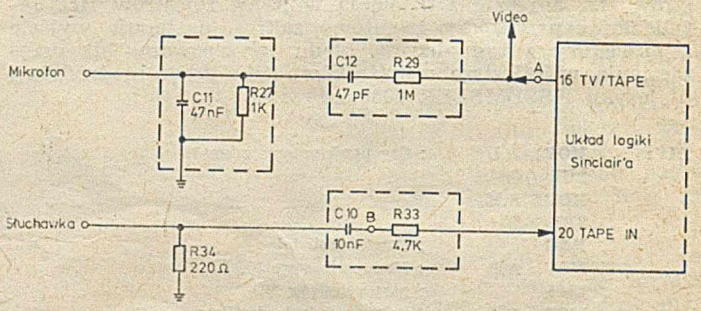
Proponowane rozwiązanie polega na wykonaniu połączeń między dwoma ZX81, z których jeden może być traktowany jako pamięć masowa dla drugiego. Możliwe jest również wiele innych zastosowań takiego połączenia: kontrola zapamiętania programu, przekazywanie danych z jednego programu do drugiego itd.

Zasada zapamiętywania programu na kasecie

Po podaniu instrukcji **SAVE**, układ logiczny ZX81 przechodzi w stan aktywny i jego wyjście TV/TAPE zmienia stan. Transmisja informacji wykonywania jest bit po bicie (ciąg impulsów o częstotliwości 3,3 kHz). Sygnał ten



nie może być podany bezpośrednio na wejście mikrofonowe magnetofonu, gdyż typowa jego czułość wynosi 5 mV. Amplituda sygnału wynosi 5 V, konieczne jest więc tysiąckrotne jej stłumienie.



Rys. 1. Filtry RC w ZX61 umożliwiające tłumienie i selekcję wykorzystywanego sygnału

ZX81 został wyposażony w filtr pasmowy (rys. 1) o częstotliwości środkowej równej 3,3 kHz. Składa się on z filtru górnoprzepustowego (szeregowy układ RC) oraz filtru dolnoprzepustowego (równoległy układ RC). Tłumienie filtru w paśmie przepustowym wynika z istnienia dzielnika tworzonego przez rezystancje R_{29} i R_{27}. Wynosi ono:

$$\frac{R_{27}}{R_{27}+R_{29}} = \frac{1}{1000}$$

jako że $R_{29} = 1000\ R_{27}$.

„Górna" częstotliwość graniczna może być wyznaczona jako:

$$F_g = \frac{1}{2\pi R_{29} C_{12}} = 3{,}38\ \text{kHz}$$

Podobnie „dolna" częstotliwość graniczna dana jest zależnością:

$$F_d = \frac{1}{2\pi R_{27} C_{11}} = 3{,}38\ \text{kHz}$$

Sygnał jest w ten sposób prawidłowo filtrowany, podczas gdy jego amplituda obniżona do 5 mV dopasowana jest do czułości wejściowej magnetofonu kasetowego.

Odtwarzanie zapisanego programu

Po podaniu instrukcji **LOAD** uaktywniony zostaje układ logiczny i wprowadzana jest informacja odpowiadająca sygnałowi wejściowemu. Sygnał ten przed dojściem do układu logicznego przechodzi przez filtr górnoprzepustowy. W stanie spoczynkowym (bez sygnału na wejściu) przez rezystancję R_{33} nie płynie prąd, gdyż kondensator C_{10} blokuje składową stałą. Wejście znajduje się „w powietrzu", tj. w stanie „1" w logice TTL.

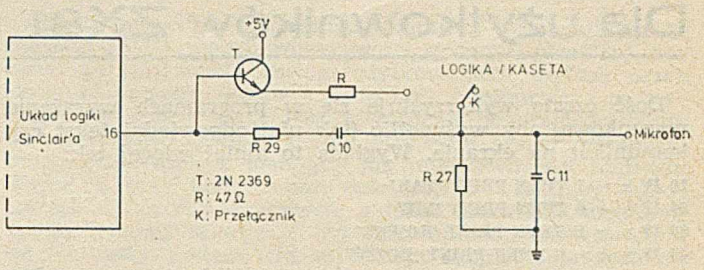
Przejście z „1" na „0" powoduje uaktywnienie wejścia układu logicznego. Konieczne jest zastosowanie sygnału o odpowiednim poziomie, a zmiany stanu powinny być bardzo szybkie. Jest to związane z pasmem transmitowanego sygnału: strome zbocza oznaczają szersze pasmo. Wskazane jest więc „podbicie" wysokich tonów w magnetofonie, w przeciwnym razie sygnał może być za słaby, a jego opadające zbocze nie wystarczająco strome, aby zmienić stan wejścia układu.

Należy przy tym uważać, aby nie podwyższyć za bardzo sygnału i nie wprowadzić zakłóceń lub zniekształceń, które mogłyby zmienić stan wejścia.

Połączenie dwóch ZX81

Wykonanie połączenia między dwoma ZX81 polega na zamontowaniu dodatkowego obwodu. Dla zapewnienia właściwego połączenia konieczne jest wzmocnienie wysyłanego sygnału (lub raczej nie tłumienie go). Obwód został zrealizowany przy użyciu tranzystora w układzie wtórnika emiterowego (wzmocnienie = 1, niska impedancja wyjściowa i wysoka impedancja wejściowa) z bazą podłączoną do wyjścia TV/TAPE.

Na rysunku 2 pokazano, że na wyjściu zastosowano rezystor 47 Ω, który ogranicza prąd płynący przez tranzystor do wartości 100 mA w przypadku zwarcia wyjścia. Zastosowany przełącznik ma dwie pozycje: jedną — dla połączenia logicznego silnych sygnałów, drugą — dla połączenia słabych sygnałów podczas nagrywania na kasetę.



Rys. 2. Przełącznik umożliwiający przesyłanie danych do magnetofonu kasetowego lub pamięci drugiego ZX81

Montaż

Tranzystor montuje się w pobliżu modulatora UHF. Bazę należy przylutować do pola znajdującego się między opornikiem R_{24} i otworem do mocowania obwodu drukowanego. Kolektor podłącza się bezpośrednio do zasilania modulatora UHF.

Przełącznik może być montowany z boku urządzenia, koło wyjścia typu „jack" sygnału wizyjnego. Wyjście przełącznika należy połączyć z opornikiem R_{27} (w pobliżu wyjścia „jack"). Opornik R podlutowuje się do przełącznika i wykonuje połączenie z emiterem tranzystora przy użyciu przewodu.

Prawidłowość działania połączenia można łatwo sprawdzić wykorzystując diodę LED, przymocowaną do wtyczki. Katoda powinna być podłączona do masy (0V), tj. do zewnętrznej metalowej części wtyczki. Anodę podłącza się do wewnętrznej części wtyczki.

Dioda powinna się świecić tylko wtedy, gdy przełącznik jest w pozycji LOGIKA. Po podaniu instrukcji SAVE „P" i nastawieniu przełącznika w pozycji LOGIKA, dioda powinna świecić się, gdy ekran jest ciemny. Podczas transmisji dioda świeci się „słabo".

Wykorzystanie drugiego ZX81

ZX81 będący „nadajnikiem" należy przełączyć w pozycję LOGIKA, a następnie wprowadzić instrukcję LOAD „P" w ZX81 spełniający rolę „odbiornika". Do ZX81 „nadajnika" wprowadzić program:

```
10 SAVE „P"
20 PRINT „DZIAŁA POPRAWNIE"
30 GO TO 20
```

i wcisnąć RUN.

Po wykonaniu transmisji oba mikrokomputery wyświetlają ten sam tekst, co oznacza, że zawierają ten sam program.

Opracował ANDRZEJ KONTKIEWICZ
na podstawie „MICRO-SYSTEMES",
marzec 1983

● Piotr Jastrzębski jest od niedawna posiadaczem mikrokomputera ATARI 600 XL. Chętnie nawiąże kontakt z innymi użytkownikami — dla wymiany doświadczeń. Kontakt: 81-365 Gdynia, ul. Abrahama 56 m. 2; tel. 20-26-77.

● Mieczysław Gołoś jest użytkownikiem mikrokomputera NEW BRAIN (model A). Dysponuje ASSEMBLEREM, FORTH'EM i wieloma informacjami technicznymi. Chętnie wymieni doświadczenia z innymi użytkownikami. Kontakt: 01-122 Warszawa, ul. Batalionu Zośka 4A m. 28; tel. p. 25-98-13, d. 36-43-86.

● W różnych ośrodkach powstają „agentury" KUM. Rozpoczynamy druk adresów osób, które będą dysponowały ankietami klubowymi, bieżącymi informacjami, itp. **KRAKÓW:** Waldemar Skorupa, 30-079, Kijowska 11 m. 31, tel. d. 37-10-25, p. 22-52-00; **POZNAŃ:** Piotr Torz, 60-621, Grudzieniec 117 m. 6, tel. p. 408-51; **SZCZECIN:** Andrzej Górecki, 71-175, Włodkowica 23 m. 3, tel. d. 737-17, p. 430-51 w. 49; **LUBLIN:** Kazimierz Rybka, 20-954, Budowlana 5, tel. p. 450-71 w. 653. (c.d.n.)

Informacje mikroKLANU prosimy przekazywać niezwłocznie pod adresem INFORMATYKI: 00-041 Warszawa, ul. Jasna 14/16 p. 244 (tel. 27-71-40). Czas płynie.

Prowadzi Andrzej J. Piotrowski (tel. p. 43-66-11 w. 42, d. 48-22-85)

Dla użytkowników ZX81

Dość często wykorzystuje się w programach wyrażenia warunkowe, aby wyświetlić (lub nie) odpowiedni tekst czy komunikat na ekranie. Wygląda to mniej więcej tak:

```
10 IF x = 1 THEN PRINT 'TAK'
20 IF x = 0 THEN PRINT 'NIE'
30 IF y = E THEN PRINT 'ROWNE'
40 IF y <> E THEN PRINT 'ROZNE'
```

To samo można zapisać w sposób może nieco mniej przejrzysty, ale zajmujący znacznie mniej miejsca w pamięci:

```
10 PRINT 'TAK' AND (x=1); 'NIE' AND (x=0),,
'ROWNE' AND (Y=E),, 'ROZNE' AND (Y<>E).
```

Uwaga: Nawet gdy trzeci warunek nie jest spełniony, interpretowany jest zapis dwóch przecinków.

Czasem zdarza się, że musimy przenumerować linie w pewnych blokach programu. Wiąże się to jednak ze skorygowaniem argumentów we wszystkich instrukcjach GOTO odnoszących się do przenumerowanego bloku. W bardziej rozbudowanych programach stwarza to możliwość wielu pomyłek. Można tego uniknąć stosując pewną regułę przy pisaniu programów. Poszczególnym częściom programu nadajemy nazwy wykorzystując instrukcję REM. Na początku programu umieszczamy deklaracje przypisujące wspomnianym nazwom numer odpowiadający linii umieszczonej bezpośrednio za linią zawierającą komentarz. Jako argument instrukcji GOTO podajemy zamiast numeru linii nazwę zmiennej. Po przenumerowaniu bloku programowego wystarczy teraz przejrzeć linie zawierające instrukcję REM i odpowiednio zaktualizować listę deklaracji. A więc koniec ze żmudnym wyszukiwaniem instrukcji GOTO!

Operatory warunkowe można w ZX81 traktować jako funkcje bolowskie, bowiem wynik sprawdzania warunku przyjmuje wartość 1 lub 0. Tak więc poprawny jest np. zapis: PRINT A = B. W tabeli zestawiono operatory warunkowe i odpowiadające im wyrażenia boolowskie.

$A < B \rightarrow \overline{A} \wedge B$	$A <> B \rightarrow A \,[+]\, B$
$A > B \rightarrow A \wedge \overline{B}$	$A = B \rightarrow \overline{A \,[+]\, B}$
$A <= B \rightarrow \overline{A} \vee B$	$A * B \rightarrow A \wedge B$
$A <= B \rightarrow \overline{A \vee B}$	$A + B \rightarrow A \vee B$

[+] — oznacza różnicę symetryczną (ang. EXOR)

Stosując wyrażenia warunkowe (co znakomicie upraszcza programy) można modelować kombinacyjne układy logiczne (tzn. sieć bramek logicznych bez sprzężeń zwrotnych). W zapisach należy jednak pamiętać, że standardowe operatory logiczne (AND, OR, NOT) mają niższy priorytet niż wyrażenia arytmetyczne i należy stosować nawiasy. Wykorzystanie wyrażenia arytmetycznego A+B ogranicza się do przypadków, gdy nie może zajść sytuacja, że równocześnie A = 1 i B = 1.

A oto przykład programu umożliwiającego drukowanie tabeli logicznych:

```
10 FOR A=0 TO 1
20 FOR B=0 TO 1
30 FOR C=0 TO 1
.
.
240 FOR Z=0 TO 1
250 PRINT A;B;...Z;'=' ; (BADANE WYRAŻENIE LOGICZNE)
260 NEXT Z
270 NEXT Y
.
.
.
500 NEXT A
```

I jeszcze przykład wyrażenia odpowiadającego 4-bitowemu generatorowi parzystości:

```
A<>(B<>(C<>D))
```

Rozszyfrowywanie barwnych pasków na opornikach nie jest zajęciem superzłożonym. Dla kilku rezystorów nie opłaca się nawet ładować programu. Jeśli jednak w ramach wiosennych porządków będziemy musieli posegregować całe pudło, to ZX81 służy pomocą. Zamieszczone poniżej programy pracują w wersji angielskojęzycznej, może jednak ktoś z Czytelników pokusi się o ich „przetłumaczenie"?

W poniższym programie wystarczy wprowadzić tylko kilka początkowych liter nazwy koloru, tak aby nie mógł on pomylić się z innym:

```
 10 LET C$='BLACK BROWN RED   ORANGEYELLOW
    GREEN BLUE  VIOLETGRAY  WHITE GOLD  SILVER'
 20 DIM A(3)
200 SCROLL
105 FOR J=1 TO 3
110 PRINT "BAND ";J;"? "
120 INPUT B$
130 PRINT B$
135 SCROLL
140 FOR K=1 TO 11
150 IF B$ =C$(K*6+1 TO K*6+LEN B$) THEN GOTO 180
160 NEXT K
170 PRINT '?'
171 SCROLL
175 GOTO 110
180 IF K=10 THEN LET K=-1
181 IF K=11 THEN LET K=-2
185 LET A(J)=K
190 NEXT J
195 SCROLL
200 PRINT 'R= ' ; (10*A(1)+A(2))*10**A(3) ; ' OHMS'
205 SCROLL
206 SCROLL
207 CLEAR
210 GOTO 10
```

Drugi program wypisuje kolory kolejnych pasków dla podanej wartości rezystancji:

```
10 DIM B(3)
20 LET C$=' BLKBRONREDORGYELGRNBLUVLTGRAWHTGLD'
25 SCROLL
30 PRINT 'OHMS? ' ;
40 INPUT X
50 PRINT X
51 SCROLL
52 IF X  >=1 THEN GOTO 55
53 PRINT 'TOO SMALL'
54 GOTO 25
55 LET F=0
57 IF X  >=10 THEN GOTO 75
58 LET X =X*10
60 LET F=1
75 LET X=INT X
80 LET B(1)=VAL((STR$ X) (1))
100 LET B(2)=VAL((STR$ X) (2))
120 LET B(3)=(LEN(STR$ X))-2
121 IF F=1 THEN LET B(3)=10
130 FOR I=1 TO 3
135 LET B(I)=B(I)* 3+1
140 PRINT C$(B(I));C$(B(I)+1);C$(B(I)+2); ' ';
150 NEXT I
160 SCROLL
165 SCROLL
175 GOTO 25
```

Spostrzeżenia dotyczące możliwości ZX81 i powyższe procedury zaczerpnięto z rocznika 1982 czasopisma PERSONAL COMPUTER WORLD

AJP

dokończenie ze str. 12

karki są stale gotowe do pracy bez konieczności korzystania z pośrednictwa operatora (tzw. Hot-Reader i Hot-Printer).

Komputer końcowy dysponuje pamięcią operacyjną o pojemności 64 K bajtów i jest wyposażony w klawiaturę, monitor ekranowy, dwie jednostki pamięci na dyskach elastycznych, a niekiedy również w drukarkę. Oprócz tego na wielu wydziałach Uniwersytetu sprzężono je z projektorami, zapewniającym śledzenie podczas wykładów całego przebiegu konwersacji na dużym ekranie ściennym. Komputery te wyposażone są w system oprogramowania podstawowego AMBOSS 1, umożliwiających następujące tryby pracy: „off-line" (lokalny), konwersacyjny z komputerem centralnym, wsadowy z przesyłaniem zbiorów do komputera centralnego, a także wypisywanie tekstów na drukarkach komputerów typu 6.660.

REALIZACJA PROJEKTU

Firma SIEMENS przyjęła na siebie wszystkie funkcje generalnego dostawcy, łącznie z wykonaniem robót budowlanych oraz instalacji klimatyzacyjnej. Istniejący już poprzednio ośrodek obliczeniowy Uniwersytetu, jako przyszły główny użytkownik nowego systemu, wydzielił do współpracy z zespołem projektowym firmy SIEMENS odpowiednią grupę specjalistów. Wyjątkowo sprawny przebieg etapów projektowania i wdrażania systemu miał swoje źródło we wzorowej współpracy obu zespołów. W czasie właściwej realizacji projektu liczebność zespołu firmy SIEMENS dochodziła okresowo do 40 osób.

PROCES ODBIORU

Sprzęt i oprogramowanie systemu były odbierane w oparciu o „Szczególne warunki umów na sprzęt komputerowy i programy" ustalone przez Federalne Ministerstwo Spraw Wewnętrznych (Bundesinnenministerium) i obowiązujące wszystkie instytucje finansowe z budżetu państwowego. Ze względu na długotrwałość tych czynności przyjęto zasadę odbiorów częściowych. W szczegółach procedura odbiorów przestawiała się następująco:

- sprawdzenie działania komputera 7.882 z lokalnymi urządzeniami peryferyjnymi pod nadzorem systemu operacyjnego BS 3000, a także działania specyficznych programowych rozwiązań problemu administrowania dostępem użytkowników do zasobów komputera centralnego
- sprawdzenie działania komputera 6.660 jako stacji zdalnego wprowadzania danych (Remote — Job — Entry — Station) we współpracy z komputerem komunikacyjnym SIEMENS 3893, a także działania takich składników systemu operacyjnego BS 3000, jak JES (Job Entry System) oraz NCP (Network Control Program)
- sprawdzenie działania komputera 6.611 w warunkach pracy „off linie" z dołączonymi jednostkami pamięci na dyskach elastycznych i drukarką, a także działania systemu oprogramowania AMBOSS 1
- sprawdzenie działania specjalizowanych urządzeń przyłączonych do komputera centralnego oraz komputerów obsługi węzłów
- eksploatacja próbna całego systemu, ze szczególnym uwzględnieniem współdziałania wszystkich jego składników, zwłaszcza zaś współdziałania z komputerem końcowym (terminalem) w trybie „on line" z użyciem specjalizowanego oprogramowania użytkowego. Niezależnie od tego w fazie tej dokonano weryfikacji wszystkich testów sprawności systemu, jakie demonstrował dostawca przed zawarciem umowy.

Pierwsze cztery sprawdziany wykonywano jednocześnie z uwzględnieniem wspomnianych warunków odbioru państwowego. Odbywały się one w dni robocze od godz 8 do 17 (9 godzin!) przez okres 30 dni. Warunkiem niezbędnym odbioru było osiągnięcie w tym okresie minimum 100 godzin pracy efektywnej, przy wskaźniku strat czasu nie przekraczającym 10%.

Podczas eksploatacji próbnej wymagane było ponowne sprawdzenie działania wszystkich części składowych systemu, nawet już sprawdzonych z wynikiem pozytywnym w ramach cząstkowych testów i to zarówno w działaniu indywidualnym, jak i we współpracy z całym systemem.

Zgodnie z zawartą umową, w czasie przeprowadzania testu całościowego należało w szczególności sprawdzić, czy wyniki uzyskane podczas symulacji działania terminali oraz oszacowane wówczas czasy przepływu informacji w sieci są zgodne z wynikami pracy w warunkach rzeczywistych. Ażeby podczas konwersacji rzeczywistej stworzyć warunki identyczne z założonym podczas symulacji, zastosowano przy terminalach zamiast obsługi operatorskiej specjalne programy-roboty, opracowane przez pracowników ośrodka uczelnianego. Badania te wykazały czasy lepsze od uzyskanych podczas symulacji, a mianowicie 1,57 s w warunkach jednoczesnego działania 150 terminali.

EKSPLOATACJA I PLANOWANY ROZWOJ SYSTEMU

Już bezpośrednio po zainstalowaniu i uruchomieniu w sierpniu 1982 r. podstawowej części systemu, powstały warunki do rozpoczęcia obsługi użytkowników. Jednak przez okres rozbudowy systemu do pełnej konfiguracji, co nastąpiło w końcu maja 1983 r. ośrodek obliczeniowy Uniwersytetu eksploatował jeszcze poprzedni system komputerowy, jakkolwiek z malejącą już intensywnością obliczeń. Wynikało to z konieczności dokończenia wcześniej rozpoczętych zadań obliczeniowych, a także ułatwienia użytkownikom trudnej zazwyczaj dla nich operacji przestawienia się na nowy system. Od tej chwili eksploatacja systemu odbywała się przez 5 dni w tygodniu na dwie ośmiogodzinne zmiany, a następnie w miarę rosnącego zapotrzebowania również przez noc oraz w dniach wolnych od pracy, lecz bez obsługi operatorskiej.

Tak intensywne wykorzystywanie systemu spowodowały dwa istotne fakty. Po pierwsze, faktyczna liczba 37 000 studentów znacznie przekroczyła ustalenia przyjęte w fazie planowania systemu, a jednocześnie wskutek coraz szybszego rozwoju zastosowań komputerów konieczne było rozszerzenie udziału zajęć z informatyki w programie dydaktycznym uczelni. Po drugie, łatwa dostępność olbrzymich zasobów komputera 7.882 spowodowała skokowy wzrost zapotrzebowania na obliczenia naukowe. Niektóre zamierzenia badawcze, zaniechane wskutek niedostatecznej mocy obliczeniowej poprzednio zainstalowanego sprzętu lub przekazane do innych ośrodków obliczeniowych, uzyskały możliwość pełnej realizacji.

Wzrastające lawinowo zapotrzebowanie na informatyczne kształcenie studentów jest zaspokajane przez szybkie rozszerzanie dotychczasowego zakresu tematycznego wykładów i seminariów. Liczba korzystających z nich studentów w semestrze zimowym 1982/1983, w porównaniu do roku poprzedniego, podwoiła się do ponad 1000 osób dziennie.

Podczas realizacji zajęć dydaktycznych w pełni sprawdziła się duża elastyczność przyjętej koncepcji konfiguracji sprzętowej systemu, głównie dzięki niezawodności komputerów końcowych 6.611, zarówno w trybie pracy „off line" (do uruchamiania programów), w tym również zdalnej pracy wsadowej i przesyłania zbiorów do komputera centralnego, jak i pracy konwersacyjnej. Stwierdzono istnienie właściwego stanu równowagi pomiędzy obu trybami pracy, co zapewnia zbliżenie się do założonego optimum wykorzystania możliwości tego komputera.

W realizacji zadań wymagających szczególnie intensywnych obliczeń, a występujących np. w dziedzinie badań meteorologicznych lub oceanograficznych, system okazał się bardzo wydajny, zwłaszcza przy wykorzystywaniu translatora FORTRAN 77. Chociaż nominalna wydajność systemu, w porównaniu do uprzednio eksploatowanego komputera, zwiększyła się ok. 10 razy, to rzeczywisty wzrost wydajności w zakresie obliczeń typowych dla wspomnianych dziedzin zastosowań jest aż dwudziestopięciokrotny. Spowodowało to znaczne ożywienie zastosowań informatyki w wielu obszarach prac badawczych Uniwersytetu. Potwierdzają to licznie zgłaszane już żądania zwiększenia liczby terminali oraz przyłączenia do systemu specjalizowanych minikomputerów. Żądania te można spełnić bez większych trudności w ramach istniejącej mocy obliczeniowej komputera głównego i przepustowości komputerów komunikacyjnych, oczywiście pod warunkiem spełnienia wspomnianych ustaleń SNA. Przy podejmowaniu decyzji zakupu dla Uniwersytetu systemu komputerowego serii SIEMENS 7.800, istotną rolę odegrała gwarancja złożona przez producenta odnośnie możliwości rozbudowy systemu do 300 terminali, a także dodatkowego powiększenia jego mocy obliczeniowej. Jest zrozumiałe, że taka rozbudowa spowodować musi zwiększenie pojemności pamięci operacyjnej, liczby komputerów komunikacyjnych i pojemności pamięci dyskowej.

Pojawiły się również propozycje bardziej efektywnego wykorzystania systemu i zwiększenia jego możliwości obliczeniowych poprzez połączenie w sieć z innym dużym komputerem o podobnym przeznaczeniu.

Oprac.
WŁADYSŁAW KLEPACZ

CSK–COMPUTER STUDIO KAJKOWSKI

Produkcja mikrokomputerów
Oprogramowanie
Doradztwo informatyczne

81-505 GDYNIA–Orłowo
ul. Balladyny 3b
(zakład)

81-651 GDYNIA
ul. Konwaliowa 10 m. 25
tel. 24-01-50

Nareszcie dostępne w kraju! Oprogramowanie użytkowe niezbędne w każdej nowoczesnej firmie. Łatwe w obsłudze nawet dla nieinformatyków.

BANK DANYCH — CSK — system zarządzania relacyjną bazą danych,
TABPLAN — CSK — pakiet do planowania, sporządzania kalkulacji, zestawień i sprawozdań,
TEKST — CSK (w opracowaniu) — system redagowania i edycji tekstów oraz programów źródłowych,
TRANSCOM — CSK — system współpracy mikrokomputera z komputerami ODRA, MERA, RIAD.

CSK dostarcza gotowe pakiety programowe, umożliwiające tworzenie użytkowych systemów przetwarzania danych w dowolnym obszarze zarządzania — dla mikrokomputerów typu ELWRO 513, IMP-85, ROBOTRON 5120, LIDIA lub IBM PC, pracujących pod nadzorem systemów operacyjnych CP/M 2.2. lub CP/M 86.

Cechą charakterystyczną tych systemów jest łatwość obsługi. Specjaliści różnych dziedzin, nie przygotowani zawodowo do pracy z komputerem — mogą tworzyć systemy informatyczne.

Przykładowe systemy użytkowe, które można zbudować przy wykorzystaniu wymienionych produktów programowych, to:

- **systemy płacowe, finansowo-księgowe, magazynowe, osobowe, bibliograficzne** — zakładane, uruchamiane i eksploatowane za pomocą BANK DANYCH — CSK,
- **systemy planowania, kalkulacji, szybkiego sporządzania ofert, sprawozdań, zestawień** — zakładane i eksploatowane za pomocą TABPLAN — CSK,
- **systemy automatycznego prowadzenia korespondencji, pisania dowolnych tekstów i ich edycji, redagowania programów źródłowych** — tworzone za pomocą TEKST — CSK,
- **systemy współpracy z komputerami ODRA 1305, MERA 400 jako teletype lub stacja ICL 7020 za pośrednictwem sieci telefonicznej lub specjalizowanych sieci telekomunikacyjnych** — tworzone za pomocą TRANSCOM — CSK.

CSK dostarcza wyżej wymienione systemy bazowe z pełną dokumentacją użytkową w cenach od 100 tys. do 400 tys. zł — w zależności od obszaru zastosowań, konfiguracji mikrokomputera i dodatkowych wymagań użytkownika. Możliwe jest także opracowanie i wdrożenie konkretnego systemu użytkowego na zlecenie klienta.

BANK DANYCH – CSK

Jest to system zarządzania relacyjną bazą danych, pomyślany jako narzędzie dla tworzenia systemów użytkowych. Został opracowany przez CSK dla szerokiej rodziny mikrokomputerów wykorzystujących mikroprocesory 8080 i Z80.

Wielkość zbiorów danych przetwarzanych w systemie bazy danych ograniczona jest następującymi parametrami:
— liczba rekordów w zbiorze — maks. 65 535,
— liczba znaków w rekordzie — maks. 1000,
— liczba znaków w polu — maks. 254,
— liczba pól w rekordzie — maks. 32,
— liczba znaków klucza indeksowego — maks. 100.

Dane mogą mieć charakter numeryczny i tekstowy.

Wymagania sprzętowe:
— mikrokomputer oparty na mikroprocesorze 8080, 8085, Z80,
— system operacyjny CP/M lub kompatybilny,
— min. 48 KB pamięci operacyjnej.

Wymagania te spełniają m.in. następujące mikrokomputery: LIDIA, ELWRO 513, IMP 85, ROBOTRON 5110 i 5120.

EO/557/K/84

TABPLAN – CSK

Jest to pakiet programowy oparty o koncepcję ELEKTRONICZNEGO FORMULARZA. Umożliwia pracę na arkuszu o wymiarach 63 kolumny i 255 wierszy. Posiadając możliwość definiowania pól tworzących dany formularz (3—32 znaki) jako opis (pole alfanumeryczne) — pozwala na łatwe tworzenie systemów informatycznych.

Użytkownik tworzy systemy bezpośrednio na swoim stanowisku pracy, wybierając jedną z wielu podanych na ekranie funkcji: edycja, usuwanie, wstawianie, kopiowanie, przenoszenie, sortowanie, formatowanie, skalowanie, definiowanie, zerowanie, drukowanie itp. Dotyczą one zarówno pojedynczych pól, wierszy, kolumn, jak i całego formularza. Zaprojektowane i wypełnione formularze (kalkulacyjne, planistyczne, sprawozdawcze itp.) można zapisać na dysku lub wydrukować na drukarce.

Wymagania sprzętowe analogiczne jak dla BANKU DANYCH — CSK.

TRANSCOM–CSK (EMULATOR TTY)

Emulator TTY umożliwia współpracę mikrokomputera z komputerami serii ODRA 1300 w trybie MOP, pod kontrolą systemu operacyjnego GEORGE-3. Mikrokomputer, pracujący pod kontrolą systemu operacyjnego CP/M, zastępuje w tym układzie urządzenie typu dalekopis, a ponadto umożliwia transmisję zbiorów:
— typu tekst (w systemie CP/M) do zbiorów typu GRAPHIC (w systemie GEORGE-3),
— typu GRAPHIC (w systemie GEORGE-3) do zbiorów typu tekst (w systemie CP/M).

Emulator TTY umożliwia użytkownikowi dostęp do dużej maszyny cyfrowej w celu wykonania na niej obliczeń wymagających zaangażowania znacznych zasobów czasu procesora, pamięci operacyjnej i pamięci zewnętrznych. Po wykonaniu przetwarzania na komputerze, wynikowy zbiór może być przesłany z powrotem do mikrokomputera.

Wymagania sprzętowe analogiczne jak dla BANKU DANYCH — CSK.

CSK specjalizując się zarówno w sprzętowych, jak i programowych problemach grafiki mikrokomputerowej — oferuje:

USŁUGI SPRZĘTOWE:

- pomoc przy wyborze i zakupie ploterów oraz urządzeń do digitalizacji,
- przyłączanie tego typu urządzeń do komputera (w tym — wszystkich krajowej produkcji),
- serwis i naprawy gwarancyjne ploterów firm:

WATANABE (Japonia, RFN)
GOERZ (Austria),

- przystosowywanie telewizorów JOWISZ i NEPTUN do funkcji monitorów kolorowych z wejściem RGB.

OPROGRAMOWANIE NARZĘDZIOWE:

- do rozszerzania dowolnego języka programowania o instrukcje graficzne i obsługi ploterów,
- do obsługi procesu automatycznej digitalizacji dokumentów, zdjęć, rysunków, szkiców itd.,
- do wspomagania procesu tworzenia grafiki „trójwymiarowej".

SYSTEMY UŻYTKOWE:

- wspomagania procesu projektowania płytek obwodów drukowanych,
- wspomagania procesu kreślenia rysunków technicznych maszynowych,
- redagowania tekstu TEKST CSK z ploterem, jako urządzeniem drukującym (edycja pisma o dowolnym kształcie i kolorze czcionki).

Oferowane oprogramowanie pracuje pod kontrolą systemów operacyjnych CP/M 2.2. oraz CP/M-86.

CSK zaprasza na Targi

56 Międzynarodowe Targi Poznańskie

10—17 czerwca 1984

pawilon 2, na piętrze

EO/557/K/84

Zagrożenia informatyki

Redakcja INFORMATYKI zwróciła się do mgr. inż. Tadeusza Mazurkiewicza — Przewodniczącego Rady Zrzeszenia Przedsiębiorstw ZETO, a jednocześnie dyrektora jednego z najstarszych i najbardziej dynamicznych przedsiębiorstw tej sieci — ZETO Gdańsk — o przedstawienie swych poglądów, które naszym zdaniem są odbiciem oceny sytuacji tych wszyskich, którym leży na sercu przyszłość rozwoju zastosowań informatyki w naszym kraju.

Red. — Aktualna sytuacja w krajowej informatyce powoduje wzrost pesymizmu w całym środowisku. Coraz mniej słyszy się wypowiedzi uspokajających, natomiast zdecydowana większość zabierających głos naświetla przyszłość w bardzo ciemnych barwach. Jak Pan, jako czołowy przedstawiciel jednego z największych, zorganizowanych odłamów tego środowiska, ocenia tę sytuację?

Tadeusz Mazurkiewicz: — Kryzys naszej gospodarki wywołuje zjawiska, które grożą głębokiem regresem w informatyce, objawiającym się postępującym zanikiem zastosowań środków techniki obliczeniowej. W wyniku takiego regresu nastąpi:

- osłabienie tempa i zmniejszenie efektywności badań naukowych
- obniżenie poziomu rozwiązań konstrukcyjnych i efektywności technologii w przemyśle
- obniżenie efektywności organizacji i zarządzania
- zmniejszenie zdolności do wprowadzania innowacji.

Suma tych zjawisk spowoduje zepchnięcie naszej gospodarki — w wymianie międzynarodowej — głównie do roli dostawcy surowców, żywności i siły roboczej, gdyż inne nasze produkty po prostu nie wytrzymują konkurencji.

Red. — Jakie są, zdaniem Pana, podstawowe przyczyny regresu naszej informatyki?

T.M.: — Regres informatyki może powstać w wyniku utrzymywania się trzech podstawowych zagrożeń:

- Utraty wykwalifikowanych kadr, często wybitnych w skali międzynarodowej, a z reguły najbardziej zdolnych i przedsiębiorczych. Najlepiej ilustruje to fakt, że w latach 1980—1982 odeszło z informatyki 10 385 osób, tj. 18% w stosunku do zatrudnienia w 1980 r., Można spodziewać się, że w 1983 r. ubytek ten wzrośnie do ok. 25%.
- Zaniku działalności badawczo-rozwojowej w informatyce pomimo pewnych sukcesów w okresie kryzysu (np. opracowanie w ZETO Gdańsk sprawnego symulatora i emulatora dla bezpośredniego wykorzystania programów ODRY na komputerach RIAD).
- Dekapitalizacji środków technicznych informatyki przy równoczesnym braku możliwości ich odtworzenia i modernizacji: w 1987 r. ok. 40% komputerów osiągnie wiek 15 lat, a ponad 50% minikomputerów — wiek 10 lat, co oznacza praktyczną niezdolność do dalszej efektywnej pracy. W liczbach bezwzględnych oznacza to ok. 300 komputerów i ok. 800 minikomputerów, w sytuacji gdy przemysł dostarcza rocznie co najwyżej kilkanaście komputerów i ok. 50 minikomputerów.

Red.: — Jakie są źródła podanych przez Pana zagrożeń?

T.M.: — Uważam, że głównymi źródłami zagrożeń są:

- Niskie średnie płace w ośrodkach informatyki w porównaniu do średnich płac w gospodarce narodowej. Od 1979 r. do 1982 średnia płaca w informatyce wzrosła tylko o 88,5%, natomiast w całej gospodarce — średnio o 118%. Tendencja ta nie ulega zahamowaniu i w 1983 r. różnica poziomów płac przekroczy już 30%. Uważam, że przyczyną niskich płac w informatyce jest traktowanie ośrodków obliczeniowych przez istniejące mechanizmy gospodarki tak samo, jak przedsiębiorstw produkujących wyroby na bardzo chłonny rynek. Oprócz tego wielu przedsiębiorstwom produkcyjnym przyznano dodatkowe preferencje wynikające np. z przynależności do określonych resortów lub gałęzi gospodarki (ulgi w obciążeniach na PFAZ, ulgi w podatku dochodowym). Warto podkreślić, że w przedsiębiorstwach usług informatycznych w latach 1980—1981 produkcja spadła nieznacznie, a zwyżki cen usług były minimalne. W tym czasie w przemyśle czy budownictwie spadek produkcji był bardzo duży, a ceny wzrosły wielokrotnie. Mechanizm reformy w latach 1981—1983 preferował jednak tych, którzy mieli znaczny spadek produkcji i mogli ją szybko zwiększyć.
- Niepełne i niekonsekwentne wdrażanie reformy gospodarczej (częste zmiany reguły gry ekonomicznej), które owocują brakiem zainteresowania przedsiębiorstw w podejmowaniu optymalnych decyzji, a w konsekwencji również brakiem zapotrzebowania na usługi informatyczne.
- Niska efektywność prac badawczo-rozwojowych w dziedzinie zastosowań informatyki oraz rozproszenie środków na ten cel.
- Niewystarczające (ilościowo i asortymentowo) dostawy sprzętu z produkcji krajowej, która nadal kierowana jest głównie na eksport, bardziej opłacalny od sprzedaży odbiorcom krajowym.

W ten sposób cenne narzędzia mogące służyć wzrostowi efektywności naszej gospodarki i zdolne wymusić tak potrzebny postęp techniczny, ekonomiczny i organizacyjny, przekazujemy innym krajom. Oznacza to, że krajom tym ułatwiamy taki postęp, eliminując jednocześnie u siebie podstawowe jego narzędzia.

- Brak koordynacji rozwoju informatyki, który powoduje, że przy bardzo małym nasyceniu gospodarki środkami techniki obliczeniowej występuje jednocześnie rażąco słabe wykorzystanie istniejących zdolności obliczeniowych (średnio mniej niż 8 g/dobę, eksploatacja głównie jednoprogramowa) przy równoczesnym występowaniu silnych dążeń do zakupów nowych komputerów za granicą (w strefie dolarowej) oraz organizowania „sieci" ośrodków obliczeniowych. Bez odpowiedniego przygotowania organizacyjnego, które wyraźnie lekceważy się, postępowanie takie musi prowadzić do rezultatów znanych już z przykładu ZUS w Warszawie, co nie zraża jednak innych do podejmowania podobnych prób.

Red.: — A jakie mogą być konsekwencje tych zagrożeń?

T.M.: — Najważniejsze konsekwencje zagrożeń prowadzących do regresu naszej informatyki można by sformułować następująco:

— stopniowe obniżenie jakości wytwarzanych systemów informatycznych
— zahamowanie postępu w dziedzinie projektowania i eksploatacji systemów informatycznych
— zmniejszenie niezawodności, a w konsekwencji wydajności sprzętu komputerowego.

Sytuacja taka spowoduje szeroki odwrót dotychczasowych użytkowników od zastosowań informatyki, a więc zapoczątkuje proces stopniowej likwidacji znacznej liczby ośrodków obliczeniowych.

Red — Czy w opisanej, dramatycznej wręcz sytuacji, możliwe są skuteczne środki zaradcze?

T.M.: — Sądzę, że regres informatyki może zahamować szybkie podjęcie odpowiednich środków zaradczych. Za najważniejsze w chwili obecnej tego rodzaju działania uważam:

Po pierwsze — zrezygnowanie z traktowania informatyki jak przemysłu wyrobów rynkowych i zaliczenie jej do podstawowych stymulatorów przyspieszenia rozwoju oraz unowocześniania życia gospodarczego i społecznego. W konsekwencji takiej decyzji konieczne będzie podjęcie następujących działań:

- udzielenie przemysłowi zamówień rządowych na produkcję sprzętu komputerowego na potrzeby krajowe, zgodnie z profilem określonym przez użytkowników
- pozostawienie całości amortyzacji w przedsiębiorstwach usług informatycznych celem stworzenia warunków samofinansowania ich rozwoju
- udzielenie przedsiębiorstwom usług informatycznych ulg w podatku dochodowym oraz zapewnienie możliwości przeznaczenia wygospodarowanych środków finansowych na własny rozwój

• zwolnienie przedsiębiorstw usług informatycznych od obciążeń na PFAZ (np. do poziomu średniej płacy w przemyśle)

Po drugie — do czasu ustąpienia barier utrudniających naturalny rozwój zastosowań informatyki, stworzenie ośrodka koordynacji tego rozwoju z szerokimi uprawnieniami (m.in. w zakresie dysponowania środkami budżetowymi, określania zakresu zamówień rządowych itp.). Szczególnie istotne jest szybkie uruchomienie mechanizmów, zapewniających:

• ukierunkowanie produkcji sprzętu komputerowego zgodnie z rzeczywistymi potrzebami użytkowników krajowych
• stymulowanie rozwoju systemów informatycznych dla preferowanych dziedzin, rodzajów zastosowań i tematów
• kontrolę efektywności wykorzystania deficytowych środków technicznych informatyki
• opiniowanie zakupów importowanych środków technicznych informatyki i materiałów eksploatacyjnych.

Red.: — Jak w najbardziej lapidarny sposób mógłby Pan określić perspektywy rozwoju informatyki w warunkach reformy gospodarczej?

T.M.: — Reformy naszej gospodarki nie dokonamy rękami ale głową, do tego zaś niezbędnym narzędziem jest informatyka.

KW

Symptom reaktywizacji

Przygotowywane od jesieni środowiskowe spotkanie łódzkich informatyków z władzami, mającymi w swej pieczy informatykę, doszło ostatecznie do skutku w połowie stycznia br. Przypominało ono trochę radę wojenną w oblężonym mieście. Sformułowano kilkanaście tez w sprawie konieczności wyjścia z zapaści informatycznej i powołano specjalny zespół autorski do opracowania stosownego dokumentu pod przewodnictwem prof. Kąckiego.

Spotkanie łódzkie miało znacznie szerszy wymiar, niż mogłoby to wynikać z jego lokalnego charakteru. Nie tyle bowiem omawiano bolączki informatyków i kryzys informatyki, ile irracjonalną komputerofobię dotychczas ogłoszonych programów i planów wychodzenia z kryzysu gospodarczego.

W słowie wprowadzającym sekretarz Komitetu Łódzkiego PZPR, Konrad Janio, zauważył, że do spotkania przygotowywano się nie z myślą o „zamknięciu głosów krytykujących informatykę", ale właśnie pod hasłem otwarcia konstruktywnej dyskusji, mając w perspektywie szerzej rozumiane efekty gospodarcze, tkwiące w komputerze — tym potencjalnym narzędziu wspomagania decyzyjnego.

Marian Polski, dyrektor ZETO—Łódź, pełniący funkcję, gospodarza spotkania, podkreślił, że choć doraźnie najważniejsze są kadry, to jednak wybiegając w przyszłość należy skupić uwagę na niepokojąco dekapitalizującym się sprzęcie komputerowym. Wprawdzie zdrowa selekcja foruje najsilniejszych, ale gdy nie ma niczego poza elekcją, to walka o przetrwanie przekreśla rozwój. Wiarygodnym partnerem gospodarki mogą być tylko sprawne ośrodki obliczeniowe. Z kolei jednak wiarygodności gospodarki nie uzyska się forując pozorną dynamikę wzrostu wielu przedsiębiorstw, które ledwo osiągają poziom produkcji z roku 1979 — z jednoczesną dyskryminacją ośrodków informatyki, nawet tych, które w tym okresie zapisały na swym koncie wzrost produkcji usług. Jak wiadomo, wystarczyłaby zatem tylko zmiana szyldu organizacyjnego, aby znalazły się ułatwienia amortyzacyjne, zmniejszenia odpisów PFAZ-owskich oraz ulgi podatkowe — jeżeli miałoby się wątpliwe szczęście podlegania jednostce po prostu niegospodarczej. Efekt zaś jest taki, że poważna część wyszkolonej kadry zawodowych informatyków odchodzi do całkowicie nieinformatycznej działalności prywatnej.

Tomasz Pawlak, dyrektor Sekretariatu Komitetu Informatyki stwierdził że liczba „komputerów statystycznych" — 827 na przełomie 1982/1983 — przestaje budzić jakikolwiek optymizm gdy przyjrzeć się bliżej niepokojącemu maleniu nakładów inwestycyjnych na informatykę w latach 1977/1982. W tym okresie nastąpił spadek z 8,2 do 2,7 ‰ dochodu narodowego, ale w odniesieniu do całości nakładów inwestycyjnych kraju wyraził się jeszcze silniej: z 3,1 do 0,6 ‰. Pewien oficjany dokument danych liczbowych zawiera tzw. optymistyczny szkic rozwoju gospodarczego PRL do progu XXI wieku. Otóż „optymizm" polega tu na założeniu, że w roku 2001 krajowy park komputerowy będzie tylko o prawie połowę mniejszy niż w roku 1981. Podobno na „większe obciążenie informatyką" gospodarki narodowej nie stać. Mimochodem nasuwa tu się sarkastyczna uwaga, że „nieobciążona informatyką" gospodarka byłaby ewenementem na miarę krajów już chyba V świata.

Rozmiary ubytku najzdolniejszych i najbardziej ofensywnych informatyków próbował oszacować (w tysiącach osób!) Tadeusz Mazurkiewicz, przewodniczący Rady Zrzeszenia Przedsiębiorstw ZETO, w numerze przedstawiamy jego autoryzowaną wypowiedź — (przyp. red).

Prace na razie dosyć jeszcze hermetycznej Komisji ds. Reformy Gospodarczej komentował prof. Andrzej Straszak, uczestniczący w pracach Zespołu I tej Komisji. Ze zrozumiałych względów nie wyjawił szczegółów, ale chyba nie zdradził tajemnicy stwierdzając, iż wśród członków wymienionego ciała informatyka nie cieszy się specjalną sympatią, podobnie jak wśród członków rządu i aparatu partyjnego. Jedynym wyjątkiem zdaje się być łódzkie środowisko partyjne. Nie ma w tym jednak nic specjalnie dziwnego, skoro jeszcze w połowie lat siedemdziesiątych jedno z plenów KC uznało informatykę za dziedzinę dla naszego kraju nierozwojową. Zatem o dynamice informatyki nie decydowały u nas względy gospodarcze: zaś skutki kryzysu pogłębiły ostro zapaść samej informatyki w jeszcze większym stopniu niż krajowego budownictwa, uważanego zwykle za najgorszą dziedzinę polskiej gospodarki.

Ale za jakieś dwa—trzy lata eksponenci reformy gospodarczej znaczną odczuwać, że nie pozorowana płacowo reforma wymusza rozwój wspierającej ją informatyki — o ile tylko zostanie uprzednio wymuszone unowocześnienie księgowości. Wpadka z ZUS — prof. Straszak przeszedł do drugiego akcentowanego tematu — może się okazać w dalszych skutkach o tyle częściowo pozytywna, bo uwidoczniła, że system masowej rewaloryzacji rent i emerytur bez komputerów i łatwo modyfikowalnych programów byłby jeszcze mniej sprawny. A jako specjalista automatyk, mówca stwierdził z całą odpowiedzialnością, że komputeryzacja energetyki, zapobiegając gigantycznym stratom jakie powstały wskutek awarii krajowej sieci, z nawiązką opłaciła w zimie stulecia wszystkie dotychczasowe „obciążenia" gospodarki narodowej komputerami i informatykami.

Ostateczny akcent położony zaś został na niezbyt jeszcze głośny fakt, że w ubiegłym roku ZSRR podjął gigantyczny program rozwoju informatyki, któremu nadano rangę równie priorytetową, jak swego czasu programowi kosmicznemu.

W dalszych wystąpieniach wyodrębił się znacząco zgodny dwugłos lokalnych bankowców i gusowców, którzy twierdzili, iż wielu eksponentów władzy nie zdaje sobie sprawy, że większość informacji sprawozdawczych jest przetwarzana w sposób komputerowy. Trzeba więc także i w aparacie kadrowym wyrabiać mechanizmy doceniania informatyki, a informatyków uczyć bezkonfliktowego współdziałania.

GIEŁDA INFORMACJI

Teleks - terminalem ODRY - 1305

W Ośrodku Informatyki Centralnego Ośrodka Badawczo-Rozwojowego Przemysłu Włókienniczego-Południe w Bielsku-Białej zaprojektowano i wykonano urządzenie umożliwiające w trybie bezpośrednim wykorzystanie teleksu jako terminala konwersacyjnego komputera ODRA-1305. Tak więc każdy posiadacz teleksu stał się potencjalnym użytkownikiem systemu cyfrowego, pracującego pod kontrolą systemu operacyjnego GEORGE-3. Wykonane urządzenie może współpracować zarówno z multiplekserem MPX-325, jak i skanerem ICL 7930.

Rozwiązanie to, wykorzystujące istniejącą sieć teleksową, daje możliwość takich zastosowań, np.: rejestracja danych, ogólnokrajowe systemy wyszukiwania informacji, praca w systemie wielodostępnym (MOP).

Bliższych informacji udziela kierownik Działu Obsługi Technicznej Ośrodka Informatyki COBR PW-Płd., mgr inż. Józef Alaszewicz; 43-300 Bielsko-Biała, ul. Nadbrzeżna 12; tel. 282-91 wew. 94.

Sprzęt

HMN „Szopienice" w Katowicach posiada na zbyciu sprawne technicznie (mało używane) maszyny:
dwie sprawdzarki EC 9018 (prod. 1977)
cztery dziurkarki EC 915 (prod. 1977)

Adres: Huta Metali Nieżelaznych „Szopienice", Zakładowy Ośrodek Informatyki, 40-337 Katowice, ul. Obr. Westerplatte 87; tel. 59-35-10.

ARKADIA

Przedsiębiorstwo EMA-PROMEL w Gliwicach oferuje system ARKADIA, obliczający stan zanieczyszczeń powietrza atmosferycznego dla obiektów przemysłowych. System operacyjny: RT11-V03 lub RT11-V04. Język programowania: FORTRAN 4 plus. Minimalny zestaw sprzętu: minikomputer MERA-60 z pamięcią operacyjną 16 KB, dysk elastyczny dwukieszeniowy, drukarka wierszowa, monitor ekranowy.

Oprogramowanie systemu spełnia m.in. następujące funkcje:
— oblicza maksymalne stężenie zanieczyszczeń gazowych i pyłów zawieszonych w osi wiatru
— oblicza średnie roczne stężenie dla wybranego zanieczyszczenia gazowego lub pyłu zawieszonego, a także — dla zespołu zanieczyszczeń
— oblicza stężenie średnie i maksymalne oraz częstotliwość przekroczeń stężeń granicznych zanieczyszczeń typu gaz-pył zawieszony
— oblicza roczne opady pyłu dla różnych klas ziarnowych.

Informacji udziela inż. Jerzy Czyż; PROMEL, ul. Kościuszki 1, 44-100 Gliwice; tel. 32-15-25 wew. 281.

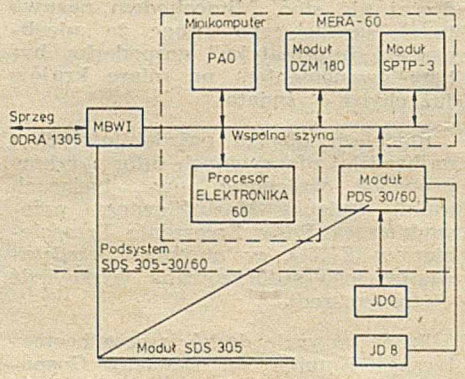
Część informacyjną spotkania zakończyło wystąpienie Lidii Jackiewicz-Kazaneckiej, posłanki uczestniczącej w pracach Sejmowej Komisji Nauki i Techniki. Komisja ta stworzyła elaborat (przesłany w połowie grudnia ub.r. do wicepremiera Messnera), w którym jest mowa m.in. o katastrofalnym powiększaniu się luki rozwojowej w informatyce nawet między PRL a innymi krajami RWPG. Dokument wypowiada się przeciwko restytuowaniu Komitetu Informatyki, popiera zaś ideę utworzenia Państwowego Komitetu Postępu Naukowo-Technicznego o szerokich uprawnieniach. Jest tam mowa o celowości pozostawiania nawet 100% odpisów amortyzacyjnych na inwestycje informatyczczne w ośrodkach obliczeniowych, a także postulowane są różne kroki mogące podnieść rangę informatyki.

ADAM B. EMPACHER

SDS 305—30/60 sterowanie pamięciami dyskowymi

W Zakładach Elektronicznych ELWRO we Wrocławiu uruchomiono w drugim półroczu ubiegłego roku system sterowania pamięciami dyskowymi SDS 305-30/60. Jest on przeznaczony do współpracy z pamięciami dyskowymi EC 5061-0 produkcji bułgarskiej (o pojemności 29 MB) i wraz z nimi stanowi podsystem pamięci dyskowej do komputera ODRA 1305. Nowa pamięć dyskowa — zarówno ze względu na swą stosunkowo dużą pojemność, jak i efektywność działania — znacznie rozszerza możliwości obliczeniowe tego komputera, a w szczególności — zapewnia efektywną pracę systemu operacyjnego GEORGE-3.

Podsystem SDS 305-30/60, którego schemat przedstawiono na rysunku 1, składa się z następujących modułów:
- minikomputera MERA-60, wyposażonego w procesor ELEKTRONIKA-60, półprzewodnikową dynamiczną pamięć operacyjną (PAO), konsolę w postaci drukarki DZM 180/KSR lub monitora MERA-7953 oraz stację taśmy papierowej SPTP 3
- bezpośredniej wymiany informacji MBWI
- sterowania pamięciami dyskowymi PDS 30/60.



Rys. 1. Schemat konfiguracji SDS 305-30/60

Moduły MBWI oraz PDS 30/60, wraz z autonomicznym zasilaniem, stanowią konstrukcyjnie wydzielony, zmontowany na panelu 19-calowym blok o symbolu SDS 305. Na identycznych panelach zmontowane są:
— procesor ELEKTRONIKA-60 wraz z PAO oraz układami sterowania konsolą i stacją taśmy papierowej SPTP-3
— stacja taśmy papierowej SPTP-3

Wymienione wyżej moduły podsystemu umieszczone są w szafie o wymiarach 590 × 902 × 113 mm, oprócz konsoli minikomputera, która znajduje się na zewnątrz tej szafy. Podsystem SDS 305-30/60 przyłącza się do ODRY 1305 za pośrednictwem standardowego sprzęgu (ang. interface) do kanału autonomicznego. Jednostki dyskowe łączone są z modułem PDS 30/60 za pośrednictwem sprzęgu typowego dla EC 5061.

Moduł MBWI stanowi adapter międzyprocesorowy, transformujący dane przesyłane z komputera ODRA z postaci znakowej na słowa dwubajtowe i odwrotnie oraz transformujący rozkazy standardowego sprzęgu ODRY 1305 na rozkazy sprzęgu MERY-60 i odwrotnie.

Moduł PDS 30/60 zawiera układy sterujące pamięciami dyskowymi, pośredniczące w przesyłaniu danych pomiędzy pamięcią operacyjną MERY-60 a dowolną z dziewięciu jednostek dyskowych, z uwzględnieniem wymaganego formatu zapisu danych na pakiecie dysku.

Procesor ELEKTRONIKA-60 wraz z programem sterującym (rezydującym w PAO) realizuje następujące funkcje:

— komunikacja z jednostką centralną ODRA 1305
— obsługa żądań transmisji
— bezpośrednie sterowanie jednostkami dyskowymi
— powtarzanie transmisji po wystąpieniu błędu
— usuwanie błędów przesuwu
— przesyłanie programu pierwotnego do jednostki centralnej
— komunikacja z operatorem poprzez konsolę.

Czytnik taśmy papierowej modułu SPTP-3 służy do wprowadzania do pamięci minikomputera programu sterującego oraz autonomicznych programów testujących.

Konsola w postaci drukarki mozaikowej DZM 180/KSR lub monitora ekranowego MERA-7953 zapewnia komunikację operatora z minikomputerem oraz sygnalizację błędów.

Wszystkie moduły SDS 305-30/60 połączone są z sobą sprzęgiem typu „wspólna szyna" i komunikują się według zasady „master-slave". Moduły PDS 30/60 i MBWI pracują w trybie bezpośredniego dostępu do pamięci (DMA).

*

Na rysunku 2 przedstawiony jest schematycznie programowy sprzęg podsystemu SDS 305-30/60. Nadrzędny system operacyjny ODRY komunikuje się z lokalnym programem sterującym za pomocą tzw. bloków sterujących, przesyłanych poprzez moduł bezpośredniej wymiany informacji MBWI. Poprzez MBWI przesyłane są także dane przeznaczone do zapisu na dysk lub odczytane z dysku, dla których buforem jest PAO. Bloki sterujące zawierają informacje sterujące, organizacyjne i kontrolne.

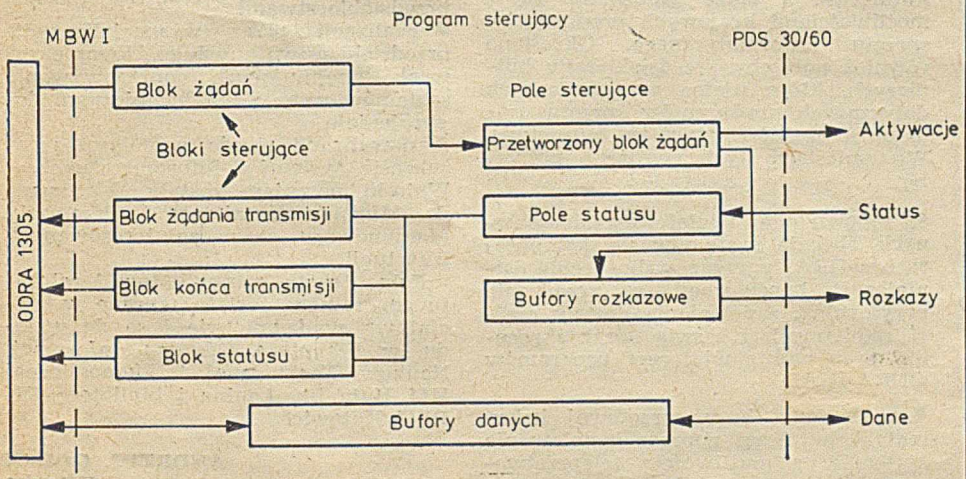
W zależności od kierunku przesyłania rozróżnia się następujące bloki sterujące:
blok żądań, przesyłany z komputera ODRA do MERY-60 i zawierający szczegółowe informacje o operacjach, jakie ma wykonać pamięć dyskowa **bloki statusu jednostki dyskowej oraz początku i końca transmisji danych**, przesyłane z MERY-60 do ODRY i zawierające informacje o realizacji żądań centralnego procesora, stanie jednostek dyskowych itp.

Każdy z procesorów może przerwać przepływ danych w celu przesłania bloków sterujących. W programie sterującym istnieje część służąca do bezpośredniej kontroli pamięci dyskowych. Do jej funkcji należą: adresowanie jednostek dyskowych, głowic i cylindrów, obsługa transmisji danych, obsługa przerwań związanych z dyskami, powtórzenia przesuwu głowic i transmisji danych w celu dokonania próby usunięcia błędów. Funkcje te realizowane są poprzez pole sterujące, bufory rozkazowe i bufory danych.

Przez bufor rozkazowy należy rozumieć pole w PAO opisujące rozkaz lub łańcuch rozkazów kanałowych, takich jak: pisz, czytaj, szukaj, kasuj. Dotyczy to boków HOME ADDRESS, LICZNIKA i DANYCH. Bufor rozkazowy zawiera typ rozkazu, długość bloku, odpowiedni do boku bajt synchronizujący, długość przerwy międzyblokowej, adres bufora danych oraz wskaźnik łańcuchowania.

Jeden łańcuch rozkazów może dotyczyć operacji w zakresie jednego cylindra. Program sterujący akceptuje dwa żądania transmisji na każdą jednostkę dyskową. Kolejność wykonania transmisji jest zgodna z kolejnością ich otrzymania.

Dla zainicjowania rozkazu kanałowego lub innych operacji dyskowych, takich jak: adresowanie jednostek dyskowych, głowic i cylindrów, odłączenie jednostki dyskowej, pytanie o status, powrót głowic na cylinder 0 — program sterujący wysyła do modułu PDS 30/60 tzw. aktywacje. Aktywacje są również wysyłane do modułu MBWI w celu zainicjowania odpowiednich operacji. Aktywacje są adresami MERY-60 w przedziale 28—32 K.



Rys. 2. Sprzęg programowy SDS 305-30/60

GIEŁDA INFORMACJI

WTRSPOOL

W Centrum Elektronicznym NBP w Warszawie zmodyfikowano program systemowego wyjścia WTR. Nowa wersja programu wypisującego zachowuje wszystkie funkcje standardowego programu.

Głównym celem tej modyfikacji było uzyskanie możliwości powtórnego wydruku wyników bez powtarzania całego zadania (częste awarie drukarek wierszowych). Powtórny wydruk zrealizowano przez jednoczesne z wydrukiem zapisywanie każdego z zadań do specjalnego zbioru wydruków (taśmowego lub dyskowego). Zapisy do tego zbioru (opcjonalne) są identyczne z drukowanymi rekordami. Obciążenie wprowadzonym dodatkowo zapisywaniem do zbioru wydruków nie powoduje zwolnienia działania programu WTR, wymaga natomiast większej pamięci operacyjnej dla Programu Systemowego Wyjścia o 2 Kb (obszar buforów dodatkowego zbioru). Nowa wersja programu wypisującego pozwala na znaczne zmniejszenie ilości drukowanego papieru, przez zastosowanie listy nazw zadań, których nie należy drukować (np. wydruki z często wykonywanych zadań operatorskich). Cena WTRSPOOL wynosi 51700 zł.

Informacji na temat oprogramowania udziela: Narodowy Bank Polski, Centrum Elektroniczne; ul. Świętokrzyska 11/21, 00-950 Warszawa; tel. 20-03-21 wew. 10-81 lub 10-77.

GEORGE-3

WPIPB „ETOB" oferuje ośrodkom eksploatującym komputery ODRA 1305 pod systemem operacyjnym GEORGE-3 procedury rozszerzające możliwości tego systemu o dynamiczny, bez wymiany pakietów dyskowych w trakcie pracy maszyny, przydział egzozbiorów mianowanych (w tym o lokalizacji dogodnej dla sortowań) — dla użytkowników. Przepustowość instalacji komputerowej wzrasta do poziomu, przy którym suma chwilowych potrzeb użytkowników na egzozbiory nie przekracza łącznej pojemności zamontowanych pakietów i jest niekiedy nawet kilkakrotnie wyższa niż w sytuacji, gdy wykorzystuje się tradycyjnie (statycznie) zalokowane egzozbiory.

Dostęp do pamięci dyskowej może być objęty systemem prioryterów, wyróżniających np. użytkowników MOP. Ponadto oferowana jest procedura realizująca koncepcję wielozbiorów w stosunku do zbiorów PZS formatu MT. Dodatkowe informacje zamieszczone zostały w INFORMATYCE nr 11, 1983, ss. 17—20.

Informacji udzielają przedstawiciele ETOB Warszawa: mgr inż. Jan Stepaniec — tel. 32-79-63 oraz mgr inż. Paweł Stasiewicz — tel. 32-02-42.

*

Dzięki zastosowaniu w podsystemie minikomputera, czas obsługi dysków przez system operacyjny ODRY 1305, jak również przeznaczony do tego celu obszar pamięci operacyjnej komputera, zostają znacznie zmniejszone. Osiągnięcia te w powiązaniu z zastosowaniem dysków o dużej pojemności pozwalają w istotny sposób zwiększyć efektywność wykorzystania komputera ODRA 1305. Podsystem jest wyposażony w bogaty zestaw testów autonomicznych i systemowych, które zapewniają jego dokładną funkcjonalną kontrolę oraz ułatwiają naprawę uszkodzeń.

Do testów autonomicznych należą testy: pamięci, przerwań, rozkazów, konsoli, urządzeń, we-wy taśmy papierowej, autonomiczne modułu PDS 30/ /60 (TPDSO1; testy 1—13) oraz testy współpracy między ODRĄ 1305 i MERĄ-60 poprzez moduł wymiany informacji TMWIO1-ESAZ; testy 1—20).

Do testów sytemowych należą (znane użytkownikom dysków 8 MB w zestawie ODRA) testy: #NFDA, #NFDB, #RDAM, #NFDE, #MPUZ. SDS 305--30/60 zapewnia zapis odpowiednich programów ładujących (ang. bootstrap) i egzekutorów na dysk oraz ich odczyt za pomocą operacji pulpitowej BOOTSTRAP komputera ODRA 1305.

Parametry techniczne i eksploatacyjne SDS 305—30/60

Maksymalna liczba dołączonych jednostek dyskowych: 9 w tym jedna zapasowa
Minikomputer lokalny: MERA-60
Długość słowa minikomputera: 16 bitów
Standard zapisu na pakiecie dysków: ICL
Liczba rekordów na ścieżce: 15
Liczba danych w rekordzie: 128 słów (384 bajty)
Maksymalna szybkość przesyłania danych: 104 K słów/s
Rodzaj pracy: całodobowy z możliwością wyłączeń
Zasilanie: 220 V (+10%—15%) częstotliwość 50 Hz ±2%
Moc pobierana: ≤1500 VA
Wymiary (mm): wysokość 1113, szerokość — 590, głębokość — 902
Masa: ok. 150 kg
Zakres temperatur pracy: 15—30°C
Wilgotność względna: 40—80%
Ciśnienie atmosferyczne: 840—1060 mbar
Zapylenie: maks. 1 g/m³, przy wielkości ziaren 0 < 3 μm
Atmosfera: stopień agresywności B wg PN-71/H-04651.

Istnieje możliwość:

- zapisu procedury ładowania egzekutora E6RM za pomocą programu #ENGF w zbiorze ICLKE6RMBOOT (w przypadku zapisu procedur dla obu egzekutorów) lub w zbiorze ICLK-HARDFILE
- zapisu procedury ładowania dla egzekutora EWG3 za pomocą samego EWG3 w zbiorze ICLKHARDFILE
- zapisu egzekutora ENG3 za pomocą programu #GIN.

Podsystem SDS 305-30/60 będzie miał możliwość zapisu na dysk odpowiedniej procedury ładowania dla programu sterującego oraz samego programu sterującego, a także odczyt programu sterującego z dysku do pamięci operacyjnej MERY-60 za pomocą operacji pulpitowej BOOTSTRAP w module SDS 305.

ANDRZEJ MROCZEK
Instytut Komputerowych Systemów Automatyki i Pomiarów Wrocław

KONFERENCJE

Kierunki rozwoju systemów informatycznych i ich bazy sprzętowej w hutnictwie

W dniach 14—15 listopada br. odbyła się w Ośrodku Wypoczynkowym KM Huta im. Lenina, w Bartkowej nad Dunajcem konferencja zorganizowana przez: Stowarzyszenie Inżynierów i Techników Przemysłu Hutniczego — Oddział Hutnictwa Żelaza i Stali w Krakowie, Kombinat Metalurgiczny Huta im. Lenina w Krakowie, Kombinat Metalurgiczny Huta Katowice, Centrum Informatyki i Badań Ekonomicznych Hutnictwa w Katowicach i Przedsiębiorstwo Usług Informatycznych METEKON w Katowicach.

W konferencji wzięli udział reprezentanci ośrodków ETO przedsiębiorstw hutniczych oraz przedstawiciele producentów sprzętu informatycznego. Na konferencji wygłoszono 15 komunikatów o działalności i zamierzeniach służb informatycznych hutnictwa oraz o zamierzeniach niektórych producentów sprzętu informatycznego. Ponadto został wygłoszony referat generalny bilansujący zamierzenia i możliwości rozwoju hutniczej informatyki.

Konferencja umożliwiła wymianę doświadczeń oraz zamierzeń dalszego rozwoju systemów informatycznych w hutnictwa, a także zapoznanie się z możliwościami krajowych producentów sprzętu informatycznego. Określono również potrzeby przedsiębiorstw hutniczych, które ujęto we wnioskach dotyczących producentów sprzętu oraz organizacji informatyki w hutnictwie. We wnioskach tych wyrażono potrzebę:

- zapewnienia ciągłości eksploatowanych obecnie systemów na komputerach ODRA, poprzez wzbogacenie istniejących konfiguracji o urządzenia peryferyjne, zapewnienie dostaw części zamiennych i serwisu oraz skrócenie prac nad emulatorem programów ODRA-RIAD
- rozbudowy bazy sprzętowej informatyki hutniczej poprzez instalowanie systemów z możliwością teleprzetwarzania i przetwarzania rozproszonego; warunkuje to rozwój systemów kierowania produkcją hutniczą
- usprawnienia serwisu technicznego producentów sprzętu
- zwiększenia dostaw materiałów eksploatacyjnych i urządzeń pomocniczych
- utworzenia hutniczej Rady Programowej Informatyki, której głównym zadaniem byłoby reprezentowanie interesów służb informatycznych hutnictwa oraz usprawnienie wymiany doświadczeń pomiędzy poszczególnymi przedsiębiorstwami
- realizacji przez wyspecjalizowane przedsiębiorstwa usług kompleksowych, obejmujących opracowywanie systemów wraz z ich technicznym dozbrojeniem
- nowelizacji prawa wynalazczego w zakresie systemów informatycznych.

Wnioski powyższe zostały skierowane do Ministra Hutnictwa i Przemysłu Maszynowego oraz zainteresowanych instytucji.

Pełne teksty komunikatów i referatu generalnego zostały zebrane w specjalnym wydawnictwie i są do wglądu w bibliotece Ośrodka Informacji Naukowo-Technicznej i Ekonomicznej KM Hutą im. Lenina i bibliotece CIBEH Katowice.

ANDRZEJ GOLEŃ
Katowice
WIESŁAW MIERZOWSKI
Kraków

Samotesty

W połowie lat siedemdziesiątych powstał w ramach amerykańskiego stowarzyszenia ACM (Association for Computing Machinery) zamysł opracowania pomocy szkoleniowych do indywidualnego użytku, umożliwiających „bez narażnia osobistego autorytetu" wykrywać luki we własnej wiedzy informatycznej. Z bezpośrednią inicjatywą przygotowania tej nowej formy sprawdzania posiadanych wiadomości wystąpiła grupa informatyków z Politechniki w Orlando (Floryda, USA), która pod przewodnictwem Terry J. Fredericka utworzyła tzw. Roboczy Komitet Samotestów.

Ideę samotestów — tym terminem będziemy określać nieco długą oryginalną nazwę Self-Assessment Procedures — przedstawiono na łamach miesięcznika Communications of the ACM, jeszcze w maju 1976, ilustrując ją gotową próbką 30 pytań samokontrolnych, dotyczących przykładowo wybranych trzech grup tematycznych [1]). Od tego czasu opublikowano już niemal tuzin samotestów tematycznych, wstępnie zweryfikowanych w różnych ośrodkach profesjonalnych, ale przewidzianych do rewizji po upływie „okresu półtrwania wiedzy informatycznej", szacowanym umownie na pięć lat. Nie jest wykluczone, że zaktualizowane samotesty będą w niedalekiej przyszłości publikowane w formie wydawnictwa seryjnego, a może nawet kolportowane w sprzedaży wysyłkowej np. przez ewentualnie utworzony w ramach ACM dział szkolenia samokontrolnego. W ankiecie dołączonej do wspomnianej publikacji w czasopiśmie ACM oprócz pytań o okres szkolenia informatycznego, długość stażu pracy, obszar zastosowań, typ pracy itp., znalazło się również zdanie: „Ile zapłaciłbyś za 50-pytaniowy samotest z zakresu intersującej cię problematyki — 5, 10, 20, 40 więcej dolarów?" Wiadomo, że w USA działalność szkoleniowa, zwłaszcza prowadzona pod auspicjami renomowanego stowarzyszenia, może stać się źródłem poważnych dochodów. Zaś samotesty ACM-owskie obejmują dosyć szeroki wachlarz problemowy: organizacja systemów, sortowanie wewnętrzne, narzędzia programistyczne, baza danych, modele kolejek systemowych, etyka informatyczna, kierowanie pracami oprogramowanionwymi, język ADA [2]) i in.

Zanim dorobimy się w Polsce własnych samotestów (a można by takie opracować przynajmniej dla podstawowych w naszym kraju komputerów i systemów) oraz zanim zaczną do nas docierać nowe wydania samotestów ACM — warto udostępnić naszym informatykom próbki aktualnie dostępnych opracowań wspomnianego Komitetu Fredericka.

Chociaż prawa autorskie wydawnictwa ACM są pedantycznie chronione odpowiednimi zastrzeżeniami — nie tylko przy okładce, ale i na każdej stronie rozpoczynającej nowy artykuł, to wyjątkowo przy informacji o samotestach umieszczono generalne upoważnienie do ograniczonego przedruku. W sumie warunki ograniczające są tylko trzy, zresztą — nie uciążliwe, a wynikające z intencji pomysłodawców samotestu jako dyskretnego sprawdzianu własnej fachowości (niefachowości) tylko przed samym sobą — oraz „niestwarzania możliwości osiągania zysków przez profesjonalne ośrodki szkoleniowe drogą odpłatnego kolportowania samotestów bez upoważnienia ACM". Trzeci warunek sprowadza się do formalnego zacytowania in extenso ośmiu wierszy samego zastrzeżenia z przytoczeniem danych bibliograficznych zawierających wyraźną datę publikacji oryginalnej.

Zanim w jednym z następnych numerów INFORMATYKI przytoczymy przykładowy samotest, warto stwierdzić różnice w stosunku do tradycyjnych sprawdzianów kwestionariuszowych oraz testów weryfikacyjnych. W tych ostatnich wybranie czy wpisanie właściwej odpowiedzi kończy procedurę — natomiast w samoteście dopiero ją rozpoczyna. Osoba korzystająca z samotestu czyta wyjaśnienia prawidłowych odpowiedzi, porównuje je z własnymi odpowiedziami, szuka pogłębienia wiadomości we wskazanych źródłach bibliograficznych, a wreszcie sama określa koniec uzupełniania wiedzy w sprawdzonej dziedzinie. Zastrzeżenie ACM wyraźnie zabrania wykorzystywania samotestu do sprawdzania posiadanej wiedzy przez osobę drugą! Nie wolno więc traktować samotestu jako narzędzia egzaminacyjnego, testu warunkującego przyjęcie na kurs szkoleniowy, ani nawet podawać choćby jednego pytania jako zadania kontrolnego do oceny postępu szkolenia informatycznego, co stanowiłoby wypaczenie intencji autorów tej metody.

Samotesty nie narzucają określonej metody postępowania — jak podkreślają autorzy we wstępie do każdego kolejnego przykładu. Można oczekiwać, że lekturę testu jedni zaczną właśnie od odpowiedzi, drudzy — od wskazówek bibliograficznych, inni zaś od listy pytań, ale np. od jej końca! Każdy sposób indywidualnego wykorzystania samotestu będzie bowiem dobry, jeżeli po jego zakończeniu zainteresowany odczuje, iż jako informatyk nauczył się czegoś nowego i istotnego w jego pracy zawodowej.

Jak podają autorzy samotestów, pytania na początku działalności komitetu były wybierane głównie z istniejących podręczników. Potem nastąpił dłuższy okres praktycznej weryfikacji zestawu pytań i ich treści, a także wariantów odpowiedzi i wskazówek wyjaśniających. W procesie weryfikacji, a następnie modyfikacji samotestów uczestniczyli zarówno autorzy wykorzystanych podręczników, jak i recenzenci spoza Komitetu, w sumie kilkaset osób.

Dotychczas opublikowane samotesty [3]) są wyraźnie określone jako próbne. Zestawy pytań samotestowych dla wszystkich możliwych do wyodrębnienia kilkuset grup tematycznych w różnych działach informatyki zostaną opracowane, o ile znajdą się na ten cel fundusze, niezbędne na opłacenie dydaktyków i weryfikatorów tak monstrualnego zadania. Jest zrozumiałe, że idea samotestów wyklucza przyjęcie jakichkolwiek limitów czasowych na opanowanie zagadnień określonych w poszczególnych pytaniach. Nawet najmniejszego samotestu nie trzeba opanowywać w sposób ciągły, można np. rozłożyć go na porcje obejmujące tylko jedno pytanie dziennie, jeżeli — oczywiście — starczy czasu na dokładne przeanalizowanie problemu oraz związanej z nim lektury uzupełniającej.

Wprawdzie z samotestów można korzystać w dowolny sposób, ale autorzy sformułowali w tym zakresie następujące sugestie:

- w miarę lektury kolejnych pytań należy zakreślać odpowiedzi uznane za najwłaściwsze
- po dojściu do końca listy pytań należy porównać udzielone odpowiedzi z wzorcowymi
- najwcześniejsza niewłaściwa odpowiedź wskazuje potrzebną literaturę naprowadzającą
- jeżeli temat nie jest zbyt interesujący, można poprzestać na krótszych wyjaśnieniach
- jeżeli podane na końcu samotestu wyjaśnienia nasuwają wątpliwości, należy nie ociągać się z dotarciem do podanej literatury uzupełniającej.

Oczywiście, osoby posiadające mniejszy zasób podstawowej wiedzy teoretycznej będą mogły powtarzać cykl samokontrolny tak długo, aż odpowiedzą na wszystkie pytania z poczuciem pełnego rozumienia problemu.

Można sądzić, że od czasu opublikowania pierwszego z omawianych samotestów, wiele środowiskowych spotkań informatyków na świecie było poświęcone omawianiu samej idei samotestów i związanej z nią dobrowolną samoweryfikacja wiedzy informatycznej. Przykładowo — wyobrażamy sobie, że tego rodzaju spotkanie być może uda się zorganizować jeszcze w bieżącym roku na terenie łódzkim, zanim przykłady pierwszych samotestów zostaną opublikowane na łamach INFORMATYKI. Jest chyba o co bić się skoro prezes ACM, Jean P. Sammet uznał samotesty za najważniejsze osiągnięcie tego stowarzyszenia w dekadzie lat siedemdziesiątych!

ADAM B. EMPACHER
JERZY L. ROSSOWSKI

---

[1]) A — Budowa programów a modularność (11 pytań), B — Indeksowanie danych i przeszukiwanie zbiorów (9), C — Buforowanie i bloki rekordów (10)

[2]) Por. INFORMATYKA nr 11, 1983, s. 2—6: Język ADA w testach

[3]) Ostatni samotest (XI) opublikowano w lipcowym numerze CACM z ub.r. — tym razem dotyczący początków historii informatyki

Komputerowe stresy

Amerykański tygodnik TIME dokonuje systematycznie wyboru człowieka roku. W styczniu 1983 zamiast człowieka pojawiła się maszyna roku — komputer. Autorzy obszernego artykułu, podkreślając rosnącą rolę maszyn cyfrowych w życiu Amerykanów, pisali jednocześnie o niepokojącym zjawisku, wzrastającej niechęci społeczeństwa do informatyzacji pracy biurowej.

Zarówno sekretarki, jak i ludzie na kierowniczych stanowiskach, są zwykle podejrzliwi w stosunku do nowego sprzętu, szczególnie gdy stanowi zagrożenie dla ich pracy. Niektórzy starzy urzędnicy opierają się używaniu klawiatury, twierdząc, że jest to poniżające. Dwaj kierownicy w pewnej wielkiej firmie odmawiali czytania jakichkolwiek tabulogramów, dopóki sekretarki nie przepisały ich w formie standardowych zestawień. TIME cytował opinie specjalistów: „Kierownicy nie korzystający z urządzeń zewnętrznych komputera mogą w ciągu trzech do pięciu lat stać się nieprzydatni z punktu widzenia organizacji pracy. Trzeba powiedzieć — bezużyteczni".

Artykuł „Leczenie cyberofobii" z marcowego (1983) numeru amerykańskiego miesięcznika PSYCHOLOGY TODAY przedstawia już dokładniejszą analizę tego zjawiska. Odkąd komputery częściej pojawiają się w biurach, coraz więcej osób na odpowiedzialnych stanowiskach jest nękanych objawami czegoś, co specjaliści są skłonni nazywać „komputerofobią" lub „cyberofobią" — lękiem, brakiem zaufania lub wręcz nienawiścią do maszyn liczących.

Sanford Weinberg z filadelfijskiego Uniwersytetu St. Joseph przeprowadził wywiady z kilkuset wykładowcami i studentami, posługującymi się komputerami, i przetestował niektórych z nich podczas pracy na terminalach — za pomocą urządzenia mierzącego reakcję skórno-galwaniczną. Stwierdził, że prawie jedna trzecia badanych cierpi na cyberofobię. Około pięciu procent zaobserwowanych zachowań wskazywało na objawy klasycznej fobii: mdłości z obrzydzenia, zawroty głowy, zimne poty, wysokie ciśnienie krwi. Jeden ze sfrustrowanych pracowników przewrócił podczas wywiadu filiżankę z kawą i rzucił popielniczkę na konsolę komputera.

David Cossey, dyrektor ośrodka obliczeniowego Uniwersytetu Pensylwania, zaobserwował wśród swoich studentów „szok terminalowy". Podejrzewa on, że określenie rozmiarów problemu jest trudne, ponieważ wielu cierpi na „ukrytą cyberfobię" z obawy przed ujawnianiem swych lęków wobec powszechnego wychwalania korzyści płynących z zastosowania komputerów.

Z badań przeprowadzonych przez firmę konsultacyjną BOOZ, ALLEN & HAMILTON wynika, że chociaż starsi przedstawiciele kadry kierowniczej mniej chętnie widzą komputery w swoich biurach, to problem nastawienia ludzi do maszyn nie jest związany wyłącznie z wiekiem. Kierownik, który ma za sobą 25 lat pracy w jednym miejscu będzie bardziej bronił się przed informatyzacją niż jego rówieśnik, który często zmieniał pracę. Podobnie poziom wykształcenia nie jest czynnikiem decydującym o stosunku do komputerów. Jest nim natomiast umiejętność posługiwania się klawiaturą, najistotniejsza przy uczeniu się pracy z terminalem. Należy pamiętać, że samo pisanie na maszynie jest tradycyjnie traktowane jako „gorsze" zajęcie, zatem sekretarki będą mniej narażone na „komputerowe" stresy.

Inna ważna obserwacja dokonana przez pracowników wspomnianej firmy konsultacyjnej dotyczy „czynnika samozadowolenia". Kierownicy, którzy wierzą, że zawsze optymalnie wykorzystują swój czas pracy, są o wiele gorzej nastawieni do komputerów niż ci, którzy stale dążą do zwiększenia efektywności swoich działań. Cyberfobia staje się chorobą kadry kierowniczej średniego szczebla, coraz częściej przekonanej, że nie podejmuje już decyzji, że staje się jedynie „przedłużeniem konfiguracji komputera", którą w dodatku można w każdej chwili zastąpić maszyną. Weinberg proponuje swym studentom przezwyciężenie lęku dzięki pracy na elektronicznych kalkulatorach, grom komputerowym oraz pisaniu prostych programów. Bostoński FIRST NATIONAL BANK stworzył ośrodek obliczeniowy, gdzie udziela się kierownikom indywidualnych porad i instrukcji. Mogą oni również wypożyczać do domu małe komputery. Podobno takie metody oswajania z informatyką przynoszą niespodziewanie dobre rezultaty.

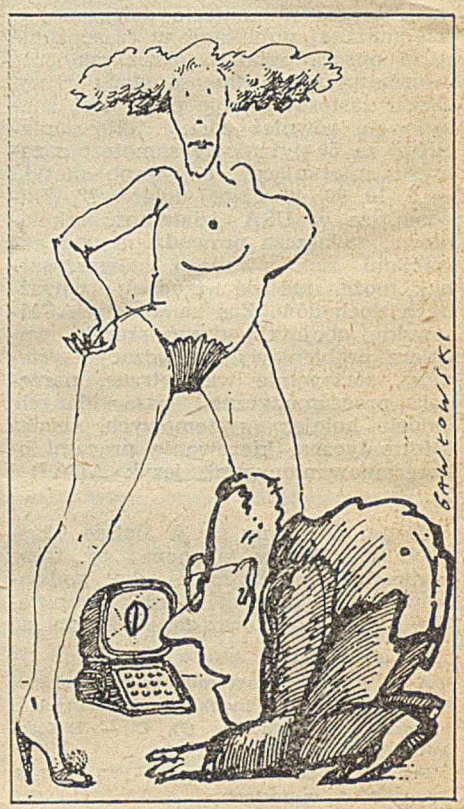
Ale są i tacy, których oswajać wcale nie trzeba. Najlepszym dowodem jest list czytelniczki zamieszczony w sierpniowym numerze z ub. r. wspomnianego miesięcznika. Pisze ona:

„Phil i ja byliśmy małżeństwem przez siedem lat, gdy to się stało. Od roku nie jest już tak jak dawniej. Ale cały czas jesteśmy razem: Phil i ja, i Radio Shack TRS-80 Model 1. Nasze małżeństwo przetrwało — domowe komputery zatrzymują mężów w domach.

Dokładnie pamiętam nasze pierwsze Święto Dziękczynienia, kiedy pojawił się Model 1. Stół był cudowny — świece, złocisty indyk, wspaniała uczta, na której przygotowanie straciłam dwa dni. Brakowało tylko mężczyzn. Stali w salonie stłoczeni wokół komputera i grali w ‚Nawiedzony dom', ‚Atak obcych' i ‚Szukanie piramid'.

Na ostatnim zjeździe rodzinnym powiedziałam kobietom, że chcę napisać o tym, co komputery robią z naszego życia domowego. Moja teściowa, cudowna kobieta, której równowagi nie zachwiało nawet wychowanie siedmiorga dzieci, stwierdziła z przekąsem: ‚Wiesz, odkąd zjawił się komputer, nie jesteście już parą'. Moja szwagierka Barbara dodała: ‚Zdaję sobie sprawę, że brzmi to nieco dziwnie, ale czasami... czasami czuję, że w domu jest jakiś intruz'. Jej mąż, Chip, jest jednym z tych zapaleńców, facetów przeważnie samotnych albo rozwiedzionych, którzy całymi nocami grzebią się w bitach i programach, pogrążeni w rozkoszach oprogramowania. Jeżeli nie siedzą przy terminalach to przyłażą do Chipa, żeby pogadać o swoich maszynach. Co więcej — telefon dzwoni od samego rana, ilekroć któremuś z nich coś strzeli do głowy. Uważają, że tak jest w porządku, skoro spędzają noce przy maszynach.

Dlaczego w takiej sytuacji kobiety zgadzają się na komputery w domach? W mojej rodzinie wszyscy mężczyźni używają maszyn do pracy w domu i zgodnie twierdzą, że są dzięki temu dwa albo trzy razy lepsi i mądrzejsi, więc pełni zachwytu stukają w klawiaturę. Czy jakakolwiek żona może odmówić mężowi czegoś, co czyni go tak szczęśliwym?". **(ki)**



Dokumentacja oprogramowania

Wydaje się, że w rubryce terminologicznej warto podać więcej szczegółów dotyczących dokumentacji oprogramowania. W numerze 3/84 zdefiniowano (za normę IEEE-729) jedynie dwa dokumenty — plan testowania i sprawozdanie z testów. Obecnie spróbuję wyjaśnić znaczenie innych dokumentów powstających w cyklu istnienia oprogramowania (por. INFORMATYKA, nr 1/84). Odpowiednie pojęcia scharakteryzowano w obszernym raporcie pt. „Software Development and Maintenance Guidelines", przygotowanym w ubiegłym roku dla amerykańskiego Electric Power Research Institute w Palo Alto, w Kalifornii (Topical Report No. E4-3089).

W całym cyklu produkcyjnym oprogramowania wyróżniono jedenaście dokumentów różnej wagi, stanowiących — łącznie z kodem źródłowym — przedmiot kompletnej dostawy oprogramowania. Dość dobrze zdefiniowane fazy, składające się na cykl produkcyjny, omówiliśmy w numerze 1/84. W trakcie początkowych faz cyklu powstaje szereg dokumentów bardzo istotnych dla procesu wytwarzania oprogramowania, choć nie przeznaczonych bezpośrednio dla użytkowników programu. Są to: specyfikacja wymagań, specyfikacja projektu wstępnego i szczegółowego oraz plan testowania.

Specyfikacja wymagań (ang. requirements specification) jest to dokument, który w sposób precyzyjny opisuje wymagania stawiane oprogramowaniu. Jest pierwszym dokumentem przygotowanym przez wytwórcę oprogramowania i stanowi podstawę do dalszych prac nad tym oprogramowaniem. Specyfikacja wymagań powinna zawierać m.in. opis problemu (np. jego teorię i równania matematyczne), opis sprzężenia użytkowego (tzn. wszystkich danych wejściowych i wyjściowych oraz sposobu współpracy użytkownika z programem), wymagania funkcjonalne (tzn. operacje, które należy wykonać na danych wejściowych, aby otrzymać dane wyjściowe), wymagania wykonawcze (ograniczenia czasowe, zajętość pamięci, dokładność, niezawodność, modyfikowalność itp.), opis testów odbiorczych i charakterystykę kryteriów odbioru. W dokumencie tym powinny znaleźć się także dane dotyczące środowiska eksploatacyjnego programu, a więc — opis sprzętu, oprogramowania podstawowego, wymaganych zabezpieczeń itp. Ponadto specyfikacja wymagań powinna zawierać plan produkcji oprogramowania, tj. m.in. harmonogram zakończenia poszczególnych etapów — od projektu wstępnego aż do dostawy i odbioru, wymagania osobowe, wymagania dotyczące zasobów i inne.

Specyfikacja projektu wstępnego (ang. preliminary design specification) jest to dokument, który opisuje projekt oprogramowania z dokładnością do modułu (podprogramu lub funkcji). Zawiera plan opracowania oprogramowania, plan testów i opis środowiska, w którym będzie wytwarzane oprogramowanie. Przedstawia hierarchiczny opis struktury programu, opis zastosowanych modeli matematycznych i algorytmów, opis danych wejściowych programu i ich zakresów, opis danych wyjściowych i prowadzących do nich operacji na danych wejściowych, opis systemu komputerowego i oprogramowania, które będzie użyte do opracowania programu, a ponadto — plany realizacji i testowania projektu. Specyfikacja projektu wstępnego powstaje na podstawie specyfikacji wymagań.

Specyfika projektu szczegółowego (ang. detailed design specification) jest dokumentem, który opisuje szczegółowy projekt programu na poziomie logicznym wewnątrz modułów. Zawiera także plan opracowania oprogramowania, plan testów i opis środowiska, w którym będzie powstawać oprogramowanie. Podstawą do opracowania tego dokumentu jest specyfikacja projektu wstępnego, w stosunku do której jest on rozszerzony o opis struktury poszczególnych modułów, ich danych wejściowych i wyjściowych wraz z parametrami i zakresami oraz — o plany realizacji i testowania pojedynczych modułów. W szczególności, opisuje się wszystkie struktury sterujące (rozgałęzienia, pętle, wybory itp.) oraz struktury danych (pliki, zmienne globalne, zmienne współdzielone itd.). Dokument ten musi stanowić pełny i dokładny opis programu oraz umożliwiać bezpośrednie opracowanie na tej podstawie programu, bez konieczności wprowadzania uzupełnień lub podejmowania dodatkowych decyzji.

Plan testowania (ang. test plan) jest do dokument, który opisuje zakładany przebieg testowania oraz plan analizy jego wyników, po opracowaniu i uruchomieniu oprogramowania. Plan testowania zawiera szczegółowy opis wykonania wszystkich testów, opis redukcji wyników oraz opis kryteriów ich oceny. W szczególności, podaje się charakterystyki i właściwości programu podlegające testowaniu oraz charakteryzuje dane wejściowe każdego testu i pożądane dane wyjściowe wraz z kryteriami ich przyjęcia lub odrzucenia. Plan testowania powstaje głównie na podstawie specyfikacji wymagań.

W dalszej kolejności scharakteryzuję inne dokumenty związane z produkcją oprogramowania, przeznaczone bezpośrednio dla jego użytkowników.

JANUSZ ZALEWSKI

KALENDARZ

Czerwiec

4—6, Nicea (Francja): kolokwium na temat predyspozycji w programowaniu — organizator: AFCET
7—8, SophiaAntipolis (Francja): konferencja na temat jakościowej oceny predyspozycji w programowaniu — organizator: AFCET
1—13, Paryż: PROLAMAT — międzynarodowa Konferencja na temat języków programowania dla obrabiarek — organizatorzy: IFIP oraz IFAC
17—21, Kopenhaga: XXVI międzynarodowa konferencja TIMS — organizator: The Institute of Management Sciences
26—29, Rzym: II światowa konferencja na temat międzynarodowego przepływu danych — organizator: Intergovernmental Bureau for Informatics

Lipiec

20—28, Montreal (Kanada): VII międzynarodowa konferencja na temat rozpoznawania obrazów — organizator: International Association for Pattern Recognition

Sierpień

20—28, Montreal (Kanada): VII międzynarodowa konferendowe sympozjum statystyki obliczeniowej — organizator: IASC
28—30, Kopenhaga (Dania): EUROMICRO'84 — 10. międzynarodowe sympozjum na temat mikroinformatyki i mikroprogramowania — organizator: EUROMICRO

Wrzesień

3—17, Londyn: INTERACT'84, międzynarodowa konferencja na temat czynnika ludzkiego w systemach informatycznych oraz współdziałania człowiek-maszyna — organizatorzy: IFIP, IFAC, IFORS oraz IEA
10—14, Paryż: VI międzynarodowy kongres cybernetyki i teorii systemów — organizator AFCET oraz World Organization of General Systems and Cybernetics
17—21, Paryż: międzynarodowa konferencja „Convention Informatique'84" oraz wystawa „SICOB'84" — organizator: SICOB
24—28, Hong Kong: międzynarodowa konferencja pn. „Technologia informatyczna — środek maksymalizacji potencjału gospodarczego krajów azjatyckich" — organizator: SEARCC (South Eeast Asia Regional Computer Confederation)

Trojniar W.: FORTH — język i system programowania (1)

INFORMATYKA 1984, nr 5, s. 1

Pierwsza część charakterystyki języka i systemu programowania FORTH, zawierająca prównanie z innymi językami programowania oraz omówienie funkcji i sposobu użycia podstawowych elementów konstrukcji tego języka.

Тройняр В.: FORTH — язык и система программирования (I)

INFORMATYKA 1984, № 5, стр. 1

Первая часть характеристики языка и системы программирования FORTH содержащая сравнение с другими языками программирования и обсуждение функций и способа применения основных элементов конструкции этого языка.

Zieliński C.: Roboty (1). Konstrukcja i możliwości zastosowań

INFORMATYKA 1984, nr 5, s. 5

Omówienie światowego rozwoju konstrukcji kolejnych generacji robotów ze wskazaniem osiągnięć krajowych oraz przykładowych możliwości zastosowań.

Зелиньски Ц.: Роботы (I). Конструкция и возможности применений

INFORMATYKA 1984, № 5, стр. 5

Обсуждение мирового развития конструкций очередных поколений роботов с указанием отечественных достижений и примеров применений.

Szuba T.: PROLOG-PASCAL: automatyczna synteza programu dla obrabiarek

INFORMATYKA 1984, nr 5, s. 8

Automatyczna synteza programu obróbki skrawaniem jako przykład nowoczesnego rozwiązania oprogramowania z wykorzystaniem metod sztucznej inteligencji oraz języków programowania PROLOG i PASCAL.

Шуба Э.: PROLOG-PASCAL автоматический синтез программы для станков

INFORMATYKA 1984, № 5, стр. 8

Автоматический синтез программы обработки резанием как пример современного решения программного обеспечения с использованием методов искусственного интеллекта и языков программирования PROLOG и PASCAL.

Trojniar W.: FORTH — the language and programming system (1)

INFORMATYKA 1984, No. 5, p. 1

First part of the FORTH language and programming system characteristics, which includes comparison with other programming languages and discussion of functions and handling methods of the language basic structural elements.

Trojniar W.: FORTH — Programmiersprache und -System (1)

INFORMATYKA 1984, Nr. 5, S. 1

Erster Teil einer Charakteristik von FORTH Programmiersprache und -System, die eine Vergleichung mit anderen Programmiersprachen, sowie eine Besprechung von Funktionen und Gebrauchsweise der Strukturgrundelemente dieser Sprache, umfasst.

Zieliński C.: Robots (1). Construction and application possibilities

INFORMATYKA 1984, No. 5, p. 5

Construction development of successive robot generations with indication for home achievements and examplary application possibilities are discussed.

Zieliński C.: Roboter (1). Konstruktion und Anwendungsmöglichkeiten

INFORMATYKA 1984, Nr. 5, S. 5

Eine Besprechung der weltweiten Konstruktionsentwicklung von aufeinaderfolgenden Robotergenerationen mit Betonung der einheimischen Leistungen und beispielhaften Anwendungsmöglichkeiten.

Szuba T.: PROLOG — PASCAL: automatic synthesis of machine tool's program

INFORMATYKA 1984, No. 5, p. 8

Automatic synthesis of cutting treatment program as an example for software solution based on artificial intelligence methods, as well as on PROLOG and PASCAL programming languages application.

Szuba T.: PROLOG-PASCAL: automatisierte Synthese des Programmes für Werkzeugmaschinen

INFORMATYKA 1984, Nr. 5, S. 8

Automatisierte Synthese des Programmes für spanende Bearbeitung als Beispiel einer modernen Softwarelösung mit Anwendung der künstlichen Intelligenz-Methoden, sowie PROLOG und PASCAL-Programmiersprachen.

robotron

Pisać na maszynie elektronicznej – to życzenie każdej sekretarki. Życzenie to spełnia Robotron S 6011, nowa elektroniczna maszyna do pisania z Kombinatu Robotron. Ona sama przekona Was najlepiej.
Elektroniczne funkcje dotyczące ukształtowania tekstu, pamiętanie formatów i stałych części tekstu oraz proste i łatwe korygowanie błędów – to tylko niektóre z czynników, umożliwiających przyjemne, łatwe i szybkie pisanie.

Zdecydujcie się

na Robotron S 6011,

bo ma on wiele

do zaoferowania

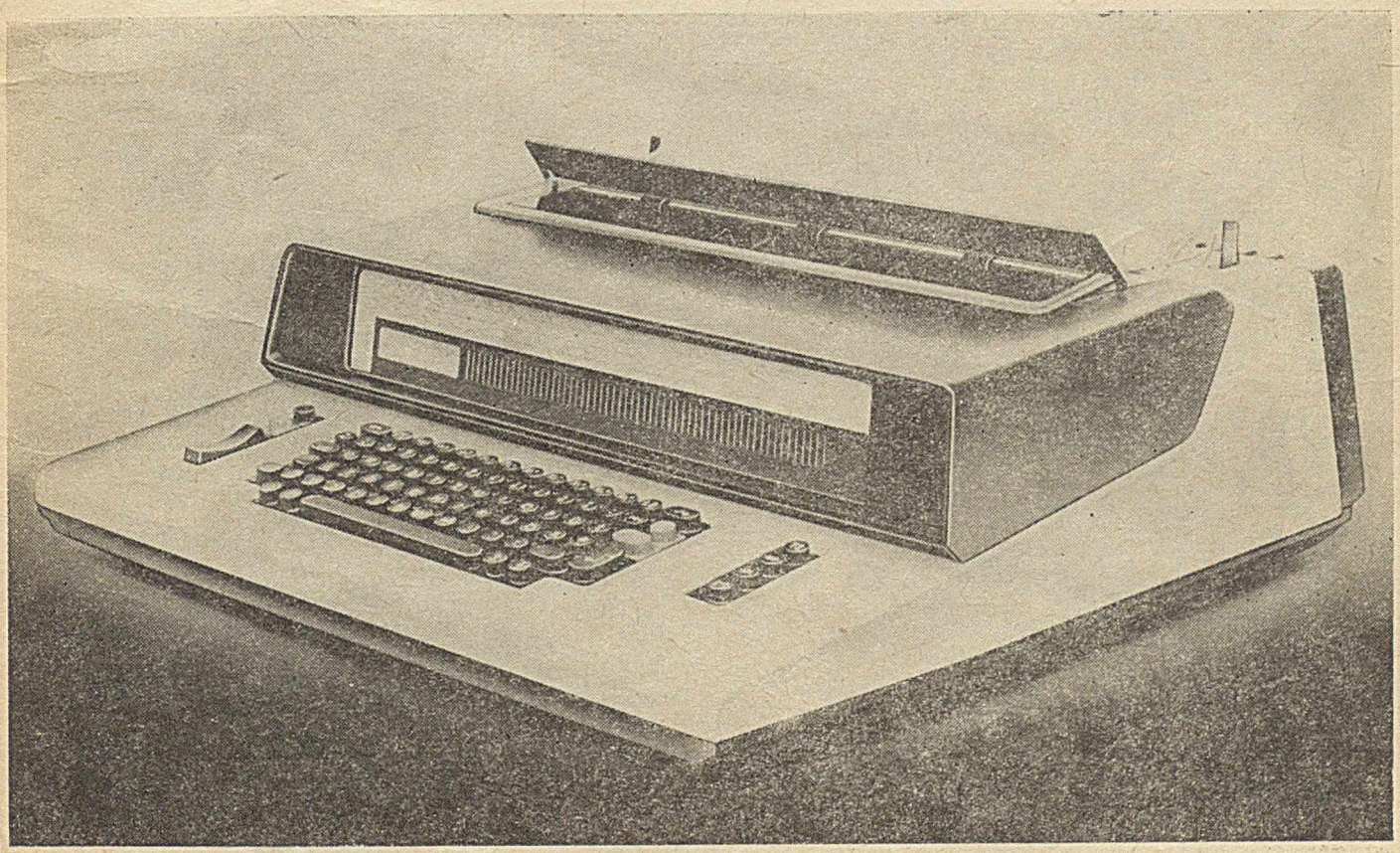
Przekonajcie się sami o komforcie pisania na maszynie Robotron S 6011.
Zapraszamy do odwiedzenia naszego stoiska (pawilon 14) na Międzynarodowych Targach Poznańskich 1984.

robotron

Robotron Export-Import
Przedsiębiorstwo
Handlu Zagranicznego
Niemieckiej Republiki
Demokratycznej
NRD – 1080 Berlin,
Friedrichstrasse 61

EO/330/K/84

S 6011



robotron

Wszędzie poszukiwani są specjaliści, zwłaszcza o wielokierunkowych kwalifikacjach. Nasz A 5120 dotrzyma. składanych przez nas obietnic na wszystkich odcinkach prac biurowych, np. w handlu (opracowanie zamówień, dyspozycje towarowe, ewidencjonowanie rachunków i dostaw), w przemyśle (obliczenia z zakresu zatrudnienia i płac, przychody i rozchody materiałów, ewidencja i kontrola wykonania zleceń, rachunkowość), w ośrodkach obliczeniowych (rejestrowanie danych masowych), w komunikacji (informowanie o połączeniach, kontrola stanu technicznego pojazdów) oraz w instytucjach finansowych. Ale to nie wszystko.
Komputer biurowy A 5120 może być stosowany również jako zdalny terminal. Składa się on z monitora ekranowego oraz jednostki pamięci na dyskach elastycznych lub pamięci kasetowej. Jego wydajność można zwiększyć przez dołączenie drukarki oraz dodatkowych jednostek pamięci zewnętrznych.

Elastyczny specjalista do prac biurowych

A 5120

Zarejestrowane dane można przesłać na nośniki urządzeń peryferyjnych albo bezpośrednio do komputera centralnego. I jeszcze jedno: dzięki koncepcji podziału ekranu na pola, A 5120 jest szczególnie dobrze przystosowany do rejestracji masowych danych. Gotowe moduły lub kompletne systemy programowe będą stale potwierdzać wyjątkowo dużą operatywność naszego specjalisty w warunkach działalności Waszego biura.
A więc przygotujcie miejsce do zainstalowania komputera biurowego A 5120!

Prosimy odwiedzić nas w pawilonie nr 14 podczas Międzynarodowych Targów Poznańskich w czasie od 10 do 17 czerwca 1984

robotron

Przedsiębiorstwo Handlu Zagranicznego NRD
Robotron Export—Import
DDR — 1080 Berlin
Friedrichstrasse 61

EO/1211/K/83-B