ISSN 0860-1941

CENA 300 zł

mikroklan

vademecum techniki mikrokomputerowej 3'87

Compaq Deskpro 386

Amstrad Joyce

GEM
system operacyjny
dla każdego

Amstrad PC 1512

MS-DOS
i jego korzenie

Kursy języków
BASIC i Pascal

ANKIETA
str. 15

Amstrad

WYDAWNICTWO NOT SIGMA

O KOMPUTERACH AMSTRAD JOYCE I PC 1512

czytaj na str. 8-9 i 12-14

mikroklan 3'87

Wprowadzenie w zagadnienia techniki mikrokomputerowej

Redaguje zespół
miesięcznika INFORMATYKA

Maciej Adamczyk, Piotr Breitkopf, Teresa Jabłońska (sekretarz redakcji), Władysław Klepacz (redaktor naczelny), Krzysztof Kontek, Jerzy Orkiszewski, Piotr Parlewicz, Maria Pawlak (sekretarz redakcji), Zbigniew Pojmański, Danuta Sot, Krzysztof Rzymkowski, Romuald Szuniewicz
Józef Kaczmarczyk (opracowanie graficzne)

WYDAWNICTWO CZASOPISM I KSIĄŻEK TECHNICZNYCH
SIGMA
PRZEDSIĘBIORSTWO NACZELNEJ ORGANIZACJI TECHNICZNEJ

00-950 Warszawa, skrytka 1004
ul. Biała 4

ISSN 0860-1941
Wydawnictwo Czasopism i Książek Technicznych NOT-SIGMA
Warszawa 1987

Drukowano na zlecenie ARS POLONA

Redakcja: 01-517 Warszawa, ul Mickiewicza 18 m. 17, tel. 39 14 34
Skład: technika fotoskładu Eurocat 150
Wydawnictwo NOT-Sigma
Druk: Bohmann Druck und Verlag GmbH & Co. KG, Wiedeń, Austria

Nakład 100 000 egz. K-87
Cena 300 zł

SPIS TREŚCI

INFORMATYCZNE ABC

OPIS

ZASTOSOWANIA

SPRZĘT

OPROGRAMOWANIE

TEST

Język C
zarządzanie pamięcią operacyjną

Coraz więcej osób posługuje się językiem C. Mimo iż zdaniem wielu informatyków jest to jeden z najważniejszych współczesnych języków programowania, jego historia jest dość długa i sięga końca lat sześćdziesiątych. W tych czasach istniało już wiele rozbudowanych języków, jak PL/I, Algol 68 i Cobol. Miały one oczywiście swoich zwolenników i przeciwników, będąc stałym przedmiotem dyskusji i porównań. W tych warunkach na Uniwersytetach w Cambridge i Londynie podjęto próbę opracowania jeszcze jednego języka, łączącego w sobie najlepsze cechy uznanych języków programowania. Język ten miał nazywać się CPL (*Combined Programming Language*) i miał być lepszy od każdego ze swych poprzedników. Niestety projekt zakończył się fiaskiem, a góra zrodziła mysz, którą nazwano później BCPL(Basic CPL).
Z języka BCPL powstał język nazwany B, a następnie – za sprawą Dennisa Ritchiego – język C. Ten ostatni został użyty do napisania większej części systemu operacyjnego UNIX i to ugruntowało jego pozycję efektywnego języka do programowania systemowego. Szereg dalszych zmian i uzupełnień uczyniło z niego język uniwersalny wysokiego poziomu, jednak przez wielu programistów jest on wciąż uważany za odmianę współczesnego asemblera. Ma to swoje uzasadnienie, ponieważ obecnie większość oprogramowania mikrokomputerów jest opracowana w tym właśnie języku, a nie w jakimkolwiek innym języku wysokiego poziomu albo asemblerowym.
Ponieważ już wkrótce ukażą się nakładem Wydawnictw Naukowo-Technicznych trzy książki poświęcone językowi C, mija się z celem przedstawienie opisu całego języka. Intencją autora artykułu jest pogłębienie wiadomości na temat języka C i zachęcenie do posługiwania się nim nie tylko użytkowników IBM PC, ale również posiadaczy tak popularnych mikrokomputerów jak ZX Spectrum i Amstrad 6128.
W szczególności programy przedstawione w niniejszym artykule zostały skompilowane i uruchomione za pomocą kompilatora HiSoft C, dostępnego w tych właśnie mikrokomputerach.

Sposób zarządzania pamięcią operacyjną w języku C zależy od implementacji. Tym niemniej we wszystkich „przyzwoitych" implementacjach są dostępne przynajmniej dwie funkcje: *calloc* i *free*.

Pierwsza z nich służy do przydzielania, a druga do zwalniania pamięci.

Funkcja calloc

Rozszerzony nagłówek: char*calloc(n,Size);
unsigned n,Size;

Wykonanie funkcji *calloc* powoduje przydzielenie obszaru pamięci operacyjnej wystarczającego do pomieszczenia *n* obiektów o rozmiarze *Size* każdy i udostępnienie wskazania na tak przydzielony obszar. Udostępnionie wskazanie jest typu (char *) i może być poddane konwersji na dowolne inne wskazanie na wspomniany obszar. Jeśli przydzielenie obszaru jest niemożliwe, to zostaje udostępnione wskazanie puste.

Przykład:
```
typedef struct{
        int re,im;
        }COMPLEX;
COMPLEX *ptr;
...
ptr=(COMPLEX*) calloc(20,sizoeof(COMPLEX));
```

Wykonanie instrukcji powoduje przypisanie zmiennej *ptr* wskazania na obszar, który może pomieścić 20 danych typu *COMPLEX*. Użycie konwersji (*COMPLEX**) jest niezbędne, ponieważ zmienna *ptr* jest typu *(COMPLEX)*, a rezultat funkcji *calloc* jest typu *(char*)*.

Funkcja free

Rozszerzony nagłówek: free(ptr)
char *ptr;

Wykonanie funkcji *free* powoduje zwolnienie obszaru pamięci operacyjnej przydzielonego za pomocą funkcji *calloc*. Argumentem funkcji *free* jest wskazanie udostępnione podczas wykonywania funkcji *calloc*.

Przykład:
```
ptr=(COMPLEX *)calloc(20,sizeof(COMPLEX));
...
free((char *)ptr);
```

Użycie konwersji jest niezbędne, ponieważ zmienna *ptr* jest typu (COMPLEX *), a parametr funkcji *free* jest typu *(char *)*.

Wydruk 1 przedstawia wykorzystanie funkcji *calloc* i *free* do zarządzania pamięcią operacyjną. Przytoczony program ilustruje zasadę działania prostego edytora, porządkującego wiersze tekstu według ich numeracji.
Wiersze tekstu są opatrzone liczbami całkowitymi bez znaku. Wprowadzenie wiersza wymaga podania liczby i tekstu wiersza. Jeśli tekst wiersza jest pusty, to jest on usuwany. Jeśli nie zostanie podana liczba ani tekst, a także wtedy, gdy zostanie wprowadzony wiersz pusty, zostaną wyprowa-

```
#include "stdio.h"

typedef struct Str{
          struct Str *next;
          int lab;
        } LIST;
typedef char *CharPtr;
typedef LIST *LISTptr;

LIST head = { NULL };



main(){
  extern LIST *find();
  int num,del;
  char line[80];
  LIST *pos;

  num = 0;
  do {
    if(!num)
      display();
    else {
      pos = find(num);
      if(pos->next && pos->next->lab == num)
        delete(pos);
      if(del != '\n'){
        getLine(line,80);
        insert(pos,num,line);
      }
    }
    getNumber(&num,&del);
  } while(del != EOF);
}



LIST *
find(num)
  int num;
{
  LIST *ref;

  ref = &head;
  while(ref->next && ref->next->lab < num)
    ref = ref->next;
  return ref;
}









delete(pos)
   LIST *pos;
{
  LIST *ptr;

  ptr = pos->next;
  pos->next = ptr->next;
  free(ptr);
}



getLine(ptr,max)
  char *ptr;
  int max;
{
  int i,c;

  for(i = 0;
      i < max - 1 && (c = getchar()) != '\n';
      i++) *ptr++ = c;
  if(c != '\n')
    while(getchar() != '\n');
  *ptr = '\0';
}



insert(pos,num,line)
  LIST *pos;
  int num;
  char line[];
{
  LIST *ptr,*ref;
  extern char *calloc();
  int len,size;

  len = 0;
  size = sizeof(LIST*tr) + sizeof(int)
                         + length(line) + 1;
  if(ptr = cast(LISTptr)calloc(1,size)){
    ref = pos->next;
    pos->next = ptr;
    ptr->next = ref;
    ptr->lab = num;
    fill(cast(CharPtr)(ptr + 1),line);
  } else
    printf("Ignored :- out of memory");
}









display()
{
  LIST *ref;

  ref = head.next;
  while(ref){
    printf("%3d %s\n", ref->lab,
                       cast(CharPtr)(ref + 1));
    ref = ref->next;
  }
}



int
length(line)
  char *line;
{
  int len;

  len = 0;
  while(*line++) len++;
  return len;
}



fill(ref,line)
  char *ref,*line;
{
  while(*ref++ = *line++);
  *ref = '\0';
}



getNumber(num,del)
  int *num,*del;
{
  int c,val;

  val = 0;
  while((c = getchar()) >= '0' && c <= '9')
    val = ((val + (val << 2)) << 1) + c - '0';
  *num = val;
  *del = c;
}
```

Wydruk 1.
Prosty edytor

```
typedef struct Str{
          struct Str *ptr;
          unsigned size;
        } HEADER;

HEADER base = { &base,0 },
       *ref = &base;


#define NULL 0
#define FAIL -1
#define TRUE 1
#define HEAPSIZE 1000

char *
sbrk(n)
  unsigned n;
{
  static char *p,
              heap[HEAPSIZE],
              *heapPtr = heap;

  if(heapPtr + n > heap + HEAPSIZE)
    return FAIL;
  return (heapPtr += n) - n;
}

char *
calloc(n,size)
  unsigned n,size;
{
  static HEADER *p,*q;
  static unsigned count;

  count = (n * size +
           (sizeof(HEADER) - 1)) /
          sizeof(HEADER) + 1;
  p = ref;
  q = p->ptr;

  while(TRUE){
    if(q->size >= count){
      if(q->size == count)
        p->ptr = q->ptr;
      else {
        q->size -= count;
        q += q->size;
        q->size = count;
      }
      ref = p;
      return (char *)(q + 1);
    }
    if(q == ref){
      if((q = (HEADER *)sbrk(count * sizeof(HEADER)))
          == FAIL) return NULL;
      q->size = count;
      free((char *)(q + 1));
    }
    p = q;
    q = q->ptr;
  }
}



free(areaPtr)
  char *areaPtr;
{
  static HEADER *p,*q;

  q = (HEADER *)areaPtr - 1;
  for(p = ref; !(q > p && q < p->ptr); p = p->ptr)
    if(p>= p->ptr && (q > p || q < p->ptr))
      break;
  if(q + q->size == p->ptr){
    q->size += p->ptr->size;
    q->ptr = p->ptr->ptr;
  }
  else
    q->ptr = p->ptr;
  if(p + p->size == q){
    p->size += q->size;
    p->ptr = q->ptr;
  }
  else
    p->ptr = q;
  ref = p;
}
```

Wydruk 2.
Implementacja funkcji calloc i free

dzone wszystkie wiersze. Przyjmuje się, że każdy wiersz jest kończony znakiem *CR*. Wprowadzenie z klawiatury znaku *Ctrl-Z* kończy wykonywanie programu.

Przykład:
Jeśli z klawiatury zostanie wprowadzony ciąg wierszy:
30 ewa
20 iza
40 jan
10 kaja
to po wprowadzeniu wiersza pustego zostanie wyprowadzony tekst:
10 kaja
20 iza
30 ewa
40 jan
Jeśli bezpośrednio po tym zostanie wprowadzony ciąg wierszy:
20
30
40 vera
to po wprowadzeniu wiersza pustego zostanie wyprowadzony napis:
10 kaja
40 vera

Przytoczony program został uruchomiony za pomocą kompilatora języka C firmy HiSoft dla mikrokomputera Amstrad 6128. Taki sam kompilator jest dostępny na ZX Spectrum. Wymieniony kompilator niemal wiernie odzwierciedla język wzorcowy i ma minimalną liczbę ograniczeń, nieistotnych dla programisty systemowego. W przytoczonym wydruku ujawniają się dwa z nich: wymaganie, aby operator konwersji był poprzedzony słowem *cast* oraz wymaganie, aby taki operator zawierał jedynie nazwę typu.

Przytoczony program jest na tyle czytelny, że nie wymaga omawiania. Tych, którzy zapoznają się z nim, zachęcam do przestudiowania wydruku 2, zawierającego przykładową implementację funkcji *calloc* i *free*, a także implementację funkcji *sbrk* zawierającą rezerwację pamięci dla sterty (ang. *heap*).

LITERATURA

[1]. Bielecki J.: Język C – interpretacja standardu, WNT 1987
[2]. Bielecki J.: Wprowadzenie do języka C, WNT (w druku)
[3]. Bielecki J.: Oprogramowanie mikrokomputerów, WKiŁ (w druku)

JAN BIELECKI

COMPAQ DESKPRO 386

Kolejna generacja komputerów osobistych zejdzie wkrótce z taśm produkcyjnych: będą to maszyny 32-bitowe, oparte na procesorze Intel 80386. Jednym z pierwszych producentów, którzy zapowiedzieli seryjną produkcję wyrobów tej klasy jest firma Compaq. Nowy komputer nazywa się Deskpro 386 i jest droższy od najszybszego PC/AT firmy IBM zaledwie o 20 procent.

Szacuje się, że obecnie eksploatowanych jest na świecie siedem do ośmiu milionów mikrokomputerów typu PC/XT oraz PC/AT, opartych na 16-bitowych procesorach Intel 8086, 8088 oraz 80286. Teraz, dzięki procesołowi 80386, na biurku użytkownika pojawia się kompatybilny z PC/AT, 32-bitowy komputer o cyklu 16 MHz i 4-gigabajtowej przestrzeni adresowej. Symulacja modeli XT oraz AT odbywa się przez programowe przyłączenie za pomocą instrukcji MODE systemu MS-DOS na cykl 4 MHz procesora 8088 lub 6 – 8 MHz, właściwy dla 80286.

14 MB RAM oraz dyski 130 MB

Pamięć operacyjna, rozszerzalna do łącznej pojemności 14 MB, w konfiguracji podstawowej ma 1 MB i jest zlokalizowana na karcie pamięci połączonej z płytą główną przez 32-bitowe złącze krawędziowe. Zamiast powszechnych dotąd układów DRAM (Dynamic RAM) zastosowano układy SCRAM (Static Column RAM) o krótszym czasie dostępu. Szczególnie cenną innowację stanowi system zarządzania pamięcią

CEMM (Compaq Expanded Memory Management). Umożliwia on stronicowanie pamięci powyżej podstawowego dla systemu operacyjnego DOS bloku 640 KB i jest zgodny ze specyfikacją LIM (Lotus-Intel-Microsoft).

W zależności od modelu komputer może być wyposażony w dysk sztywny o pojemności 40 lub 130 MB. Średni czas dostępu wynosi odpowiednio 30 i 25 milisekund (dla PC/AT – 40 milisekund), co uniemożliwia przesyłanie danych z prędkością 5 i 10 megabitów na sekundę. Do archiwizacji danych służy 40 megabajtowa stacja pamięci taśmowej o prędkości transmisji wynoszącej 500 kilobitów na sekundę.

Na płycie głównej znajduje się siedem gniazd szczelinowych (cztery – szesnastobitowe i trzy – ośmiobitowe) na karty rozszerzające. Jest miejsce na koprocesor arytmetyczny, ale wbrew oczekiwaniom nie służy ono do wstawienia procesora 80387, lecz takiego samego jak w PC/AT Intela 80287 o częstotliwości 4 lub 8 MHz.

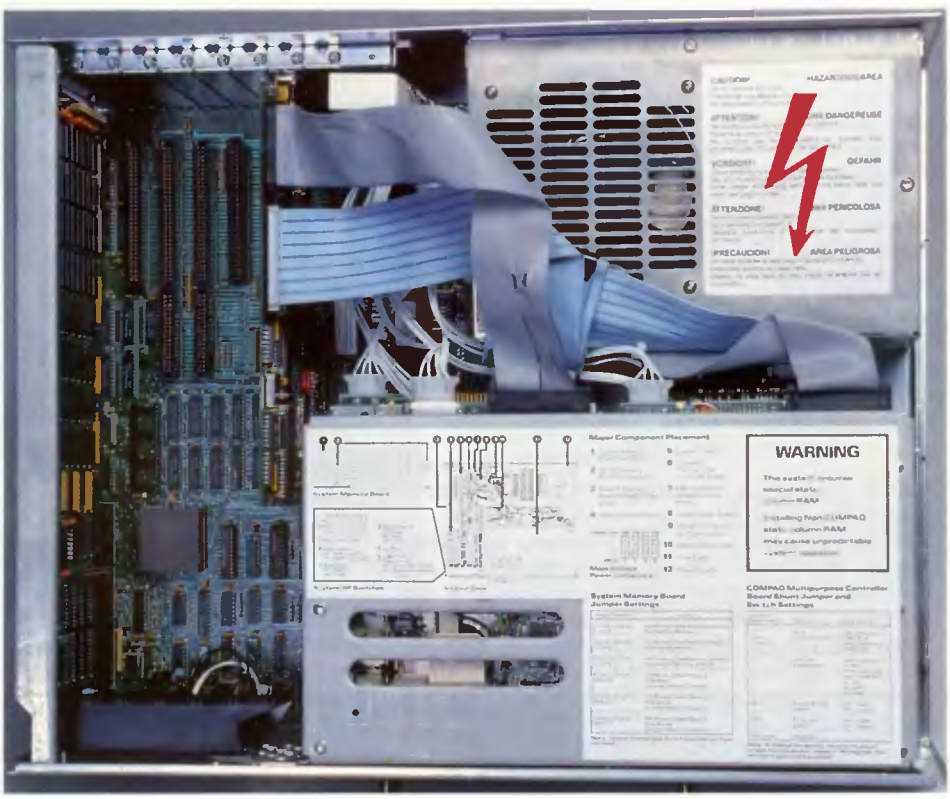
Karta graficzna systemu o rozdzielczości 640*350 z możliwością wyświetlania 16 spośród 64 kolorów, jest nie tylko zgodna ze standardem EGA (Enhanced Graphics Adapter), ale również dzięki relokacji zawartości 8-bitowej pamięci ROM do 32-bitowej pamięci RAM, zapewnia w przybliżeniu czterokrotny wzrost szybkości operacji graficznych. Do możliwości karty ECGB (Enhanced Color Graphics Board) dostosowany jest 13-calowy kolorowy monitor firmy Compaq.

Obecnie dostarczany z komputerem system operacyjny MS-DOS w wersji 3.1, nie pozwala jeszcze wykorzystać w porównaniu z PC/AT potencjalnych możliwości przyspieszenia pracy o jeden rząd wielkości. Nastąpi to dopiero po opracowaniu nowego dostosowanego do architektury systemu 32-bitowego.

CAD/CAM oraz AI

Potencjalne obszary zastosowań Deskpro 386 to komputerowe wspomaganie projektowania (CAD) i wytwarzania (CAM). Szybkość 4 mln operacji na sekundę oraz liniowa przestrzeń adresowa procesora będą z całą pewnością bardzo przydatne przy opracowywaniu dużych arkuszy kalkulacyjnych oraz symulacyjnych. Oczekuje się, że dzięki komputerom opartym na procesorze 80386 stanie się możliwe powszechne wykorzystywanie systemów ekspertowych i innych elementów sztucznej inteligencji (AI). Szybkie pamięci masowe niewątpliwie okażą się szczególnie przydatne nie tylko w rozwoju oprogramowania, ale również przy realizacji konfiguracji sieciowych.

Sceptyczni obserwatorzy podkreślają dwa powody uzasadniające brak pełnego entuzjazmu. Przede wszystkim, pełne wykorzystanie mocy obliczeniowej komputera stanie się możliwe dopiero po wprowadzeniu nowego, wielozadaniowego systemu operacyjnego. Po drugie, zagrożenie dla Deskpro 386 leży w sferze pozamerytorycznej. Decydująca o standardach firma IBM może pokusić się o opracowanie własnego PC/386, niekompatybilnego ze sprzętem konkurencyjnym, natomiast kompatybilnego z PC/XT oraz PC/AT. W takiej, mało prawdopodobnej z powodu trudności technicznych, sytuacji Deskpro 386 pozostanie jedynie szybszym AT.



Compaq Deskpro 386

Procesor:
- 32-bitowy procesor Intel 80386 z cyklem zegara 16 MHz

Pamięć oparacyjna:
- RAM o pojemności 1 MB z możliwością rozszerzenia do 14 MB
- 32-bitowa magistrala pamięci

Pamięć masowa:
- Stacja dyskietek o pojemności 1,2 MB (opcjonalnie: 360 KB)
- dysk sztywny o pojemności 40 MB i średnim czasie dostępu poniżej 30 ms lub o pojemności 130 MB i średnim czasie dostępu poniżej 25 ms

Interfejsy:
- równoległy do drukarki
- asynchroniczny dla komunikacji

Gniazda szczelinowa dla rozszerzań:
- 6 gniazd w modelu z dyskiem 40 MB
- 5 gniazd w modelu z dyskiem 130 MB

Klawiatura:
- 101-klawiszowa typu IBM RT
- 84-klawiszowa zgodna z IBM PC/AT

Dodatkowe opcje:
- 40 MB pamięć taśmowa
- koprocesor 80287 – 8 MHz
- monitor kolorowy (standard EGA)
- monitor monochromatyczny (bursztynowy lub zielony)
- rozszerzenie pamięci (1, 2, 4, 8 MB)

Cana:
- model 40 (dysk 40 MB) = 6499 dolarów USA
- model 130 (dysk 130 MB) = 8799 dolarów USA

micro

Mikroprocesory serii MC68000 (2)

W procesorze MC68000 jest dostępnych 14 trybów adresowania. Tryb adresowania jest pewnym sposobem obliczania tzw. adresu efektywnego. Jest on wykorzystywany przez rozkaz w sposób zależny od samego rozkazu. W tabeli 1 przedstawiono te tryby, ich nazwy oraz sposób obliczania adresu efektywnego. W dokumentacji firmowej wyszczególniono tryby dopuszczalne dla każdego rozkazu. Dla wielu rozkazów (niestety nie dla wszystkich), dostępne są wszystkie sensowne kombinacje trybów. Przykładowo, rozkaz MOVE jako argumentów wymaga efektywnych adresów źródła i przeznaczenia. Dopuszczalne jest, na przykład, użycie rozkazu MOVE $10, $16, tzn. przepisz zawartość komórki o adresie 10H do komórki o adresie 16H, jak również rozkazu MOVE #$100, $1000, tzn. wpisz stałą 100H do komórki pamięci o adresie 1000H.

OBSŁUGA PRZERWAŃ

Przerwania w procesorach MC680xx są czymś więcej niż tylko przerwaniami od urządzeń zewnętrznych. Pod tym pojęciem kryje się również np. obsługa rozkazów TRAP, TRAPV, CHK oraz dzielenie przez 0.

Obsługa przerwań odbywa się zazwyczaj w trybie pracy systemowej. Jeśli użytkownik nie pracuje w tym trybie, to jedyny sposób przejścia do niego polega na wywołaniu przerwania (np. procedury systemu operacyjnego mikrokomputera Atari 520 są wywoływane właśnie przez instrukcję TRAP.)

Każde z poniżej opisanych zdarzeń jest obsługiwane własnym oddzielnym przerwaniem systemowym:

- sięganie do słów rozpoczynających się od nieparzystych adresów,
- rozkazy nielegalne,
- rozkazy niezaimplementowane,
- próba dostępu do pamięci nieistniejącej lub chronionej sprzętowo (Bus Error, czyli błąd na magistrali)
- dzielenie przez 0,
- przepełnienie arytmetyczne (instrukcja TRAPV),
- zawartość rejestru poza dopuszczalnym zakresem (instrukcja CHK),
- fałszywe przerwanie.

Ponadto, w procesorze 68010 zrealizowano koncepcję maszyny wirtualnej. W takim systemie użytkownik ma złudzenie, że dysponuje dostępem do pełnej przestrzeni adresowej procesora, tzn. że ma dołączone np. 16 MB pamięci operacyjnej. Jeśli faktycznie dołączone jest tylko 2 MB, to przy próbach dostępu do pozostałych 14 MB procesor wstrzymuje swoje działanie, a sprzęt odczytuje odpowiedni blok pamięci z pamięci zewnętrznej. Po odczytaniu informacje są modyfikowane w układzie zarządzania pamięcią (MMU ang. *Memory Management Unit*), a użytkownik nie musi nawet wiedzieć czy aktywność dysku wynika z długości jego programu, czy też z innych przyczyn. W każdej chwili, kiedy procesor sięga do pamięci operacyjnej, umieszcza jednocześnie na trzech liniach specjalny kod informujący układ zarządzania pamięcią o wewnętrznym stanie procesora (patrz tabela 2). Umożliwia to np. ochronę zmiennych systemowych przed przypadkową lub celową zmianą przez osobę niepowołaną.

W pamięci operacyjnej, pod adresami od 0 do $3FF znajduje się tablica wektorów obsługi przerwań (patrz tabela 3). Po wywołaniu przerwania procesor automatycznie wprowadza na stos pewne informacje. W wypadku procesora 68000 jest to tylko poprzednia zawartość licznika rozkazów oraz rejestru stanu procesora SR. Dla procesora 68010 wpisywane są na stos: poprzednia wartość licznika rozkazów, SR, dane o klasie dodatkowych informacji, wyrównanie (ang. *offset*) informujące o wielkości tych informacji oraz same informacje.

Opisane tu skrótowo możliwości procesora 68000 są jedynie namiastką procesora 68020. Do dodatkowych możliwości należą: bardzo szybka wewnętrzna pamięć buforowa zwana notatnikową (ang. *cache memory*) o pojemności 256 bajtów, dodatkowe rozkazy związane z obsługą tzw. pól bitowych (ang. *bit fields*), większa maksymalna częstotliwość zegara, 32-bitowe szyny: danych i adresowa oraz dodatkowe cztery tryby adresowania.

Wszystkie te cechy umożliwiają komputerom opartym na tym procesorze osiąganie szybkości porównywalnej z minikomputerami rodziny VAX.

Poniżej podano krótkie charakterystyki pozostałych członków rodziny mikroprocesorów MC68000 i układów towarzyszących do obsługi operacji we-wy.

RODZINA MIKROPROCESORÓW MC68000

68008 – jednostka różniąca się od MC68000 szerokością szyn: 8-bitowa szyna danych i 20-bitowa szyna adresowa. Umożliwia to bardziej efektywne wykorzystanie stosowanych w systemach 8-bitowych układów towarzyszących, a tym samym znaczne obniżenie ich kosztów. Jednocześnie dzięki szybkiemu zegarowi (maks. 10 MHz) układ ten stanowi konkurencję dla systemów 16-bitowych;

68012 – różni się od MC68010 szerokością szyny adresowej. Jest ona 31-bitowa i daje możliwość adresacji pamięci do pojemności aż 2 GB;

68881 – koprocesor zmiennoprzecinkowy. Realizuje normę IEEE dla operacji zmiennoprzecinkowych w językach wyższego poziomu (dokładność obliczeń: 80 bitów). Może współpracować z 68020 jako koprocesor lub z pozostałymi procesorami rodziny jako bezpośrednie urządzenie zewnętrzne (ang. *tightly coupled peripherial)*.

Układy towarzyszące

68120 – *Intelligent Peripherial Controller* (IPC). Sterownik oparty na procesorze 6801 – zapewnia połączenie między układami rodziny 68000 a układami rodziny 6800. Układ wyposażony jest we własną dwuwejściową pamięć RAM (o pojemności 128 bajtów) i ma wiele trybów pracy;

68153 – *Bus Interrupter Module* (BIM). Lokalny sterownik przerwań, przeznaczony do obsługi czterech źródeł przerwań. Zgodny ze specyfikacją VME. Obudowa DIL 40;

68172 – *Bus Controller* (BUSCON). Układ interfejsu między procesorem lub DMA i szyną VMS (VMS – szyna szeregowa, stanowiąca część specyfikacji VME);

68230 – *Parallel Interface/Timer* (PI/T). Zawiera dwa porty o programowanym trybie pracy i kierunku przesyłania. Porty mogą pracować jako 16-bitowe we-wy lub razem, jako port 16-bitowy. Dodatkowo zawiera 24-bitowy zegar/licznik mogący generować przerwania. Ma mechanizmy generowania wektorów przerwań. Obudowa DIL 48;

68430 – *Direct Memory Access Interface* (DMAI). Sterownik DMA mający możliwość prowadzenia 32-bitowych transferów. Dostosowany do szyny VME;

68440 – *Dual Direct Memory Access* (DDMA). Umożliwia szybkie przesyłanie bloków danych, między pamięcią a urządzeniami zewnętrznymi, dwoma niezależnymi kanałami bez ingerencji procesora;

68450 – *Direct Memory Access Controller* (DMAC). Ma wszystkie cechy układu 68440 z tym, że transmisja może odbywać się czterema niezależnymi kanałami. Dzięki rozbudowanym trybom adresowania układ wykonuje ponadto wyrafinowane operacje na danych. Obudowa DIL 64;

Tabela 1.
Tryby adresowania

Tryb	Sposób wyliczania
Bezpośredni przez rejestr danych *(Data register direct)*	AE=Dn
Bezpośredni przez rejestr adresowy *(Address register direct)*	AE=An
Bezwzględny krótki *(Absolute short)*	AE=(następne słowo)
Bezwzględny długi *(Absolute long)*	AE=(dwa następne słowa)
Względny z 16-bitowym wyrównaniem *(Relative with offset)*	AE=(PC)+d16
Względny indeksowy z 8-bitowym wyrównaniem *(Relative with Index and Offset)*	AE=(PC)+ +(Xn)+d8
Pośredni rejestrowy *(Register Indirect)*	AE=(An)
Pośredni rejestrowy z postinkrementacją *(Postincrement Register Indirect)*	AE=(An): An=An+N
Pośredni rejestrowy z predekrementacją *(Predecrement Register Indirect)*	*An*=An−N: AE=(An)
Pośredni rejestrowy z 16-bitowym wyrównaniem *(Register Indirect with offset)*	AE=(An)+d16
Pośredni indeksowy przez rejestr z 8-bitowym wyrównaniem *(Indexed Register Indirect with Offset)*	AE=(An)+(Xn)+ +d8
Prosty (dane bezpośrednie) z długimi danymi *(Immediate)*	DATA=N następnych słów
Prosty z krótkimi danymi	

Znaczenie użytych symboli:
AE – adres efektywny
(An) – komórka pamięci wskazywana przez An
Xn – rejestr danych lub adresowy jako rejestr indeksowy
dn – n bitowe wyrównanie
N – długość argumentu w bajtach
An – rejestr adresowy
Dn – rejestr danych

Tabela 2.
Klasyfikacja dostępów do pamięci operacyjnej

Stan wyjść kodu funkcji			Klasa dostępu
FC2	FC1	FC0	
0	0	0	(nie określone)
0	0	1	dane użytkownika
0	1	0	program użytkownika
0	1	1	(nie określone)
1	0	0	(nie określone)
1	0	1	dane nadzorcy
1	1	0	program nadzorcy
1	1	1	potwierdzenie przyjęcia przerwania

Tabela 3.
Wektory obsługi przerwań

Lp.	Adres HEX	Przestrzeń	Znaczenie
0	000	SP	Początkowy wskaźnik stosu nadzorcy
1	004	SP	Początkowy licznik rozkazów
2	008	SD	Błąd na magistrali (*Bus Error*)
3	00C	SD	Błąd w adresowaniu (*Address Error*)
4	010	SD	Nielegalny rozkaz (*Illegal ins.*)
5	014	SD	Dzielenie przez zero (*Zero divide*)
6	018	SD	Obsługa rozkazu CHK
7	01C	SD	Obsługa rozkazu TRAPV
8	020	SD	Próba wykonania rozkazu dostępnego tylko dla nadzorcy przez użytkownika (*Privilege violation*)
9	024	SD	Obsługa trybu pracy krokowej (*Trace*)
10	028	SD	Emulacja rozkazów $A ---
11	02C	SD	Emulacja rozkazów $F ---
12	030	SD	Nie przydzielony, zarezerwowany
13	034	SD	Nie przydzielony, zarezerwowany
14	038	SD	Błąd formatu (tylko dla 68010)
15	03C	SD	Wektor przerwań niezainicjalizowanych
16 .23	040 05F	SD	Nieprzydzielony, zarezerwowany
24	060	SD	Fałszywe przerywanie (wykonywane, jeśli w trakcie obsługi przerwania nastąpił Bus Error)
25	064	SD	Autowektor przerwania zewnętrznego poziomu 1 (wykonywane jeśli użytkownik nie podał własnego wektora)
26. .31	068 07F	SD	Autowektor przerwania zewnętrznego dla kolejnych poziomów
32. .47	080 0BF	SD	Obsługa rozkazu TRAP # n
48. .63	0C0 0FF	SD	Nie przydzielony, zarezerwowany
64. .255	100 3FF	SD	Wektory obsługi przerwań użytkownika

dokończenie na str. 14

AMSTRAD JOYCE czyli przetwarzanie tekstów dla początkujących

Po niewątpliwych sukcesach swych komputerów domowych firma Amstrad postanowiła zaatakować rynek sprzętu profesjonalnego wprowadzając stosunkowo tani model PC 8256 o nazwie Joyce, przeznaczony głównie do przetwarzania tekstów. Test przeprowadzony przez miesięcznik *micro* stara się w sposób obiektywny ocenić wartość tego modelu.

Amstrad przystępując do ataku na rynek profesjonalny nie zdecydował się na zastosowanie w Joyce bardziej nowoczesnego procesora i pozostał przy dobrze sprzedawanym Z80A. Gdy wszędzie już mówi się i pisze o procesorach 16- i 32-bitowych, na rynku sprzętu profesjonalnego odradza się nagle procesor odstawiony prawie do lamusa. Jednak bardzo niska cena Joyce jest podstawowym atutem przeciw argumentacji zwolenników nowoczesności konstrukcji.

Koncepcja

Trzeba na wstępie zastrzec, że Joyce nie może być zestawiany z systemami komputerowymi klasy AT. Prezentując jakikolwiek mikrokomputer powszechnie podkreśla się, że może on być np. zastosowany jako system przetwarzania tekstów albo system zarządzania bazami danych. Amstrad przyjął natomiast całkowicie odmienną koncepcję, oferując rozwiązanie bardzo konkretne, a jednocześnie uniwersalne. Aby wyrazić to innymi słowy, Joyce może być np. urządzeniem przeznaczonym do przetwarzania tekstów lub systemem dla eksploatacji baz danych, a przy okazji (!) również pełnowartościowym komputerem z systemem operacyjnym CP/M. Czy filozofia ta jest słuszna, pokaże przyszłość. Koncepcja taka może bardzo szybko osiągnąć sukces w przedsiębiorstwach, w których „wielka" informatyka jest nieopłacalna, a także w niewielkich zakładach, poszukujących rozwiązania jednego konkretnego problemu. Również tzw. rozwiązania wyspowe, a więc wyizolowane problemy w większych przedsiębiorstwach, mogą stać się dla Joyce doskonałym obszarem efektywnych zastosowań. Dlatego test przeprowadzono z tego właśnie punktu widzenia.

Ogólna charakterystyka

W skład konfiguracji Joyce, oprócz monitora z wbudowaną stacją 3-calowych dyskietek, wchodzi przestawna klawiatura oraz moduł drukarki. Ten ostatni składnik nazwano modułem, ponieważ prawie całą elektronikę systemu umieszczono w obudowie monitora.

W testowanym egzemplarzu pojemność pamięci RAM wynosiła 256 KB. Prawie połowa tej pamięci (112 KB) używana jest jako tzw. dysk RAM (RAM – Disk – dysk w pamięci RAM). Obszar ten należy traktować jak stację dyskietek, a więc nie powinno się tu przechowywać danych w sposób ciągły. Przy inicjowaniu programu następuje najpierw wczytanie na dysk RAM wszystkich istotnych, często używanych podprogramów. Ze względu na wyjątkowo szybki dostęp do dysku RAM rozwiązanie to powoduje praktyczne wyeliminowanie powszechnie występujących, w sprzęcie opartym na CP/M, stosunkowo długich czasów oczekiwania na wprowadzenie podprogramów.

Do wymienionych elementów nabywca komputera otrzymuje, oprócz pakietu przetwarzania tekstów, cztery dalsze pakiety programowe: CP/M 3.0, Dr. Logo, GSX (*Graphic System eXtension*) oraz *Mallard Extended Basic*.

Eksploatacja

Joyce działa zgodnie z podstawowym wymaganiem niewykwalifikowanego użytkownika: „włączam urządzenie i pracuję". Operacja inicjowania systemu trwa ok. 8 sekund, natomiast 20 następnych sekund – wprowadzenie programu LocoScript. Po jego załadowaniu program zarządzający dyskiem (*Disk-Manager*) informuje o stanie systemu. Tak prezentowany poziom obsługi użytkownika może wydawać się początkowo zbyt skomplikowany wskutek znacznego nadmiaru podawanych informacji. Jednak z drugiej strony zbędne staje się wielokrotne wywoływanie pytań uzupełniających. Przyszłość pokaże, czy użytkownicy takie rozwiązanie zaakceptują. Z pewnością nie jest to złe, ale wymaga trochę przyzwyczajenia. Z chwilą zgłoszenia się Disk-Managera można rozpocząć pisanie tekstu, wybierając jeden z dwu trybów zapisu.

W trybie bezpośrednim każdy wiersz programu jest przesyłany bezpośrednio na drukarkę. Głowica drukująca może być wówczas ustawiana na poziomie pojedynczych znaków, co pozwala np. na łatwe wypełnianie formularzy.

W trybie drugim LocoScript zachowuje się tak, jak inne programy przetwarzania tekstów. Ten, kto jeszcze nigdy nie korzystał z komputera lub programu przetwarzania tekstów, nie posunie się w tym trybie dalej bez podręcznika lub odpowiedniego poinstruowania, natomiast osoby mające już doświadczenie w korzystaniu z komputerowego przetwarzania tekstów, odnotowują po kilku zaledwie próbach pierwsze sukcesy.

Klawiatura 82-klawiszowa – przestawna w granicach długości przewodu łączącego – odpowiada rozszerzonej klawiaturze maszyny do pisania. Rozszerzenie klawiatury

Krótka charakterystyka JOYCE

Procesor:
- Z80A, częstotliwość zegara 4 MHz

Pamięć operacyjna:
- 256 KB, z czego ok. 112 KB wykorzystywane jako dysk RAM

Pamięć zewnętrzna:
- jeden napęd dyskietek 3-calowych o pojemności 250 KB (danych niesformatowanych), umieszczony w obudowie monitora
- opcjonalnie: napęd FD-2 o pojemności 1 MB (danych niesformatowanych)

Ekran:
- o dużej rozdzielczości, monochromatyczny, zielony, 32 wiersze po 90 znaków

Klawiatura:
- rozszerzona klawiatura maszyny do pisania

Drukarka:
- mozaikowa o maksymalnej szybkości 90 znaków/s

Opcje:
- interfejs CPS 8256 (RS 232 oraz Centronics)

Oprogramowanie dostarczane z komputerem:
- CP/M 3.0
- LocoScript
- GSX (interfejs dla użytkowników programów grafiki)
- Dr. Logo, Mallard Extended Basic

Podręczniki:
- User Guide – CP/M Logo and Wordprocessor Manual, Basic Manual

wydaje się wyraźnie dostosowane do pakietu LocoScript (lub odwrotnie!). Sama klawiatura może być z całym przekonaniem nazwana profesjonalną. Zintegrowana z klawiszem CAPS-LOCK świecąca na czerwono dioda wskazuje aktualny stan systemu.

W przeciwieństwie do komputerów rodziny CPC, Joyce nie przewiduje prezentacji barwnej. Monochromatyczny zielony monitor o dużej rozdzielczości wyświetla 32 wiersze po 90 znaków. Stanowi to odpowiednik ok. 1 1/2 strony tekstu napisanego na papierze formatu A.4. Wiersz 90-znakowy jest wprawdzie nietypowy, ale ma głębszy sens. Przez odpowiedni wybór postępowania w jednym z menu typu „pull-down" można np. w wydruku uwidaczniać również instrukcje sterujące. W większości wypadków prezentacja odbywa się za pomocą ciągu znaków odpowiadających skróconej postaci instrukcji. Dzięki temu nie uruchamiając drukarki można oglądać i kształtować obraz późniejszego dokumentu.

Technika okien bez myszy

Dobrze rozwiązano wywoływanie i wykorzystywanie w technice okien różnych rodzajów menu. Za pomocą kursora oraz klawiszy plus i minus następuje wybór określonych parametrów. W tym zakresie McIntosh firmy Apple walnie przyczynił się do stworzenia podejścia „przyjaznego" dla użytkownika mikrokomputera (ang. user friendly). W przeciwieństwie do tego komputera (np. również do komputera Atari 520 ST) sterowanie w Joyce zrealizowano wyłącznie przez klawiaturę. Oznacza to, że mysz jest zbędna, tym bardziej że jej zastosowanie, zwłaszcza w systemie tekstowym, nie jest uzasadnione.

Podczas testowania jako rozwiązanie niezadowalające uznano wbudowaną stację dyskietek 3-calowych. Powodem krytyki nie jest jednak napęd, lecz pojemność dyskietek, która dla systemu profesjonalnego jest zbyt mała. Przy zapisie w formacie CP/M dyskietki te mają na jednej stronie pojemność zaledwie 169 KB. Na rozwiązanie takie miały wpływ prawdopodobnie dążenia zmierzające zarówno do uzyskania możliwie niskiej ceny systemu, jak i kompatybilność z poprzednimi produktami Amstrada. Możliwość przyłączenia dodatkowego podwójnego napędu dyskietek o identycznej pojemności rozszerza dyspozycyjny obszar pamięci zewnętrznej do ok. 1 MB (danych niesformatowanych).

Dobra jakość druku

Maksymalna szybkość drukarki w trybie niekorespondencyjnym wynosi 90 znaków/s, natomiast w trybie korespondencyjnym (NLQ) szybkość ta zmniejsza się do 20 znaków/s. Różne rodzaje pisma, jak tekst wytłuszczony, górna i dolna frakcja, a także pismo proporcjonalne, zróżnicowane odstępy między znakami są drukowane w sposób bardzo czytelny. Komu to nie wystarcza, może rozszerzyć możliwości przez zastosowanie opcjonalnego interfejsu CPC 8256. Interfejs ten oferuje zarówno złącze typu RS 232, jak i Centronics. Umożliwia to przyłączenie również innych urządzeń wyjściowych.

W drukarce można stosować zarówno pojedyncze arkusze papieru, jak i taśmę ciągłą. Automatyczny podajnik pojedynczych arkuszy działa sprawnie i nie wymaga każdorazowego korygowania ustawienia. W wypadku stosowania papieru w postaci taśmy ciągłej konieczne jest użycie dostarczanej wraz z drukarką zębatki wymuszającej przesuw.

Lektura nie na jeden wieczór

Sądząc z zawartości dwu obszernych angielskich podręczników, obejmujących łącznie z podręcznikiem BASICA ponad 1000 stron, obsługa Joyce powinna być łatwa do opanowania. Przyszłość pokaże, czy tłumaczenia podręczników zachowają tę cechę. Jest jednak wątpliwe, czy każdy użytkownik będzie skłonny „przerobić" tak obszerny materiał dokumentacyjny. Dlatego producentowi można doradzić dodatkowe sporządzenie skróconej instrukcji dla tych użytkowników, którzy zamierzają wykorzystywać ten komputer „z marszu".

Joyce jest bardzo interesującym systemem dla określonego kręgu nabywców przede wszystkim ze względu na szczególnie atrakcyjną cenę. Powszechnie stosowany wskaźnik ekonomicznej efektywności systemu (stosunek ceny do wydajności) jest dla tego modelu niezwykle korzystny. Na przykładzie tego systemu Amstrad wykazał również, że komputery 8-bitowe nie są jeszcze sprzętem przestarzałym. Za specjalizowany procesor tekstów o analogicznych parametrach wydajności trzeba obecnie zapłacić wielokrotność ceny Joyce. Należy wreszcie podkreślić, że możliwość korzystania z systemu operacyjnego CP/M-Plus w istotny sposób rozszerza zakres zastosowań, przez udostępnienie wielu interesujących produktów programowych.

micro

GEM system operacyjny dla każdego

Rosnąca liczba komputerów osobistych spowodowała, że w firmach projektujących oprogramowanie zaczęto w coraz większym stopniu zwracać uwagę na tworzenie oprogramowania „przyjaznego" dla użytkownika (ang. *user friendly*), tzn. ułatwiającego pracę z komputerem osobistym osobom bez przygotowania informatycznego. W efekcie powstała znaczna liczba programów, które informują użytkownika o tym, jakie zadania programu są możliwe do wykonania w aktualnym momencie jego działania. Stale rozwijana forma graficzna takich informacji doprowadziła do powstania pojęć jak „menu", tzn. wyświetlanych na ekranie informacji o dozwolonych zleceniach dla programu w danym momencie jego wykonywania, czy też „okna", prostokątnego fragmentu ekranu obrazującego praktyczny obszar pracy użytkownika, do którego odnoszą się bieżące zlecenia operatora (główną funkcją okien jest wyświetlenie aktualnej zawartości zbiorów). W konwersacji użytkownika z komputerem zaczęto stosować wejściowe urządzenia graficzne, takie jak pióro świetlne (ang. *light pen*) czy myszka (ang. *mouse*).

Ale powstały nie tylko tego typu programy aplikacyjne. Podjęto próby stworzenia w podobnym stylu systemów operacyjnych. Jednym z najbardziej znanych i efektownych był system ukierunkowany graficznie, zaprojektowany przez firmę Apple dla komputera Macintosh. System ten miał jednak istotną wadę – nie był przenośny na inne komputery. Pozostawiało to wolne pole do działania dla innych. Właśnie tę lukę wypełnił GEM (*Graphics Environment Manager*) – graficzny system operacyjny. Opracowała go firma Digital Research, mająca już duże doświadczenie w tworzeniu systemów operacyjnych dla mikrokomputerów m.in. jako twórca słynnego CP/M. Pierwszym komputerem, na który wprowadzono GEM był IBM PC (1985 r.). Następnie system ten stał się dostępny dla bardziej zaawansowanych modeli komputerów firmy Atari. W wersji proponowanej dla komputera Atari 1040 ST wyeliminowano pewne wady wersji pierwotnej. Wprowadzone zmiany nie naruszają jednak zgodności logicznej z pierwowzorem i użytkownik znający system GEM z wcześniejszych doświadczeń może bez większych trudności posługiwać się nową wersją.

Obecnie na rynek krajowy wchodzą inne mikrokomputery wyposażone w GEM. Ale tym, o którym chcielibyśmy powiedzieć najwięcej, jest Amstrad PC 1512. Ma on kilka cech szczególnie istotnych dla efektywnego użytkowania systemu GEM. Użytkowane w Polsce zestawy komputerowe kompatybilne z IBM PC, bardzo rzadko są wyposażone w myszkę. Co prawda z systemu GEM można korzystać także implementując myszkę za pomocą klawiatury, ale jest to rozwiązanie dalekie od ideału.

Spośród wielu osób korzystających z systemu GEM znam tylko kilka, które posługują się nim za pomocą klawiatury. Pozostałe osoby, brak myszki skutecznie zniechęca do korzystania z systemu.

Amstrad PC 1512, przy rewelacyjnie niskiej cenie, ma myszkę w wyposażeniu standardowym, a firma Digital Research przygotowała dla niego nową, rozszerzoną wersję systemu GEM, która wchodzi w skład dołączanego do każdego zestawu standardowego oprogramowania.

Dla komputera Amstrad PC 1512 opracowano jedenaście programów aplikacyjnych, z których dwa: GEM Desktop i GEM Paint zawarte są w oprogramowaniu standardowym, natomiast pozostałe zapewne już niedługo będą krążyły wśród polskich użytkowników tego komputera.

A oto krótka charakterystyka tych programów:

GEM Desktop – komunikuje się z użytkownikiem, organizuje i równocześnie stanowi jego obszar pracy. Jest on dotychczasowym biurkiem z książkami, kartkami papieru do pisania i rysowania, kalkulatorem, zegarkiem i kalendarzem. Służy także do uruchamiania programów, zarówno aplikacyjnych opartych na systemie GEM, jak i innych – np. systemów DOS czy CP/M.

GEM Paint – pozwala na użycie ekranu komputera jak płótna do malowania, na którym można tworzyć obrazy przy użyciu różnych narzędzi: pędzli, pistoletu do malowania różnych wzorów, palety barw itp.

GEM Write – jest sprawnym i szybkim procesorem tekstów, łatwym w użytkowaniu przy współpracy z innymi aplikacjami systemu GEM. Pozwala na umieszczenie w tekście obrazów, rysunków, wykresów, diagramów, oferuje różne wzory czcionek oraz współpracuje z wieloma popularnymi drukarkami i ploterami.

GEM Draw – pozwala rysować wykresy, diagramy, schematy, projekty i ilustracje przy użyciu różnych elementów rysunku podanych w menu lub własnych użytkownika. Może również wybierać obraz z jednego okna i wykorzystywać go jako element obrazu w innym oknie. Specjalne funkcje tworzące na obszarze roboczym siatkę oraz linie, ustawiające w osi, skalujące itd. pozwalają osiągnąć dużą precyzję ustawienia elementów rysunku.

GEM Graph – pozwala tworzyć różnego typu diagramy wysokiej jakości (liniowe, słupkowe, kołowe i inne), w tym także trójwymiarowe. Są one ujęte w menu, a dane dla nich wprowadza się do specjalnej tabeli lub też korzysta z podanych wcześniej w innych programach, takich jak Lotus 1-2-3, Symphony czy dBase III.

GEM WordChart, GEM Draw Business Library – wspomagają inne aplikacje systemu GEM przy tworzeniu bardziej skomplikowanych wykresów i diagramów.

GEM Fonts & Drivers Pack – zawiera „drivery" i wysokiej jakości wzory znaków dla drukarek i ploterów najwyższej klasy.

GEM Diary – komputerowy notatnik z „wiecznym" kalendarzem i zegarem-budzikiem, dostępny zawsze w czasie działania systemu GEM. Pozwala to w każdej chwili sprawdzić datę z dowolnego miesiąca przeszłości, teraźniejszości i przyszłości, zobaczyć lub zapisać informacje, np. o ustalonym spotkaniu, a także ustawić zegar, tak aby przypominał o niektórych lub wszystkich spotkaniach.

GEM Font Editor – pozwala w prosty sposób projektować własny wzór znaków np. liter dla szczególnie ważnych dokumentów.

GEM Programmer's Toolkit – pozwala na tworzenie i łatwe dołączanie własnych programów mających wszelkie atrybuty aplikacji GEM, tzn. ikon, menu, okien, skrzynek dialogowych i informacyjnych. Zawiera interfejs służący do łączenia własnych aplikacji napisanych w asemblerze lub języku C (zalecany jest kompilator Lattice C) oraz symboliczny program uruchomieniowy (debugger) do efektywnego testowania, poprawiania i uruchamiania programów. Szczególnie godne podkreślenia jest to, że wszystkie aplikacje mogą wzajemnie współpracować, tworząc zwarty i logiczny system.

Wypada obecnie wyjaśnić kilka podstawowych pojęć oraz zasad operowania systemem GEM.

Kursor – strzałka wskazująca miejsce na ekranie, którego dotyczy dana operacja. W czasie wykonywania operacji kursor przyjmuje kształt klepsydry. Do przesuwania kursora służy myszka przenosząca przyrosty współrzędnych, wynikające z przesuwania jej po biurku, na przyrosty współrzędnych położenia kursora na ekranie.

Operacje związane z myszką:

- pstryk, „klik" (ang. *click*) – naciśnięcie i szybkie zwolnienie lewego przycisku myszki,

● podwójny „klik" (ang. *double click*) – szybkie dwukrotne naciśnięcie lewego przycisku myszki, służące do wybrania i otwarcia katalogu (wykonania programu), z czym jest związane otwarcie nowego okna pokazującego zawartość katalogu lub wyniki programu,
● przycisk (ang. *pressing*) – trzymanie przez dłuższy czas naciśniętego lewego przycisku myszki w celu przesunięcia (ang. *dragging*) wybranej ikony lub okna na ekranie,
● „SHIFT-klik" – trzymanie przyciśniętego prawego przycisku myszki i szybkie naciśnięcie lewego, służy np. do zamykania okna lub jednoczesnego wybierania kilku ikon plików (nazwa wynika z podobnego efektu jaki powstaje przy użyciu klawisza Shift zamiast prawego przycisku),
● ikona, wizerunek (ang. *icon*) – obrazek symbolizujący obiekt lub operację (np. stację dysków, zbiór, katalog, kosz na śmieci, ołówek, pędzel). Wybranie ikony polega na przesunięciu na nią kurosora i wykonaniu „kliku", co powoduje inwersyjne wyświetlenie potrzebnego wizerunku oraz anulowanie poprzedniego wybranego.

Menu – tytuły menu dostępnych w danej chwili są wyświetlane zawsze u góry ekranu nad oknem. Po wskazaniu tytułu (przez ustawienie na nim kursora) następuje rozwinięcie wybranego menu. Każdy wiersz menu zawiera jedno zlecenie; zlecenia możliwe w danej chwili do wykonania są wyświetlane ciemnymi znakami, natomiast pozostałe są w tym czasie niedostępne. Wiele zleceń staje się dostępnych dopiero po uprzednim wybraniu ikony katalogu, dokumentu czy programu. Niektóre mają różne opcje zaznaczone za nazwą zlecenia przez kombinacje rombu i litery – opcje wybiera się po wskazaniu zlecenia przez naciśnięcie klawisza ALT i właściwej litery. Zlecenie wybiera się przez ustawienie na nim kursora, po czym następuje inwersyjne wyświetlenie zlecenia, a „klik" powoduje jego wykonanie.

Obszar roboczy – całkowita powierzchnia pracy. Na ogół w oknie na ekranie widoczna jest tylko część obszaru roboczego, natomiast dzięki możliwości przesuwania okna po obszarze można przejść całą jego powierzchnię.

Okno – na ekranie można wyświetlić jednocześnie najwyżej cztery okna. Każde okno jest otwierane dla jednej ikony po jej wskazaniu i wykonaniu operacji otwarcia. W danej chwili może być tylko jedno okno, wyróżnione jako aktywne ciemnym kolorem ścieżki w linii tytułowej (okna nieaktywne mają linię tytułową złożoną z jasnych znaków). Nowe okno można otworzyć przez podwójny „klik" (wybranie i otwarcie ikony dysku, katalogu czy programu), a także przez jeden „klik" (wybranie) i wykonanie zlecenia „Open" z menu „File", co jest szczególnie istotne przy kłopotach związanych z odpowiednio szybkim wykonaniem podwójnego „kliku"

Linia tytułowa – zawiera ścieżkę, która prowadzi do zbioru pokazanego w danym oknie (zgodnie z hierarchicznym systemem plików zastosowanym w wersjach 2.x systemu DOS lub nowszych pod nadzorem których działa GEM). Przesunięcie okna na ekranie odbywa się przez ustawienie kursora na linii tytułowej. Następnie, trzymając przyciśnięty lewy przycisk myszki, przesuwamy kursor do nowego miejsca wybranego dla linii tytułowej.

Linie „przesuń okno w obszarze roboczym" – symbolizują całą powierzchnię obszaru roboczego. Widoczne białe pola określają położenie widocznej w oknie części obszaru, a ich rozmiary odzwierciedlają proporcje obszaru. Wskazując część linii obok białego pola i wykonując „klik" można przesuwać okno po całym obszarze i oglądać całą jego zawartość. Do przesuwania okna po obszarze roboczym służą także strzałki znajdujące się w małych polach na krawędziach okna (ang. *arrow boxes*) – przesuwa się wskazując odpowiednią strzałkę i wykonując „klik"

Zmiany rozmiarów okna wykonuje się za pomocą pola ze znakiem (ang. *full box*) w prawym górnym rogu okna lub pola ze znakiem (ang. *size box*) w prawym dolnym rogu okna. Operację „full box" wykonuje się przez „klik" na obszarze jej pola. „Full box" zmienia czasowo rozmiary okna do wielkości całego ekranu lub przywraca poprzednie. Operację „size box" wykonuje się przez wskazania pola „size box" kursorem, przyciśnięcie lewego przycisku myszki i przesunięcie kursora do miejsca wybranego dla prawego dolnego rogu okna.

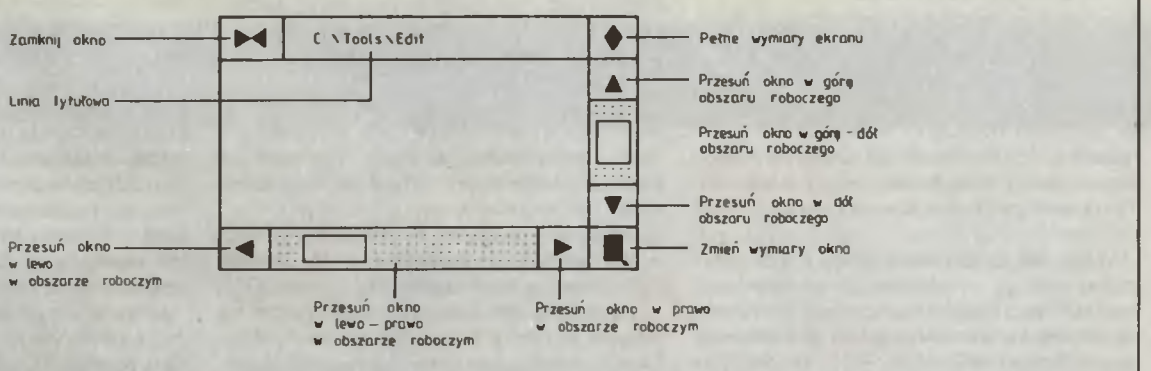
Zamknięcie okna wykonuje się przez „klik" na polu ze znakiem (ang. close box), znajdującym się w lewym górnym rogu okna, przez zlecenie typu „Quit" z menu „File" lub przez operacje „SHIFT-klik" na linii tytułowej okna. Okno nieaktywne uaktywnia się poprzez „klik" na jego niezakrytej powierzchni. Jeśli okno było częściowo ukryte pod innymi, to zostanie odtworzona jego brakująca część.

Do przekazywania informacji użytkownikowi systemu GEM są stosowane specjalne skrzynki informacyjne (ang. *dialog boxes*), które czasem służą jedynie do przekazywania informacji (ang. *alert boxes*), a czasem żądają także wyboru określonej opcji lub pliku.

Jak łatwo zauważyć, GEM nie tylko uwalnia od żmudnego wprowadzania zleceń z klawiatury, udostępnia narzędzie wspomagające programowanie złożonych operacji graficznych (dzięki grafice zgodnej z systemem GKS – uznanym za standard światowy), umożliwia wykonywanie wielu operacji systemowych (wyszukiwanie, kopiowanie, usuwanie zbiorów, przeglądanie katalogów itp.), ale przede wszystkim organizuje i upraszcza pracę użytkownika. Jest kolejnym krokiem swobodnej konwersacji człowieka z komputerem. Wprawdzie komputery nie rozumieją jeszcze poleceń wydawanych głosem, ale znają już język obrazów graficznych.

Powyższy artykuł jedynie sygnalizuje możliwości, jakie oferuje GEM. Dotyczy to również zasad jego użytkowania. Jeżeli chodzi o zasady, to powinna wystarczyć jedna: „myśl i używaj myszki!"

ROBERT LIWIŃSKI



Amstrad PC 1512

Po olbrzymim sukcesie w dziedzinie komputerów domowych firma Amstrad zaatakowała rynek komputerów profesjonalnych oferując sprzęt kompatybilny z IBM PC, a więc wyposażony w system operacyjny MS-DOS. Atak był frontalny. Osiem oferowanych wariantów PC 1512 pozwala na wybranie konfiguracji najodpowiedniejszej dla potrzeb i ... zawartości kieszeni. Modele różnią się monitorem (monochromatyczny lub barwny) i pamięcią masową (1x360 KB, 2x360 KB, 1x360 KB + 10 MB Winchester, 1x360 KB + 20 MB). Prywatni użytkownicy zainteresują się raczej modelami tańszymi. Nie są one jakościowo gorsze : monitor mono w wydaniu Amstrada nie jest zielony czy bursztynowy, lecz tzw. paper-white i odzwierciedla barwy jako poziomy szarości, a jeden napęd w systemie symuluje, podobnie jak w CPC 6128, obecność dwóch napędów dyskietek. Do tematu PC 1512 będziemy powracać. Wszystkich, którzy mają już własne doświadczenia z nowym Amstradem, zapraszamy na łamy MIKROKLANU. W szczególności problem kompatybilności PC 1512 z IBM-owskim pierwowzorem budzi wiele emocji. Zajmiemy się nim wkrótce.

Obszerny i napisany w sposób zrozumiały podręcznik stanowi istotną pomoc dla użytkownika

Perspektywy sukcesu rynkowego są dla nowego modelu raczej pomyślne. Popyt na komputery osobiste jest nadal duży, a firma Amstrad ma już ugruntowaną pozycję na rynku europejskim. Bardzo niska cena w zestawieniu z parametrami eksploatacyjnymi jest szczególnie mocnym argumentem w stosunku do sprzętu głównych konkurentów, którymi są firmy Commodore oraz Atari.

Dokumentacja

Jeśli na czoło prezentacji wysuwamy dokumentację, to nie bez powodu: niezmiernie istotne dla użytkownika jest otrzymanie wraz ze sprzętem dobrego i kompletnego podręcznika, a nie (jak dotąd często bywało) prowizorki w postaci tzw. wersji wstępnej. Tym razem jest to oobszerna książka o objętości ponad 700 stron.

W podręczniku wiele miejsca (ok. 200 stron) zajmują wyjaśnienia programów systemu GEM. Szczegółowo potraktowano również zestaw instrukcji wersji 3.2 systemu operacyjnego MS-DOS. Stosunkowo niewiele miejsca poświęcono językowi programowania Locomotive-Basic 2, którego podręcznik stanowi osobną publikację.

Klawiatura

Klawiatura jest ściśle powiązana z jednostką centralną nie tylko poprzez etykietę, ale również przez to, że wtyczka jej przewodu łączącego pasuje tylko do gniazda PC 1512, do którego z kolei nie można przyłączyć klawiatury żadnego z komputerów kompatybilnych.

Gabaryty klawiatury odbiegają swymi wymiarami od rozwiązań powszechnie spotykanych. Mniejsza o 3 cm szerokość w połączeniu z nieco wyżej nadbudowanym polem klawiszy powoduje wrażenie formy bardziej zwartej. Same klawisze wystają tylko niecały centymetr i w związku z tym wymagają nieco innego ułożenia ręki podczas pisania. Oczywiście nachylenie klawiatury może być regulowane.

Dzięki 85 klawiszom można dysponować, w porównaniu do klawiatury IBM, dwoma dodatkowymi klawiszami: DEL - - oraz DEL -. Pozwalają one skasować znak położony ponizej lub na prawo od kursora. Funkcje tych klawiszy uzależnione są jednak od odpowiedniego wspomagania programowego. Oprócz tego blok kursora został rozszerzony o klawisz ENTER.

W klawiaturze znajduje się gniazdo, do którego może być dołączony joystick ze standardem przemysłowym (IBM). W podręczniku pokazano wymagane ustawienie wtyku, stwarzające użytkownikowi mozliwość sprawdzenia prawidłowości tej operacji. Gniazdo przeznaczone do przyłączenia klawiatury znajduje się z boku obudowy jednostki centralnej komputera.

Jednostka centralna

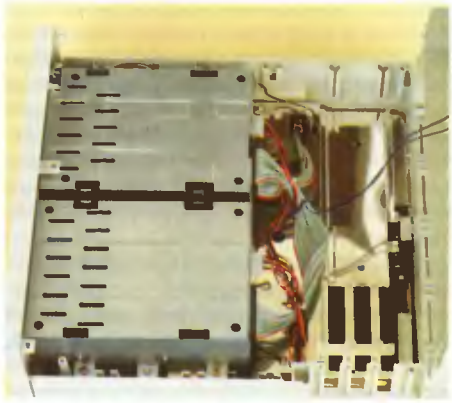
Jednostka ceentralna, zawierająca dwa wbudowane napędy dyskietek, jest wyjątkowo lekka: waży zaledwie 6 kg. Wynika to m.in. z rezygnacji z obudowy metalowej, ale zastosowanie do tego celu tworzywa wymaga od użytkownika zwiększonej ostrożności. Do zredukowania ciężaru przyczyniło się również przeniesienie zasilacza sieciowego z jednostki centralnej do obudowy monitora. Pozytywnym skutkiem ubocznym tej decyzji jest brak wentylatora.

Oczywiście ciężar jednostki centralnej nie moze być kryterium wydajności komputera. W wypadku PC 1512 wydajność ta wynika z zastosowania układów o wielkim stopniu scalenia w najnowocześniejszym, bardzo oszczędnym wykonaniu (tzw. *gate arrays*).

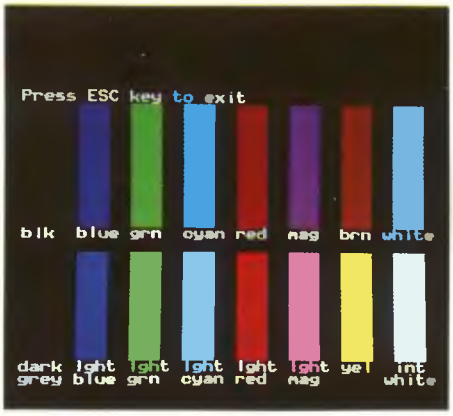
W jednostce centralnej nie użyto stosowanego powszechnie w tej klasie sprzętu mikroprocesora Intel 8088. Jej podstawą jest 16-bitowy mikroprocesor 8086 z zegarem 8 MHz, który umożliwia uzyskanie dużych szybkości przetwarzania. Cechę tę dodatkowo wzmacnia możliwość wprowadzenia koprocesora arytmetycznego 8087.

Wprowadzanie do jednostki centralnej dodatkowych kart może jednak powodować dla użytkownika pewne trudności, ponieważ dojście do płytki głównej jest dość skomplikowane. Nie wystarczy bowiem tylko zdjęcie wierzchu obudowy – konieczne jest również wyjęcie napędów dyskietek. Należy przy tym uważać, aby przy tej operacji nie wpadły do wnętrza obie śruby mocujące, umieszczone pomiędzy napędami. Odstęp między nimi jest wyjątkowo mały, co powoduje, że usunięcie zwolnionych z obudowy śrub wymaga użycia pincety lub obcążek.

Również przy rozszerzeniu pamięci operacyjnej z 512 do 640 KB konieczne są pewne operacje montażowe. Gniazda dla dalszych



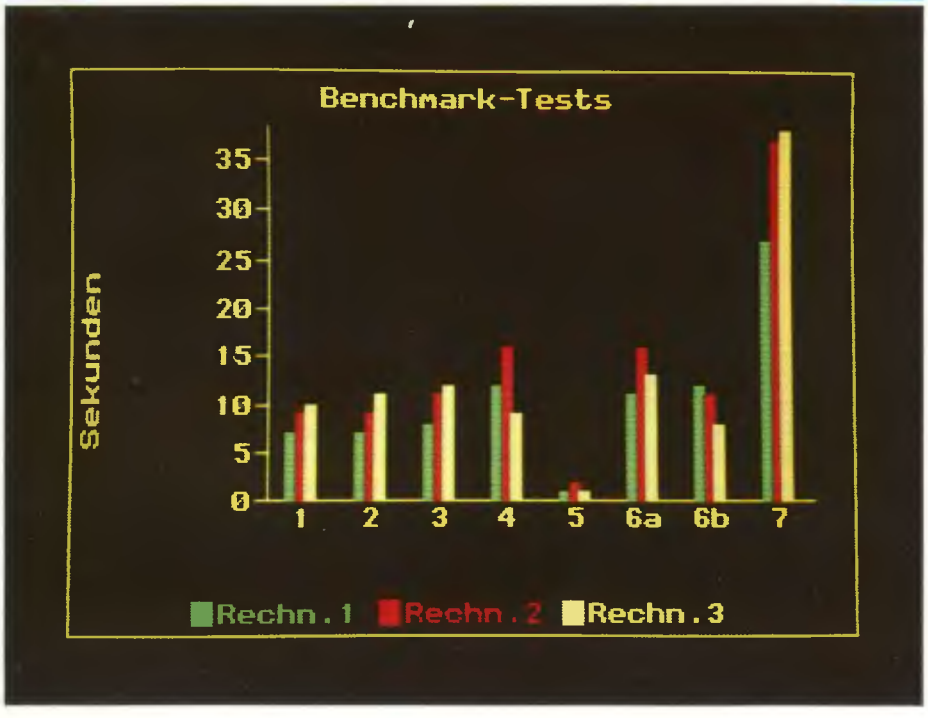
Dostęp do płytki głównej blokują dwa napędy dyskietek oraz przymocowane śrubami metalowe przykrycie



Doskonałe odtwarzanie barw jest dużym plusem monitora

Komputer 1: Amstrad PC 1512DD (procesor 8086)
Komputer 2: Wyrób importowany (procesor 8088-2)
Komputer 3: Wyrób importowany (procesor 80286)

	Operacja	Wartość zmiennych	Liczba przebiegów
Program wzorcowy 1:	A=2xX	X = 10	5000
Program wzorcowy 2:	A=SOR(X)	X = 10	5000
Program wzorcowy 3:	A=2x5 + 10 – 20		5000
Program wzorcowy 4:	A=COS (X)	X = 0,5	1000
Program wzorcowy 5:	PRINT X	X = 1	200
Program wzorcowy 6a:	zapisać na drukarce wartość X		
Program wzorcowy 6b:	odczytać z dyskietki wartość X	X = 10	2000
Program wzorcowy 7:	uzyskać liczbę liczb pierwszych w przedziale od 2 do 5000 (sito Eratostenesa)		1



układów pamięci znajdują się na płytce głównej, ale dostęp do nich z lewej strony blokuje napęd dyskietek.

Na płytce głównej znajdują się gniazda tylko dla trzech dodatkowych kart. Należy przy tym uwzględnić fakt, że w PC 1512 zbędne jest stosowanie niektórych dodatkowych kart.

Niepotrzebna jest więc często stosowana karta zegarowa (zegar czasu rzeczywistego umieszczono na płycie głównej). To samo odnosi się do kart z interfejsami: równoległym i szeregowym, które zawarte są w wyposażeniu standardowym. Umieszczone na tylnej stronie obudowy gniazda mają przewody wyprowadzone z płytki głównej.

Również na próżno można poszukiwać odrębnego adaptera dla monitora. Został on umieszczony na płytce głównej, natomiast gniazdo do przyłączenia monitora można znaleźć na tylnej stronie komputera, podobnie jak złącza dla myszy. Należy ona również do wyposażenia standardowego, ma swoje gniazdo obok gnizda klawiatury oraz pokrętła regulatora siły głosu.

Odtwarzanie barw

Szczególną pozycję w PC 1512 zajmuje monitor, ponieważ, podobnie jak w poprzednich modelach tej firmy, służy on nie

tylko do odtwarzania obrazu, ale również do zasilania w prąd całego systemu. Bez monitora nie można więc nic zrobić, w każdym razie do chwili, gdy handel nie zaoferuje odrębnego zasilacza. Oznacza to, że użytkownik musi mieć włączony firmowy monitor również wtedy, gdy będzie korzystał poprzez specjalną kartę adaptera z innego monitora, np. o dużej rozdzielczości (EGA, Herkules).

Testowany przez nas komputer był wyposażony w 14-calowy monitor kolorowy, którego bardzo naturalne odtwarzanie barw nie nasuwało żadnych zastrzeżeń. Potwierdza to załączone zdjęcie ekranu, zawierające 16 słupków w ośmiu kolorach po dwa stopnie intensywności barw.

Podczas eksploatacji negatywnie oceniono odbicie obrazu: jest ono bardzo odczuwalne przy intensywnym korzystaniu z ekranu. Natomiast doskonałym rozwiązaniem jest zagłębienie w górnej części obudowy jednostki centralnej, przeznaczone do mocowania ruchomej podstawy monitora. Pozwala to stosować zmienne nachylenie ekranu.

Napędy dyskietek 5,25" o pojemności po 360 KB wyróżniają się bardzo cichą pracą. Zamiast jednego napędu dyskietek, komputer można wyposażyć w dysk o pojemności 10 lub 20 MB.

Eksploatacja

Zakres dostawy obejmuje cztery dyskietki programowe. Za pomocą dyskietki systemowej można wprowadzić system operacyjny MS-DOS 3.2 w nowej wersji sieciowej. Wersja ta zapewnia eksploatację wszystkich znanych programów MS-DOS, przy czym w indywidualnych wypadkach konieczne jest ładowanie pliku ANSI.SYS za pomocą pliku CONFIG.SYS.

Jednakże zakłada się, że programy te są instalowane dla trybu Color Graphics Card i działają niezależnie od tego, czy aktualnie korzysta się z monitora monochromatycznego czy kolorowego. Dopiero po zainstalowaniu odpowiedniej dodatkowej karty można przyłączyć dodatkowy monitor, co pozwala zastosować również programy przeznaczone do wyprowadzania obrazów o podwyższonej rozdzielczości. Nie stwierdzono jakichkolwiek trudności związanych z kompatybilnością.

Za pomocą dyskietki systemowej DOS Plus 1.2 wprowadza się system operacyjny firmy Digital Research, kompatybilny z MS-DOS, który umożliwia wykorzystanie również programów eksploatowanych w systemie CP/M 86. Dodatkową zaletą jest emulowanie przez DOS Plus dwóch typów terminali: powszechnie stosowanego terminala ANSI oraz VT 52.

Sterownik VT 52 jest jak wiadomo stosowany w IBM PC. W porównaniu do terminala ANSI, VT 52 ma wiele dodatkowych funkcji, wyróżniających się prostą budową rozkazów. Pod nadzorem systemu DOS Plus następuje prawidłowa interpratacja sekwencji ESCAPE zarówno ANSI, jak i VT 52. Zbędne jest dodatkowe ładowanie ANSI.SYS pod nadzorem DOS Plus.

Pod nadzorem systemu DOS Plus można wykonywać równocześnie wiele programów, a oprócz aktualnie wykorzystywanego oprogramowania, (które jest wyświetlane na ekranie), mogą być w drugim planie wykonywane inne programy. Pod warunkiem, że programy te odpowiadają typowi danych CMD.

Dwa specjalne programy drugiego planu wraz z PRINT.CMD oraz ALARM.CMD znajdują się na dyskietce systemowej. Podczas gdy PRINT.CMD odpowiada programowi PRINT.COM w systemie MS-DOS, w wypadku ALARM.CMD chodzi o elektroniczny terminarz. Notatki wprowadzane do tego terminarza są w odpowiednim momencie wyprowadzane na ekran w formie komunikatu, oczywiście przy założeniu, że nie zapomni się nastawić terminarza na ON.

Dyskietka inicjująca GEM ładuje na początku również DOS Plus. Za pomocą dyskietki GEM Desktop można zorganizować poziom użytkownika GEM. Dzięki temu można dysponować również GEM Paint oraz językiem programowania Locomotive Basic 2, który jest tu przeznaczony do zastosowań GEM.

Za pomocą programu konfigurującego NVR (*Non-Volatile-RAM*) można przechowywać zmienne systemowe w zasilanym bateryjnie RAM. Należą do nich m.in. data i godzinowy czas, kody klawiszy, ustawienie interfejsu szeregowego, tryb standardowy ekranu oraz wielkość dysku RAM zorganizowanego jako napęd C:. Jednakże w wypadku wyłączenia komputera programy umieszczone na dysku RAM traci się całkowicie.

W czasie normalnej eksploatacji zwraca uwagę duża szybkość PC1512. Dlatego bardzo interesujące może być porównywanie z innymi mikrokomputerami. Na podstawie testu przeprowadzonego za pomocą programów wzorcowych (ang. *benchmark test*), PC 1512 w zestawieniu z dwoma innymi wyrobami zajął pierwsze miejsce, przy czym w tzw. sicie Erastotenesa jego przewaga jest szczególnie wyraźna. Oczywiście widać tu wpływ zastosowania procesora 8086.

Ceny

Za zaledwie ok. 500 funtów (poniżej 800 dolarów) można otrzymać konfigurację podstawową, obejmującą jeden napęd dyskietek oraz monitor monochromatyczny. Egzemplarz testowany (PC 1512DDCH) kosztował 720 funtów (poniżej 1100 dolarów). W zestawieniu z jego wydajnością, ceny te są szczególnie niskie.

Reasumując, za stosunkowo niską cenę użytkownicy otrzymują komputer o bogatej konfiguracji, niezawodny i poręczny. Wydaje się, że również w Polsce renoma, jaką zdobył sobie Amstrad poprzednimi modelami, sprawi, że oczekiwać możemy sporego napływu tego komputera do kraju.

micro

Mikroprocesory

(dokończenie ze str. 7)

68451 – *Memory Management Unit* (MMU). Podstawowy układ zarządzania pamięcią. Umożliwia translacje adresów oraz zabezpieczenie całej przestrzeni adresowej. Wspomaga organizację pamięci wirtualnej, zapewnia obsługę 32 różnej wielkości segmentów pamięci. Istnieje możliwość łączenia wielu MMU w celu uzyskania żądanej liczby segmentów;

68452 – *Bus Arbitation Module* (BAM). Układ sterownika dostępu dla ośmiu procesorów 680xx. Zgodny ze specyfikacją VME. Obudowa DIL 28, technologia ALS, czas arbitrażu – maks. 52 ns;

68454 – *Intelligent Multiple Disc Controller* (IMDC). Układ z możliwością jednoczesnego sterowania czterema napędami dysków elastycznych lub sztywnych z interfejsem ST506 lub SA1000;

68459 – *Disk Phase Lock Loop* (DPLL). Układ wspomagający implementację sterownika IMDC;

68483 – *Advanced Graphic/Alfanumeric Controller* (AGAC). Generuje wektory, łuki i okręgi oraz dowolny krój liter. Sprzętowe wypełnianie obszarów dowolnymi wzorami. Zarządza własną pamięcią obrazu: maks. 16 płaszczyzn o rozdzielczości 2048 na 2048 pikseli, 256 kolorów. Rejestry przesuwające wewnątrz układu. Obudowa z 68 wyprowadzeniami;

68562 – *Dual Universal Serial Communications Controller* (DUSCC). Zawiera dwa kanały transmisji. Realizuje protokoły BISYNC DDCMP X.21, HDLC, ADCCP, SDLC, X.25. Może współpracować z kanałem DMA. Wewnętrzny programowany licznik (*timer*);

68564 – *Serial Input/Output* (SIO). Uniwersalny układ we-wy szeregowego. Może działać w trybie asynchronicznym, synchronicznym zorientownym bitowo (HDLC, SDLC) oraz synchronicznym zorientowanym bajtowo (IBM BiSync). Dwa wewnętrzne generatory zegarowe. Maksymalna szybkość transmisji – 1,25 Mb/s. Obudowa DIL 48;

68605 – *X.25 Protocol Controller* (XPC). Układ sterownika transmisji X.25. Dostosowany do współpracy z 68020 (32-bitowa szyna adresowa). Maksymalna szybkość transmisji – 10 Mb/s. Obudowa PLCC z 84 wyprowadzeniami;

68652 – *Multi-Protocol Communications Controller* (MPCC). Sterownik transmisji synchronicznych. Maksymalna szybkość transmisji 2 Mb/s. Obudowa DIL 40;

serii MC68000 (2)

68653 – *Polynominal Generator Checker* (PGC). Układ wykrywania i korekcji błędów transmisji. Dostosowany do współpracy z MPCC i EPCI;

68661 – *Enhanced Programmable Communications Interface* (EPCI). Uniwersalny sterownik szeregowej transmisji synchronicznej lub asynchronicznej. Istnieją trzy wersje, różniące się zestawem wewnętrznie generowanych częstotliwości transmisji. Obudowa DIL 28;

68681 – *Dual Asynchronous Receiver/ /Transmitter* (DUART). Zawiera dwa niezależne kanały transmisji. Maksymalna szybkość transmisji – 1 Mb/s. Wewnętrzny generator zegara transmisji. Programowany format transmisji. Dodatkowo: 6-bitowy port wejściowy. Obudowa DIL 48;

68901 – *Multi-function Peripherial* (MFP). 8-bitowy port we-wy, sterownik dla 16 wektorów przerwań (*Interrupt Controller*), cztery programowane liczniki, mogące generować zegar dla wewnętrznego asynchronicznego sterownika szeregowego we-wy. Obudowa DIL 48;

68905 – *Basic Memory Access Controller* (BMAC). Układ sterownika pamięci dynamicznej dla systemów opartych na szynie typu 68000;

68930 – *Digital Signal Processor* (DSP). Procesor sygnałowy z własną pamięcią ROM. Czas cyklu – 160 ns. Architektura typu Harvard 2 × 128 × 16 RAM, 512 × 16 ROM do zapamiętywania współczynników. DSP mogą być łączone kaskadowo. Przykładowo: FFT na ciągu 256 próbek typu zespolonego – 2 ms; autokorelacja dziesiątego rzędu na 240 próbkach (wynik 32-bitowy) – 1,8 ms. Obudowa DIL 48;

68931 – *Digital Signal Processor* (DSP). Jest to wersja 68930 bez wewnętrznej pamięci ROM, ale za to z zewnętrzną szyną danych/adresów dla pamięci ROM. Obudowa LCCC z 84 wyprowadzeniami.

Artykuł jest tylko zasygnalizowaniem możliwości procesorów rodziny 68000. Zainteresowanych dokładniejszymi informacjami odsyłamy do literatury specjalistycznej.

JERZY ORKISZEWSKI
PIOTR PARLEWICZ

Ankieta

Wypełnienie i wysłanie tej ankiety wpłynie na lepsze dostosowanie treści i sposobu redagowania MIKROKLANU do Twoich potrzeb i oczekiwań, a także przyczyni się do poprawy dystrybucji czasopisma. Konstrukcja ankiety maksymalnie ułatwia jej wypełnienie: wystarczy tylko zakreślić numer wybranej odpowiedzi, a w niektórych punktach – wpisać drukowanymi literami informację rozszerzającą. Z wdzięcznością przyjmiemy również dodatkowe uwagi, życzenia i propozycje, które prosimy dołączać do ankiety na osobnej kartce.

Redakcja

Wszyscy Czytelnicy, którzy w terminie do 31 lipca 1987 r. nadeślą:

1) kompletnie wypełnioną ankietę

2) kupon (narożnik) wycięty z numeru 2'87, naklejony w zaznaczonym miejscu,

wezmą udział w losowaniu drukarki GP 50S oraz 10 nagród pocieszenia w postaci uprawnień do bezpłatnego otrzymywania w 1987 r. numerów MIKROKLANU.

1. Ile masz lat? . . . . . .

2. Jakie masz wykształcenie?
2.1. Wyższe
2.2. Średnie
2.3. Podstawowe

3. Gdzie mieszkasz?
3.1. W mieście wojewódzkim
3.2. W innym mieście
3.3. Na wsi

4. Czy masz przygotowanie informatyczne?
4.1. Nie mam przygotowania
4.2. Informatyczne studia wyższe
4.3. Informatyczna szkoła pomaturalna
4.4. Informatyczne kursy
4.5. Samokształcenie i praktyka zawodowa

5. Czy korzystasz z mikrokomputera?
5.1. Nie
5.2. Tak (od . . . . . roku)

6. Gdzie i z jakiego typu mikrokomputera korzystasz?
6.1. W pracy . . . . . .
6.2. W szkole . . . . . .
6.3. W klubie . . . . . .
6.4. U rodziny, przyjaciół . . . . . .
6.5. Mam własny . . . . . .

7. Czy Twoje zainteresowania tematyką mikrokomputerową dotyczą
7.1. Konstrukcji sprzętu komputerowego
7.2. Oprogramowania
7.3. Zastosowań – podaj dziedziny . . . . . .
7.4. Innych zagadnień – podaj jakich . . . . . .
7.5. Nie są ukierunkowane

8. Czy masz praktyczne umiejętności
8.1. Budowy i naprawy sprzętu komputerowego
8.2. Programowania w języku . . . . . .
8.3. Korzystania z systemu operacyjnego . . . . . .
8.4. Nie mam

9. Czy treść MIKROKLANU jest dla Ciebie
9.1. Zrozumiała
9.2. Trudna
9.3. Bardzo trudna

10. Czy uważasz, że należy zwiększyć liczbę artykułów na temat
10.1. Sprzętu
10.2. Oprogramowania
10.3. Zastosowań – podać w jakiej dziedzinie . . . . . .
10.4. Na inne tematy – podać jakie . . . . . .
10.5. Obecna struktura jest właściwa

11. Czy nabycie MIKROKLANU w Twoim miejscu zamieszkania jest
11.1. Łatwe
11.2. Trudne
11.3. Niemożliwe

miejsce na wklejenie kuponu z nr 2/87

WYPEŁNIĆ DRUKOWANYMI LITERAMI

imię i nazwisko

ulica, nr domu, nr mieszkania

kod pocztowy, miejscowość

mikroklan

Redakcja Mikroklanu

00-950 Warszawa

skrytka 1004

miejsce na znaczek

Interfejs szeregowy

Wydruk 1

```
*HISOFT GENS3M2 ASSEMBLER*
      ZX SPECTRUM

CoPyri9ht (C) HISOFT 1983,4
All ri9hts reserved

Pass 1 errors: 00

          10 *C-
FA00      20         ORG  64000          ; Poczatek Programu
          30
          40 ;************************
          50 ;
          60 ;  > INICJACJA <
          70 ;
          80 ;************************
          90
FA00     100 INICJ:  LD   HL,(CHANS)
FA03     110         LD   BC,3*5
FA06     120         ADD  HL,BC
FA07     130         LD   BC,OUT
FA0A     140         LD   (HL),C
FA0B     150         INC  HL
FA0C     160         LD   (HL),B
FA0D     170         XOR  A
FA0E     180         LD   A,15
FA10     194         JP   OUT
         200
5C4F     210 CHANS:  EQU  #5C4F
         220
         230 ;************************
         240 ;
         250 ; > OUTPUT I ZNAK <
         260 ;
         270 ;************************
         280
FA12     290 OUT     DI   ; wylacz Przerwania
FA13     300         CPL  ; zaneGuj zaw. danych
FA14     310         PUSH HL             ; zachowaj wartosci
FA15     320         PUSH DE             ; rejestrow na stosie
FA16     330         PUSH BC
FA17     340         PUSH AF
FA18     350         RRCA ; ustaw Pierwszy bit
FA19     360         RRCA ; danych na Pozycji D3
FA1A     370         RRCA ; przez cykliczne obroty
FA1B     380         PUSH AF             ; zach. biez.konfi9uracje
FA1C     390         LD   L,9            ; ustaw licznik bitow
FA1E     400         LD   B,255          ; ustaw adres Portu
FA20     410         LD   C,254
FA22     420         LD   A,%00011111    ; bit START i border 7
FA24     430 START   OUT  (C),A          ; wyslij bit startu
FA26     440         CALL DEL            ; skok do Petli oPozn.
FA29     450         NOP  ; skorY9owanie czasowe
FA2A     460         NOP
FA2B     470         NOP
FA2C     480         NOP
FA2D     490         NOP
FA2E     500 DATA    POP  AF             ; odzyskaj biez.kont. A
FA2F     510         RRCA ; ustaw nastePny bit
FA30     520         PUSH AF             ; zach.biez.konf.danych
FA31     530         DEC  L              ; zmniejsz licznik bitow
FA32     540         JP   Z,STOP         ; skok jesli koniec danych
FA35     550         NOP  ; skorY9owanie czasowe
FA36     560         AND  %00010000      ; wybierz bit D4
FA38     570         LD   D,A            ; ustaw wartosc D4 tez
FA39     580         RRC  D              ; na Pozycji D3
FA3B     590         OR   D
FA3C     600         OR   %00000111      ; border 7
FA3E     610         OUT  (C),A          ; wyslij bit danych
FA40     620         CALL DEL            ; skok do Petli oPozn.
FA43     630         JP   DATA           ; nastePny bit danych
FA46     640 STOP    POP  AF             ; odzyskaj zaw.akumulatora
FA47     650         LD   A,%00000111    ; bit STOP i border 7
FA49     660         OUT  (C),A          ; wyslij bit STOP
FA4B     670         CALL DEL            ; Pierwszy
FA4E     680         CALL DEL            ; i drugi
FA51     690         POP  AF             ; odzyskaj rejestry
FA52     700         POP  BC             ; ze stosu
FA53     710         POP  DE
FA54     720         POP  HL
FA55     730         EI   ; wlacz Przerwania
FA56     740         RET  ; koniec Procedury
         750
FA57     760 DEL:    PUSH HL             ; Poczatek Petli
FA58     770         PUSH DE             ; oPoznia jacej
FA59     780         LD   HL,926         ; wartosc licznika dla
FA5C     790         LD   DE,1           ; 150 bodow
FA5F     800         SCF
FA60     810         CCF
FA61     820 LOOP:   SBC  HL,DE
FA63     830         JP   NZ,LOOP
FA66     840         POP  DE
FA67     850         POP  HL
FA68     860         RET  ; koniec Petli

Pass 2 errors: 00

Table used:   102  from   200
```

y do ZX Spectrum

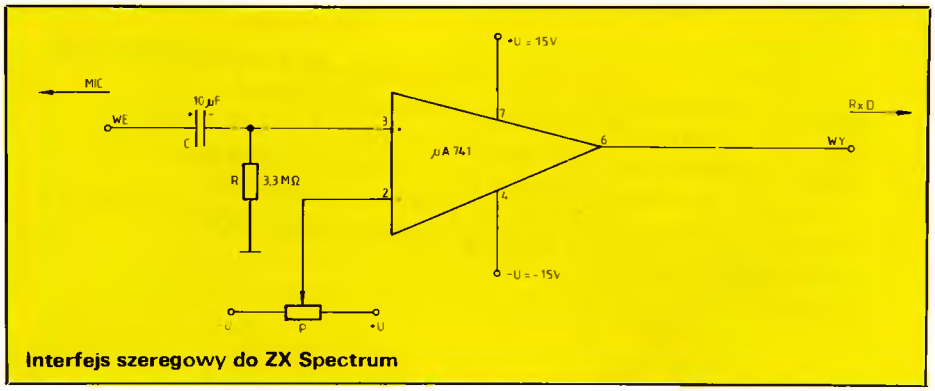
Większość użytkowników ZX Spectrum, po początkowym okresie zabawy, zaczyna poważniej traktować swój sprzęt, chce poznać, wykorzystać i rozszerzyć jego możliwości. Wówczas okazuje się, że niezbędnym elementem nawet najprostszego systemu jest drukarka, z przyłączeniem której często wiążą się określone problemy.

Wśród wielu używanych obecnie drukarek część stanowią drukarki z portem szeregowym pracującym w systemie V24. Przedstawiony poniżej projekt dotyczy prostego przyłączenia takiej drukarki do ZX Spectrum. Zastosowano w nim metodę programowego przesyłania danych. Europejski standard V24 przewiduje następującą formę przesyłania danych:

- znormalizowane prędkości przesyłania 75, 150, 300, 600, 1200, 2400, 4800 i 9600 (bitów na sekundę),
- jeden bit START (logiczne 0),
- osiem bitów danych,
- jeden lub więcej bitów STOP (logiczne 1),
- poziomy logiczne V24:
  1 --(−3V do −15V)
  0 --(+3V do −15V).

Wydruk 2

```
   1 REM PROGRAM TWORZY KOD
MASZYNOWY "OUT" I ZAPISUJE NA
TASMIE
   2 REM "OUT" URUCHAMIA SIE
PRZEZ RANDOMIZE USR 64000
   4 CLEAR 63999
   5 LET S=0
  10 LET CS=12859
  20 FOR T=64000 TO 64104
  30 READ A: POKE T,A
  40 LET S=S+A
  50 NEXT T
  60 IF S<>CS THEN PRINT "POPRAW
 DANE W LINIACH DATA": STOP
  70 SAVE "OUT"CODE 64000,105
  80 STOP
1000 DATA 42,79,92,1,15,0,9,1,18
1010 DATA 250,113,35,112,175,62
1020 DATA 15,24,0,243,47,229,213
1030 DATA 197,245,15,15,15,245
1040 DATA 46,9,6,255,14,254,62
1050 DATA 31,237,121,205,87,250
1060 DATA 0,0,0,0,0,241,15,245
1070 DATA 45,202,70,250,0,230,16
1080 DATA 87,203,10,178,246,7
1090 DATA 237,121,205,87,250,195
1100 DATA 46,250,241,62,7,237
1110 DATA 121,205,87,250,205,87
1120 DATA 250,241,193,209,225
1130 DATA 251,201,229,213,33,158
1140 DATA 3,17,1,0,55,63,237,82
1150 DATA 194,97,250,209,225,201
```



Interfejs szeregowy do ZX Spectrum

W podanym przykładzie prędkość transmisji jest określona programowo na 150 bodów. Wykorzystano fakt, że częstotliwość zegara w Spectrum jest stabilizowana i wynosi 3,5 MHz. Znając czas wykonania każdej instrukcji, można tworzyć pożądaną częstotliwość z zupełnie wystarczającą dokładnością rzędu 0,1%, organizując odpowiednią pętlę opóźniającą. O wyborze prędkości 150 bodów zadecydowały dwa fakty:

- brak informacji o stanie drukarki, a przy 150 bodach bufor danych nigdy nie będzie przepełniony, ponieważ szybkość drukowania jest nieco większa,
- ograniczony czas narastania napięcia wyjściowego wzmacniacza µA741 (slew rate), który przy 9600 bodach daje duże zniekształcenia.

Jako port wyjściowy służy wewnętrzny port Spectrum o adresie 254, którym ustawia się stan na wejściach głośnika i MIC poprzez bity D3, D4. Istnieją tu tylko dwa problemy. Po pierwsze w systemie V24 spoczynek oznacza się stałą jedynką lub brakiem napięcia; ponieważ układ RC nie pozwala na utrzymanie stałego napięcia na wyjściu MIC, to zawartość danych należy najpierw logicznie zanegować, a następnie użyć µA741 w konfiguracji wzmacniacza nieodwracającego! Po drugie, przy wysyłaniu bitu danych korzystnie jest wysterować nim zarówno wejście MIC, jak i głośnik. Co prawda drukowaniu towarzyszy ciche burczenie, ale uzyskujemy większą sprawność działania, związaną ze sprzężeniem obu tych wyjść i ponad pięciokrotnym wzmocnieniem sygnału w gniazdku MIC oraz brakiem zniekształceń.

Do uruchomienia układu jest potrzebny jedynie dobór napięcia odniesienia komparatora za pomocą potencjometru montażowego. W tym celu jest przydatny oscyloskop. Układ został zmontowany i uruchomiony, a pokazany wydruk był obsługiwany w opisany sposób. Podobnie pracuje również port szeregowy mikrokomputera Meritum. Koszt urządzenia jest niewielki, co jest istotną zaletą w porównaniu z ceną Interface I lub koniecznych portów Intela 8251, 8253. Komputer jest zabezpieczony przed zwarciem wejścia MIC, a przyłączanie do niebuforowanej szyny nieraz już skończyło się wymianą pamięci.

Jeszcze słowo o programie obsługi. Program OUT wysyła, zgodnie z normą V24, kod znaku zawarty w akumulatorze. Istotne jest wyłączenie przerwań ze względu na częstą ingerencję ULA, zakłócającego równomierną pracę procesora i zależności czasowe.

Przy obsłudze programu w BASICU Spectrum konieczny jest jeszcze program obsługi związany ze specyfiką oznaczeń słów kluczowych, UDG, znaków sterujących itd. Takimi programami dysponują kluby, firmy i użytkownicy. Natomiast jest przedstawiony sposób adresowania kanału drukarki. Po wykonaniu wstępnej procedury w zmiennych systemowych zostaje wprowadzony adres, od którego rozpoczyna się obsługa znaku wysyłanego na drukarkę. Program w przedstawionej formie może być stosowany do drukowania łańcuchów i liczb w BASICU, dowolnych wydruków Pascala i asemblera napisanych na Spectrum. Sam program OUT może być wykorzystany przy układaniu własnej procedury drukowania zawartości ekranu. Jedyną niedogodnością jest konieczność wyłączania układu podczas wczytywania programów z taśmy, ale wszystkiego nie można mieć za darmo. Niewątpliwą zaletą projektu jest możliwość bezpiecznej i owocnej zabawy.

TOMASZ CIERPISZ

MS-DOS i jego korzenie

System operacyjny „ożywia" sprzęt – bez niego komputer jest tylko zbiorem układów elektronicznych. Jakie zadania ma ten pomocny duch i jak je realizuje – wiedzą w szczegółach tylko nieliczni użytkownicy. W kolejnych odcinkach wyjaśnimy znaczenie oraz słabe i mocne strony różnych systemów operacyjnych.

W świecie dużych komputerów, a często również minikomputerów, obowiązuje zasada, że są one sprzedawane z własnym, opracowanym przez producenta sprzętu, systemem operacyjnym. Często możliwości takiego systemu mają zasadnicze znaczenie przy podejmowaniu decyzji zakupu komputera. Na rynku mikrokomputerów można tymczasem zaobserwować dwa interesujące zjawiska. Po pierwsze – istnieją tu systemy, które działają na całkowicie różnych komputerach (np. CP/M lub Unix), po drugie – dla niektórych komputerów oferuje się wiele różnych systemów operacyjnych.

Pierwszy szlagier – CP/M

Historia mikrokomputerowych systemów operacyjnych rozpoczęła się w 1972 r., gdy producent półprzewodników – firma Intel – wprowadziła na rynek pierwszy procesor 8-bitowy o symbolu 8008. Założona przez Gary Kildalla firma MAA (Microcomputer Applications Associates), poprzednik Digital Research Institute (DRI), na zlecenie Intela zdefiniowała i zaimplementowała na 8008 nowy język programowania systemów komputerowych o nazwie PL/M (Programming Language for Microprocessors).

Wkrótce Kildall zaproponował Intelowi, aby jako uzupełnienie PL/M opracować system operacyjny o nazwie CP/M (Control Program for Microprocessors). Pomysł ten został jednak przyjęty przez firmę Intel bez entuzjazmu. MAA zdecydowała się wówczas opracować CP/M na własny koszt oraz samodzielnie wprowadzić go na rynek, bez adresowania dla konkretnego modelu komputera.

CP/M został wprawdzie zaprojektowany dla mikrokomputerów wyposażonych w procesor 8080, ale działa również na systemach z procesorem Z80. Wymagał wprawdzie stacji dyskietek – wówczas bardzo nowoczesnego rozwiązania pamięci masowej – ale równocześnie oferował funkcje niezbędne dla organizowania zawartości dyskietek. W szczytowym okresie ery mikrokomputerów 8-bitowych CP/M stał się dominującym systemem operacyjnym. Został on zaakceptowany i nadal jest jeszcze stosowany przez większość producentów komputerów. W 1981 r. istniało już ok. 200 tys. instalacji na ok. 3000 różnych konfiguracji sprzętowych.

Aby zrozumieć sukces rynkowy CP/M, należy porównać strukturę oprogramowania komputerów opartych na CP/M z konwencjonalnymi rozwiązaniami opartymi na języku BASIC, stosowanymi przykładowo na mikrokomputerach Apple II lub podobnie ukierunkowanych modelach firmy Tandy czy Commodore. W tych ostatnich producent ukrył system operacyjny w interpreterze języka BASIC. Polecenia systemu operacyjnego (np. wyprowadzenie jednego wiersza na ekran albo odczytanie rekordu ze zbioru na dyskietce) są zazwyczaj wywoływane za pomocą instrukcji języka BASIC.

CP/M znacznie ułatwia pracę tym programistom, którzy nie stosują języka BASIC. Zażądanie od dowolnego programu CP/M polecenia systemu operacyjnego, wymaga wykonania skoku przez podprogram do stałego adresu, gdzie jest zakodowana odpowiednia funkcja. W przeciwieństwie do tego, w systemach operacyjnych zawartych w języku BASIC dla każdego polecenia istnieje inny adres podprogramu, który w dodatku w różnych wersjach może się zmieniać. Pod nadzorem CP/M jest możliwe napisanie programu użytkowego w znacznym stopniu niezależnego od konkretnego modelu komputera. Oznacza to, że jeżeli program taki nie korzysta ze sprzętu poza systemem operacyjnym, to będzie działał również na każdym innym komputerze.

Dalszym mocnym punktem CP/M jest jego zdolność dostosowywania się do całkowicie różnych systemów komputerowych, jeżeli spełniają one wspomniane warunki sprzętowe. Z CP/M wywodzi się BIOS (Basic Input/Output System), który odnaleźć można w wielu innych systemach operacyjnych. CP/M jest zbudowany modularnie, przy czym wszystkie zależne od urządzeń zewnętrznych części programu oraz parametry są ujęte w odrębnym module, którym jest właśnie BIOS.

Dostosowanie CP/M do innego mikrokomputera dotyczy tylko BIOS. Pozostałe części systemu operacyjnego, a mianowicie interpreter poleceń CCP (Console Command Processor) oraz właściwe jądro systemu operacyjnego BDOS (Basic Disc Operating System), pozostają niezmienione. Adaptacja CP/M jest znacznie mniej pracochłonna niż pełna implementacja systemu operacyjnego. Ta właśnie cecha spowodowała, że CP/M został gorąco przyjęty przez użytkowników mniejszych komputerów.

Szybkie rozpowszechnienie CP/M doprowadziło do pojawienia się prawdziwej powodzi programów użytkowych do zastosowań biurowych, które z kolei przyczyniły się do dalszych sukcesów CP/M. Dzięki temu powstały tak efektywne programy – bestsellery, jak Wordstar lub Dbase II, których sukces absolutnie nie byłby możliwy bez niezależnego od producenta systemu operacyjnego.

Najbardziej rozpowszechnionym obecnie wariantem CP/M dla komputerów 8-bitowych jest wersja 2.2, znana również jako CP/M-80. Wprawdzie firma Digital Research wypuściła jeszcze dalszą wersję – 3.0 (CP/M Plus), ale może być ona prawidłowo wykorzystywana wyłącznie na specjalnie skonstruowanych komputerach, które za pomocą tzw. przełączania banków pamięci przełamują obowiązującą w obszarze 8-bitowców granicę 64 KB. Jednak CP/M Plus nie zdołał dotąd osiągnąć istotnych sukcesów rynkowych.

Co to jest system operacyjny?

Zanim głębiej wejdziemy w historię rozwoju systemów operacyjnych, należy zrobić małą dygresję w zagadnienia informatyki i wyjaśnić, o co właściwie chodzi w systemie operacyjnym. Mówiąc najprościej, jego zadania można porównać z obowiązkami działu organizacji przedsiębiorstwa. Dział ten nie pracuje bezpośrednio na rzecz produkcji, ale musi stale dbać o bezkolizyjne współdziałanie różnych komórek własnego przedsiębiorstwa. Analogicznie do tego system operacyjny, nazywany czasem programem organizacyjnym, organizuje wzajemną współpracę poszczególnych części składowych komputera.

Jeżeli spytamy informatyka o funkcję systemu operacyjnego, to zwykle otrzymamy zwięzłą, ale niezbyt precyzyjną odpowiedź: *System operacyjny ma za zadanie zarządzać zasobami operacyjnymi komputera*. Pojęcie *zasoby operacyjne* oznacza jednostkę centralną (CPU – Central Processing Unit), pamięć operacyjną oraz całość *peryferii*: klawiaturę, monitor ekranowy, drukarkę, pamięci masowe (dyskietki i dyski sztywne), modemy itp. urządzenia. Z pojęciem *za-*

rządzanie wiąże się wiele funkcji, spośród których należy wymienić niektóre najważniejsze:

● Operacja inicjowania, a więc pierwotne wprowadzenie systemu operacyjnego z dyskietki lub dysku sztywnego po włączeniu komputera. Jest to procedura, za pomocą której komputer – przy pustej jeszcze pamięci operacyjnej – zaopatruje się sam w rozkazy potrzebne do rozpoczęcia działania systemu. Musi on, podobnie jak niegdyś baron Münchausen, wydobyć się z bagna za własną czuprynę. Operacja ta, w języku angielskim określana jest terminem *bootstrapping*. Do wykonania tej operacji zawsze potrzebna jest pamięć stała ROM (Read Only Memory). W prostych komputerach domowych cały – „zapakowany" w języku BASIC – system operacyjny znajduje się w ROM, co oczywiście eliminuje operację wprowadzania systemu. W komputerach z dyskiem sztywnym pamięć ROM zawiera niewielki program wczytujący system operacyjny z dysku.

● Po załadowaniu i rozpoczęciu działania systemu, inicjalizacja wprowadza zasoby komputera w zdefiniowany stan początkowy (skasowanie zawartości ekranu, ustawienie napędów dyskietek na pierwszą ścieżkę).

● Organizowanie współpracy między jednostką centralną i urządzeniami peryferyjnymi, polegające m.in. na koordynacji w czasie operacji wejścia i wyjścia. Jeżeli np. na dyskietce trzeba zapisać dane, to system operacyjny powinien dopilnować, aby operacja zapisu oczekiwała tak długo, aż właściwy sektor dyskietki znajdzie się pod głowicą zapisu.

● Przyjmowanie przez system operacyjny instrukcji użytkownika, który tym sposobem wywołuje potrzebne procedury systemu, inicjuje wykonanie programów lub włącza urządzenia. Tę część działań określa się jako poziom lub interfejs użytkownika. W tym celu system operacyjny daje użytkownikowi do dyspozycji język instrukcji, który może być bardzo prosty – przykładowo sześć poleceń CCP systemu CP/M – albo skonstruowany w sposób wyjątkowo efektywny, jak np. Unix-Shell. Do poleceń systemowych zalicza się m.in. listowanie skorowidza dyskietki oraz zmianę nazw i kasowanie zbiorów danych.

● Wywoływanie poleceń systemu z poziomu programu za pomocą wywołań podprogramów zwanych wywołaniami systemowymi (ang. *system calls*). W ten sposób program użytkowy może np. odczytywać dane z dyskietki lub wprowadzać je na ekran. Szczególne znaczenie wywołania systemowego polega na tym, że programista nie musi zajmować się szczegółami sprzętowymi (może np. pomijać adres, za pomocą którego ma być wywołana drukarka czy nazwy poszczególnych instrukcji dla sterownika dysku sztywnego). Mówi się więc, że system operacyjny „ukrywa" szczegóły sprzętu. Właściwość ta, w wypadku zmian sprzętu, pozwala uniknąć modyfikacji wszystkich programów. Przykładowo, gdy stację dyskietek zastąpi się dyskiem sztywnym, to sterowanie takiego dysku przejmuje odpowiednio zaadaptowany system operacyjny. Oznacza to, że istniejące programy użytkowe będą współpracować z tym dyskiem tak samo, jak poprzednio z dyskietkami. Podobnie wygląda zastosowanie programu na innym komputerze z tym samym systemem operacyjnym.

Powyższa charakterystyka zawiera istotne cechy typowego mikrokomputerowego systemu operacyjnego klasy CP/M lub MS-DOS. Systemy operacyjne dla minikomputerów lub dużych komputerów (do których zalicza się m.in. Unix) dysponują jeszcze innymi możliwościami, które w systemach z podziałem czasu pozwalają realizować wieloprogramowość i eksploatację wielodostępną.

BIOS, BDOS, i CCP

Systemy operacyjne mają najczęściej budowę modularną, przy czym poszczególne moduły udostępniają możliwości systemu komputerowego na różnych poziomach. Zasadę tę można dobrze zilustrować na przykładzie modułu BIOS, w którym funkcje związane ze sprzętem umiejscowiono w tzw. generatorach. One to uruchamiają przyłączone do komputera urządzenia, przy czym dla każdego z nich muszą być osobno programowane. Generatory realizują swe funkcje za pomocą bardzo prostych operacji. Przykładowo za pomocą generatora obrazu BIOS można w CP/M na monitorze zapisać zawsze tylko jeden znak. Generator ten reaguje ponadto na znaki specjalne, takie jak „nowy wiersz" lub „skasować ekran". Za pomocą generatora dyskietek można na nich zapisać lub odczytać zawsze tylko jeden sektor, którego położenie powinno być określone za pomocą numerów ścieżki i sektora.

BDOS opiera się na BIOS i udostępnia możliwości systemu na wyższym poziomie, np. pozwala wyprowadzić na ekran cały ciąg znaków. Szczególne znaczenie ma logiczne zarządzanie dyskiem, za które jest właśnie odpowiedzialny BDOS. Grupuje on zawartość dysku w pliki (ang. *files*) i prowadzi ich skorowidz (ang. *directory*). Dzięki temu program nie potrzebuje każdorazowo adresować danych na dysku za pomocą numerów ścieżek i sektorów. Wystarczy bowiem podać tylko nazwę pliku oraz pozycję wewnątrz tego pliku, aby natychmiast uzyskać dostęp do potrzebnych danych. Polecenie BDOS może więc wyglądać następująco: „Czytaj jedenasty rekord pliku o nazwie MICRO.INH" lub „Dołącz dane z podanego buforu do pliku BETRSYST.TXT". Dzięki stosowaniu nazw plików możliwe jest zapamiętywanie przez różne programy swoich danych na wspólnej dyskietce lub dysku.

Podczas gdy funkcje BDOS mogą być realizowane tylko przez wywołania systemowe z programu, to zadaniem interpretera poleceń CCP jest współpraca z użytkownikiem. Pobiera on instrukcje z klawiatury, analizuje polecenia i przekształca w wywołania BDOS, a następnie je wykonuje.

Istnieją również rozszerzenia CP/M, które przez zastąpienie modułów CCP implementują własny interpreter poleceń o zwiększonej mocy. Należy tu wymienić Microshell, który nakłada na CP/M podobny jak w systemie Unix poziom obsługi z obszernym językiem poleceń, przeadresowywaniem operacji wejścia-wyjścia i przetwarzaniem potokowym.

Wróćmy teraz do historii mikrokomputerowych systemów operacyjnych. Opierając się na sukcesie CP/M, podjęto w Digital Research próbę zdobycia nowego obszaru rynku języków programowania i translatorów. Szczególnie dużo zainwestowano w opracowanie kompilatora dostosowanej do mikrokomputerów wersji języka PL/I. Tymczasem konkurencyjna firma Microsoft osiągnęła już znaczne sukcesy w rozwoju języków programowania, zwłaszcza języka BASIC.

Przeadresowywanie i przetwarzanie potokowe programów

```
DIR > DIR.DANE
```

Przeadresowywanie wyprowadzenia polecenia „DIR" do zbioru „DIR.DANE"

```
PROG < DANE.WEJŚĆ
```

Przeadresowywanie wprowadzeń do programu „PROG" ze zbioru „DANE WEJŚĆ"

```
PROG < DANE.WEJŚĆ > PRN
```

Przeadresowywanie wprowadzeń do programu „PROG" ze zbioru „DANE.WEJŚĆ" i wyprowadzeń na drukarkę (PRN)

```
DIR | SORT | MORE
```

Przetwarzanie potokowe programów DIR, SORT, i MORE: SORT sortuje wyprowadzenie DIR (a więc skorowidze); wyprowadzenie SORT (posortowany skorowidz) jest wyświetlane na ekranie przez MORE.

MS-DOS: wielki interes

Praktycznie każdy interpreter BASICA, który jest dostarczany łącznie z mikrokomputerem, pochodzi z firmy Microsoft, założonej w 1977 r. przez 21-letniego Bill Gatesa. Atak Digital Research okazał się posunięciem błędnym, które wprowadziło okresowo tę firmę w bardzo poważny kryzys. Koncentrując się na językach programowania, przeoczono właściwie nowe tendencje w rozwoju mikrokomputerów.

W 1978 r. Intel wprowadził na rynek 16-bitowe procesory 8086 i 8088. Cały świat oczekiwał, że Digital Research opracuje dla tych procesorów nową wersję CP/M. Na wiosnę 1979 r. firma Seattle Computer Products skonstruowała komputer jednoukładowy oparty na 8086. Pierwsze kontakty z Digital Research w sprawie CP/M-86 nie dały rezultatu i Seattle Computer Products zaprezentowała wspomniany komputer w listopadzie 1979 r. z niezależnym interpreterem BASICA firmy Microsoft (a więc z wbudowanym systemem operacyjnym).

Gdy w następnym okresie nadal nic się nie działo z CP/M-86, firma Seattle ponownie zaktywizowała się i w ciągu dwóch miesięcy skonstruowała system operacyjny podobny do CP/M, któremu nadano znamienną nazwę QDOS (Quick and Dirty Operating System). W sierpniu 1980 r. ukazał się on na rynku jako wersja 0.10. Wywołania systemowe w QDOS w znacznym stopniu były zbieżne z rozwiązaniami CP/M, natomiast wprowadzono w nim całkowicie nową organizację dyskietek.

W końcu 1980 r. Digital Research wyszedł na rynek z CP/M-86. Mniej więcej w tym samym czasie Seattle Computer Products dostarczył już wersję 0.3 QDOS, który został przemianowany na 86-DOS. W końcu Microsoft, w lipcu 1981 r. uzyskał wyłączność praw do systemu 86-DOS 1.14 i rozpoczął jego sprzedaż pod nazwą MS-DOS 1.0 (Microsoft Disk Operating System).

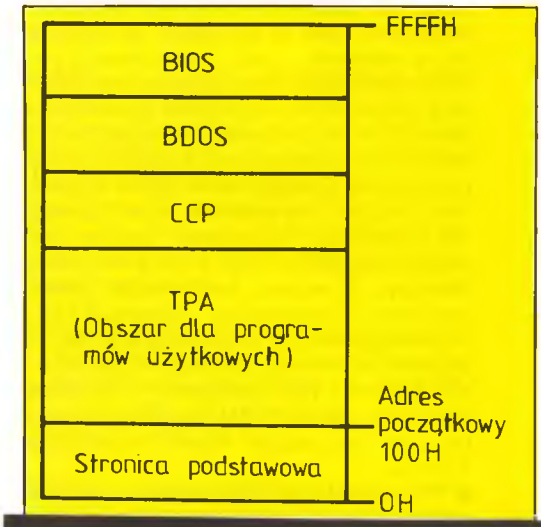
Dla producenta sprzętu, jakim był Seattle Computer Products, wydawało się to dobrym interesem, ponieważ firma Microsoft dysponowała znacznie lepszymi możliwościami sprzedaży. Jednak wkrótce nastąpiło zaskoczenie, gdy Microsoft zawarł bardzo korzystną transakcję z IBM. MS-DOS miał być sprzedawany w wersji dostosowanej do IBM PC pod nazwą PC-DOS.

Można tylko spekulować odnośnie tego, czy kierownictwo firmy Microsoft widziało rewelacyjny interes już wtedy, gdy kupowało 86-DOS. Można również domyślać się, dlaczego IBM przez współpracę z Microsoft wziął pod uwagę fakt, że jego konkurenci będą mogli stosować bardzo podobny do PC-DOS systemu MS-DOS i dzięki temu łatwiej produkować komputery kompatybilne.

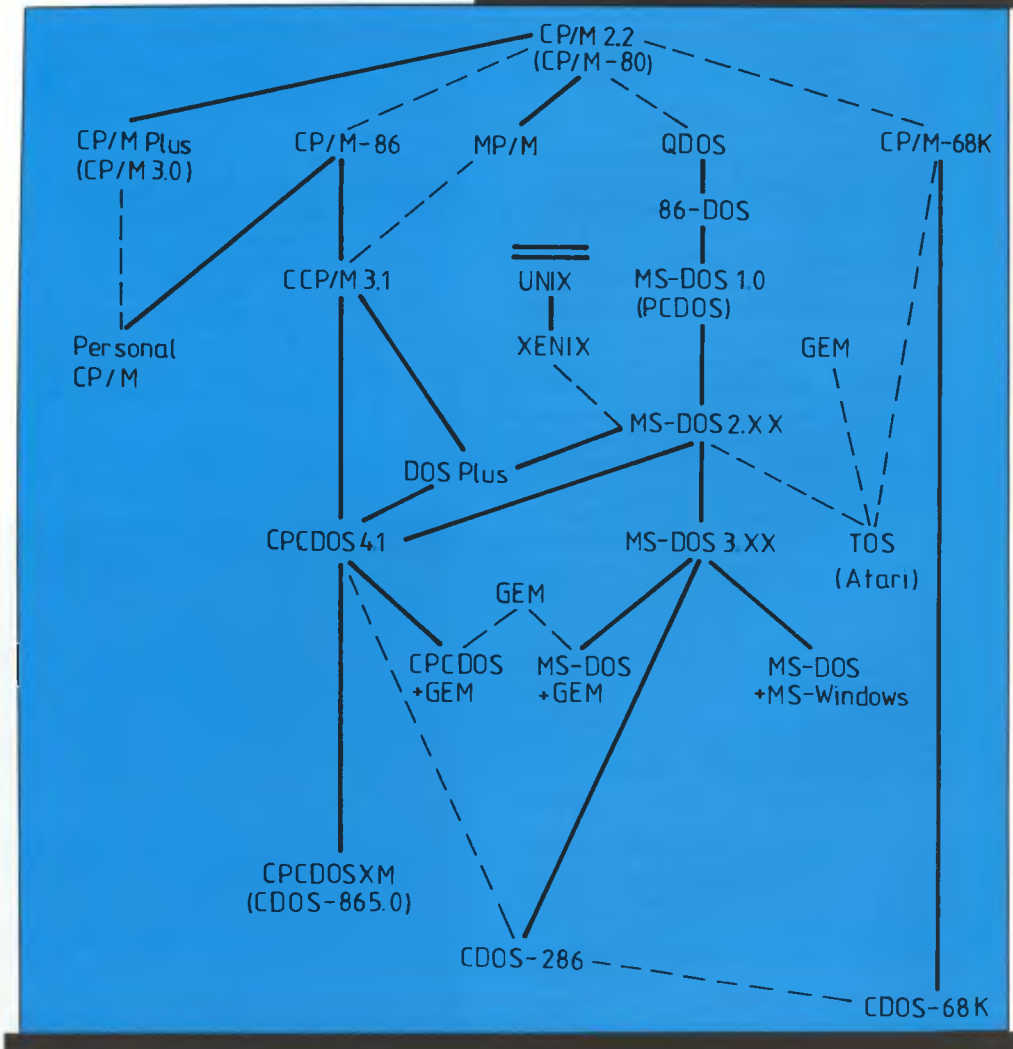
Zachowanie takie było całkowicie sprzeczne z dotychczasowymi zwyczajami IBM w obszarze sprzedaży dużych komputerów i minikomputerów, gdyż firma ta zawsze mocno trzymała w ręku oprogramowanie systemowe w tym celu, aby maksymalnie utrudnić życie producentom sprzętu kompatybilnego. Czy więc potężny IBM nie miał możliwości opracowania własnego systemu operacyjnego dla PC, a więc realizacji zadania, które dla niewielkiej firmy Seattle Computer Products wymagało zaledwie kilku miesięcy pracy?

W konsekwencji takiego rozwoju sytuacji Digital Research znalazł się w trudnej sytuacji, ponieważ IBM nie wykazywał w stosunku do CP/M-86 żadnego zainteresowania. Skutkiem tego system ten nie mógł zdobyć znaczącej pozycji na rynku amerykańskim. Natomiast firmie Microsoft udało się rozszerzyć swą wiodącą pozycję w dziedzinie języków programowania na obszar systemów operacyjnych i tym samym wysunąć się na pozycję lidera wśród producentów oprogramowania systemowego mikrokomputerów.

Microsoft postanowił intensywnie rozbudować MS-DOS. Ponieważ w tej samej firmie pracowano nad Xenixem, odmianą systemu Unix, uzyskane przy realizacji tego zadania doświadczenia można było zainwestować w dalszy rozwój MS-DOS. W wyniku tych prac w 1983 r. zaprezentowano wersję 2.11 tego systemu. Można w niej było stwierdzić wiele cech Unixa, m.in. hierarchiczny system plików oraz możliwość prze-



Schematyczna struktura CP/M i jego rozmieszczenie w pamięci mikrokomputera



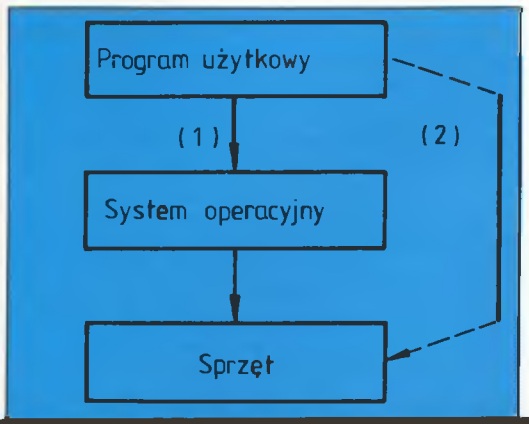
Drzewo genealogiczne rodziny CP/M-MS-DOS: linie ciągłe odpowiadają pełnej kompatybilności w górę, połączenia przerywane – kompatybilności funkcjonalnej o zróżnicowanym zakresie

adresowywania (ang. *redirection*) operacji wejścia i wyjścia, a także przetwarzania potokowego (ang. *pipelining*) programów.

Przeadresowywanie wyprowadzenia programu polega na tym, że dane przeznaczone do wyprowadzenia na ekran, są przesyłane do zbioru dyskowego albo innego urządzenia zewnętrznego (zwykle drukarki). Odwrotnie dzieje się przy przeadresowywaniu wprowadzania: dane pochodzące ze zbioru dyskowego lub z innej części mikrokomputera zastępują ich wprowadzenie z klawiatury. Przetwarzanie potokowe programów łączy obie możliwości: dane wyjściowe z jednego programu kierowane są najpierw do przejściowego zbioru na dysku, a dopiero stamtąd przechodzą jako dane wejściowe do programu drugiego.

Przy odrobinie zręczności powyższe cechy MS-DOS 2.11 można wykorzystać bardzo efektywnie. Przetwarzanie potokowe programów odgrywa w praktyce nieco mniejszą rolę, ponieważ ma sens tylko wtedy, gdy przeznaczone do przetwarzania programy spełniają określone wymagania. Mówi się wtedy o filtrach, które zostały napisane w stylu Unixa.

Niestety MS-DOS w wydaniu standardowym oferuje tylko filtry FIND, MORE oraz SORT. Oprócz tego niektóre narodowe klawiatury komputerów kompatybilnych z IBM PC nie zawierają potrzebnego do przetwarzania potokowego programów znaku „!", co wymaga każdorazowego programowego przełączania klawiatury na wersję amerykańską.



Dostęp programu użytkowego do sprzętu odbywa się za pośrednictwem systemu operacyjnego (1); rozwiązania „nieczyste" omijają program organizacyjny (2)

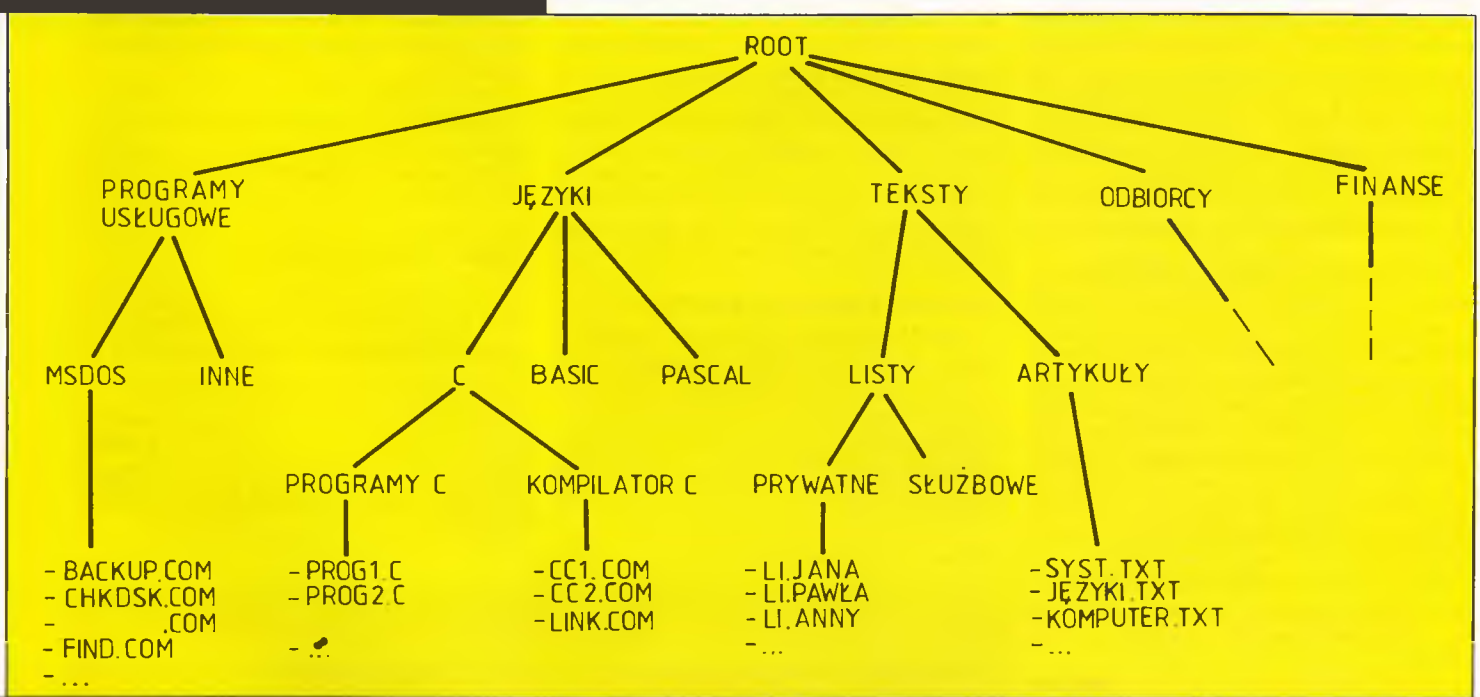
Zbiory danych

Największe znaczenie wśród nowych rozwiązań wersji 2.11 ma hierarchiczny system plików. Pomysł pochodzi z Unixa i stamtąd wszedł do wielu nowoczesnych systemów operacyjnych. Pomysł jest prosty, a jednocześnie elegancki: każda dyskietka lub dysk sztywny otrzymuje skorowidz (ang. *directory*), w którym wyspecyfikowane są pliki oraz ich położenie na dysku. Takie rozwiązanie występuje również w „płaskich" systemach plików, które przykładowo zostały zastosowane w CP/M lub Apple-DOS. Zdarza się jednak, że uporządkowany hierarchicznie skorowidz główny może zawierać również podskorowidze (ang. *subdirectories*), które z kolei wyposażone są w dalsze podskorowidze. W wyniku tego powstaje hierarchicznie uporządkowane „drzewo" plików, przez którego rozgałęzienia można znaleźć drogę do poszukiwanego pliku. Drzewo to składa się z nazw wszystkich skorowidzów umieszczonych pomiędzy skorowidzem głównym (ang. *root directory*), a więc „korzeniem" drzewa, a skorowidzem docelowym z poszukiwanym plikiem. Określenia te w Unixie są podzielone przez ukośnik „/". Niestety dla MS-DOS ustalono jako znak dzielący „\" lub ukośnik z lewym nachyleniem („\").

W przeciwieństwie do CP/M, który pozwala na podział dysku sztywnego na maksymalnie 16 obszarów użytkownika (ang. *user aresa*), za pomocą hierarchicznego systemu plików można zarządzać w sposób elegancki, logiczny i przejrzysty również dyskami sztywnymi o wielkiej pojemności. Poszczególne skorowidze mogą być niewielkie i przejrzyste, a pliki w różnych skorowidzach mogą otrzymywać takie same nazwy, nie doprowadzając przez to do sytuacji konfliktowych.

Dla każdej dziedziny prac można założyć odrębny skorowidz oraz nadać mu nazwę określającą zawartość. Ten, kto pracował na dużym komputerze z wieloma użytkownikami i musiał przestrzegać skomplikowanych reguł nadawania nazw, odczuje zalety struktury drzewiastej.

MS-DOS, który, podobnie jak CP/M, dzieli się na jądro systemu operacyjnego BIOS oraz interpreter poleceń, różni się jednak w dwóch istotnych punktach. Po pierwsze, BIOS MS-DOS można rozszerzyć. Polega to na tym, że przy dużym obciążeniu systemu do stałych generatorów można dołączyć dalsze (ładowane dynamicznie),



Hierarchiczne drzewo zbiorów:
do listu Jana dochodzi się korzystając kolejno ze ścieżek o nazwach:
TEKSTY, LISTY, PRYWATNE, LISTJANA

przez co MS-DOS staje się bardziej elastyczny. Natomiast włączanie do CP/M nowych generatorów zawsze wymagało generowania nowego systemu operacyjnego. Po drugie – interpreter poleceń (według wzorca Unix) nie został zintegrowany z systemem operacyjnym i ma postać normalnego programu aplikacyjnego. Daje to w efekcie znaczne uproszczenie przejścia na inny poziom obsługi operatorskiej, np. sterowanie za pomocą menu.

Podczas gdy przejście MS-DOS z wersji 1.xx do wersji 2.xx było dużym skokiem jakościowym, to wersja 3.xx (1984 r.) charakteryzuje się tylko niewielkimi ulepszeniami. Brak w niej tak istotnej innowacji, jaką było wprowadzenie w wersji 2.11 właściwości Unixa. Stąd niezbyt usprawiedliwiony jest nowy numer wersji. MS-DOS 3.xx zawiera m.in. dalsze polecenia i umożliwia wielokrotne użycie plików, i to nawet pochodzących z różnych komputerów sieci mikrokomputerowej.

Niestety często nie ma możliwości udowodnienia przewagi MS-DOS. Wiele programów aplikacyjnych „zachowuje się nieprawidłowo" i realizuje dostęp do sprzętu z pominięciem systemu operacyjnego, najczęściej ze względu na dążenie do zwiększenia szybkości przetwarzania. Programy te wykorzystują MS-DOS właściwie tylko jako system zarządzania plikami. Oczywiście nie jest to winą MS-DOS, jednak taka tendencja podważa jego rację bytu jako systemu operacyjnego. System taki, jak już wyjaśniano, powinien bowiem kryć szczegóły rozwiązań sprzętowych i w ten sposób przezwyciężać większość problemów kompatybilności sprzętu.

Takie programy użytkowe, jak Lotus 1-2-3 lub słynny symulator lotu Microsoft stwarzają wiele trudności. Po pierwsze – wymagają one sprzętu całkowicie kompatybilnego i stawiają producentów komputerów kompatybilnych z IBM wobec problemów, które powinny i mogą być rozwiązane właściwie przez omawiany system operacyjny. Po drugie – utrudniają dalszy rozwój tego systemu operacyjnego, np. w kierunku możliwości pracy wielozadaniowej lub techniki okien. W zastosowaniu wspomnianych programów nie można ponadto wykorzystać takich właściwości systemu operacyjnego, jak np. przeadresowywanie operacji wejścia i wyjścia.

micro

DODATKOWA PAMIĘĆ RAM DO AMSTRADA

Początkowy okres entuzjazmu po nabyciu komputera Amstrad CPC 464 mija dość szybko, gdy jego użytkownik uświadomi sobie dwa istotne ograniczenia tego modelu: stosunkowo małą pojemność pamięci RAM oraz powolną zewnętrzną pamięć taśmową. Wyjściem z tej sytuacji, oprócz zakupu nowszych modeli, są przystawki oferowane przez producenta albo niezależne firmy.

Nabycie stacji dysków elastycznych otwiera użytkownikowi tego komputera możliwości systemu operacyjnego CP/M niestety, podobnie jak w wypadku modelu CPC 664, tylko częściowo. Po prostu, można korzystać z programów o długości nie przekraczającej 39 KB, co zamyka dostęp do ponad 90% oferowanego na rynku oprogramowania działającego w systemie CP/M.

Niewiele lepiej wygląda sytuacja przy pracy z interpreterem języka BASIC. W tym wypadku, z 64 KB pamięci RAM dla użytkownika pozostaje tylko 42 KB.

Radykalnym rozwiązaniem tych problemów może być zakup specjalnej przystawki serii *SP*, produkowanej przez zachodnioniemiecką firmę Vortex Computer Systeme. W najbardziej rozbudowanej wersji rozszerza ona pamięć CPC 464 o 512 KB, udostępniając jednocześnie wiele nowych możliwości.

CHARAKTERYSTYKA SPRZĘTU

W odróżnieniu od innych rozszerzeń sprzętowych, które zwykle są dołączane na zewnątrz komputera, przystawka firmy Vortex jest kartą umieszczaną wewnątrz obudowy bezpośrednio nad główną płytką. Na karcie tej, oprócz 16 układów RAM po 256 KB, pamięci EPROM 16 KB oraz dwóch 40-stykowych podstawek, znajduje się także ponad 20 układów scalonych umożliwiających bezkonfliktową pracę całego systemu z innymi urządzeniami zewnętrznymi, przewidzianymi przez producenta.

Wszystkie elementy półprzewodnikowe są wykonane w technologii CMOS i pobierając mniej niż 400 mA prądu, nie stanowią poważniejszego obciążenia dla zasilacza komputera.

Zamontowanie przystawki jest przeróbką dość prostą, nie wymagającą żadnego lutowania. Po zdjęciu obudowy wyjmuje się z komputera mikroprocesor *Z80* oraz układ *ULA*. Następnie oba te układy wkłada się w podstawki na karcie, samą zaś kartę umieszcza na głównej płytce i łączy z nią dwoma 40-stykowymi wtykami wkładanymi w podstawki po wyjętych układach scalonych. Dodatkowe połączenie między płytkami stanowią dwa przewody sygnałów *RAMDIS* i *ROMDIS*.

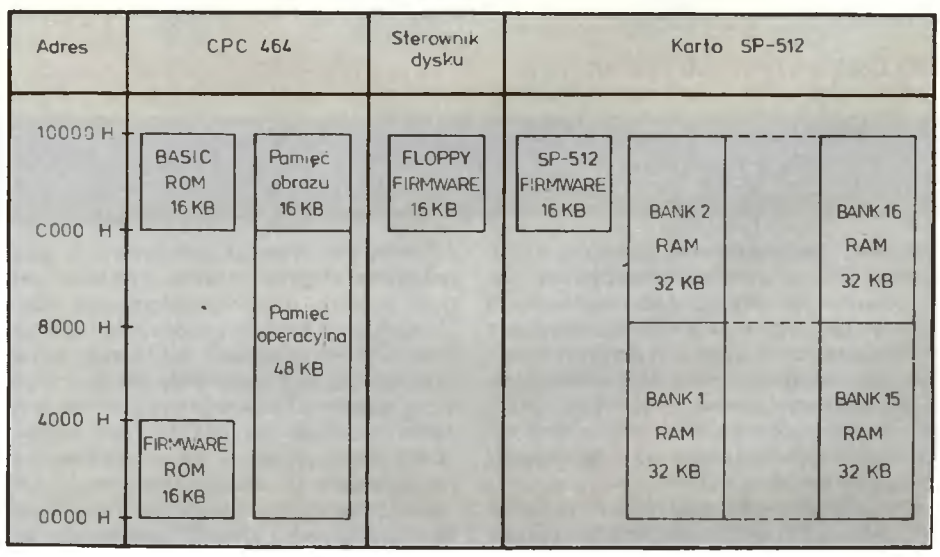
Po zamknięciu obudowy dysponujemy sprzętem nie różniącym się od pierwowzoru, ale o możliwościach znacznie przewyższających nowsze modele Amstrada: CPC 664 i CPC 6128.

ORGANIZACJA PAMIĘCI

Procesory 8- bitowe, a w szczególności *Z80*, mają 16-bitową szynę adresową i są w stanie bezpośrednio zaadresować tylko 64

dokończenie na str. 32

Mapa pamięci systemu



Poufność informacji

Dążenie do ograniczania kręgu odbiorców informacji jest tak stare, jak mowa. Oprócz względów etycznych i ekonomicznych największy wpływ na rozwój metod zabezpieczania informacji mają cele wojskowe. Zaostrzenie współzawodnictwa ekonomicznego sprawiło, że ostatnio coraz bardziej wzrosła rola aspektu gospodarczego. Przyczyną tego jest nie tylko zwiększenie liczby danych, ale również ich wartości.

Jednym z najstarszych historycznie przykładów szyfrowania jest tekst zapisany pismem klinowym, zawierający recepturę polewy garnków glinianych. Stosowano wówczas przede wszystkim tzw. metodę substytucyjną, polegającą na zastępowaniu każdego zapisywanego znaku innym znakiem, którego znaczenie było znane tylko piszącemu oraz odbiorcy wiadomości. Metoda ta zeszła na drugi plan z chwilą wykrycia faktu, że wskutek zróżnicowanej częstotliwości występowania liter w danym języku, zdeszyfrowanie tekstu jest stosunkowo łatwe. Zamiast niej zaczęto więc stosować metodę graficzną[1]).

Dążenie do zatajenia informacji jest ściśle związane z wysiłkami strony przeciwnej, aby złamać szyfr. Zawsze metody deszyfracji zdobywały przewagę nad metodami szyfrowania. Podczas drugiej wojny światowej do deszyfracji stosowano już bardzo zaawansowane metody oraz maszyny liczące, a przechwytywanie treści tajnych komunikatów było jednym z istotnych czynników uzyskania przewagi nad przeciwnikiem.

Włamywacze i kopiści

Obecnie, oprócz nie malejącego znaczenia wojskowego, wiele innych czynników wpływa na to, że systemy szyfrowania wzbudzają zainteresowanie również w sferze gospodarki. Z chwilą olbrzymiego wzrostu sprzedaży tanich komputerów domowych, producenci atrakcyjnych gier usiłowali przez monopol sprzedaży przechwycić całość kieszonkowego, jakim dysponowała olbrzymia rzesza nastolatków. Niestety młodzi hobbiści zręcznie obeszli zabezpieczenia firmowe programów gier, złamali szyfry i wkrótce rozpoczął się błyskawiczny rozwój wymiennego handlu bezpłatnie wykorzystywanym oprogramowaniem. Próby zrekompensowania strat poniesionych przez producentów gier (przy pomocy organów ścigania) i uzyskane na tej drodze sporadyczne sukcesy nie mogły jednak przysłonić ich totalnej klęski.

Znacznie lepiej zabezpieczyli swe interesy producenci oprogramowania komputerów osobistych. Wprawdzie istnieje np. edytor tekstu Wordstar w dalszym ciągu bez jakiejkolwiek ochrony przez kopiowaniem (a więc praktycznie bezpłatny), jednak programy są stopniowo coraz lepiej zabezpieczane i wyposażane w klauzulę zakazu kopiowania. Również tu „włamywacze" przystąpili do działań zmierzających do ominięcia zabezpieczeń. Z chwilą, gdy na rynku ukazuje się nowe zabezpieczenie, natychmiast pojawia się program, który łamie użyty szyfr. Nawet kropelka kwasu zmieniająca fizycznie dyskietkę, zaopatrując ją w rodzaj odcisku linii papilarnych palca, zostaje skopiowana i znajdzie się w skorowidzu nowej dyskietki jako logiczna kropelka kwasu. Zniszczony sektor zostaje przebadany i zarejstrowany na kopii programu. Jednak wydaje się, że wysiłek, jaki trzeba włożyć w to, aby obejść współczesne zabezpieczenia, jest jeszcze dostateczny duży, aby odstraszyć większość włamywaczy. Wiele blokad kopiowania pozostaje nadal nie przelamanych, i nawet najbardziej przebiegli złodzieje muszą złożyć broń.

Wstęp do ośrodka obliczeniowego

Wspaniałą rzeczą jest ośrodek obliczeniowy. Wystarczy ekran, telefon i modem – a nawet tylko tzw. sprzęg akustyczny – i już do niego wejdziesz. Do wielkiego ośrodka ze wszystkimi programami i danymi, którymi on dysponuje. Warunkiem jest oczywiście znajomość hasła, które mówi komputerowi o posiadaniu uprawnienia do korzystania z zasobów ośrodka. Znajomość hasła nie zawsze jest synonimem faktycznego uprawnienia i dlatego liczne ośrodki obliczeniowe na świecie, a zwłaszcza w USA, mają jeszcze dodatkowe zabezpieczenia. W tym względzie szczególnie uczulone są ośrodki naukowo-badawcze oraz banki, ponieważ chodzi tu o wykradzenie albo sfałszowanie danych. Taka sytuacja doprowadziła w ostatnich latach do opracowania szyfrów, które powinny zapewnić bezpieczne przechowywanie i wymianę danych.

System DES

W latach 1976–1977 IBM opracował system o nazwie DES (Data Encryption Standard), który nadal jest uznawany jako pewny sposób szyfrowania danych. Nawet jeżeli dysponuje się dowolnie długim tekstem oryginału i jego zaszyfrowaną wersją, wykrycie zasady szyfru jest niemożliwe. Sama metoda szyfrowania jest w tym wypadku powszechnie znana: przesyłany tekst dzieli się na bloki długości 64 bitów[2]). Każdy taki blok opracowuje się w dwóch 32-bitowych połówkach. Takie w zasadzie identyczne przekształcenie przeprowadza się szesnastokrotnie, na przemian w lewej i prawej połówce bloku. Opierając się na ustalonym algorytmie następuje logiczne powiązanie danych z 64-bitowym szyfrem, przy czym funkcja przekształcania zależy również od treści przeciwległych połówek bloku. Wynik można ponownie przekształcić w tekst czytelny za pomocą odwrócenia całej procedury, jednak pod warunkiem, że zna się wspomniany 64-bitowy szyfr. Metoda ta nie została jeszcze podważona, ale znawcy przedmiotu uważają, że złamanie szyfru nastąpi najpóźniej za cztery do pięciu lat.

Systemy typu public-key

Dalszą, nienaruszoną jeszcze odmianą zabezpieczania danych są tzw. szyfry nieodwracalne. Ich koncepcja zakłada, że każdy nadawca może zaszyfrować tekst za pomocą powszechnie znanego szyfru, natomiast rozszyfrować tylko ten, kto będzie dysponował

1) Patrz charakterystyki w ramkach.

2) Wyjaśnienie uproszczone

deszyfratorem. Na pierwszy rzut oka wydaje się to nieprawdopodobne, ale istnieje już wiele metod, spośród których dokładniej opiszemy jedną, uznawaną dotąd za „nie złamaną". System ten od nazwisk swych wynalazców: Rivesta, Shamira i Adlemanna otrzymał nazwę RSA.

System RSA

Dla zainteresowanych podajemy w ramkach dokładny algorytm RSA. Najważniejszym szczegółem w tym systemie jest możliwość odwrócenia zaszyfrowania, a tym samym deszyfracji tekstu. Tekst pierwotny można otrzymać również wtedy, jeśli zostanie on najpierw zdeszyfrowany, a dopiero potem zaszyfrowany.[3]

Ponieważ klucz deszyfrujący jest znany wyłącznie nadawcy, metoda ta jest idealna również do zastosowania w elektronicznych podpisach. Ta metoda jest teoretycznie również możliwa do złamania: obie liczby pierwsze [p] oraz [q][4]) można znaleźć za pomocą rozłożenia na czynniki liczby [a]. Oczywiście wymaga to dłuższego czasu, nawet przy zastosowaniu bardzo szybkiego komputera oraz współczesnych metod rozkładania dużych liczb na czynniki pierwsze. Przykładowo dla liczby 50-cyfrowej potrzeba tylko ok. 3 sekund, ale już dla liczb 75-cyfrowych aż 90 dni, a 100-cyfrowych – 60 lat. Ponieważ komputery są coraz szybsze, a wiele z nich można zartudnić do rozwiązania tego samego problemu w różnych przedziałach liczb, to istnieje możliwość złamania również metody RSA.

Użytkownik wybiera dwie bardzo duże liczby pierwsze [p oraz q] jakie mu może podać własny komputer (zaleca się długość 100 cyfr) i tworzy się z nich liczbę [a]. Liczba ta wspólnie z dalszą liczbą losową [s] zostaje ujawniona jako klucz szyfru, natomiast obie liczby pierwsze [p] oraz [q] pozostają jako tajne.

Nadawca przekształca swój tekst w ciągi liczb (np. według kodu ASCII) i koduje te liczby [L] zgodnie z wzorem [K = L^s modulo a]. Powstały w ten sposób kod [K] wskutek użycia funkcji modulo nie zwiększył długości w porównaniu do [L]. Odbiorca poznaje teraz dalszą liczbę [e], charakteryzującą się tym, że w porównaniu wyniku operacji ([p] −)* ([q] − 1) jest względną liczbą pierwszą, (obie te liczby nie mają największego wspólnego dzielnika). Natomiast [e] jest znana tylko odbiorcy, który może ją doprowadzić do pierwotnej postaci przez zastosowanie wzoru [L = K^e modulo a]. [s] otrzymano pierwotnie z [p], [q] oraz[e]. Jest to wielokrotna inwersja wyrazenia:

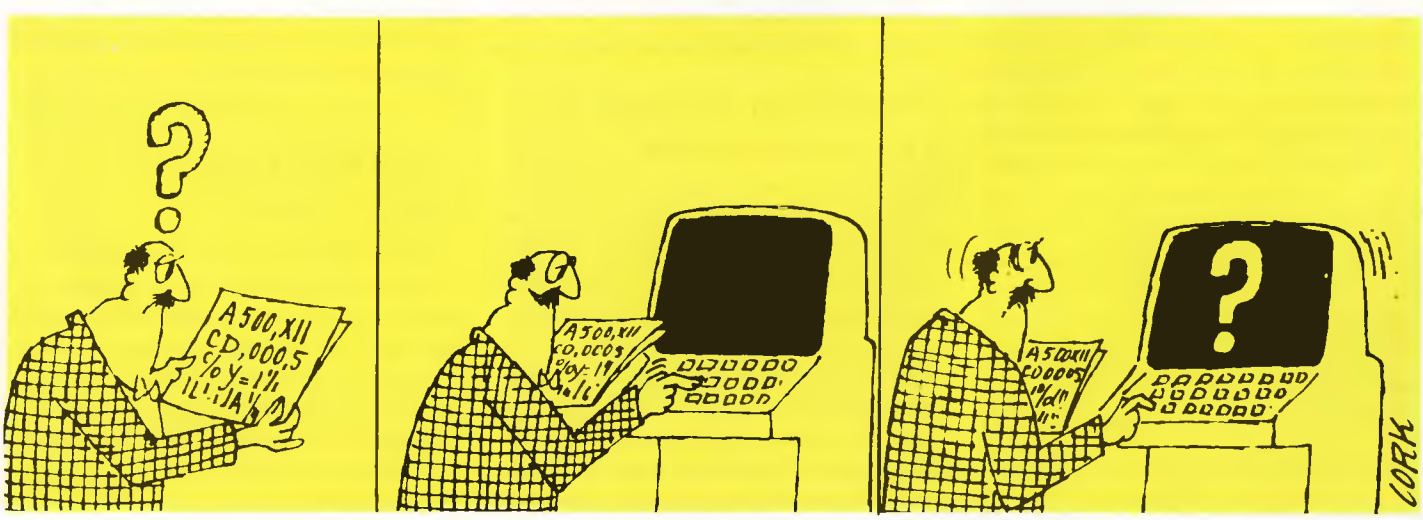
[e mod (([p] − 1) . ([q] −1))]

[3] Właściwość ta jest jedną z tych, które Diffie i Hellmann sformułowali następująco:
a) zaszyfrowanie i zdeszyfrowanie informacji Z daje ponownie informację Z:
V (E[Z]) = Z
b) funkcja powyższa musi być również odwracalna:
E (V[Z]) = Z
c) zarówno V jak i E należy stosować pojedynczo,
d) ujawnienie V nie pozwala na poznanie procedury deszyfrowania E. Jedynie posiadacz E może ten tekst ponownie przekształcić w Z:
V ≠ E⁻¹

[4] Patrz wyjaśnienie w ramkach.

Ochrona profesjonalna

Niebezpieczeństwo ujawnienia utajnionych informacji, a także nielegalnego użycia programów spowodowało podjęcie wysił-



Prosty system substytucyjny

Alfabet przekształca się np. za pomocą słowa szyfrującego:

ABCDEFGHIJKLMNOPQRSTUVWYZ
WINDROSEABCFGHIJKLMPQTUVXYZ

Dzięki znajomości tylko jednego słowa (WINDROSE) możliwe jest proste przetłumaczenie (odszyfrowanie) utajnionego tekstu.

Szyfrowanie cyfrograficzne

Jak wyżej, jednak nowy alfabet zostaje sprowadzony do macierzy 5x5 (I=J).

WINDR
OSEAB
CFGHK
LMPQT
UVXYZ

Tekst pierwotny dzieli się na pary liter i zostaje przetłumaczony parami. Należy przy tym stosować się do następujących trzech reguł:

1. Obie litery znajdują się w tym samym wierszu → , zawsze bierze się obie sąsiednie prawe (np IR=NW, BE=OA, LP=MQ).
2. Obie litery znajdują się w tej samej kolumnie → , zawsze bierze się literę położoną poniżej (np. SM=FV, HY=QD).
3. Jeżeli nie występuje 1. ani 2., to w macierzy tworzy się prostokąt, który jako dwa ze swych czterech punktów narożnych ma obie litery pierwotne. Litery tekstu zostają zastąpione przez dwa inne punkty narożne prostokąta, przy czym zostanie pobrany ten sam wiersz (np. IH=DF).

ków zmierzających do opracowania systemów szyfrowania, które w granicach obecnych ludzkich możliwości (oraz w rozsądnym wymiarze czasu) będą niemożliwe do złamania. Jednym z nich był wyżej wspomniany system RSA. Powstaje jednak coraz więcej przedsiębiorstw zajmujących się wyłącznie problemem ochrony danych. Należy tu w pierwszym rzędzie wymienić firmę Vault Corporation, z Kalifornii, która za pomocą przemyślnej techniki programowania oferuje swym klientom wszelkie rodzaje ochrony, poczynając od zabezpieczeń dyskietek (PROLOK) poprzez zabezpieczanie danych (FILELOK) aż do zabezpieczeń telekomunikacyjnych (TELELOK).

System szczególnie pewny w działaniu zrealizowano również w znanej austriackiej firmie VOEST-ALPINE.

Soft Seal

Rozwiązanie to opiera się na opracowanej przez firmę Honeywell Bull karcie CP-8. Na karcie tej, wykonanej z plastiku i przypominającej rozwiązania powszechnie używane do wprowadzania w szczeliny urządzeń automatycznych, znajduje się 8-bitowy komputer jednoukładowy (wymiary 3x4 mm, grubość poniżej 1 mm). Ochrona danych lub programów, czy też zapobieganie nielegalnemu dostępowi do komputera, jest dla tego systemu sprawą niezwykle prostą. Część programu i (lub) szyfr zostają zapamiętane w pamięci karty. Każdy program jest niekompletny, a więc niemożliwy do wykonania, brak mu bowiem części zawartej w karcie. Obejścia blokady dostępu, teoretycznie możliwe w innych systemach zabezpieczania, są tu oczywiście bezskuteczne. Za pomocą jednej karty można ochronić wiele programów. Z handlowego punktu widzenia słabością tego systemu jest konieczność użycia specjalnego czytnika karty, który powoduje odczuwalny wzrost jego ceny. Realizacja tego rozwiązania w postaci Eurokarty albo wtyku, dostosowanego do gniazd interfejsów szeregowych, byłaby z pewnością rozwiązaniem znacznie tańszym. Mimo tych zastrzeżeń wydaje się, że ten rodzaj zabezpieczenia jest obecnie najbardziej przyszłościowy, ponieważ pojedyncze urządzenie zapewnia ochronę wszystkich obszarów: programów, danych oraz samego hasła.

micro

BASIC dla początkujących

W poprzednich odcinkach naszego kursu omówiliśmy instrukcje, które wystarczają już do napisania prostych programów o coraz bardziej skomplikowanej strukturze, gdzie pomocne okaże się użycie operatorów logicznych. Najpierw jednak pomówimy o obliczeniach numerycznych.

Na wydruku 1 pokazano rozwiązanie ćwiczenia z poprzedniego odcinka. Wiersze 200 do 320 tworzą podprogram służący do omówionego już dokładnie przejrzystego sposobu formatowania wyświetlanych liczb. Zastąpiono jedynie *A$* przez *X$* oraz *A* przez *X*. Nieco zmienił się także program główny, składający się przede wszystkim z wywołań wspomnianego podprogramu do formatowania liczb.

Niektórzy z Was będą może zdziwieni, że nigdzie nie występuje znak przejścia do nowego wiersza. Otóż okazuje się to niepotrzebne! Na ekranie wyświetlanych jest bowiem pięć wartości liczbowych, z których każda zajmuje osiem pól znakowych i w ten sposób jest wypełniany cały wiersz ekranu (przyporządkowanie ośmiu pól każdej wyświetlonej liczbie następuje w wierszu 300). Na wydruku 2 pokazano wynik działania programu.

DATA I READ

Na wydruku 3 przedstawiono program, który akceptuje także liczby ujemne oraz wyświetla znak liczby nie przed nią, lecz za nią. Odpowiedni podprogram zajmuje wiersze 100 do 230. Wiersze 10 do 65 tworzą program główny. Zawiera on dwie nowe instrukcje.

Instrukcja **DATA** umożliwia wprowadzenie danych do programu. Dane te, oddzielono od siebie przecinkami, mogą być zarówno liczbami, jak i ciągami znaków. Instrukcja **READ** służy do odczytywania danych zawartych w instrukcji **DATA** i musi zawierać nazwy zmiennych, do których mają być przesłane.

Instrukcje **DATA** i **READ** pozwalają więc na sztywne wbudowanie w program odpowiednich danych i pobieranie ich w czasie jego wykonywania. Dane mogą być zgrupowane w jednym miejscu programu, co znacznie poprawia jego czytelność i ułatwia jego uruchamianie oraz testowanie.

Wróćmy do wydruku 3. Trzy pierwsze wiersze zawierają dane, które w pętli zajmującej wiersze 25 do 60 są pobierane, odpowiednio formatowane i w końcu wyświetlane na ekranie. W instrukcji **PRINT** w wierszu 35 istotny jest przecinek, który powoduje, że po wywołaniu podprogramu druga liczba w danym wierszu ekranu jest wyświetlana w polu zaczynającym się zawsze w tej samej kolumnie (na początku tego pola znajdują się spacje – wydruk 4). Aby program przetestować dla liczb ujemnych, w wierszu 45 pobrana wartość mnożona jest przez −1. Podprogram formatujący jest wywoływany w wierszu 40 dla liczb nieujemnych, a w wierszu 55 – dla liczb ujemnych.

Przyjrzyjmy się teraz, jak działa podprogram formatujący wyświetlane liczby (wiersze 100 do 230). Wiersz 100 jest nam już znany. W wierszu 110 z ciągów znaków jest pobierany i umieszczany w zmiennej *VZ$*

A	B	OBWÓD	POWIERZCHNIA	PRZEKĄTNA
1,00	1,00	4,00	1,00	1,41
1,00	2,00	6,00	2,00	2,24
1,00	3,00	8,00	3,00	3,16
1,00	4,00	10,00	4,00	4,12
1,00	5,00	12,00	5,00	5,10
1,00	6,00	14,00	6,00	6,08
1,00	7,00	16,00	7,00	7,07
1,00	8,00	18,00	8,00	8,06
1,00	9,00	20,00	9,00	9,06
1,00	10,00	22,00	10,00	10,05
1,00	11,00	24,00	11,00	11,05
1,00	12,00	26,00	12,00	12,04
1,00	13,00	28,00	13,00	13,04
1,00	14,00	30,00	14,00	14,04
1,00	15,00	32,00	15,00	15,03
1,00	16,00	34,00	16,00	16,03
1,00	17,00	36,00	17,00	17,03
1,00	18,00	38,00	18,00	18,03
1,00	19,00	40,00	19,00	19,03
1,00	20,00	42,00	20,00	20,02
2,00	1,00	6,00	2,00	2,24
2,00	2,00	8,00	4,00	2,83
2,00	3,00	10,00	6,00	3,61
2,00	4,00	12,00	8,00	4,47
2,00	5,00	14,00	10,00	5,39
2,00	6,00	16,00	12,00	6,32
2,00	7,00	18,00	14,00	7,28
2,00	8,00	20,00	16,00	8,25
2,00	9,00	22,00	18,00	9,22
2,00	10,00	24,00	20,00	10,20
2,00	11,00	26,00	22,00	11,18

Wydruk 2. Wynik działania programu z wydruku 1

znak pierwszy od lewej, czyli znak liczby (spacja dla liczb nieujemnych albo minus dla liczb ujemnych). W wierszu 120 zmienna *X$* jest pozbawiana tego znaku. Przetwarzanie liczb ujemnych i nieujemnych przebiega w identyczny sposób: oblicza się ich wartość

Wydruk 1. Rozwiązanie zadania z części V

```
100  PRINT"     A       B  OBWÓD  POW.   PRZEKĄTNA"
101  PRINT"-------------------------------------------"
105  FOR A=1 TO 20
110  FOR B=1 TO 20
115  U=2*A+2*B
120  FL=A*B
125  DI=SQR(A*A+B*B)
130  X =A:GOSUB 200
135  X =B:GOSUB 200
140  X =U:GOSUB 200
145  X =FL:GOSUB 200
150  X =DI:GOSUB 200
155  NEXT B
160  NEXT A
199  END
200  I =INT(100*X+.5)/100
210  I$=STR$(I)
220  L=LEN(I$)
230  IF L<3 GOTO 260
240  IF MID$(I$,L-2,1)="." THEN I$=LEFT$(I$,L-3)+","+RIGHT$(I$,2):GO
TO 270
250  IF MID$(I$,L-1,1)="." THEN I$=LEFT$(I$,L-2)+","+RIGHT$(I$,1)+"0
":GOTO 270
260  I$=I$+",00"
270  L=LEN(I$)
280  IF X>999.99 THEN I$=LEFT$(I$,L-6)+"."+RIGHT$(I$,6)
290  IF X>999999.99 THEN I$=LEFT$(I$,L-9)+"."+RIGHT$(I$,10)
300  I$=RIGHT$("          "+I$,8)
310  PRINT I$;
320  RETURN
```

Wydruk 3. Program formatujący liczby będące kwotami pieniężnymi

```
10 DATA 0,0.1,0.01,1,1.2,23,23.6,123.89
15 DATA 1234,1234.5,12345.67
20 DATA 123456.56,1234567.5,12345678
25 FOR I=1 TO 14
30 READ X
35 PRINTX,
40 GOSUB100
45 X=X*(-1)
50 PRINTX,
55 GOSUB100
60 NEXT
65 END
100 X$=STR$(X)
110 VZ$=LEFT$(X$,1)
120 X$=RIGHT$(X$,LEN(X$)-1)
130 Y=ABS(X)
140 IF Y<1 AND Y>0 THEN X$="0"+X$
150 IF Y-INT(Y) < 0.001 THEN X$=X$+".00" :GOTO170
160 IF Y*10-INT(Y*10) < 0.001 THEN X$=X$+"0"
170 X$=RIGHT$("              "+X$,14)
180 X$=LEFT$(X$,11)+","+RIGHT$(X$,2)
190 IF Y>999.99 THEN X$=LEFT$(X$,8)+","+RIGHT$(X$,6)
200 IF Y>999999.99 THEN X$=LEFT$(X$,5)+","+RIGHT$(X$,10)
210 X$=X$+VZ$
220 PRINTX$
230 RETURN
```

bezwzględną (funkcja *ABS(X)*,) po czym się ją odpowiednio przetwarza.

W wierszu 140 następuje sprawdzenie, czy wartość bezwzględna liczby zawiera się w przedziale (0, 1). Jeśli tak, przed przecinkiem dostawiane jest 0. Następnie program sprawdza, czy w liczbie wprowadzanej występuje część ułamkowa. Jeśli nie, to różnica pomiędzy liczbą i przyporządkowaną jej wartością funkcji INT jest zerem. W takim wypadku program sam dostawi kropkę dziesiętną i dwa zera. W wierszu 150 wystarczy jednak, żeby różnica była mniejsza od 0,001 (a nie równa 0). Wynika to z pewnych problemów pojawiających się przy obliczeniach numerycznych, które omówimy później.

W wierszu 160 następuje podobne sprawdzenie, ale dla wartości pomnożonych przez 10. Dzięki temu obejmowane są prrzypadki, gdy po kropce dziesiętnej w danych wejściowych była tylko jedna cyfra. Wówczas po kropce i tej cyfrze dodawane jest tylko jedno zero.

W następnym wierszu ciąg znaków ograniczany jest do 14. Na koniec następuje zastąpienie kropki dziesiętnej przez przecinek dziesiętny (wiersz 180), wprowadzenie kropek oddzielających liczbę tysięcy (wiersz 190) i milionów (wiersz 200) oraz dodanie na końcu liczby jej znaku (wiersz 210). W wierszu 220 liczba jest wyświetlana na ekranie.

0	0,00
0	0,00
.1	0,10
-.1	0,10-
.01	0,01
-.01	0,01-
1	1,00
-1	1,00-
1.2	1,20
-1.2	1,20-
23	23,00
-23	23,00-
23.6	23,60
-23.6	23,60-
123.89	123,89
-123.89	123,89-
1234	1.234,00
-1234	1.234,00-
1234.5	1.234,50
-1234.5	1.234,50-
12345.67	12.345,67
-12345.67	12.345,67-
123456.56	123.456,56
-123456.56	123.456,56-
1234567.5	1.234.567,50
-1234567.5	1.234 567,50-
12345678	12.345.678,00
-12345678	12.345.678,00-

Wydruk 4. Wynik działania programu z wydruku 3

Wydruk 6. Wynik działania programu z wydruku 5

PROBLEMY NUMERYCZNE

Komputer zawsze liczy z dokładnością do ustalonej liczby pozycji (np. 8 – 10 pozycji).

Liczne funkcje komputera – nawet te najprostsze – powstają w wyniku potęgowania. Obliczenia te przeprowadzane są z pewną ograniczoną dokładnością. Wskutek tego np. z liczb całkowitych mogą się zrobić liczby ułamkowe. Może to prowadzić do zupełnie odmiennego od zamierzonego działania programu, o czym przekona Cię króciutki przykład z wydruku 5. Na wydruku

```
100 FOR I=1 TO 50
110 PRINT I,I*I,I↑2
120 NEXT
```

Wydruk 5. Program testujący dokładność obliczeń

6 pokazano wynik jego działania. Program wyświetla obok siebie liczby, które powinny być sobie równe. Pierwsza powstaje bowiem z pomnożenia zmiennej strującej pętli przez samą siebie, a druga – przez podniesienie jej do kwadratu. Jak widać, obliczenia na dwa sposoby dla liczb 7,9,21,22 do 24, 31 nie dały tych samych wyników.

1	1	1
2	4	4
3	9	9
4	16	16
5	25	25
6	36	36
7	49	49.0000001
8	64	64
9	81	81.0000001
10	100	100
11	121	121
12	144	144
13	169	169
14	196	196
15	225	225
16	256	256
17	289	289
18	324	324
19	361	361
20	400	400
21	441	441.000001
22	484	484.000001
23	529	529
24	576	576.000001
25	625	625.000001
26	676	676.000001
27	729	729.000001
28	784	784.000001
29	841	841
30	900	900
31	961	961.000001
32	1024	1024
33	1089	1089
34	1156	1156
35	1225	1225
36	1296	1296
37	1369	1369
38	1444	1444
39	1521	1521
40	1600	1600
41	1681	1681
42	1764	1764
43	1849	1849
44	1936	1936
45	2025	2025
46	2116	2116
47	2209	2209
48	2304	2304
49	2401	2401
50	2500	2500

```
100 A=1620
110 B=INT(A)
120 C=A/10
130 D=INT(C)
140 E=A/100
150 F=INT(A/100)
160 PRINTA,B,A-B
170 PRINTC,D,C-D
180 PRINTE,F,E-F
200 PRINTINT(45.6*10),INT(456)
```

Wydruk 7. Program testujący dokładność obliczeń

```
1620          1620          0
162           162           0
16.2          16            .200000003
455           456
```

Wydruk 8. Wynik działania programu z wydruku 7

Kolejny program przykładowy i wynik jego działania przedstawiono na wydrukach 7 i 8. Liczba 1620 jest tu najpierw przekształcana na różne sposoby (funkcja INT, dzielenie przez 10 i przez 100). Następnie wyświetlane są dane i różnice pomiędzy nimi. W wierszu 200 takie same, zdawałoby się, dane wyjściowe powodują wyświetlenie różnych wartości. Wszystkie te przykłady pokazujemy po to, aby pokazać Ci, że jeśli komputer nie robi dokładnie tego, czego po nim oczekujesz, to błąd może wynikać właśnie z ograniczonej dokładności przy obliczeniach numerycznych.

OPERACJE LOGICZNE

W naszym drugim programie do formatowania liczb wprowadziliśmy operator AND. Oprócz AND („i") istnieją jeszcze operatory OR („lub") i NOT („nie"). Omówimy je teraz nieco dokładniej.

Najpierw zajmiemy się najmniejszą jednostką logiczną w komputerze, czyli bitem. Bit może przybierać dwie wartości: *0* albo *1* i może oznaczać przepływ prądu lub jego brak, napięcie niskie lub wysokie itp. Na tym prostym rozdzieleniu dwóch stanów opierają się najbardziej skomplikowane systemy programowe oraz sposoby organizacji danych. Weźmy dwie zmienne *A* i *B*, które mogą przybierać tylko wartości *0* albo *1*. Kiedy zapiszemy teraz operację logiczną AND w postaci:

C=A AND B

C będzie miało wartość *1* wtedy i tylko wtedy, gdy jednocześnie *A* i *B* będą równe *1*. Często wartości *0* i *1* nazywane są wartościami logicznymi. *0* to „fałsz", a *1* to „prawda".

Operację OR możemy zapisać w postaci:

D = A OR B

Operacja NOT wymaga tylko jednego argumentu i powoduje zmianę jego wartości na przeciwną. W wyniku operacji:

E = NOT A

jeśli *A* było równe *0*, to *E* będzie równe *1* i na odwrót.

Wartości logiczne mogą również powstawać w wyniku użycia operatorów porównania. Przedstawiono to w tabeli, zawierającej wszystkie możliwe wartości zmiennych logicznych *A* i *B* oraz wyniki operacji AND, OR, NOT, = (równość) i ¯ (nierówność).

Podstawową dziedziną zastosowań omówionych operatorów logicznych są instrukcje warunkowe IF ... THEN ... Zamiast $\bar{A}B$ można też napisać NOT (*A*=*B*). Operatory mają różne priorytety: AND ma wyższy priorytet niż OR. Dlatego dwa sposoby przeprowadzenia operacji:

A AND B OR C AND D
A AND (B OR C) AND D

prowadzą do różnych razultatów. W drugim przykładzie najpierw wykonana będzie operacja w nawiasach, czyli OR, zaś w pierwszym przykładzie – operacja o wyższym priorytecie, czyli AND.

AND służy do przeprowadzania logicznych operacji „i".

OR służy do przeprowadzania logicznych operacji „lub".

NOT służy do przeprowadzania logicznych operacji „nie".

Przy wykonywaniu kilku złożonych operacji logicznych najpierw wykonywane są operacje w nawiasach, następnie operacje NOT, potem AND i na końcu OR.

A	B	A AND B	A OR B	NOT A	A = B	A < > B
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	0	1
1	1	1	1	0	1	0

Tabela prawdy dla operacji logicznych i operacji porównania

W swoim komputerze możesz też działać operatorami logicznymi na innych liczbach niż 0 i 1. Trzeba w tym miejscu odwołać się do sposobu przedstawienia liczby w postaci dwójkowej (binarnej). Wynik działania operatorem logicznym na dwie dowolne liczby powstaje w ten sposób, że odpowiednia operacja logiczna przeprowadzana jest oddzielnie na każdej parze bitów o takich samych numerach kolejnych pobranych z obu liczb, a otrzymane w wyniku wartości (0 albo 1) stają się odpowiednimi bitami liczby wynikowej. W wypadku operacji NOT negowany jest każdy bit liczby. Przykłady operacji logicznych na liczbach ośmiobitowych pokazano na wydruku 9. Jednak jest to istotne jedynie dla tych czytelników, którzy pragną głębiej poznać istotę operacji logicznych.[1] Na koniec uwaga praktyczna: staraj się nie komplikować wyrażeń logicznych. Operacje logiczne powinny wiązać się z logiczną konstrukcją i przebiegiem programu. Spróbuj rozwiązać następujące zadanie: znaleźć najprostsze przedstawienie następujących wyrażeń logicznych:

[1] Fragment ten nie jest prawdziwy dla wszystkich dialektów BASICA.

```
                  128  64  32  16   8   4   2   1
                 ---------------------------------
                  2^7  2^6 2^5 2^4 2^3 2^2 2^1 2^0
                 ---------------------------------
A = 164   = 1     0    1   0   0   1   0   0
          = 128 + 32       +   4
B = 36    = 0     0    1   0   0   1   0   0
          =           32           +4

A AND B = 36     10100100
164 AND 36 = 36  00100100
                 00100100 = 36

A OR B = 164     10100100
164 OR 36 = 164  00100100
                 10100100 = 164

A = 164
B =  72 = 64 + 8 = 01001000
A AND B =        10100100
                 01001000
                 --------
                 00000000 = 0

A OR B =         10100100
                 01001000
                 --------
                 11101100 = 236 = 128 + 64
                                + 32 + 8 + 4
```

Wydruk 9. Przykład operacji logicznych na liczbach 8-bitowych

A AND (B OR C) AND C AND (A OR B)
A OR B OR (C AND B)
A AND C OR B AND C

Być może pomoże Ci w tym następująca tabela:

A B C ...
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

do której można wpisać wyniki.

micro

Powtórzenie wiadomości z części I – V

- Pętla programowa jest wykonywana przynajmniej jeden raz.
- Zawartość zmiennych można wyświetlać w trybie bezpośrednim za pomocą polecenia PRINT.
- Interpreter pozwala w dowolnej chwili zatrzymać program i wyświetlić zawartość zmiennych.
- Funkcja INT () może służyć do obliczenia części ułamkowej liczby dziesiętnej. Wynikiem jest liczba całkowita.
- Liczby całkowite nie są zaokrąglane.
- Wewnątrz nawiasów za nazwą funkcji INT może znajdować się wyrażenie arytmetyczne złożone z liczb i zmiennych oraz operatorów arytmetycznych.
- Jeśli argumentem funkcji INT () jest ujemna liczba dziesiętna z częścią ułamkową, to wynikiem będzie najbliższa liczba całkowita większa od niej (o mniejszej wartości bezwzględnej).
- Przy wejściu do podprogramu zmienne mają wartości nadane im w programie wywołującym.
- Podprogramy mogą być w sobie zagnieżdżone.
- Za pomocą funkcji LEN (zmienna) można ustalić liczbę znaków wchodzących w skład ciągu znaków przechowywanego w zmiennej.
- Znak „+" może służyć do łączenia ze sobą ciągów znaków.
- Wyrażenie LEFT$ (zmienna, długość) pozwala wydzielić lewą część ciągu znaków. Pierwszy parametr jest nazwą zmiennej tekstowej, zaś drugi mówi, ile znaków licząc od początku należy wydzielić.
- Wyrażenie RIGHT$ (zmienna, długość) pozwala wydzielić określoną liczbę znaków licząc od prawej strony.
- Wyrażenie MID$ (zmienna, numer znaku, długość) pozwala wydzielić ze zmiennej tekstowej określoną liczbę znaków licząc od określonego miejsca.
- Zamiast zmiennych tekstowych w funkcjach LEFT$, RIGHT$ i MID$ mogą występować złożone wyrażenia tekstowe, np. połączenia ciągów znaków.
- W funkcjach LEFT$, RIGHT$ i MID$ parametry liczbowe (ostatni parametr w LEFT$ i RIGHT$ oraz dwa ostatnie parametry w MID$) mogą być także zmiennymi liczbowymi lub innymi wyrażeniami liczbowymi.
- Funkcja VAL (zmienna tekstowa) służy do ustalenia wartości liczbowej symbolizowanej przez tekst.
- Za pomocą funkcji STR$ (zmienna liczbowa) można przekształcić liczbę w odpowiadający jej ciąg znaków. Uwzględniony będzie przy tym znak liczby.

Uczmy się Pascala!

W trzeciej części kursu języka Pascal omawiamy możliwości sterowania przebiegiem programu.

W pierwszych dwóch odcinkach wprowadziliśmy pojęcie typu danych w Pascalu oraz omówiliśmy liczby rzeczywiste: *REAL*, całkowite: *INTEGER* oraz typ znakowy: *CHAR*. Przykładowe programy miały jedną wspólną cechę. Instrukcje były wykonywane kolejno, niezależnie od wartości wprowadzanych danych. Jeżeli chcieliśmy wykonać jakąś czynność wielokrotnie, to nie pozostawało nam nic innego, jak wypisać ciąg instrukcji odpowiednią liczbę razy. Postępowanie takie nie jest jednak praktyczne, gdyż liczba powtórzeń w ogólnym wypadku może być znaczna, a co bardziej istotne, często pożądane jest, aby była ona w programie zmienna.

Instrukcje warunkowe

Do sterowania przebiegiem programu stosuje się tzw. *instrukcje sterujące*. Wykorzystują one zmienne logiczne typu BOOLEAN. Tak jak zmienne pozostałych typów, zmenne typu BOOLEAN deklarujemy w segmencie rozpoczynającym się słowem *VAR*. Jak pamiętamy, istotną cechą zmiennej jest zakres przyjmowanych wartości oraz dopuszczalne dla danego typu operacje. Zmienne BOOLEAN mogą przyjmować tylko dwie wartości: prawdy – *TRUE* oraz fałszu – *FALSE*.

Zmienne logiczne wykorzystujemy do budowania warunków, które realizuje się poprzez porównanie wartości zmiennej, nie znanej w chwili pisania programu, z pewną wartością kontrolną. Wynik takiego porównania jest właśnie typu BOOOLEAN. Warunek spełniony ma wartość logiczną *TRUE*, warunek niespełniony przybiera wartość *FALSE*.

Załóżmy przez moment, że pewien fikcyjny podatek naliczany jest według następujących zasad: jeżeli suma do opodatkowania jest niższa niż sto złotych, to podatku nie nalicza się, jeżeli przekracza tysiąc – to podatek wynosi dwadzieścia, a w pozostałych wypadkach – dziesięć procent. Odpowiedni program przedstawiono na wydruku 1.

W programie tym wykorzystaliśmy instrukcję warunkową. Jej działanie polega na tym, że sprawdzany jest warunek występujący po słowie *IF*. Instrukcja następująca po słowie *THEN* jest wykonywana tylko w wypadku spełnienia tego warunku. Z reguły korzystamy z faktu, że wynik porównania jest typu BOOLEAN, w związku z czym zamiast deklarować zmienne logiczne w sposób jawny, wstawiamy wyrazenia logiczne bezpośrednio do instrukcji warunkowej (wydruk 2).

Warunki dla operacji logicznych

Dla wszystkich przykładów obowiązuje następująca konwersja zmiennych: **VAR ch : CHAR; it : INTEGER;**

oraz przyporządkowanie:

ch := 'A';
it := 10;

Za pomocą TRUE określa się stan, w którym warunek jest spełniony, natomiast FALSE – ten, w którym warunek nie jest spełniony. W wypadku typu danych CHAR kolejność odpowiada właściwej wartości kodu ASCII (patrz druga część kursu).

Operator	Opis	Przykład	Efekt
=	TRUE, jeżeli lewy i prawy argument są równe	ch = 'X' it = 10	FALSE TRUE
< >	TRUE, jeżeli lewy i prawy argument nie są równe	ch 'X' it 10	TRUE FALSE
> =	TRUE, jeżeli lewy argument jest większy (równy prawemu)	ch > = 'X' it > = 10	FALSE TRUE
< =	TRUE, jeżeli lewy argument jest mniejszy (równy) prawemu	ch < = 'X' it < = 10	TRUE TRUE
>	TRUE, jeżeli lewy argument jest większy od prawego	ch > 'X' it > 10	FALSE FALSE
<	TRUE, jeżeli lewy argument jest mniejszy od prawego	ch < 'X' it < 10	TRUE FALSE
IN	TRUE, jeżeli lewy argument jest zawarty w zbiorze	ch **IN** ['a'...'x'] it **IN** [0,5...15]	FALSE TRUE
NOT	TRUE, jeżeli argument jest FALSE	**NOT** (ch='X') **NOT** (it=10)	TRUE FALSE
AND	TRUE, tylko jeżeli prawy i lewy argument jednocześnie są TRUE	(ch='X') **AND** (it=10) (ch='A') **AND** (it=10)	FALSE TRUE
OR	TRUE, jeżeli jeden z argumentów lub oba są TRUE	(ch='X') **OR** (it=10) (ch='A') **OR** (it=10)	TRUE TRUE

Dodatkowo mozna stosować jako warunki zmienną typu BOOLEAN.

Wydruk 1

```
PROGRAM wymiar1;
VAR i:INTEGER;
    a,b,c:BOOLEAN;
BEGIN
WRITE('podaj podstawe opodatkowenia: ');
READLN(i);
a := i < 100;
b := i > 1000;
c := (i >= 100) and (i <= 1000);
IF a THEN WRITELN('suma zwolniona od podatku');
IF b THEN WRITELN('podatek obliczony wg stopy 20% wynosi: ',0.2*i);
IF c THEN WRITELN('obliczony wg stopy 10% wynosi: ',0.1*i);
END.
```

Wydruk 2

```
PROGRAM wymiar1 krotszy;
VAR i:integer;
BEGIN
WRITE('podaj podstawe opodatkowania: ');
READLN(i);
IF i < 100 THEN
   WRITELN('suma zwolniona od podatku');
IF i > 1000 THEN
   WRITELN('podatek obliczony wg stopy 20% wynosi: ',0.2*i);
IF (i >= 100) AND (i <= 1000) THEN
   WRITELN('podatek obliczony wg stopy 10% wynosi: ',0.1*i);
END.
```

Do działań na zmiennych typu BOOLEAN stosuje się znane ze szkoły średniej operatory logiczne koniunkcji, alternatywy oraz negacji. W Pascalu nazywają się one odpowiednio *AND, OR* oraz *NOT*. Wprawdzie po chwili zastanowienia każdy mógłby sam przypomnieć sobie poniższe tabelki, jednak są one na tyle przydatne i często stosowane, że warto je przytoczyć:

AND	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

DR	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

Reasumując: koniunkcja daje wartość prawdy tylko wtedy, gdy oba argumenty są prawdziwe, zaś alternatywa daje wartość fałszu tylko dla obu fałszywych argumentów. Negacja prawdy daje fałsz, zaś negacja fałszu – prawdę.
W tabeli zamieszczono różne przykłady operacji logicznych i wykorzystano nie znany nam dotychczas typ danych *SET*.

SET oznacza po prostu zbiór wartości. Zbiór można podać przez określenie ciągu nie jego elementów. Zapis:

[1..100]

oznacza zbiór wszystkich liczb całkowitych od jednego do stu. Dwie kropki pomiędzy liczbami zastępują wszystkie pośrednie wartości. Zbiór można też określić przez wyliczenie jego elementów. Zapis:

[1, 2, 3, 5, 7, 11]

oznacza zbiór pierwszych sześciu liczb pierwszych. Dopuszczalne jest łączenie obydwu rodzajów zapisu, np.:

[1, 3, 50..100].

Do zbiorów wrócimy jeszcze w dalszym ciągu kursu. Pokażemy wtedy, jak realizuje się znane najmłodszym czytelnikom ze szkoły podstawowej (a najstarszym ze studiów wyższych) pojęcie zbioru pustego, a także operacje sumy, przecięcia i różnicy zbiorów. Okaże się również, że elementami zbioru mogą być nie tylko liczby całkowite, ale także litery, a nawet jabłka i ołówki. Jednak będzie to wymagało wprowadzenia pojęcia typu danych definiowanego przez użytkownika typu danych. Na razie powiemy tylko o operacji *IN*, sprawdzenia przynależności do zbioru. Operacja ta daje w wyniku wartość logiczną. Możemy więc jej użyć do zbudowania instrukcji warunkowej. Zamiast pisać:

IF (i = 1 OR i = 2 OR i = 3 OR i = 4 OR i = 5) THEN WRITE ('wartość i wynosi 1 lub 2 lub 3 lub 4 lub 5');

piszemy po prostu:

IF i IN [1..5] THEN
WRITE ('i jest jedną z pierwszych pięciu liczb naturalnych');

Taki zapis ma dwie zalety. Przede wszystkim unikamy długich ciągów alternatyw, a wyniki badań statystycznych prowadzonych wśród programistów mówią, że największa liczba błędów jest popełniana właśnie w złożonych wyrażeniach logicznych. Dodatkową zaletą jest uzyskanie bardziej spójnego i jaśniejszego pojęciowo programu.

Dobre zwyczaje w programowaniu polegają między innymi na takim wykorzystywaniu bogactwa języka, aby uczynić zapis jak najbardziej przejrzysty i oszczędny. Wróćmy więc jeszcze na chwilę do naszego programu „wymiar". Przypomnijmy sobie, że trzecia część naszego fikcyjnego przepisu podatkowego rozpoczynała się od słów *‚w przeciwnym razie'*. Program moglibyśmy napisać więc tak:

podaj i
jeżeli i < 100 to podatek: = zero
w przeciwnym razie, jeżeli i > 1000 to podatek: = 20%
w przeciwnym razie podatek: = 10%

W Pascalu zamiast *‚w przeciwnym razie'* piszemy krótko: *ELSE*, a nasz program przybiera postać przedstawioną na wydruku 3.

Zwróćmy uwagę, że przed słowem *ELSE* nie piszemy średnika, ponadto skorzystaliśmy z możliwości opuszczenia średnika przed słowem *END*.

Przypominając sobie wiadomości o operatorze *DIV* z poprzedniego odcinka, możemy napisać program „wymiar" z wykorzystaniem badania przynależności do zbioru (wydruk 4).

Instrukcja *IF...THEN...* wybiera spośród dwóch możliwości. Konieczność wyboru spośród większej liczby wariantów prowadzić może do stosowania całego szeregu kolejnych instrukcji warunkowych, co wymaga niezwykle starannego sprawdzania programu i zawsze prowadzi do zawiłego kodu.

Wybór spośród skończonej liczby alternatyw można czytelniej zrealizować za pomocą instrukcji *CASE*. Składnia instrukcji *CASE* jest zbliżona do *IF... THEN...* z ostatniego przykładu. Dla jej lepszego zrozumienia przepisaliśmy program „wymiar" w jeszcze inny sposób (wydruk 5).

Zwróćmy uwagę na kończące instrukcję CASE słowo *END;* a przede wszystkim na sposób przyporządkowania poszczególnych instrukcji programu odpowiednim podzbiorom wartości parametru sterującego. Sposób wyliczania elementów zbioru jest analogiczny do omówionego wyżej dla typu *SET*, pomijamy jedynie nawiasy kwadratowe i stawiamy dwukropek. Problem z instrukcją *CASE* polega na tym, że poza TURBO Pascalem oraz Pascalem MT+ implementacje z reguły nie dopuszczają wariantu *ELSE*. Czytelnik powinien sprawdzić za pomocą swojego kompilatora, czy program z powyższej wersji nie będzie prowadził do błędu. Jeżeli tak, to znaczy, że należy posłużyć się efektywnym zapisem lub zamiast ELSE napisać np. ‚10..100:' co da ten sam rezultat kosztem ograniczenia zakresu wartości zmiennej „i" do mniejszych od 10000.

W powyższych rozważaniach sterowanie rozciągnęło się na jedną instrukcję, bezpośrednio następującą po instrukcji warunkowej. Chcemy często, aby sterowanie rozciągało się na całe ciągi, czyli bloki instrukcji. Struktura blokowa umożliwia projektowanie programu, poczynając od jego struktury logicznej i stopniowe przechodzenie do szczegółów, często z wykorzystaniem struktury blokowej na niższym poziomie. W ten sposób możemy programować praktycznie bez skoków. Ci Czytelnicy, którzy zetknęli się

Wydruk 3

```
PROGRAM wymiar2;
VAR i:integer;
BEGIN
WRITE('podaj podstawe opodatkowania: ');
READLN(i);
IF i < 100 THEN
   WRITELN('suma zwolniona od podatku')
ELSE IF i > 1000 THEN
   WRITELN('podatek obliczony wg stopy 20% wynosi:',0.2*i)
ELSE
   WRITELN('podatek obliczony wg stopy 10% wynosi:',0.1*i)
END.
```

Wydruk 4

```
PROGRAM wymiar3;
VAR i :integer;
BEGIN
WRITE('podaj podstawe opodatkowania ');
READLN(i);
j := i DIV 100;
IF j = 0 THEN
   WRITELN('suma zwolniona od podatku')
ELSE IF j IN [ 1 .. 9 ] THEN
   WRITELN('podatek obliczony wg stopy 20% wynosi: ',0.2*i)
ELSE
   WRITELN('podatek obliczony wg stopy 10% wynosi: ',0.1*i);
END.
```

Wydruk 5

```
PROGRAM wymiar4;
VAR
i,j:INTEGER;
BEGIN
WRITE('podaj podstawe opodatkowania...');
READLN(i);
j := i DIV 100;
CASE j OF
  0:
    WRITELN('suma zwolniona od podatku');
  1 .. 9:
    WRITELN('podatek obliczony wg stopy 20% wynosi:',0.2*i);
  ELSE
    WRITELN('podatek obliczony wg stopy 10% wynosi:',0.1*i)
END;
END.
```

Wydruk 6

```
PROGRAM wymiar5;
VAR i,j:INTEGER;
BEGIN
WRITE('podaj podstawe opodatkowania:');
READLN(i);
IF i < 0 THEN WRITELN('blad - program zatrzymany')
  ELSE
  BEGIN
  j := i DIV 100;
  CASE j OF
    0:
     WRITELN('suma zwolniona od podatku');
    1 .. 9:
     WRITELN('podatek obliczony wg stopy 20% wynosi:',0.2*i);
    ELSE
     WRITELN('podatek obliczony wg stopy 10% wynosi:',0.1*i)
  END;
  END;
END.
```

z BASIC-em będą mieli prawdopodobnie początkowo trudności z unikaniem instrukcji skoku *GOTO*, która wprawdzie w Pascalu istnieje, jednak jest niezwykle rzadko stosowana. Blok rozpoczyna słowo *BEGIN* a kończy *END*. Tak więc dowiadujemy się, że wszystkie nasze dotychczasowe programy posiadały strukturę blokową, lecz zawierały tylko jeden blok główny, którego koniec oznacza *END*. (z kropką).

Załóżmy, że chcemy zmodyfikować program „wymiar" tak, by wykazywał błąd, jeżeli wczytamy omyłkowo wartość ujemną. Część czynną programu opiszemy właśnie jako blok i zezwolimy na jej wykonanie tylko pod warunkiem, że użytkownik wczyta poprawnie podstawę opodatkowania (wydruk 6).

Z punktu widzenia instrukcji *IF...THEN...* cały blok programu od drugiego słowa *BEGIN* do przedostatniego *END* stanowi jedną instrukcję i jest pomijany w wypadku wczytania ujemnej danej.

Instrukcje iteracyjne

Dotychczasowe wyjaśnienia dotyczyły metod sekwencyjnej realizacji programu, polegających na sukcesywnym wykonywaniu instrukcji. Niezwykle ważną grupę technik wykorzystuje możliwość budowania tzw. pętli do wielokrotnego wykonywania bloków programu.

Pascal przewiduje trzy rodzaje pętli: *REPEAT, WHILE,* oraz *FOR*. Gdybyśmy w naszym programie chcieli zmusić użytkownika do wczytywania wartości *i* tak długo aż nie poda wartości dodatniej, to możemy wykorzystać instrukcję *REPEAT...UNTIL. REPEAT* znaczy dosłownie: *powtarzaj,* zaś UNTIL: *do momentu gdy...* . Komputer będzie powtarzał ciąg instrukcji zawarty pomiędzy słowem *REPEAT* a *UNTIL,* aż do momentu spełnienia warunku wypisanego po słowie *UNTIL,* w naszym wypadku do wczytania dodatniego *i*.

```
REPEAT
        WRITE('podaj liczbę dodatnią');
        READLN (i)
UNTIL i > 0;
```

Zauważmy, że instrukcja:

```
REPEAT
        WRITELN ('zapętliłem się');
UNTIL FALSE;
```

spowoduje wypisanie napisu na ekranie. Taka sytuacja jest najczęściej niezamierzona przez programistę. Przy pisaniu instrukcji iteracyjnych należy więc pamiętać o niebezpieczeństwie przypadkowego „zapętlenia się" programu.

Wydruk 7

```
WRITE('podaj liczbe dodatnia');
READLN(i)
WHILE i < 0 DO
     BEGIN
          WRITELN('miałaś podac liczbe dodatnia
          WRITE('sprobuj jeszcze raz');
          READLN(i)
     END;
```

Drugi sposób iterowania polega na wykorzystaniu instrukcji *WHILE... DO... . WHILE* oznacza: *tak długo jak...* zaś *DO: wykonuj instrukcję*. Różnica pomiędzy *REPEAT* oraz *WHILE* polega na tym, że w *REPEAT* warunek sprawdzany jest na końcu, co oznacza, że pętla zostanie zawsze wykonana co najmniej raz, również gdy warunek jest spełniony. W pętli *WHILE,* jeżeli warunek zakończenia jest spełniony, to pętla nie jest wykonywana ani razu (wydruk 7).

Jeżeli pomimo kolejnych upomnień konsekwentnie będziemy wprowadzać liczby ujemne to tak napisany program nigdy się nie skończy. Trzeci rodzaj pętli umożliwia wykonanie ciągu instrukcji określoną liczbę razy.

Pętla *FOR* składa się z części sterującej i z bloku do wykonania. Parametrami *FOR* są trzy liczby: licznik, jego początkowa i końcowa wartość.

FOR licznik: = początek TO koniec DO... .

Wykonanie *FOR* rozpoczyna się nadaniem zmiennej *licznik* wartości *początek*. Następnie sprawdza się, czy *licznik* nie jest większy niż *koniec*. Jeżeli nie, to wykonany jest blok instrukcji, licznik zwiększa się o jeden i operację powtarza. Gdy wartość *licznik* przekroczy *koniec,* wykonywana jest kolejna instrukcja programu.

Wydruk 8

```
PROGRAM test petli;
VAR licznik,poczatek,koniec:INTEGER;
BEGIN
WRITE('podaj dwie liczby: poczatek, koniec ...');
READLN(poczatek,koniec);
FOR licznik := poczatek TO koniec DO
     BEGIN
          WRITELN('licznik=  ',licznik);
          WRITELN('poczatek= ',poczatek);
          WRITELN('koniec=   ',koniec);
          WRITELN
     END
END.
```

Wydruk 9

```
PROGRAM plus;
VAR i, licznik :INTEGER;
BEGIN
     WRITE('podaj liczbe dodatnia...');
     READLN(i);
          FOR licznik := 1 TO 3 DO
            IF i <= 0 THEN
               BEGIN
                    WRITELN('miales podac liczbe dodatnia');
                    WRITE('sprobuj jeszcze raz...');
                    READLN(i);
                    WRITELN
               END;
     IF i > 0 THEN
               WRITELN('gratulacje, ',i,' jest liczba dodatnia')
          ELSE
               WRITELN('szkoda, nie rozumiesz')
END.
```

Najprościej to sprawdzić śledząc przebieg programu na wydruku 8.

Zmienna *licznik* jest osiągalna wewnątrz pętli i można ją wykorzystywać, nie należy jej jednak zmieniać, gdyż może to spowodować trudne do przewidzenia skutki. Instrukcja *WRITELN* bez argumentów służy do wyprowadzania pustego wiersza celem uzyskania bardziej przejrzystego wydruku.

Pętla *FOR... TO...* powoduje inkrementację, czyli zwiększanie licznika. Jeśli chcemy uzyskać kolejne przebiegi ze zmniejszającym się licznikiem, tj. z dekrementacją, to zmieniamy *FOR... TO...* na *FOR... DOWNTO...* . Zapis:

FOR licznik := 3 DOWNTO 1 DO...

byłby w poniższym programie *plus* równoważny z:

FOR licznik := 1 TO 3 DO... .

Program *plus* zgodnie z zasadą „do trzech razy sztuka" daje użytkownikowi trzykrotnie szansę wprowadzenia liczby dodatniej (wydruk 9).

Oprac. PIOTR BREITKOPF

micro

DODATKOWA PAMIĘĆ RAM DO AMSTRADA

dokończenie ze str. 22

KB pamięci. Obsługa większego obszaru wymaga stosowania specjalnych technik sprzętowo-programowych, polegających na stronicowaniu pamięci. Rozwiązania tego rodzaju można spotkać w wielu wcześniejszych komputerach, ale w wypadku CPC 464 sytuacja jest bardziej złożona. Uwzględniając sterownik dysku i kartę *SP-512* mamy do czynienia z systemem, który zawiera 64 KB ROM i 576 KB RAM. Pamięć ROM jest podzielona na 4 strony po 16 KB, a dodatkowo 512 KB RAM zawiera 16 stron po 32 KB.

Nowe lub zmodyfikowane instrukcje interpretera języka BASIC dostępne dzięki karcie SP-512

BANK	BASIC	BOS	CALL	COMMON
DEV	FAST	FRAME	GOSUB	GRAPER
GPEN	ID	LIST	LOAD	MASK
MON	NEW	PEEK	POKE	RAMCLOSE
RAMFIELD	RAMOPEN	RAMREAD	RAMWRITE	RECORDS
RETURN	RUN	SAVE	SCREEN.IN	SCREEN.OUT
SCREENS	SLOW	SPOOL.ON	SPOOL.OFF	UNMASK
VIDEO.ON	VIDE.OFF			

Przełączanie stron, dostępne programowo, odbywa się na najniższym poziomie za pomocą rozkazów *OUT* mikroprocesora *Z80*. Mapa pamięci całego systemu jest przedstawiona na rysunku.

BASIC

Dla użytkownika posługującego się językiem BASIC, Amstrad CPC 464 wyposażony w kartę *SP-512* jest systemem umożliwiającym pisanie programów o maksymalnej długości 288 KB oraz dysponowanie dodatkowym obszarem danych o pojemności do 256 KB. Poszczególne fragmenty programu mieszczą się w 9 blokach BANK, dostępnych za pomocą instrukcji *BANK* z odpowiednim numerem. Komunikacja między blokami odbywa się przy użyciu następujących zmodyfikowanych instrukcji interpretera BASICA:

|GOTO, BANK, nr linii
|GOSUB, BANK, nr linii
|RETURN

Zmienne wewnątrz bloku są lokalne, a przekazywanie ich do innych bloków jest realizowane za pomocą instrukcji | COMMON. Niestety nie dotyczy ona macierzy, ale i w tym wypadku nie jesteśmy bezradni. Można umieścić macierz we wspólnym obszarze danych i wtedy będzie ona dostępna w każdym bloku. Obszar danych może być także wykorzystany do przechowywania 16 pełnych obrazów, które mogą pojawiać się na ekranie co 1/3 sekundy.

Dodatkową możliwość stwarza tzw. *SPOOLER*, czyli dodatkowy bufor drukarki, który jest uaktywniany instrukcją *SPOOL ON*. Drukowanie tekstu nie przerywa wtedy działania innego programu.

Należy podkreślić, że instrukcje operujące blokami pamięci są uruchamiane dopiero po użyciu instrukcji | BOS. Natomiast przed użyciem tej instrukcji komputer zachowuje pełną zgodność z oprogramowaniem działającym w wersji nierozszerzonej.

W tabeli zestawiono 37 nowych instrukcji interpretera BASICA, które umożliwiają operowanie blokami pamięci, zwiększają możliwości graficzne, obsługują *SPOOLER* drukarki oraz udostępniają bardzo wygodny monitor, podobny do działającego w systemie CP/M programu *DDTZ*.

CP/M

Sama karta *SP-512* nie pozwala na pracę w systemie CP/M, ale po dołączeniu stacji dyskietek, znacznie zwiększa jego możliwości.

W tak rozszerzonym zestawie uzyskuje się obszar 58 KB pamięci przeznaczony do wykorzystywania na własne programy w systemie CP/M (ang. TPA – Transient Program Area). Za pomocą programów *MOVCPM* i *SYSGEN* możliwe jest zwiększenie tego obszaru do 62 KB.

Bardzo użyteczne są dwa dalsze programy: *SPOOLL* oraz *RAMDISC*. Pierwszy z nich uruchamia bufor drukarki (32 KB), natomiast drugi – formatuje pozostałą część pamięci, tworząc w ten sposób wirtualny dysk *C* o pojemności 448 KB. Dysk ten, odznaczający się wieloma zaletami, może być używany zamiast drugiej fizycznej stacji dyskietek.

***

Reasumując, przystawka *SP-512* jest bardzo udaną konstrukcją i stanowi doskonałe uzupełnienie nabytego wcześniej sprzętu. Pozwala uzyskać system przewyższający możliwościami dotychczas oferowane na rynku modele Amstrada. Pewną wadą może być jednak jej wysoka cena, bliska cenie samego komputera.

JAROSŁAW MŁODZKI

MIĘDZY LONDYNEM A WARSZAWĄ

Dyrektora znanej już dobrze na naszym rynku firmy wysyłkowej POLANGLIA Ltd, pana Andrzeja Łukomskiego, spotkaliśmy na warszawskim Okęciu, na chwilę przed powrotem do Londynu. W imieniu MIKROKLANU rozmawiał Jerzy Orkiszewski.

– Panie Dyrektorze, śledzi Pan uważnie brytyjski rynek komputerowy, w szczególności miejsce na tym rynku Amstrada – firmy, z którą związane jest Pańskie przedsiębiorstwo. Proszę powiedzieć nam, z jakim przyjęciem spotkał się w Wielkiej Brytanii nowy model Amstrada, PC 1512?

– Nowy Amstrad stał się prawdziwą sensacją. Anglicy w pełni docenili zalety tego komputera. Nie bez wpływu pozostawała doskonała marka, jaką wyrobił sobie Amstrad dzięki poprzednim modelom. Dość powiedzieć, że tylko w ciągu września zeszłego roku, czyli w ciągu 30 dni po prezentacji PC 1512, przedpłaty, jakie napłynęły do Amstrada, osiągnęły zawrotną kwotę 100 milionów funtów.

– Czy Amstrad był przygotowany na taki popyt?

– Amstrad jest firmą niezwykle dynamiczną i elastyczną, dla której duży popyt nie stanowi problemu. Wszystkie przedpłaty zostały już*) zrealizowane.

– Jaki udział w sukcesach zawdzięcza Amstrad swoim szefom a pańskim znajomym, panom Alanowi Sugarowi i Joe Oki?

– Ogromny. Są to ludzie wprost stworzeni do prowadzenia interesów, mający jednocześnie ogromne wyczucie rynku i koniunktury, szczególnie Alan Sugar, który mimo powszechnie znanego szorstkiego i bezceremonialnego obejścia jest niezwykle popularną postacią. W zeszłym roku Sugar otrzymał zaszczytne miano Young Businesman of the Year („młody biznesmen roku" – przyp. red.).

– A jak ocenia Pan szanse PC 1512 na polskim rynku?

– Myślę, że również polscy specjaliści i wszyscy użytkownicy docenią zalety tego komputera. Przecież jest to wprost wymarzony komputer do wszelkich zastosowań: jest szybki, zintegrowany, wygodny w obsłudze i jak wszystkie produkty Amstrada, niezawodny i tani.

– Póki co, w instytutach i przedsiębiorstwach dominują tajwańskie składanki.

– Tak. Jednak biorąc pod uwagę konfigurację, w jakiej sprzedawany jest PC 1512, czy są one naprawdę tańsze? Nie bez znaczenia, szczególnie w polskich warunkach, powinna też być jakość kupowanego sprzętu. Pod tym względem Amstrad jest bez zarzutu.

– Porozmawiajmy o POLANGLII. Jak dawno istnieje firma?

– Powstała w 1984 roku.

– Czym się zajmujecie?

– Jesteśmy wyłącznym przedstawicielstwem Amstrada w Polsce. Zajmujemy się także wysyłkową sprzedażą innego sprzętu. Są to m.in. drukarki firmy Star.

– Wasze ceny są bardzo konkurencyjne, jeżeli nie najniższe z podawanych przez różne firmy wysyłkowe, dość przecież liczne. Jak to osiągacie?

– Jako jedyni posiadamy status pośrednika Amstrada w sprzedaży detalicznej do Polski. Należy wspomnieć, że sam Amstrad w ogóle nie zajmuje się sprzedażą detaliczną. Działa za pośrednictwem dystrybutorów dla poszczególnych krajów, w tym również POLANGLII. Dla nas oznacza to, że po pierwsze – kupujemy po cenach fabrycznych, dzięki czemu możemy być tani. Po drugie – otrzymujemy komputery bezpośrednio w magazynów Amstrada; są one opatrzone fabrycznym znakiem „Quality Control". Inne firmy oferują sprzęt z trzeciej ręki, często jest on po reperacjach. Od początku zdecydowaliśmy się zarabiać raczej liczbą sprzedawanych komputerów, a nie wysoką marżą na pojedynczych sztukach. Poza tym POLANGLIA jest małą firmą, stąd nieduże są koszty własne.

– Tym niemniej ostatnio ceny PC 1512 poszły u was w górę...

– Istotnie, ale wyłącznie na skutek podwyższenia cen przez samego Amstrada. Zresztą i my pobieraliśmy przedpłaty; wszyscy, którzy wpłacili do 31 października 1986 roku według starych cen, otrzymają bądź już otrzymali komputery bez dopłat, mimo że my kupowaliśmy je już po nowych cenach.

– To już prawie działalność charytatywna.

– Nie, po prostu dbamy o dobrą opinię.

– Czy nie obawia się Pan konkurencji ze strony Schneidera? Co to właściwie jest firma Schneider?

– Chociaż to może wydać się nieprawdopodobne, Schneider ma wobec Amstrada taki sam status jak POLANGLIA. Schneider jest po prostu pośrednikiem Amstrada na RFN. Tyle że jego zamówienia są tak duże, że Amstrad zgadza się na oznakowanie egzemplarzy przeznaczonych do RFN pod nazwą „Schneider". To tylko kwestia wielkości zamówień. Gdybyśmy zamówili jutro np. 50 000 sztuk PC 1512, to niedługo pojawiłby się w Polsce „nowy komputer" POLANGLIA PC 1512, kompatybilny z Amstradem PC 1512. Mógłby on mieć kolor żółty, instrukcje po polsku i złącza pośrednie, wszystko według życzenia. Jest takie pupularne angielskie powiedzonko „business is business..." Wszystkie „Schneidery" produkowane są w Korei, w fabryce Amstrada. Wracając do konkurencji. Oczywiście wiele osób kupi sprzęt w RFN. Jest on tam jednak znacznie droższy. Tak na marginesie: Amstrad wielokrotnie miał z firmą Schneider zatargi cenowe. Poza tym warto zaznaczyć, że Schneider nie ma praw eksportu do Polski; istnieje taki „gentelman's agreement", że cała oficjalna korespondencja z Polski jest przesyłana do POLANGLII. My z kolei wysyłamy do Schneidera wszystko, co dotyczy rynku niemieckiego.

– Podkreślił Pan niezawodność sprzętu Amstrada. Wie Pan jednak doskonale, że zakup u Pana to dla przeciętnego Polaka ogromna pozycja w budżecie. Obawy o ewntualny serwis są więc zrozumiałe.

– Oczywiście. Między innymi plonem mojego obecnego pobytu w kraju jest załatwienie tej sprawy. Mogę już powiedzieć, że serwisem zakupionego u nas sprzętu zajmuje się firma Refleks z Warszawy. Za gwarancję serwisową pobieramy dodatkową opłatę.

– Czy dostępne są już modele PC wyposażone w Winchester?

– Tak. Wokół tych dysków powstało wiele plotek. Prawda jest taka, że Amstrad realizował najpierw zamówienia na mniejsze konfiguracje. Było to zgodne z zapowiedziami z sierpnia zeszłego roku. Już wtedy wszystkie problemy techniczne były rozwiązane.

– A jak sprzedaje się amstradowe Spectrum?

– Spectrum liczy sobie kilka lat. Nowa jego wersja ma znacznie ułatwioną obsługę (wbudowany magnetofon, podobnie jak CPC 464), jest bardzo tania i oczywiście istnieje moc napisanego dla Spectrum oprogramowania. Ma bardzo wielu odbiorców.

– Wspomniał Pan o drukarkach Star. Najpopularniejsze w Polsce modele SG 10 i SG 15 nie są już produkowane. Co proponuje Pan w zamian?

– W ramach sprzętu firmy Star – drukarki NL 10. Chciałbym jednak zwrócić uwagę na nową, doskonałą drukarkę Amstrada: DMP 4000. Jest ona kompatybilna ze wszystkimi komputerami Amstrada, od CPC 464 po PC 1512 HD20 CM. Jest również tańsza „siostra" DMP 4000 – drukarka DMP 3000. Być może czytelników MIKROKLANU zainteresuje fakt, że Amstrad planuje osiągnięcie na rynku drukarek takiej pozycji, jaką ma na rynku komputerów. Jeżeli wszystko przebiegać będzie zgodnie z planami, to za rok Amstrad stanie się drugim producentem drukarek w świecie.

– Rozmawiamy po polsku. Czy urodził się Pan w Polsce?

– Jestem Polakiem. Tyle że urodziłem się w Argentynie. Oboje rodzice są Polakami, których losy rzuciły po wojnie do Ameryki Południowej. W Londynie natomiast mieszkam od pięciu lat. W Polsce mam rodzinę i wielu przyjaciół.

– I ostatnie pytanie. Czego życzyłby Pan sobie w 1987 roku.

– Aby był pod każdym względem tak udany jak ubiegły.

– A Czytelnikom MIKROKLANU?

– Wszystkim – Amstrada PC 1512.

– Z przyjemnością dołączamy się do obu życzeń. Dziękujemy za rozmowę i do zobaczenia w kraju.

*) Rozmowę przeprowadzono na początku stycznia.

W następnym numerze:

DRUKARKI