młody TECHNIK

informik

MAGAZYN KOMPUTEROWY „MŁODEGO TECHNIKA"



Nasz komentarz:

NA WIRAŻU

Historia mikrokomputera osobistego, zwłaszcza rozumianego, na podobieństwo motoryzacji, jako czynnik cywilizacyjny, osiągnęła kolejny, drugi już zakręt. Za pierwszy zakręt można uznać wkroczenie IBM i pojawienie się PC. Teraz pora na kolejny wiraż, chociaż pisk opon jest jakby cichszy. Wracając do komunikacyjnej analogii: wiraż, zwłaszcza ostry, jest tym miejscem, z którego widać obydwa odcinki drogi: ten, który już przebyliśmy od ostatniego zakrętu, oraz ten, na który właśnie wkraczamy. Warto więc troszeczkę zwolnić, by nie przegapić okazji do ostatniego ostrego spojrzenia przez ramię. Kończy się oto era sprzętu, umownie określanego jako 16-bitowy (XT i AT), a zaczyna epoka osobistych komputerów 32-bitowych. Jest to wydarzenie znacznie ważniejsze, niż wprowadzenie przez IBM rodziny PS/2 i mikrokanału. Tym razem nie chodzi bowiem o technologiczną kosmetykę, ale o fundamentalną koncepcję.

Kiedy nie tak dawno pojawił się 32-bitowy mikroprocesor Intel 80386, stał się on egzotycznym nadzieniem kosztownego sprzętu klasy „super-PC". Upłynął pewien czas, i oto pojawiły się jednak dwa kolejne procesory, należące do tej samej rodziny: 80386SX oraz 80486. Stosunek między klasycznym procesorem 80386 a jego wersją SX jest taki, jak między 8086 a 8088. Wersja SX ma podobną architekturę wewnętrzną jak zwykły 386, ale zewnętrzną magistralę tylko 16-bitową, co pozwala konstruować sprzęt znacznie prostszy, tańszy i przy pełnym wykorzystaniu istniejących już rozwiązań PC/AT. Sam procesor także kosztuje o wiele mniej od „pełnej" wersji 32-bitowej. Co prawda wydajność jest mniejsza, ale użytkownik dysponuje pełnymi właściwościami programowymi procesora 80386. To ważne, gdyż procesor ten jest pierwszym w rodzinie, który naprawdę nadaje się do systemów wielozadaniowych z wirtualną gospodarką pamięcią i ochroną zasobów.

Procesor 80486 to krok w przeciwnym kierunku. Ponad milion tranzystorów w jego strukturze dzieli się między procesor zgodny z 80386, integralny koprocesor zgodny z 80387, ultraszybką pamięć podręczną o pojemności 8 KB oraz zaawansowane systemy sterowania pamięcią. Procesor nie jest zwyczajną „kalką" starego 386: realizuje te same rozkazy (plus kilka nowych), ale w inny sposób — sprzętowo, a nie za pomocą mikroprogramu, tak, że większość rozkazów mieści się w jednym—dwóch cyklach zegara systemowego. Takie połączenie koncepcji CISC z kuchnią RISC daje kilkakrotny wzrost wydajności w porównaniu z 80386.

Tym sposobem nareszcie istnieje dziś rodzina procesorów pozwalająca konstruować sprzęt różnej klasy — od komputerów biurkowych do bardzo wydajnych stacji roboczych klasy mini-komputera — ale o zbliżonych podstawowych właściwościach programowych. Sytuacja ta jest zupełnie inna w porównaniu z rodziną PC/XT/AT, gdzie chęć zachowania zgodności z najmniejszym członkiem rodziny skutecznie utrudniała wykorzystanie pełnych możliwości procesora 80286, a z jego zaawansowanych właściwości wykorzystywana była w zasadzie tylko jedna: większa szybkość pracy.

Rozszerzoną pamięć AT wykorzystywano w najlepszym razie jako dysk wirtualny, co zresztą często prowadziło do pozornie niewytłumaczalnych kłopotów. Na czas dostępu do pamięci rozszerzonej blokowana była bowiem obsługa przerwań. Winę ponosi tu zresztą kalectwo 80286, który łatwo było przełączyć w naturalny dla niego tryb adresacji wirtualnej, ale oficjalna droga powrotna nie istniała: trzeba było podeprzeć się „sztuczkami". Natomiast 80386 jest już procesorem naprawdę dojrzałym. Wiedzą o tym ci, którym zdarzyło się zetknąć chociażby ze znanym już od jakiegoś czasu systemem WINDOWS 386 czy systemem operacyjnym MOS386.

Procesorem nr l w statystycznym komputerze osobistym wczesnych lat dziewięćdziesiątych będzie więc zapewne 80386 w której z mutacji, a pamięć operacyjna będzie się liczyć w megabajtach. Stworzone zostały w tym kierunku solidne podstawy w postaci masowej produkcji „kostek" pamięci 1 MB, a wkrótce zapewne i 4 MB. Inne szczegóły architektury są już mniej ważne, poza tym można liczyć na zdrowy instynkt wytwórców sprzętu, którzy samorzutnie dostosowują się do ustabilizowanych na rynku standardów — wszystko jedno, czy będzie to mikrokanał czy też konkurencyjna magistrala EISA.

W warunkach rozpowszechnienia procesorów klasy 386 gwałtownie wzrośnie w komputerach osobistych znaczenie systemu operacyjnego UNIX, dla którego sprzęt klasy XT i AT/286 był o dwa numery za mały — chociażby z powodu braku mechanizmów skutecznego rozgraniczenia równoległe wykonywanych procesów. UNIX jest klasycznym systemem wielozadaniowym i wielodostępnym, rozwiązującym z natury mnóstwo problemów, które w systemie DOS kosztowały hektolitry potu całą rzeszę programistów. Procesor 80486 jest właśnie wyraźnym krokiem firmy Intel w kierunku „osowjenia" UNIX-a, który dotychczas spośród popularnych mikroprocesorów najlepiej czuł się na MC68000/10/20. Inaczej niż PS/2, UNIX nie jest nowym systemem — istnieje dla niego mnóstwo dojrzałych aplikacji, sprawdzonych wcześniej na stacjach roboczych i mini-komputerach.

Czy spojrzenie przez ramię za zakręt dostarcza jasnego obrazu? Wszystko zależy od tego, czy zdobędziemy się na konsekwentny, bezmiarty obiektywizm i spróbujemy odrzucić na chwilę żywiołową fascynację postępem technicznym. Pojawienie się nowej — tym razem naprawdę nowej — generacji sprzętu nie może prowadzić do potępiania istniejących komputerów PC/XT i AT. Będą one produkowane jeszcze przez wiele lat, coraz tańsze i w wielu zastosowaniach, zwłaszcza biurowych, zapewne optymalne. Nowy sprzęt otwiera natomiast drogę do jakościowo nowych aplikacji i możliwości. Warto uwzględniać to już obecnie, kreśląc plany różnych mniej lub bardziej długofalowych przedsięwzięć informatycznych. Wiele można osiągnąć w tym kierunku metodą drobnych kroczków, chociażby sięgając szerzej do języka C i programując tak, by zapewnić zgodność z realizacjami tego języka w systemie UNIX.

Nawet jeśli nowa generacja mikrokomputerów jest dla nas na razie piosenką przyszłości, już teraz można zrobić wiele, aby jej przyjęcie nie odbywało się na zasadzie: „kupujemy nowe — stare na śmietnik". Nie powtarzajmy znów błędów, które popełniono już raz, przesiadając się ze ZX Spectrum na IBM-PC.

Roland Wacławek



młody TECHNIK informik

MŁODY TECHNIK INFORMIK

MAGAZYN KOMPUTEROWY „MŁODEGO TECHNIKA" NR IV [12] ROCZNIK III

SPIS TREŚCI

FELIETONY:

NA WIRAŻU — Roland Wacławek . . . . II s. okł.

O KOMPUTERZE — NIEINFORMATYCZNIE — Jerzy Klawiński 1

ARTYKUŁY:

TURBO PASCAL 4.0: TPFiles: MODUŁ POMOCNICZY PODPROGRAMÓW OBSŁUGI PLIKÓW — Jacek Rauk . 2

HISTORIA SPRZED PÓŁ WIEKU O PEWNYM GARAŻU, MONECIE I 538$ — Grzegorz Zalot 13

ZAGĘSZCZENIE WYDRUKU GRAFIKI — Roland Wacławek 16

KRZEMOWE SUPERMÓZGI — Krzysztof Wiśniewski . 20

INTERFEJS — BUFOR DO ZX SPECTRUM — Dariusz A. Przygoda 23

STAŁE DZIAŁY:

NOWE I NAJNOWSZE . . SEMINARIUM INFORMIKA: ZX-y INACZEJ! — Janusz Młodzianowski, Tadeusz Zaleski 25

KOMPUTER W SZKOLE — opr. Ireneusz Włodarczyk . 30

Z IBM PS/2 W „MIKROKANAŁACH" — Roland Wacławek . . . III i IV s. okł.

Zdjęcia w numerze: Władysław P. Jabłoński, Grzegorz Zalot, „BYTE".

Numer ilustrował: Jacek Nowicki.

O KOMPUTERZE – NIEINFORMATYCZNIE

Przypominam sobie następującą scenę: na pewnych targach dwóch młodzieńców rozmawia z pewnym dystyngowanym panem o niepozornej, szarej szafie, na której stoi plakieta symbolizująca przyznany złoty medal targowy.

— I to ma być komputer? — dziwi się jeden z młodych ludzi. — Bez klawiatury i monitora?

*— Ależ tak, zapewniam pana — odpowiada specjalista. — Jest to komputer, tyle że specjalizowany, **nieinformatyczny**...*

*Zadumałem się, słysząc coś takiego — do tej pory nie zastanawiałem się **jakie** mogą być komputery...*

Definicja komputera, zawarta w „Małym ilustrowanym leksykonie technicznym" (Wydawnictwa Naukowo-Techniczne, Warszawa 1982), głosi, iż jest to maszyna matematyczna — zestaw automatycznie działających urządzeń do przetwarzania danych. Ani słowa o zastosowaniach, budowie, odmianach itp.

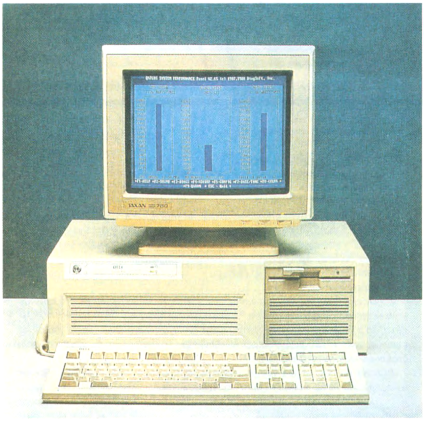
*Inaczej patrzą na ten temat Amerykanie, którzy każdy **mikroprocesor** uważają (i liczą sobie!) jako **mikrokomputer**. Nie umiem do końca powiedzieć dlaczego, ale taka koncepcja **specjalizowanego** komputera najbardziej mi odpowiada. Może dlatego, że przecież w pełni wyczerpuje szeroko „zakrojoną" definicję małego leksykonu?*

W naszej świadomości społecznej komputer jawi się nam w swej postaci z naiwnych reklam naszej telewizji: barwny ekran, płyta główna i klawiatura z obowiązkowo doczepioną do niej uroczą panienką. O innych formach tego urządzenia zazwyczaj się nie wie. A czyż inżynierski kalkulator z rozbudowanymi działaniami i pamięcią kilku działań nie jest wyspecjalizowaną formą komputera? A breloczek odpowiadający na gwizdnięcie dźwięczną melodyjką? A komputer pokładowy samochodu wyświetlający aktualne dane bez informatycznej klawiatury? A zegarek z programowanym budzikiem i kalkulatorem?

*Są fakty, zjawiska czy rzeczy nie poddające się zabiegowi definiowania. Są jednak i takie, które dałyby się zapewne zdefiniować, ale... Inna będzie ich definicja dla specjalisty, inna dla ludzi spoza branży. W tej sytuacji należy postawić pytanie, na czym polega tworzenie definicji **uniwersalnych**, zrozumiałych i zaakceptowanych przez wszystkich. Chyba nie na tych przesadnej szerokości interpretacyjnej, a na celnym wykazaniu **głównej idei** danego urządzenia, bez zbędnych ograniczeń (np. jak bez klawiatury i nie do celów matematycznych — to nie komputer!), które i tak bardzo szybko życie przenosi do lamusa.*

*Fakt istnienia komputerów specjalizowanych, czy też nieinformatycznych skłania mnie — a może skłoni i osoby zajmujące się określaniem co jest, a co nie jest komputerem — do przemyślenia na nowo pewnych starych i może już nieco zdezaktualizowanych definicji. Może warto pomyśleć nad umiejętnym **poszerzeniem** definicji? Tylko czy piorąca wówczas w automatycznej pralce gospodyni domowa będzie miała czas sprawdzić w nowej edycji „Małego ilustrowanego leksykonu techniki", czy pierze w pralce z **komputerem?***

JERZY KLAWIŃSKI



JACEK RAUK

Turbo Pascal 4.0: TPFiles/ — MODUŁ POMOCNICZYCH PODPROGRAMÓW OBSŁUGI PLIKÓW

Jednym z podstawowych problemów przy pisaniu bardziej rozwiniętych programów obliczeniowych (programy statystyczne, przetwarzanie dużych zbiorów liczb, iteracja funkcji itp.) jest zapewnienie właściwego, czyli wygodnego dla użytkownika i odpornego na jego pomyłki sposobu wprowadzania danych. Nie jest oczywiście problemem zabezpieczenie programu przed pomyłką polegającą na wpisaniu przecinka zamiast kropki dziesiętnej czy liczby rzeczywistej zamiast całkowitej. Służy do tego dyrektywa {**$I**−} włączająca automatyczną kontrolę poprawności operacji wejścia/wyjścia i funkcja **IOResult** (przykłady ich wykorzystania podałem w artykule *Turbo.Lib — biblioteka procedur Turbo Pascala 3.0* zamieszczonym w „InforMiku" 1/1989). Nie można jednak w ten sposób zabezpieczyć się przed pomyłką użytkownika polegającą na wpisaniu złej — chociaż semantycznie poprawnej — wartości (błąd taki jest bardzo częsty, zwłaszcza tam, gdzie wprowadzać należy długie kolumny liczb). Przy dużej ilości danych wczytywanie ich „z klawiatury" jest ponadto bardzo uciążliwe.

Niedogodności tych uniknąć można wykorzystując zbiory dyskowe. (W tekście artykułu pod pojęciem **„plik"** rozumiem zmienną plikową, zaś zbiory fizyczne istniejące na dysku określam jako **„zbiory"**, **„zbiory zewnętrzne"** i **„zbiory dyskowe"**; w komentarzach na listingach w obu przypadkach używam nazwy **„plik"**). Dają one możliwość dokładnego sprawdzenia wszystkich danych i ich poprawy, dopisania opuszczonej linii czy korekty popełnionej pomyłki.

Turbo Pascal 4.0 dysponuje bogatym zestawem podprogramów ułatwiających przeprowadzanie operacji na plikach i zbiorach. Ich dokładny opis znaleźć można

w licznych podręcznikach tego języka (np. Jan Bielecki: *Rozszerzony Turbo Pascal wersja 4.0*; WKŁ; Warszawa 1988). Szczególnie dogodne jest wykorzystanie plików tekstowych (**text**), łatwych w tworzeniu i przeglądaniu. Zestawienie najważniejszych podprogramów, przydatnych przy pracy z plikami tekstowymi i zbiorami znajduje się w **tabeli 1**. Jeszcze bogatsze możliwości oferuje **Turbo Pascal 5.0 (tabela 2)**. Wykorzystanie ich ułatwia znakomicie pisanie programów „przyjaznych" wobec użytkownika (tzw. ang. *user friendly*) i wygodnych w obsłudze. Standardowe procedury i funkcje nie rozwiązują jednak wszystkich problemów. W czasie pisania programów denerwowało mnie bardzo np. to, że błąd w pliku, nawet „zauważony" przez procedurę czytania danych, powodował konieczność przerwania programu, wywołania edytora, korekty i ponownego rozpoczęcia pracy. Stanowiło to poważne utrudnienie, zwłaszcza dla początkujących użytkowników komputera. Drażniące było też przechowywanie nazw plików zewnętrznych w postaci zmiennej typu **string**, z której dopiero należało pracownice „wyłuskiwać" nazwę zbioru, rozszerzenie, numer stacji dysków i ścieżkę dostępu. Dla ułatwienia sobie życia opracowałem kilka podprogramów i trików do korzystania z plików, którymi podzielić chciałbym się z Czytelnikami. Podprogramy zgrupowane są w moduł **TPFiles (listing 1)**, który dołączyć należy **po uprzednim dołączeniu modułów CRT i DOS**. Oto jego opis.

Tabela 1. Ważniejsze standardowe podprogramy Turbo Pascala 4.0 przydatne przy obsłudze plików tekstowych.

Nazwa	Moduł	Typ *)	Opis
		Operacje zapisu i odczytu	
Assign	System	proc	skojarzenie pliku ze zbiorem;
AssignCrt	Crt	proc	skojarzenie pliku z konsolą;
SetTextBuf	System	proc	związanie z plikiem jawnego buforu;
Append	System	proc	otwarcie pliku do dopisywania;
Reset	System	proc	otwarcie pliku do czytania;
Rewrite	System	proc	otwarcie pliku do zapisu;
Read	System	proc	wprowadzenie danych z pliku;
Readln	System	proc	wprowadzenie danych z pliku; przejście do nowej linii;
Write	System	proc	wprowadzenie danych do pliku;
Writeln	System	proc	wprowadzenie danych do pliku; przejście do nowej linii; "obniżenie" pliku;
Truncate	System	proc	rozpoznanie końca pliku;
Eof	System	f boolean	rozpoznanie końca pliku;
SeekEof	System	f boolean	rozpoznanie końca pliku z pominięciem spacji;
Eoln	System	f boolean	rozpoznanie końca wiersza;
SeekEoln	System	f boolean	rozpoznanie końca wiersza z pominięciem spacji;
IOResult	System	f word	funkcja poprawności operacji we/wy; "zerowanie" bufora;
Flush	System	proc	zamknięcie pliku;
Close	System	proc	
		Operacje na zbiorach, dyskach i katalogach	
FindFirst	Dos	proc	wyszukanie pierwszego zbioru o podanych cechach;
FindNext	Dos	proc	wyszukanie następnego zbioru o podanych cechach;
GetFAttr	Dos	proc	określenie atrybutów zbioru;
SetFAttr	Dos	proc	ustawienie atrybutów zbioru;
GetFTime	Dos	proc	ujawnienie czasu ostatniej modyfikacji zbioru;
SetFTime	Dos	proc	wymuszone ustawienie czasu modyfikacji zbioru;
Erase	System	proc	usunięcie zbioru;
Rename	System	proc	zmiana nazwy zbioru;
DiskSize	Dos	f longint	wyznaczenie pojemności dysku;
DiskFree	Dos	f longint	wyznaczenie wolnej przestrzeni dysku;
GetDir	System	proc	podanie nazwy katalogu;
ChDir	System	proc	zmiana bieżącego katalogu;
MkDir	System	proc	utworzenie podkatalogu;
RmDir	System	proc	usunięcie pustego podkatalogu

*) - w rubryce Typ oznaczono: procedury: proc, funkcje: f/typ.

typ FileName —
— jest to typ służący do przechowywania nazw zbiorów. Jego poszczególne pola oznaczają:
— **correct: Boolean** — przyjmuje wartość true, gdy nazwa podana jest poprawnie. Jeśli nie (np. występują znaki niezgodne z regułami DOS-a) ma wartość false;
— **drive: word** — numer stacji dysku w konwencji przyjętej w funkcjach standardowych **DiskSize** i **DiskFree** (1 = A, 2 = B itd.);
— **path: string[63]** — ścieżka dostępu;
— **name: string[8]** — nazwa zbioru;
— **ext: string[3]** — rozszerzenie nazwy zbioru.

zmienne InpN, OutN typu FileName —
— służą do przechowywania nazw zbiorów danych i wyników. Ich wartości określane są przez procedurę **ReadIONames**.

zmienne InpF, OutF typu text —
— pliki wejścia/wyjścia. Moduł NIE KOJARZY ich z nazwami plików **InpN, OutN**. Dokonać tego musi piszący program w wybranym przez siebie momencie: **Assign (InpF,FName(InpN));**

funkcja Verify (jej odpowiednikiem w **TP 5.0** jest procedura **GetVerify**) —
nagłówek: **function Verify:byte;**
— pozwala na sprawdzenie znacznika weryfikacji zapisu. Jeśli jest on ustawiony, funkcja przyjmuje wartość **ON (1)**, jeśli nie — **OFF (0)**. Stałe **ON** i **OFF** są zdefiniowane w interfejsie modułu.

procedura SetVerify (jej odpowiednikiem w **TP 5.0** jest procedura **SetVerify**) —
nagłówek: **procedure SetVerify (status:byte);**
— pozwala na ustawienie znacznika weryfikacji zapisu (**1 = ON, 0 = OFF** — można wykorzystać stałe **ON** i **OFF**). Przy pracy ze zbiorami zewnętrznymi należy na wszelki wypadek ustawić wskaźnik na **ON**, dobry zwyczaj nie zmieniania parametrów ustalonych przez właściciela komputera nakazuje jednak po zakończeniu pracy programu powrócić do zastanego stanu wskaźnika (patrz: **listing 2**).

procedura StrtoFN (jej odpowiednikiem w **TP 5.0** jest procedura **FSplit**) —
nagłówek: **procedure StrtoFN (var FN:FileName, NM: string);**
— działanie procedury polega na zamianie nazwy zbioru zewnętrznego **NM** podanej w postaci **string** na zmienną **FN** typu **FileName**. Jednocześnie przeprowadzona jest uproszczona kontrola poprawności nazwy. Jeżeli zamiana nie napotka przeszkód, pole **correct** zmiennej **FN** przyjmie wartość **true**, zaś pozostałe pola odpowiednio:
drive: numer podanego w nazwie napędu dyskietek (patrz: opis typu **FileName**) lub gdy litera oznaczająca napęd nie wchodzi w skład łańcucha **NM** — numer aktualnego napędu;
path: jeżeli w zmiennej **NM** bezpośrednio po literze oznaczającej napęd dysków lub na początku nazwy występuje znak '\' pole **path** zawiera podaną w nazwie **NM** ścieżkę dostępu. Jeśli nie, podana ścieżka dostępu

Tabela 2. Dodatkowe podprogramy obsługi zbiorów dyskowych wprowadzone przez wersję Turbo Pascal 5.0.

Nazwa	Moduł	Typ *)	Opis
FSplit	Dos	proc	podział nazwy zbioru na logiczne składowe;
FExpand	Dos	f PathStr	rozwinięcie nazwy zbioru do pełnej nazwy kwalifikowanej;
FSearch	Dos	f PathStr	poszukiwanie nazwy zbioru w katalogach;
GetVerify	Dos	proc	pobranie stanu znacznika weryfikacji zapisu;
SetVerify	Dos	proc	ustawienie stanu znacznika weryfikacji zapisu.

*) - w rubryce Typ oznaczono: procedury: proc, funkcje: f/typ.

```
{                LISTING 1                    }

{IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM;}
{:              TPFiles     v 0.5              :}
{:      modul pomocniczych procedur            :}
{:            obslugi plikow                   :}
{:                              J.Rauk    1989 :}
{HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<}
Unit                                    TPFiles;
{*************************************}
                 Interface
{*************************************}
Uses
   Dos,Crt;
Const
 {          Verify / SetVerify              }
 ON = 1;   OFF =0;
Type
 FileName = record
 { typ obrazujacy nazwe zbioru             }
 {poprawnosc} correct  :  Boolean;
 {naped       } drive    : word;
 {sciezka     } path      : string[63];
 {nazwa       } name      : string[8];
 {rozsz.      } ext       : string[3]
              end;
Var
   InpN,OutN   :  FileName;
   {  nazwy   plikow   wejscia - wyjscia }
   InpF,OutF   :  text;
   {  pliki wejscia - wyjscia              }

function                        Verify : byte;
{ zwraca znacznik weryfikacji zapisu     }
{ 0 = OFF, 1 = ON                         }

procedure    SetVerify ( status : byte );
{ ustawia znacznik weryfikacji zapisu    }
{ 0 = OFF, 1 = ON                         }
procedure
 StrtoFN ( var FN:Filename; NM:string  );
{ procedura zamienia nazwe pliku          }
{ (string) na rekord typu FileName   i   }
{ bada jej zgodnosc z regulami DOS-u     }

function     FName ( FN:Filename): string;
{ zamienia rekord FN na string            }

function
  PathExist ( FN : Filename ) : Boolean;
{ bada istnienie sciezki dostepu do       }
{ pliku FN                                }

function
    FileExist ( FN :FileName ): Boolean;
{  funkcja bada istnienie pliku FN        }

procedure        BackUp ( FN : FileName );
{ procedura  zmienia  rozszerzenie        }
{ nazwy  pliku  FN na  BAK.               }

function              CurrentDDrive : word;
{ zwraaca nr aktualnego napedu dyskow    }
{ 1 = A, 2 = B, 3 = C ...                 }

function
    DDriveOK ( Drive : word ) : boolean;
{ testuje gotowosc do pracy stacji        }
{ dyskow HDD, FDD, RAM-Disk               }

procedure ReadIONames      ( Def:string );
{ procedura ustala nazwy plikow we-wy    }
{ oraz wykonuje automatycznie Backup     }
{ pliku wyjscia o ile istnial             }

procedure   Edit   (Name:string;Li:integer);
{ Mini-Edytor danych                      }

              Implementation
 {*************************************}
 Const
  { KOMUNIKATY :                          }
  inf01  =  'Input    file  [';
  inf02  =  'Output  file  [';
  inf03  =  '] ==>';
  inf04  =  'Illegal  file  name  !';
  inf05  =  'Drive  ';
  inf06  =  ' is  not  ready  .';
  inf07  =  'Correct  it  and  hit  [ENTER].';
  inf08  =  'Path  ';
  inf09  =  '  doesn''t  exist  .';
  inf10  =  'Give  the  file  name  correctly  !';

  CorrectChar :                set of char =
  { zbior znakow dopuszczalnych w          }
  { nazwach zbiorow DOS-u                  }
  ['!','#','$','&'..')','-','.','0'..':',
      '@'..'Z',','_','a'..'(',')','~'];


 function                        Verify : byte;

 {*************************************}
  var

   r : registers;          { pseudorejestry }

  begin
```

```
  r.AH  :=  $54;
  MsDos ( r );
  Verify  :=  r.AL
 end;                                  ( Verify )

procedure    SetVerify ( status : byte );
(*************************************)
 var
  r : registers;          ( pseudorejestry )
 begin
  r.AH  :=  $2E;
  r.AL  :=  status;
  MsDos ( r )
 end;                                ( SetVerify )

procedure
 StrtoFN  (  var  FN:Filename;  NM:string  );
(*************************************)
 var
  x : byte;                          ( licznik )
  a : char;                          ( zm. pom.)
 begin
  For x:=1 to Length ( NM) do
   begin
    NM[x]:=UpCase(NM[x]);
    If not (NM[x] in CorrectChar) then
     begin
      FN.correct  :=  false;
      Exit
     end
   end
  With FN do
   begin
    correct  :=  true;
    x  :=  Pos  (':',NM);
    If  x=0  then
     drive  :=  CurrentDDrive
     else
      If  x<>2  then
       begin
        correct  :=  false;
        Exit
       end
      else
       begin
        a  :=  NM[1];
        drive  :=  Ord(a)-64;
       end;
    NM  :=  Copy  (  NM,x+1,Length  (NM)  );
    If    (Pos(',NM)<>1)
    and  DDriveOK(drive)  then
     begin
      GetDir  (  drive,path  );
      path:=Copy    (path,3,Length(path));
      If  path<>'  then  path:=path+'
     end
    else
     path  :=  '';
    x:=  Pos  (',NM);
    While  x<>0  do
     begin
      path:=path+Copy  (  NM,1,x  );
      NM:=Copy  (  NM,x+1,Length  (  NM  )  );
      x:=  Pos  (',NM);
      If  x=1  then
       begin
        correct  :=  false;
        Exit
       end
     end   ;
    x  :=  Pos  (':',NM);
    If  x<>0  then
     begin
       name  :=  Copy  (  NM,1,x-1  );
       ext   :=  Copy  (  NM,x+1,3  );
     end
    else
     begin
      name  :=  NM;
      ext   :=''
     end;
   end;
```

```
function     FName ( FN:Filename): string;
{*****************************************}
 var
  x   : byte;                    { licznik }
  a   : char;                    { zm. pom.}
  nm  : string;                  { zm. pom.}
 begin
  With FN do
   If not correct then
    nm  :=''
   else
    begin
     a  := Char ( drive+64 );
     nm := Concat
      (a,':',path,'\',name,'.',ext)
    end;
  For x :=1 to Length ( nm) do
   begin
    nm[x]:=UpCase(nm[x]);
    If not ( nm[x] in CorrectChar ) then
     nm  :=''
   end;
  FName := nm
 end;                            { FName }

function
  PathExist ( FN : Filename ) : Boolean;
{*************************************}
 var
  pname,          { nazwa sciezki (string) }
  actpath : string; { aktualny katalog }
 begin
  PathExist  :=  false;
  GetDir   (0,actpath);
  With FN do
   If correct  then
    begin
     pname   :=   Char(drive+64)+':'+path;
{$I-} { wylaczenie kontroli we_wy          }
     ChDir ( pname );
{$I+} {  wlaczenie kontroli we_wy          }
     If  IOResult=0  then
      begin
       ChDir ( actpath );
       PathExist  := true
      end
    end
 end;                            { PathExist }

function
    FileExist ( FN :FileName ): Boolean;
{*************************************}
  var
   pl : file;                    { zm. pom.}
  begin
   Assign  ( pl,FName(FN) );
{$I-} { wylaczenie kontroli we_wy          }
   Reset ( pl );
{$I+} {  wlaczenie kontroli we_wy          }
   If IOResult = 0 then
    begin
     Close  ( pl );
     FileExist  :=  true
    end
   else
    FileExist  :=  false
   end;                            { FileExist }

 procedure       BackUp ( FN : FileName );
 {*************************************}
   var
    pl    : file;                  { zm. pom.}
```

SZANOWNI CZYTELNICY!

Oddajemy do Waszych rąk ostatni numer "InforMika" drukowany w formie oddzielnego czasopisma. Nie zaniechamy jednak kontynuowania tematyki informatycznej - szukajcie jej w naszym podstawowym czasopismie - "Młodym Techniku"!

```
    name : string;                    { zm. pom.}
   begin
    If FileExist ( FN ) then
    begin
     name := FName(FN);
     FN.ext  :='BAK';
     If FileExist ( FN ) then
      begin
       Assign ( pl,FName(FN) );
       Erase ( pl )
      end;
     Assign ( pl,name );
     Rename ( pl, FName(FN) )
    end
  end;                                { BackUp }

function              CurrentDDrive : word;
{****************************************}
 var
  r : registers;          { pseudorejestry }
 begin
  r.AH := $19;
  MsDos ( r );
   CurrentDDrive := r.AL+1;
 end;                          { CurrentDDrive }

 function
     DDriveOK ( Drive : word ) : boolean;
{****************************************}
 begin
  If DiskSize(Drive)=-1 then
    DDriveOK := false
  else
    DDriveOK := true
 end;                               { DDriveOK }

 procedure ReadIONames     ( Def:string );
{****************************************}
 var
    x     : integer;                { licznik }
    name  : string;                 { zm.pom. }
```

```
 procedure                          ReadIName;
 { czytanie nazwy pliku wejscia              }
  begin
   If Def<>'' then
    name := Def
   else
    name := 'NONAME';
   StrtoFN ( InpN,name );
   x := Pos ( '.',name);
   If x=0 then
     InpN.ext := 'INP';
   Write ( inf01,FName(InpN),inf03 );
   Readln ( name );
   If name<>'' then
    begin
     StrtoFN ( inpN,name );
     If InpN.correct then
      begin
       x := Pos ( '.',name);
       If x=0 then
        InpN.ext := 'INP'
      end
     else
      begin
       Writeln (inf04);
       ReadIName;
       Exit
      end
    end
  end;                                { ReadIName }

procedure                           ReadOName;
{ czytanie nazwy pliku wyjscia              }
  begin
   OutN.correct := InpN.correct;
   OutN.drive   := InpN.drive;
   OutN.path    := InpN.path;
   OutN.name    := InpN.name;
   OutN.ext     := 'OUT';
    Write     (inf02,FName(OutN),inf03);
```

```
  ReadIn ( name );
  If  name<>''  then
   begin
    StrtoFN ( OutN,name );
    If  InpN.correct  then
     begin
      x  :=  Pos  (  '.',name);
      If  x=0  then
       InpN.ext   :=   'OUT'
     end
    else
     begin
      Writeln   (inf04);
      ReadOName;
      Exit
     end
   end
 end;                                  { ReadOName }

begin
  If ParamCount>0 then
    begin
      name  :=  ParamStr  (1);
      StrtoFn  (  InpN,name  );
      x  :=  Pos  (  '.',name);
      If  x=0  then
       InpN.ext   :=   'INP';
      If  not  InpN.correct  then
        ReadIName;
        ReadOName
     end
   else
    begin
     ReadIName;
     ReadOname
    end;
  With InpN  do
   begin
    While not DDriveOK ( drive ) do
     begin
      Writeln
       (   inf05,Char(drive+64),inf06    );
      Writeln  (  inf07  );
      ReadIn
     end;
    While not PathExist ( InpN ) do
     begin
      Writeln     (inf08,Char(drive+64),
      ':',path,inf09);
      Writeln    (inf10);
      ReadIName
     end
   end;
  With OutN do
   begin
    While not DDriveOK ( drive ) do
     begin
      Writeln
       (   inf05,Char(drive+64),inf06    );
      Writeln  (  inf07  );
      ReadIn
     end;
    While not PathExist ( OutN ) do
     begin
      Writeln     (inf08,Char(drive+64),
      ':',path,inf09);
      Writeln    (inf10);
      ReadOName
     end
   end;
  Backup ( OutN )
 end;                                  { ReadIONames}

procedure   Edit   (Name:string;Li:integer);
{       rozwiazanie   prowizoryczne          }
{*************************************}
 begin
  Exec     ('c:ur4urbo.exe',Name)
 end;                                  { Edit }
end.                                  { modulu }
```

poprzedzona jest ścieżką dostępu od „korzenia" do aktualnie otwartego katalogu napędu podanego w **drive**. Zabezpieczona to przed zgubieniem dostępu do zbioru w czasie zmian katalogów i napędów (np. procedurą **ChDir**);

name: nazwę zbioru;

ext: rozszerzenie nazwy zbioru.

Jeśli w trakcie zamiany wystąpią przeszkody uniemożliwiające jej poprawne wykonanie, pole **correct** przyjmie wartość **false**, zaś pozostałe pola – wartości nieokreślone. Dla przykładu:

— string **'TPFiles.pas'** nada polom rekordu **FN** wartości: **correct** = true, **drive** = nr aktualnej stacji, **path** = = ścieżka dostępu do zbiorów w aktualnym katalogu, **name** = 'TPFILES', **ext** = 'PAS'.

— string **'b:\jakis\noname.txt':correct** = true, **drive** = **2**, **path** = '\JAKIS', **name** = 'NONAME', **ext** = = 'TXT'.

— string **'c:\jak < s' :correct** = false, **drive**, **path**, **name**, **ext** — wartości nieokreślone.

funkcja FName (jej odpowiednikiem w **TP 5.0** jest procedura **FExpand**)

nagłówek: **function FName (FN:FileName): string;**

— dokonuje operacji odwrotnej niż procedura **StrtoFN**, tzn. zamienia zmienną typu **FileName** na string, będący nazwą zbioru (czytelną dla procedur **Assign**, **AssignCrt**). Jeżeli pole **correct** zmiennej **FN** ma wartość **true** funkcja zwraca otrzymany string, jeśli nie — zwraca pusty łańcuch znakowy.

funkcja PathExist —

nagłówek: **function PathExist (FN:FileName): Boolean;**

— funkcja zwraca wartość true, gdy ścieżka dostępu do zbioru o nazwie **FN** istnieje lub **false** — w przeciwnym wypadku. Jej zastosowanie omówię przy opisie procedury **ReadIONames**.

funkcja FileExist —

nagłówek: **function FileExist (FN:FileName): Boolean;**

— funkcja ta zwraca wartość **true**, gdy zbiór o nazwie **FN** istnieje lub **false** — w przeciwnym wypadku. Procedurę tę w innej, nie wykorzystującej typu **FileName** formie, opisałem już uprzednio w bibliotece procedur **Turbo.Lib**.

procedura BackUp —

nagłówek: **procedure BackUp (FN:FileName);**

— podobnie jak poprzedni i ten podprogram ma swój pierwowzór w bibliotece **Turbo.Lib**. Wprowadzenie typu **FileName** umożliwiło napisanie tej procedury w dużo prostszy sposób. Jej działanie polega na zmianie rozszerzenia nazwy istniejącego zbioru dyskowego na **'.BAK'**. Jej wywołanie przed każdym otwarciem pliku procedurą **Rewrite** zabezpiecza program przed przypadkowym zniszczeniem starej zawartości zbioru skojarzonego z tym plikiem.

funkcja CurrentDrive —

nagłówek: **function CurrentDrive:word;**

— zwraca numer aktualnego napędu dysków w konwencji: 1 = A, 2 = B, 3 = C itd.

Informacje o aktualnym napędzie uzyskać wprawdzie można poprzez procedurę **GetDir**, wymaga to jednak pracochłonnego analizowania otrzymanego łańcucha znaków.

funkcja DDriveOK —

nagłówek: **function DDriveOK (Drive:word): Boolean;**

— dzięki wykorzystaniu rozszerzeń implementacyjnych **Turbo Pascala 4.0** możliwe było napisanie tej, znacznie uproszczonej i ulepszonej w stosunku do wersji podanej w bibliotece **Turbo.Lib**, funkcji. Testuje ona nie tylko gotowość do pracy stacji dyskietek elastycznych, ale też i dysku twardego i RAM-dysku. Jeżeli stacja jest gotowa do odczytu zbiorów, funkcja zwraca wartość *true*, jeśli nie (uszkodzenie, stacja o podanym numerze nie istnieje, nie włożono dyskietki itp.), wartość *false*. Numer stacji dysków podawany jest jako parametr wywołania **Drive** zgodnie z konwencją przyjętą w standardowych procedurach **DiskSize** i **DiskFree**, tzn. 0 = aktualny napęd, 1 = A, 2 = B itd. Kontrola wartości tej funkcji przed każdą operacją dyskową zabezpieczy program przed przerwaniem działania w wypadku uszkodzenia napędu (lub błędu użytkownika) i utratą wyników (w wypadku obliczeń numerycznych stracić można w ten sposób dużo czasu i nerwów).

procedura ReadIONames —

nagłówek: **procedure ReadIONames (Def:string);**

— w przeważającej większości programów wystarczające jest stosowanie jednego pliku danych i jednego pliku wyników. Sądzę, że opracowanie stałego sposobu określania przez użytkownika ich nazw oszczędza znacznie pracę. Przedstawiona procedura jest rozwiązaniem bardzo prostym. Wczytuje ona nazwę pliku wejścia z linii wywołania programu (umożliwia to pracę w trybie wsadowym, sterowanym programem typu ***.BAT**) i tworzy z niej nazwę pliku wyjścia przez zmianę rozszerzenia na **.OUT**. Jeżeli nazwa podana w linii wywołania nie ma rozszerzenia oraz nie jest zakończona kropką, przyjęte zostanie rozszerzenie **.INP**. Jeżeli w linii wywołania programu nie podano nazwy albo nazwa nie spełnia wymogów DOS-a, procedura przystępuje do wczytania nazw plików z klawiatury, proponując „defaultową" nazwę zbioru wejścia na podstawie parametru **Def**, oraz „defaultową" nazwę zbioru wyników na podstawie nazwy zbioru danych. Procedura dokonuje ponadto uproszczonej kontroli poprawności podanych nazw, sprawdza czy istnieje dostęp do podanych w nazwach napędów dysków (**DDriveOK**) i czy istnieją podane w nazwach ścieżki dostępu (**PathExist**). Kontrola ta pozwala wyeliminować błędy popełnione przez użytkownika. Podanie nazwy nieistniejącego pliku danych przy istniejącej ścieżce dostępu NIE JEST TRAKTOWANE JAKO BŁĄD, lecz jako prośba o „pomoc" przy stworzeniu zbioru danych (patrz: **listing** 2). Na koniec wywoływana jest procedura **BackUp** dla zbioru wyników. Opisaną procedurę sugeruję potraktować jako kanwę do stworzenia własnych, efektowniejszych rozwiązań — np. wyświetlenia listy dostępnych plików z rozszerzeniem **.INP** i wybrania właściwego przez podświetlenie (sposób podświetlania fragmentów ekranu opisałem w artykule *TPScreen: moduł dodatkowych procedur ekranowych* zamieszczonym w „InforMiku" 3/1989).

procedura Edit —

nagłówek: **procedure Edit (Name:string; Li:integer);**

— procedura ta jest właściwie wewnętrznym edytorem danych. Jej wykorzystanie nie musi ograniczać się do zabezpieczenia programu przed błędem w zbiorze danych, ułatwienia tworzenia zbioru danych czy wyświetlania wyników. Może ona z powodzeniem służyć do wyświetlenia dłuższych informacji na temat działania programu (instrukcji użytkownika) lub innych, przygotowanych wcześniej plików tekstowych. Możliwości jej wykorzystania przedstawia program **Demo** (**listing 2**). Pierwszy parametr — **Name** — przedstawia nazwę zbio-

```
{                LISTING 2                    }

{$M   64000,0,0}
{ ustawienie rozmiaru sterty i stosu      }
{ konieczne z powodu prowizorycznego      }
{ rozwiazania procedury EDIT              }

program                                DEMO;
{ Demonstruje wykorzystanie modulu        }
{ TPFiles do wczytywania danych           }
{ i udostepniania wynikow                 }
 uses
  Crt,Dos,{...,}TPFiles;

 const
  msg0 = 'END OF PROGRAM.';
  msg1 = 'PLEAS WAIT...';
  msg2 = 'INPUT FILE ';
  msg3 = 'OUTPUT FILE ';
  msg4 = ' DOESN''T EXIST.';
  msg5 = 'CREATE IT? [Y]es/[N]o ';
  msg6 = 'NOT ENOUGH DISK SPACE FOR ';
  msg7 = 'SELECT A NEW DISK DRIVE ';
  msg8 = 'UNEXPECTED END OF INPUT FILE';
  msg9 = 'DATA READING ERROR IN LINE ';

  MaxDat     = 50;
  { dopuszczalna liczba danych            }
  MaxIFSize = 20000;
  { max. rozmiar pliku we. (w bajtach) }
  MaxOFSize = 50000;
  { max. rozmiar pliku wy. (w bajtach) }

 type
  Data = record
        T,P : array [1..MaxDat] of real;
         end;
  {...}
 var
  A                  : Data; { dane        }
  VerStat            : byte;
  { znacznik weryfikacji zapisu           }
  FirstResultsLine  : integer;
  { nr pierwszej lini wynikow             }
  { ... }

 procedure
     CharReadln ( var a:char;b:string );
{ wczytuje znak z zakresu okreslonego }
{ przez wymienienie w zmiennej b        }
 var
  i,                 { zm. pom.            }
  max,min,           { zakres liter        }
  x,y   : integer; { poiozenie kursora }
  OK    : Boolean; { zm. pom.             }
 begin
  x := WhereX; y := WhereY;
  If y=25 then Dec(y);
  Repeat
   Readln ( a ); a := UpCase ( a );
   i := Pos ('..',b);
   If i = 0 then
    OK := Pos(a,b) > 0
   else
    begin
     max := Ord (b[i+2])+1;
     min := Ord (b[i-1])-1;
     OK:= (Ord(a)<max) and (Ord(a)>min)
    end;
   If not OK then
    begin
     Write (#7);
     GotoXY ( x,y );
     ClrEoL;
     Write (#7)
    end
  until OK
 end;                          { CharReadln }
procedure        GetData ( var a:Data );
{ wczytuje dane do dalszych obliczen   }
 var
  i        : integer;        { zm. pom    }
  verd     : char;           { zm. pom    }
  comment : string[80];      { komentarz }
  EoD      : Boolean;
  { znacznik konca danych                }
 begin
  If not FileExist ( InpN ) then
   begin
    While
    DiskFree(InpN.drive)<MaxIFSize    do
     begin
      Writeln    (msg6,msg2,'.');
      Write ( msg7 );
      CharReadln    (verd,'A..Z');
      InpN.drive    :=Ord(UpCase(verd))-64;
      InpN.path    := ''
     end;
    Writeln    (msg2,FName(InpN),msg4);
    Write (msg5);
    CharReadln ( verd,'YN' );
```

```
  If  verd='Y'then
   begin
    Writeln  (msg1);
    BackUp ( InpN );
    Assign  ( InpF,FName(InpN));
    Rewrite ( InpF );
    Writeln  ( InpF,'Succes');
    Writeln ( InpF );
    Writeln
     ( InpF,'PIMPal'10,'TIKI'10      );
    For i:=1 to MaxDat do
     Writeln(InpF,0:0:10,2,0:0:10:2);
    Close ( InpF );
    Edit  ( FName(InpN),1)
   end
  else
   begin
    Writeln  (msg0);
    Halt;
   end
 end;
While
DiskFree(OutNdrive)<MaxOFSize   do
 begin
  Writeln     (msg6,msg1,'');
  Write ( msg2 );
  ChatKeydin      (verd,'A'..'Z');
  OutNdrive   =   OrdUpCase(verd)-64;
  OutNpath    =  '';
 end;
Assign  ( InpF,FName(InpN) );
Reset  ( InpF );
Assign  ( OutF,FName(OutN) );
Rewrite ( OutF );
For i:=1 to 3 do
 begin
  Readln  ( InpF,comment );
  Writeln ( OutF,comment )
 end;
EoD  = SeekEoF ( InpF );
If EoD then
 begin
  Close     (InpF);Close(OutF);
  Writeln  (msg8);
  Edit    (FName(InpN),3);
  GetData ( a );          ( Rekurencja )
  Exit
 end;
i:=1;
While not EoD do
 begin
```

```
{$I-} ( wylaczenie kontroli we_wy          )
      Readln       (InpF,a.Pil1,a.Til);
{$I+} ( wlaczenie kontroli we_wy           )
      If not (IOResult=0)then
       begin
        Close     (InpF);Close(OutF);
        Writeln     (msg2,i+3,#7#7);
        Delay    (2000);
        Edit     (FName(InpN),i+3);
        GetData ( a );          ( Rekurencja )
        Exit
       end;
      EoD  := ( SeekEoF (InpF))
              or  (i=MaxDat);
      Writeln
      (OutF,a.Pil1:10:2,a.Til:10:2);
      Inc  ( i )
     end;
    FirstResultsLine   =   i+2;
    Close   (InpF);
    Close  (OutF)
   end;                          ( GetData )

 procedure              Work ( A:Data(...) );
 ( "makieta" procedury obliczeniowej      )
  begin
 ( Tu dokonuje sie obliczen ...            )
  end;                                ( Work )

 procedure                PutResults ((...));
 ( "makieta: procedury zapisujacej wyn)
  begin
   Append ( OutF );
   Writeln  ( OutF,'Results  :');
   Writeln ( OutF );
 ( Tu zapisuje sie wyniki ...              )
   Close ( OutF );
   Edit    (FName(OutN),FirstResultsLine);
   Writeln ( msg0 )
  end; ( PutResult )

 begin
  VerStat  := Verify;
  SetVerify ( ON );
  ReadIONames   ('DEMO');
  GetData ( A );
  Work  ( A  (...));
  PutResults     ((...));
  SetVerify ( VerStat )
 end.                                ( DEMO )
```

ru w postaci stringu (można tu wykorzystać funkcję **FName**), drugi nr linii, w której znaleźć winien się kursor po wywołaniu. Ze względu na dużą objętość procedury w **listingu 1** podałem „zaślepkę" (rozwiązanie nieeleganckie i prowizoryczne) polegającą na wykorzystaniu edytora **Turbo Pascala** (podobnie wykorzystać tu można dowolny inny edytor nie dopisujący znaków sterujących np. **Kedit**). W wersji tej parametr **Li** nie ma oczywiście znaczenia. Wady tego rozwiązania to np. konieczność założenia, iż potencjalny użytkownik ma wybrany przez nas edytor i że znamy katalog, w którym go przechowuje lub też konieczność dostarczenia go z naszym programem (**a co z prawami autorskimi?!**). Napisanie własnego edytora jako podprogramu nie jest zbyt trudne, choć dosyć pracochłonne. W najbliższym czasie postaram się opisać sposób jego budowy.

Wykorzystanie modułu i sposób korzystania z plików tekstowych omówię pokrótce na przykładzie programu **Demo** (**listing 2**).

Nagłówek programu poprzedzony jest ustawieniem rozmiaru stosu i sterty. Konieczne jest to ze względu na zastosowanie procedury **Exec** w podprogramie **Edit**. Napisanie własnego edytora danych pozwala uniknąć tej niedogodności. Program **Demo** „udaje" program obliczeniowy, szukający zależności pomiędzy ciśnieniem podanym w megapaskalach a temperaturą w skali Kelvina. Plik danych składać ma się z 2 linii komentarza (np. zapisanie źródła danych), 1 linii opisu (nie ma on większego znaczenia dla programu, ale ułatwia pracę użytkownikowi) i do 50 linii danych P-T (dwie kolumny liczb rzeczywistych nie mniejszych od zera). Ponieważ przedstawianie metodyki obliczeń nie było moim celem, procedury **Work** i **PutResults** są jedynie „makietami". W programie pozostawiłem tylko fragmenty istotne dla demonstracji korzystania z plików danych i wyników. Program rozpoczynają definicje stałych, będących komunikatami ekranowymi (zgrupowanie wszystkich komunikatów na początku pozwala na szybką zmianę wersji językowej programu — można też wykorzystać mechanizm kompilacji warunkowej), oraz 3 stałych:

MaxDat — określająca maksymalną dopuszczalną liczbę danych T-P, konieczna, gdy nie stosujemy zmiennych dynamicznych,

MaxIFSize — oszacowanie maksymalnego rozmiaru pliku danych w bajtach,

MaxOFSize — oszacowanie maksymalnego rozmiaru pliku wyników w bajtach.

Następnie zdefiniowany jest typ **Data** służący do przechowywania i przesyłania danych, oraz zadeklarowane są zmienne **A** (dane), **VerStat** (przechowujący zastany przez program stan znacznika weryfikacji zapisu) i **FirstResultLine** — licznik linii w pliku wyników. Zdefiniowana jest również procedura **CharReadln**, służąca do ułatwienia komunikacji z użytkownikiem. Następna procedura: **GetData** jest podprogramem do wczytania danych.

Działanie samego programu rozpoczyna zapamiętanie zastanego stanu flagi weryfikacji zapisu i ustawienie go na **ON**. Następnie wczytywane są nazwy plików wejścia/wyjścia (dzięki uprzedniemu rozwiązaniu tego problemu w module **TPFiles** nie stanowi to kłopotu). Teraz program wywołuje procedurę **GetData**. Procedura ta:

1. Sprawdza istnienie zbioru dyskowego **InpN** (jeżeli zbiór nie istnieje, procedura traktuje to jako prośbę o pomoc przy stworzeniu go).
2. Sprawdza, czy wolne miejsce na dysku wystarcza do założenia zbioru danych i — jeśli miejsca jest zbyt mało — prosi o podanie nazwy nowego napędu dysków.
3. Pyta użytkownika, czy założyć zbiór danych. Odpowiedź przeczącą przerywa program, potwierdzenie powoduje zapisanie na dysku o numerze **InpN.drive** szkieletu zbioru **InpN** z wolnym miejscem na komentarz i dwiema kolumnami zer. Zbiór ten następnie udostępniany jest do uzupełnienia (procedura **Edit**). Tu właśnie przydaje się pozornie niepotrzebna 3 linia (**'P[MPa](...)T[K]'**). Ułatwia ona orientację użytkownikowi. (Jeśli zbiór już istniał, czynności 2 i 3 są opuszczane.)
4. Sprawdza, czy dysk **OutN.drive** ma dość wolnego miejsca na zapis zbioru wyników i ewentualnie prosi o wybranie innego dysku.
5. Dokonuje skojarzenia pliku danych i wyników z nazwami zbiorów i ich otwarcia odpowiednio do odczytu i zapisu.
6. Przepisuje trzy linie tekstu z pliku **InpF** do **OutF** bez kontroli.
7. Czyta wartości ciśnień i temperatur. Jeżeli wartości obarczone są błędem semantycznym (zły typ) lub logicznym (wartości ujemne), pojawia się odpowiedni komunikat i dokonywana jest edycja zbioru danych. Po poprawie ponownie rozpoczyna się czytanie danych (rekurencja). Uproszczone rozwiązanie procedury **Edit** uniemożliwia oczywiście ustawianie kursora w błędnej linii. Pełne działanie programu możliwe będzie dopiero po zmianie tej procedury.
8. Po wczytaniu danych, program zapamiętuje numer linii, od której rozpoczynają się wyniki obliczeń i zamyka plik **InpF**.
9. Mimo iż plik **OutF** jest jeszcze niekompletny, jest on również zamykany (radzę każdorazowo zamykać plik przed dłuższą przerwą w zapisie i otwierać go ponownie procedurą **Append**, lub stosować procedurę **Flush** do wymiecenia bufora — uniemożliwi to utratę części informacji w wypadku nieprzewidzianego przerwania pracy programu).

Po wczytaniu danych dokonywane są obliczenia (procedura **Work**), a następnie wywoływany jest podprogram **PutResults**. Jego działanie polega na:

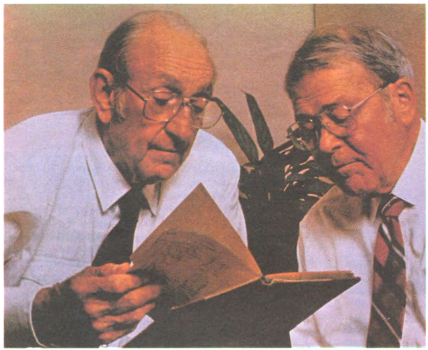
1. Ponownym otwarciu pliku **OutF** (tym razem do dopisywania);
2. Wpisaniu wyników obliczeń;
3. Zamknięciu pliku wyników;
4. Wyświetleniu zbioru wyników (procedura **Edit**);
5. Wyświetleniu komunikatu o zakończeniu programu.

Jeżeli przewidujemy pracę programu w trybie wsadowym, warto dopisać opcję pozwalającą z linii wywołania wyłączyć wyświetlanie wyników.

Mam nadzieję, że przedstawione procedury i sposób ich wykorzystania ułatwią Czytelnikom pisanie programów łatwych w obsłudze i pozwolą, poprzez wyeliminowanie pisania od nowa powtarzających się fragmentów programów, skoncentrować się na istocie rozwiązywanych problemów.

W numerze III/89 „InforMika" w listingu na str. 15 dwukrotnie zreprodukowano **trzy** wersy listingu. Aby program działał poprawnie wystarczy **wykreślić trzy pierwsze wersy lewego lamu**.

Przepraszamy!



Bill Hewlett (po prawej) i Dave Packard w chwili zadumy nad starą książką finansową firmy z pierwszych miesięcy jej istnienia

Grzegorz Zalot

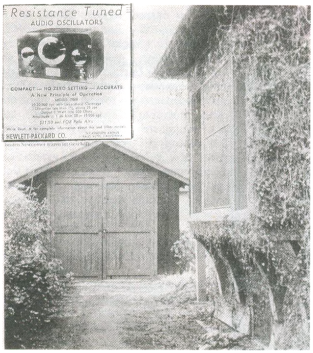
HISTORIA SPRZED PÓŁ WIEKU O PEWNYM GARAŻU, MONECIE i 538 $

Jeszcze przed wybuchem drugiej wojny światowej dwóch młodych inżynierów, kolegów z Uniwersytetu Stanford w Palo Alto w Kalifornii, postanowiło połączyć swoje umiejętności i założyć niedużą firmę elektroniczną. Dave Packard, wówczas już żonaty, wynajął niewielki domek, na zapleczu którego ulokował swojego kolegę, cieszącego się jeszcze urokami stanu wolnego, Billa Hewletta. Najważniejszy był jednak garaż — tam miała znaleźć swoje lokum nowo powstająca firma.

Zaczęło się od rzutu jednodolarówką. Wynik rozstrzygnął, czy firma będzie się nazywała Packard-Hewlett, czy Hewlett-Packard. Wynik znamy. Pierwsze wydatki zaczęły się we wrześniu 1938 roku. Były to 24 dolary i 84 centy na „środki produkcji". W grudniu doszło jeszcze 5,08 dolara na materiały biurowe. W pierwszym miesiącu działalności firmy (jej kapitał założycielski wyniósł równo 538 dolarów), czyli w styczniu 1939, mamy już znacznie większe koszty — $93,98 na materiały, a także niewielkie koszty $1,50 na samochód, $2,00 na energię elektryczną i dwa dolary dla „publicity". Rachunki z drobiazgową dokładnością prowadzi w tym czasie Lucille Packard. Pierwsze koszty wyposażenia biura, to $4,64 na używane biurko i $30 na maszynę do pisania. To w sierpniu 1939 — trzy miesiące później — firma inwestuje 70 dolarów w ogłoszenie o swoim pierwszym produkcie. Był to generator akustyczny HP-200A, o bardzo nowoczesnej na owe czasy konstrukcji. W stosunku do porównywalnych urządzeń na rynku miał on przy niższej cenie większy zakres częstotliwości, był także mniej wrażliwy na zakłócenia. Pierwszym większym odbiorcą urządzeń na kwotę $536,71 było studio filmów dźwiękowych Walta Disneya.

Na początku drugiego roku działalności zatrudniono już pracownika za całe 25 dolarów tygodniowo. W styczniu dochody wyniosły 907,71 dol. przy wydatkach 835,31 dol., a już trzy miesiące później, przy trzech pracownikach 2550 dol., przy wydatkach jedynie 2000. Drugi rok działalności to obrót aż 34 tysiące dolarów, trzech pracowników, przeprowadzka do nowego lokalu. Dwa lata później firma ma już własny budynek przy 395 Page Mill Road w Palo Alto. W tym roku opracowano również woltomierz o nie znanej dotychczas stabilności i niezawodności.

W roku 1950 HP poczyniło znaczący postęp w zakresie mikrofalowych urządzeń testujących. Rok później powstał HP-524A — licznik wielkiej częstotliwości. Wyniki otrzymywano za jego pomocą po niespełna dwóch sekundach, podczas gdy urządzenia produkowane przez konkurencję potrzebowały na to około dwóch minut. Po ośmiu latach (w roku 1958) firma miała już 51 milionów dolarów obrotu, 373 typy wyrobów i zatrudniała 1778 pracowników. Rok później nastąpił skok do Europy — w Genewie powstało biuro sprzedaży, w Boeblingen (Niemcy Zachodnie) pierwsza filia produkcyjna licząca 16 pracowników.



Początki firmy HP — niepozorny garaż na zapleczu drewnianego domku. W lewym górnym rogu — generator HP200B, pierwsze urządzenie produkowane seryjnie

Jednym z ważniejszych etapów rozwoju firmy było opracowanie w 1960 r. oscyloskopu do obserwacji przebiegów cyfrowych do zastosowania w technice komputerowej. Trzy lata później powstał pierwszy syntetyzer częstotliwości, używany w automatycznych systemach pomiarowych. W roku 1964 skonstruowano HP-5060A — generator wzorcowy o rewelacyjnej na owe czasy dokładności 10^{-6} s. Powstały również pierwsze urządzenia przeznaczone dla medycyny, bazujące na technice ultradźwiękowej. Nastąpił burzliwy rozwój firmy. W 1966 roku utworzono laboratorium HP. Powstał pierwszy komputer, HP-2116A. A oto kilka ważniejszych dat:

1967 r. — Urządzenie do monitorowania tonów serca noworodka podczas fazy porodu. Opracowane przez HP zegary atomowe instalowane są w charakterze wzorców czasu.

1968 r. — Pierwszy programowany kalkulator stołowy, HP-9100A.

1969 r. — HP oferuje swój pierwszy system operacyjny pozwalający na wielodostęp 16 użytkowników.

1971 r. — Prezentacja pierwszego mini-komputera programowanego przez użytkownika.

1972 r. — Pierwszy kalkulator inżynierski HP-35 powoduje istną rewolucję na rynku kalkulatorów.

1973 r. — Powstaje mini-komputer HP-3000.

1974 r. — Pierwszy mini-komputer posiadający pamięć operacyjną 4 KB.

1978 r. — HP prezentuje pierwsze programy na bazie sztucznej inteligencji.

1980 r. — Już trzy miliardy obrotu i 57 tysięcy zatrudnionych.

1981 r. — HP prezentuje pierwszą strukturę półprzewodnikową zawierającą 600 tysięcy tranzystorów wykonaną w technologii NMOS-II.

1982 r. — HP-9000 rozpoczyna erę 32-bitowych układów scalonych.

1983 r. — Prezentacja idei „Touch screen".

1984 r. — Powstaje pierwsza na świecie drukarka atramentowa — słynny HP-Inkjet.

1985 r. — Pierwsza drukarka laserowa HP Laserjet — mechanikę opracowano wspólnie z japońską firmą Canon.

1987 r. — Bill Hewlett odchodzi na emeryturę. Do grona dyrektorów wchodzą synowie założycieli firmy, Walter Hewlett i David Woodley Packard.

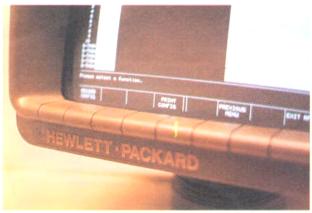
1988 r. — HP w magazynie „Fortune" znajduje się na liście 50 największych firm świata. Równo 10 miliardów dolarów obrotu, 87 tysięcy zatrudnionych, w tym 5500 w RFN.

Maj 1989 r. — Akcje przedsiębiorstwa są oficjalnie wprowadzone na giełdę we Frankfurcie.

Wydawałoby się, że koncern HP niemal przebojem toruje sobie miejsce na rynku. Tak jednak nie jest. Przez pierwsze dziesięciolecia receptą na sukces była metoda „inżynierowie budują dla inżynierów". Jej sens zawierał się w analogicznych potrzebach inżynierów HP i inżynierów innych firm — potencjalnych odbiorców. Przykładowo, jeżeli pewien zespół w HP potrzebował do nowych prac lepszy, doskonalszy oscyloskop, to z bardzo dużym prawdopodobieństwem urządzenie to było potrzebne również inżynierom w innych firmach. Ta myśl przewodnia szefów HP była skuteczna jednak tylko do pewnego momentu. Jej konsekwencją był fakt dużego rozdrobnienia firmy i brak odpowiedniej informacji

Jeden z mikrokomputerów HP z kilkuletnią już historią. Maszyny te, produkowane dobrych kilka lat temu, charakteryzowały się niezwykłą wprost dbałością o formę i ergonomię. Klawiatura zintegrowana z wygodnym biurkiem, interesujący pod względem kształtu monitor, oczywiście z możliwością przesuwania wzdłuż blatu. Po prawej ręce wygodnie dostępne są kieszenie napędów dyskowych





HP pierwsze wprowadziło tzw. *Touch screen* — specjalne klawisze o definiowanej funkcji zintegrowane z monitorem. W połączeniu z uproszczonym systemem okienek można było wygodnie tworzyć systemy prowadzące użytkownika, nie wymagające żmudnego przedzierania się przez kilka tomów dokumentacji. Dużym ułatwieniem jest standardowe wyposażenie każdego systemu HP w bazę danych

o produktach rozwijanych w innych firmach. W pewnym momencie zaczął być odczuwany brak profesjonalnego marketingu.

Inną ciekawostką z historii HP, jest fakt istnienia na początku lat osiemdziesiątych aż pięciu różnych rodzin komputerów HP, zupełnie ze sobą niezgodnych programowo z uwagi na różne systemy operacyjne. Wykorzystanie oprogramowania oferowanego przez wiele firm było również utrudnione. Wtedy ówczesny szef HP, John Young, postawił znaczną część posiadanych środków na jedną kartę: koncepcję architektury RISC (ang. *Reduced Instruction Set Computer*) — procesorów o zredukowanej liczbie prostych instrukcji, jednak wykonywanych bardzo szybko. Koncepcja ta była znana już wcześniej. Zajmowano się nią w IBM — jednak do tych prac nie przywiązywano zbyt wielkiej wagi. Young sprowadził od IBM jednego z najlepszych pracowników, fizyka Joela S. Birnbauma i rozpoczął szeroko zakrojone i bardzo żmudne prace nad nową koncepcją procesora. Dodatkowym utrudnieniem była konieczność zapewnienia pro-

Wygodne stanowisko pracy można zbudować także przy niezależnych monitorach. Podstawą jest odpowiednia konstrukcja biurka łączącego się (strona prawa) z jednostką centralną wraz ze stacjami dysków oraz (strona lewa) z drukarką. Oczywiście nie bez znaczenia jest odpowiednie oświetlenie miejsca pracy oraz oprawa plastyczna pomieszczenia programistów. Dodajmy jeszcze dla wyjaśnienia pewnych wątpliwości, że zdjęcie to nie zostało wykonane w kraju, choć być może i u nas ktoś już zaczyna dbać o estetykę miejsca pracy



gramowej zgodności z oprogramowaniem dla maszyn wcześniejszych. Kilkuletnie prace zostały w roku 1987 uwieńczone sukcesem. Nowy model HP-3000/950 znacznie przewyższał poprzednie pod względem wydajności (7 MIPS), zakresu zastosowań, a także poboru mocy. System ten jest obecnie oferowany w nowszej wersji 955 o wyższej wydajności (11 MIPS), można podłączyć do niego do 400 końcówek przy pracy wielodostępnej oraz wieloprogramowej.

Marka Hewlett-Packard kojarzy się nam na ogół z solidnością, niezawodnością, najwyższymi parametrami technicznymi oraz wysoką ceną. Cena ta jest jednak wynikiem zastosowania najnowocześniejszych, niezawodnych technologii wykonania. Dokładna kalkulacja udowadnia, że w wielu sytuacjach opłacalne jest zastosowanie sprzętu droższego, jednak bardziej niezawodnego.

Jak solidny jest sprzęt HP, wiemy bardzo dobrze. Jest on do tego stopnia trwały, że powstał obecnie rynek używanych urządzeń elektronicznych tej firmy. Pełnosprawny system, mający za sobą już kilka lat eksploatacji, można kupić (z gwarancją!) po cenie wyraźnie niższej od ceny porównywalnego systemu nowego, także innej marki. Oczywiście może się wydawać, że maszyny takie są już znacznie zużyte moralnie — pamiętajmy jednak, że wiele mechanizmów znanych nam choćby z popularnych IBM-ów zastosowanych było już wiele lat temu w komputerach HP. Przykładowo technika „Windows" zastosowana była w HP już w 1978 roku! To, co może być dla potencjalnych polskich użytkowników komputerów HP najważniejsze, to ochrona przed utratą i zniszczeniem danych. Jest to problem nękający wielu użytkowników mikrokomputerów klasy PC — krótkotrwały zanik napięcia w sieci może kosztować nas wiele godzin żmudnej pracy. Zresztą w niektórych zastosowaniach (np. w bankowości) bez odpowiednich mechanizmów ochrony danych w ogóle nie można stosować sprzętu komputerowego. Ten problem przy odpowiednim skompletowaniu maszyn HP nie występuje.

Na pewnym etapie komputeryzacji zakładów przemysłowych nieodzowne staje się połączenie poszczególnych komputerów w odpowiednią sieć — tylko wtedy osiągamy pełne możliwości wykorzystania systemu komputerowego oraz centralnego lub rozproszonego banku danych. Jednak w naszych warunkach opieramy się głównie na IBM-ach i bardzo drogich sieciach. Nie zawsze rozwiązanie takie jest optymalne, czasem też przy większej liczbie użytkowników jednostka centralna (ang. *server*) nie nadąża z przetwarzaniem danych. Alternatywą jest zastosowanie systemu wielodostępnego z wydajnym komputerem centralnym oraz tanimi terminalami. System taki jest bardzo wskazany ponadto tam, gdzie operujemy dużą ilością danych typu magazynowego. Poza tym często zdarza się, że któryś z użytkowników uruchomi na swoim komputerze grę z wirusem (czy jakiś inny „zarażony" program), w wyniku czego porażeniu ulega cała sieć. Problem ten znika przy zastosowaniu systemów z licencjonowanym oprogramowaniem.

Panowie Hewlett i Packard są dzisiaj już na zasłużonej emeryturze. Stworzony przez nich koncern rozwija się nadal, mimo pewnych zmian w filozofii firmy. Pozostał jednak w jej produkcji trwały akcent na niezawodność i solidność sprzętu oraz komfort pracy. Może zatem kiedyś i my skorzystamy z tych zasad, które innym przyniosły niezaprzeczalny sukces?



ROLAND WACŁAWEK

ZAGĘSZCZENIE WYDRUKU GRAFIKI

Większość programów produkujących jakiegokolwiek typu informacje graficzne przewiduje możliwość wydruku grafiki. Często okazuje się jednak, że na typowych drukarkach 9-igłowych jakość druku jest niezadowalająca, zwłaszcza w przypadku, gdy taśma barwiąca jest już nieco zużyta. Odpowiedzialność za to ponosi w największym stopniu mała rozdzielczość grafiki w pionie. Podczas gdy w poziomie można wybrać w typowej drukarce rozdzielczość 60, 120 lub 240 punktów na cal, to w pionie rozdzielczość jest dyktowana odstępem między igłami, który w drukarkach 9-igłowych wynosi zawsze 1/72 cala. Przy wydruku grafiki stosowana jest rozdzielczość pozioma przynajmniej 120 punktów/cal, co prowadzi do wniosku, że rozdzielczość pionowa jest ok. 1,7 raza gorsza. W efekcie, jeżeli tylko tasiemka barwiąca nie jest mocno nasączona tuszem, w wydruku wyraźnie widoczna jest liniowa struktura rysunku. Zjawisko to jest jeszcze bardziej dokuczliwe przy wydruku grafiki 9-igłowej na drukarce 24-igłowej w trybie emulacji 9 igieł. Z powodu cieńszych igiełek liniowa struktura jest jeszcze bardziej widoczna, a kontrast — gorszy.

Podwyższenie jakości wydruku grafiki 9-igłowej można osiągnąć, stosując program, który drukuje każdą linię wielokrotnie, dokonując przy każdym kolejnym przebiegu nieznacznego przesunięcia papieru, tak aby kolejne uderzenia igieł wypadały w innym miejscu, wypełniając mniej kontrastowe obszary. W praktyce zadowalające wyniki uzyskuje się przy jedno-, a co najwyżej dwukrotnym powtórzeniu. Minusem opisanej metody jest oczywiście wydłużenie czasu pracy drukarki, ale w zamian uzyskuje się wydruki o zatartej strukturze liniowej i znacznie bardziej kontrastowe, co jest widoczne zwłaszcza przy zużytej taśmie barwiącej.

Niniejszy artykuł-program zawiera szkic rozwiązania tego problemu. Zaprezentowany tu program został przewidziany do współpracy z graficznym środowiskiem MS-WINDOWS 1.03 i może obsługiwać praktycznie dowolną drukarkę 9-igłową standardu EPSON lub IBM-GP. Po ewentualnej, niewielkiej adaptacji program może współpracować z praktycznie dowolnym programem graficznym (przeróbka sprowadza się głównie do zaimplementowania interpretacji innych kodów sterujących). System WINDOWS został tutaj wybrany dlatego, że w trybie graficznym wysyła bardzo oszczędne dane, złożone jedynie z bloków grafiki, odpowiadających poszczególnym liniom, oraz z rozkazów przesunięcia papieru w przód o zadany odcinek. Rozkazy te realizowane są przez sekwencję znaków ⟨*ESC*⟩⟨*J*⟩⟨*parametr*⟩, gdzie *parametr* oznacza liczbę jednostek, o które należy przesunąć papier. Jednostka liczy 1/216 część cala.

```
                        Zaawansowane wydruku grafiki
;**** MULTIDU V1.0 Roland Waclawek, Sieminanowice 81. 1986 ****

Ecd_prog        SEGMENT

MS_DOS          EQU     21H     ;numer przerwania wywolujacego DOS
Drukarka        EQU     17H     ;numer przerwania obslugi drukarki
Czyta/Wekt      EQU     35H     ;MS-DOS - odczyt wektora przerwania
Ustaw_Wekt      EQU     25H     ;MS-DOS - zapis wektora przerwania
Poczatek        EQU     5TH     ;numer przerwania integracji progr.
Ecd_ESC         EQU     80H     ;kod specjalny
Ecd_SPACE       EQU     20H     ;kod spacji
Ecd_WTAB        EQU     09      ;kod tabulatora poziomego
CR              EQU     13      ;kod znaku konca wiersza (CR)
LF              EQU     10      ;kod znaku nowego wiersza (LF)

                ASSUME  CS:Ecd_prog, DS:Ecd_Prog

                ORG     1F9
Parametry       LABEL   BYTE    ;pole parametrow wywolania z  DOS

                ORG     256
Inicjacja:      JMP     Zainstaluj; przeskocz do procedury instalacji

Print_Adr       DW      0       ;offset pierwotnego handlera druk.
Print_Segm      DW      0       ;segment      pierwotnego   handlera
Status:         DB      0       ;bajt stanu programu MULTIDUK:
                                ;Bit 7:1- obsluga sekwencji  ESC\
                                ;Bit 6:1- obsluga argumentu <ESC>
                                ;Bit 5:1- ostatni  bajt  argumentu
                                ;Bit 4:1- bajt przedostatni argum.
                                ;Bit 3:1- obsluga wcisniecia  ESC\AH
                                ;Bit 0:1- aktywny  tryb  graficzny


Licznik:        DW      0       ;licznik bajtow trybu graficznego
Ecd_graf:       DW      0       ;offset kodu  operacji  graficznej
Il_rzes         DW      1       ;liczba powtorzen wydruku wiersza

Bajt&tat:       MOV     BYTE PTR Licznik,AL ;zapamietaj nowy!  AL
                TEST    BYTE PTR Status,00000100B ;czy to kod 1AH?
                JNZ     Obs_kAH         ;tak, to rozpoczyna wysyske
                TEST    BYTE PTR Status,1   ;czy aktywna grafika?
                JZ      Esc             ;nie  -  to  nie  grafika
                MOV     AL,Ecd_ESC      ;wypowiedz  znak  <ESC>
                CALL    Wyp_dodut       ;do  bufora  wiersza
                MOV     AL, BYTE PTR Ecd_graf ;wypowiedz rozkaz graf
                CALL    Wyp_dodut       ;do  bufora  wiersza
                MOV     AL,BYTE PTR Licznik   ;dlugosc bajt licznika
                CALL    Wyp_dodut       ;do  bufora  wiersza
                MOV     AL,BYTE PTR Licznik+1 ;starszy bajt licznika
                CALL    Wyp_dodut       ;do  bufora  wiersza
Esc:            AND     BYTE PTR Status,00000001B ;kasuj obsluge ESC
                IRET    Koniec_wyp

Obs_kAH:        PUSH    AX              ;zapisz dlugosc wyjsciu  na stos
                CALL    Druk_linii      ;wydruku akt. zawartosc bufora
                MOV     AL, Ecd_ESC     ;wypowiedz do drukarki znak ESC
                CALL    Wyp_druk        ;
                MOV     AL,'W'          ;wyslij do drukarki  rozkaz &AH
                CALL    Wyp_druk        ;
                POP     AX              ;odtworz w  AX  dlugosc  wyjscu
                SUB     AX,WORD PTR Il_rzes ;redukuj ilosc wydr. o liczbe
                                        ;powtorzen  wydruku wiersza
                JNZ     Esc             ;wyjsc  dane do drukarki
                AND     BYTE PTR Status,00000001B ;kasuj flagi ESC
                IRET    Koniec_wyp

Obslug_ESC      TEST    BYTE PTR Status,10H ;czy ust. bajt rezerwuj ESC?
                JNZ     Przedostat      ;nie  -trzeci z rzedu bajt jest ESC
                TEST    BYTE PTR Status,20H ;przedostatni bajt rezerwuj.?
                JNZ     Bajt&tat        ;tak, obsluga przedost. bajtu
                JMP     Koniec_ESC      ;nie,to obsluga ostatniego bajtu

Przedostat:     AND     BYTE PTR Status,11101111B ;kasuj wskaznik bitu 3
                MOV     BYTE PTR Licznik,AL ;zapamietaj jako mlodszy bajt
                JMP     Koniec_wyp      ;wyprowadz  do  drukarki  znak

Inicjuj:        MOV     CS:BYTE PTR Status, 0   ;skasuj wszystkie flagi
                MOV     CS:WORD PTR Licznik,0   ;skasuj  licznik bajtow
                MOV     CS:WORD PTR Wsk_bufora, OFFSET Bufor_in
                MOV     CS:WORD PTR Licz_bufora,0
                JMP     Do_BIOS

Wyjscznik:      DB      0               ;flaga  wyjsciowa  programu

                ASSUME  CS:Ecd_prog, DS:NOTHING     ;to po rozkazie INT 17H
Obs_17H:        TEST    BYTE PTR CS:Wyjscznik,OFFH;czy  ust  wyjsciowy?
                JNZ     Do_BIOS         ;tak, nalezy do wyjsc. dalej
                CMP     AH,1            ;nie - czy inicjalizacja drukarki?
                JZ      Inicjuj         ;tak - to  inicjalizacja  drukarki
                CMP     AH,0            ;czy funkcja drukowania znaku?
                JZ      Druku_Zn        ;tak    ;wtym wypowiadanie znaku
Do_BIOS:        JMP     CS:DWORD PTR Print_Adr ;nie, do standard. handlera

Druku_Zn:       TEST    CS:BYTE PTR Status,80H ;czy to trzy obslugi ESC?
                JZ      Tryb_Norm       ;nie -zwyczajna interpretacja
                PUSH    DS              ;przechowaj  wzywane  rejestry
                PUSH    SI              ;ten  fragment  interpretuje
                PUSH    BX              ;sekwencje znakow rozpoczete
                PUSH    CX              ;kodem sterujacym ESCAPE (1BH
                PUSH    CS              ;ustaw  segment  danych  stale
                POP     DS              ;ze segmentem kodu
                ASSUME  CS:Ecd_prog, DS:Ecd_Prog
                TEST    BYTE PTR Status,40H ;czy to pierwszy znak po ESC?
                JZ      Obslug_ESC      ;jesli nie, obsluga sekw. ESC
Pierwszy:       AND     BYTE PTR Status, 10111111B  ;kasuj bit argumentu
                MOV     BX,OFFSET Tab1_ESC ;adres tab. kodow sekw. ESC
                MOV     CL,(OFFSET Wsk_bufora-OFFSET Tab1_ESC)/3;il. elem.
Nastepny:       CMP     AL,[BX]         ;kod znaku szukany  w  wezorach?
                JE      Znalez_Kod      ;tak - zakonczs przeszukiwanie
                ADD     BX, 3           ;ustaw adres kolejnego wiersza
                DEC     CL              ;CL:=licznik elementow tablicy
                JNZ     Nastepny        ;przy nie koniec, szukaj dalej
Koniec_ESC      AND     BYTE PTR Status,00000001B ;kasuj flagi trybu ESC
                JMP     Koniec_wyp      ;wyjdz znak Z AL na drukarke

Znalez_Kod:     JMP     [BX+1]          ;skocz do adresu z tablicy

Gratika_&B      OR      BYTE PTR Status,00000001B ;ustaw  flage  grafiki
                MOV     BYTE PTR Ecd_graf, AL   ;przechowaj kod grafiki
Czytery_Bty     OR      BYTE PTR Status, 10110000B ;ustaw flagi trybu 4-B
                JMP     Koniec_wyp

Ecd_&AH:        OR      BYTE PTR Status, 00001000B ;ustaw flage kodu &AH
Trzy_Bajty      OR      BYTE PTR Status, 10100000B ;ustaw flagi trybu 3-B
                JMP     Koniec_wyp

Ecd_36H:        MOV     CS:WORD PTR Licznik,0   ;wyzeruj licznik znakow
                MOV     CS:WORD PTR Wsk_bufora, OFFSET Bufor_in
                MOV     AL,Ecd_ESC      ;wypisz znak do drukarki
                CALL    Wyp_druk        ;wypisz znak do drukarki
                MOV     AL,'6'          ;wypisz '6&' do drukarki
                CALL    Wyp_druk
                AND     BYTE PTR Status,00000001B ;kasuj flagi trybu ESC
                JMP     Koniec_wyp

                ASSUME  CS:Ecd_prog, DS:NOTHING
Tryb_Norm:      TEST    BYTE PTR Status,00000001B ;czy to tryb graficzny?
                JNZ     Tryb_Graf       ;tak    to bajt bloku grafiki
                CMP     AL,Ecd_ESC      ;czy  to  jest  znak  <ESC>?
                JE      Escape          ;jesli tak,  wlacz tryb ESC\
                CALL    Wyp_dodut       ;jesli nie, wyslij  znak  do
                JMP     Koniec_wyp      ;bufora wiersza

Wyprowadz:      JMP     DWORD PTR Print_Adr ;skocz do pierwotn handlera

Tryb_Graf:      DEC     WORD PTR Licznik ;odlicz  wypowiadany  bajt
                JNZ     Wyp_but         ;jesli nie ostatni wypowiadz
                AND     BYTE PTR Status, 11111110B ;skasuj bit grafiki
Wyp_but:        CALL    Wyp_dodut       ;wyslij znak do bufora wiersza
                JMP     Koniec_wyp

Wyp_dodut:      PUSH    BX              ;
                MOV     BX, WORD PTR CS:Wsk_bufora ;wskaznik bufora do BX
                INC     WORD PTR CS:Wsk_bufora ;przesun na nast. bajt
                MOV     [BX],AL         ;wpisz znak do bufora  wiersza
                INC     WORD PTR CS:Licznik ;zwieksz licznik znakow
                POP     BX              ;wpisz znak  do bufora
                RET

Escape:         MOV     BYTE PTR Status, 11000000B ;ustaw  bit  trybu ESC
                JMP     Koniec_wyp      ;wyslij znak ESC na drukarke

Druk_linii:     ASSUME  CS:Ecd_prog, DS:Ecd_Prog
                CALL    Emisja_1        ;wydruj  bufor  do  drukarki
                MOV     CX, WORD PTR Il_rzes ;liczba powtorzen do rej. CX
Petla_di:       PUSH    CX              ;zachowaj  licznik  na  stosie
                MOV     SI, OFFSET Wprowd_ot ;przesun papier o 1/216 cala
                CALL    Emisja_S
                CALL    Emisja_1i       ;wydruj  bufor  do  drukarki
                POP     CX              ;odtworz licznik powtorzeb do CX
                LOOP    Petla_di        ;ewentualnie  powtorz wydruk
Nie_drukuj      MOV     CS:WORD PTR Licznik,0 ;wyzeruj licznik bajtow buf
                MOV     CS:WORD PTR Wsk_bufora, OFFSET Bufor_in
                MOV     CS:WORD PTR Licz_bufora,0
                RET

Emisja_S:       MOV     CX, 3           ;program wysyla do drukarki trojke

Emisja_1:       MOV     BX,OFFSET Bufor_in ;adres bufora pocz. to powrot
                JCXZ    Emis_koniec     ;jesli bufor pusty, to powrot
Nast_bajt:      MOV     AL, BYTE PTR [BX] ;AL:=bajt  z  bufora  wiersza
                LODSB                   ;adr. do  kolejnej  znak z bufora
                PUSH    CX              ;zapamietaj  rejestry  na stosie
                PUSH    SI
                PUSH    BX
                MOV     AH,0            ;AH:=kod funkcji dr.  wysyla znaku
                PUSHF                   ;przygot. stos  dla  rozkazu IRET
                CALL    CS:DWORD PTR Print_Adr ;wywolaj pierwotny handler
                POP     BX              ;odtwrz rejestry zapisane na stos
                POP     SI
                POP     CX
                LOOP    Nast_bajt       ;powtarzaj  do  opr6nienia bufora
Emisjini        RET                     ;
                RET

Wypr_druk       MOV     AH,0            ;AH kod funkcji dr.  wysyla znaku
                PUSHF                   ;przygot. stos  dla  rozkazu IRET
                CALL    CS:DWORD PTR Print_Adr ;wywolaj pierwotny handler
                RET

Koniec_Wyp:     POP     CX              ;odtworz zawartosc  uzywanych rej.
                POP     BX
                POP     SI
                POP     DS
Koniepyp        MOV     AH,0            ;odczytaj  bieiacy status drukarki
                INT     17H             ;do  CWORD PTR Print_adr
                RET

Wprowd_ot       DB      1BH, 4AH, 1     ;sekwencja sterujaca ruchu o 1/216

Tab1_ESC        DB      '6'
                DW      OFFSET Ecd_36H
                DB      '3'
                DW      OFFSET Ecd_&AH
                DB      'K'
                DW      OFFSET Gratika_&B
                DB      'L'
                DW      OFFSET Gratika_&B
                DB      'Y'
                DW      OFFSET Gratika_&B
                DB      'Z'
                DW      OFFSET Gratika_&B

Licz_bufora:    DW      0               ;licz. bajtow wpisanych w bufor
Wsk_bufora:     DW      OFFSET Bufor_in ;wskaznik do akt. pozycji bufora
Bufor_in        DB      4000 DUP (0)    ;total bufor na wiadczy bufor

Zainstaluj:     MOV     SI, OFFSET Parametry ;offset parametrow wywolania
Instal_P:       LODSB                   ;nast. bajt bloku param do AL
                CMP     AL, CR          ;czy to koniec bloku parametrow?
                JNE     Kontyn_0        ;to moze  poz. opcje wywolania?
                JMP     Kontyn_1        ;nie ma juz wiecej zadnej opcji
```

```
Kontyn_O    CMP     AL,Kod_SPACE    ;czy biezacy znak  jest spacja?
            JE      Instal_T        ;jezeli  tak,   to  go  zignoruj
            CMP     AL, Kod_HTAB    ;czy to znak poziomej tabulacji?
            JE      Instal_F        ;jezeli  tak,   to  go  zignoruj
            CMP     AL,'S'          ;czy  jest  to  moze  cyfra  'S'
            JE      Trzy_razy       ;tak-przejscie na dwa powtorzenia
            PUSH    AX
            OR      AL, 20H         ;zamien ew. duza litere na mala
            CMP     AL,'s'          ;czy akt. znak byl litera 's'?
            JE      Test_wylacz     ;tak -sprawdz, czy to opcja 'sy'
            CMP     AL,'z'          ;czy akt. znak byl litera 'z'?
            JE      Test_zalacz     ;tak -sprawdz, czy to opcja 'za'
            CMP     AL,'x'          ;czy akt. znak byl litera 'x'?
            JE      Test_testu      ;tak -sprawdz, czy to opcja 'xt'
            POP     AX
Blad:       MOV     DX, OFFSET Blad_Opcji ;niezwany znak - wygeneruj
            JMP     Meld_Bledy      ;odpowiednik komunikat o bledzie
Trzy_razy:  MOV     WORD PTR Il_repe,2  ;zapisz liczbe powtorzen = 2
            JMP     Instal_F
Test_testu: MOV     CH,10000000B
            MOV     BL,'t'
            JMP     ZalWyl
Test_zalacz:MOV     DX, OFFSET Meld_Zalacz
            MOV     CH, 0
            MOV     BL, 'a'
            JMP     ZalWyl
Test_wylacz:MOV     DX, OFFSET Meld_Wylacz
            MOV     CH, OFFH
            MOV     BL, 'y'
ZalWyl:     POP     AX              ;odtworz pierwotny stan stosu
            LODSB                   ;zaladuj do AL  kolejny  znak
            CMP     AL, CR          ;czy to juz znak konca linii?
            JE      Blad            ;jezeli tak - sygnalizuj blad
            CMP     AL,SUB          ;przepsb  duza litere na mala
            CMP     BL, AL          ;test  zgodnosci  ze  wzorcem
            JNE     Blad            ;jezli rozne, sygnalizuj blad
            PUSH    CX
            PUSH    DX
            CALL    Test_Obecn      ;czy program juz rezyduje?
            POP     DX
            JCXZ    Istnieje        ;jezeli CX=0, rezyduje
            POP     CX
            MOV     DX,Offset Niecbecny;DX:offset meldunku
            MOV     AH, 9           ;9-funkcja DOS emisji meldunku
            INT     MS_DOS          ;wyprowadz komunikat na ekran
            MOV     AX, WORD PTR Print_Segm
            CMP     AX, OFFOOH      ;wektor wskazuje do BIOS?
            MOV     DX, OFFSET Meldbrk
            JNZ     Meld_Bledy
            MOV     DX, OFFSET MeldDisp
            JMP     Meld_Bledy
Istnieje:   POP     CX
            MOV     BX, WORD PTR Print_Segm
            MOV     ES, WORD PTR Print_Adr
            DEC     BX              ;BX:offset zmiennej  Wylacznik
            CMP     CH, 10000000B
            JZ      Testuj
            MOVB    PTR ES:[BX], CH
            JMP     Meld_Bledy
Testuj:     TEST    BYTE PTR ES:[BX],OFFH
            MOV     DX, OFFSET Stan_Sw
            JZ      Meld_Bledy
            MOV     DX, OFFSET Stan_wyl
            JMP     Meld_Bledy
Kontyn_I    CALL    Test_obecn      ;MULTIDRUK juz w pamieci?
            JCXZ    Instalacja      ;tak -nie wykonuj instalacji
Wektory:    MOV     DX, OFFSET Obs_17H
            MOV     AH,Ustaw_Wekt   ;przestawianie wektora prze-
            MOV     AL,Drukarka     ;rwania drukarki  nr  17H
            INT     MS_DOS          ;zapisz nowa wartosc wektora
            MOV     DX,Offset Meldunek
            MOV     AH,9
            INT     MS_DOS
            MOV     DX,OFFSET Zainstaluj
            INT     Prosatan        ;wyjdz program  rezydujacym
Juzistnieje:MOV     DX,Offset Komunikat
Meld_Bledy  MOV     AH, 9
            INT     MS_DOS
            INT     20H
Test_obecn: MOV     AH,Czyta_Wekt
            MOV     AL,Drukarka
            INT     MS_DOS          ;adr do ES:BX  wektor  17H
            MOV     WORD PTR Print_Adr,BX;przesun.w segment i offset
            MOV     WORD PTR Print_Segm,ES;tego wektora przerwania
            MOV     CX, 15          ;porownaj 15 bajtow  programu
            MOV     SI, OFFSET Obs_17H ;bajtow program. z akt. za-
            MOV     DI, BX          ;wartoscia pam.  operacyjnej
            REPE    CMPSW           ;porownuj az  do konca bloku
            RET                     ;lub stwierdz  niezgodnosci
Komunikat:  DB CR,LF,7
            DB 'MULTIDRUK juz jest zainstalowany!',CR,LF,'$'
Niecbecny:  DB CR,LF,7
            DB 'MULTIDRUK jeszcze nie zainstalowany!'
MeldFckr:   DB CR,LF,'albo przestal byc dostepny!',CR,LF
MeldDisp:   DB '!',CR,LF,'$'
Stan_zal:   DB CR,LF,'MULTIDRUK jest aktywny!',CR,LF,'$'
Stan_wyl:   DB CR,LF,'MULTIDRUK nie jest aktywny',CR,LF,'$'
Meld_zalacz DB CR,LF,'MULTIDRUK zostal uaktywniony!',CR,LF,'$'
Meld_wylacz DB CR,LF,'MULTIDRUK zostal wylaczony',CR,LF,'$'
Blad_Opcji: DB CR,LF,7
            DB 'W skreconiu wystapila bledna opcja!',CR,LF,'$'
Meldunek:   DB CR,LF,'Program zainstalowany w pamieci',CR,LF
            DB '$'

Kod_Prog    ENDS
            END     Inicjacja
```

Aby powtórzyć wydruk linii, musimy przechwycić i przechować w pamięci kompletną linię grafiki. Zakładając maksymalną gęstość wydruku 240 znaków na cal i drukarkę o długości wałka 15 cali uzyskujemy minimalną pojemność bufora 3600 bajtów. Dodając 10% rezerwy, potrzebujemy bufora liczącego 4000 bajtów. Nasz program zostanie zainstalowany w pamięci, przejmie przerwanie programowe obsługi drukarki i będzie śledzić wysyłane do drukarki znaki, wychwytując istotne kody sterujące. W naszym modelowym przykładzie będą trzy grupy kodów: rozkaz ⟨ESC⟩⟨6⟩, wywołany przez **WINDOWS** na początku wydruku każdego dokumentu, wspomniany już rozkaz ⟨ESC⟩⟨J⟩, wywołujący przesuw papieru oraz rodzina rozkazów graficznych ⟨ESC⟩⟨K⟩, ⟨ESC⟩⟨L⟩, ⟨ESC⟩⟨Y⟩ i ⟨ESC⟩⟨Z⟩, zwiastujących następujący po nich ciąg bajtów graficznego obrazu. Po każdej z tych sekwencji następują dwa bajty, określające długość wysyłanego za nimi bloku graficznego. Rozkaz ⟨ESC⟩⟨6⟩ wykorzystamy do inicjacji programu, natomiast ⟨ESC⟩⟨J⟩ będzie sygnałem do opróżnienia bufora.

Program przechwytuje wszystkie wysyłane do drukarki znaki i pakuje je do bufora, wyszukując jednak zarazem wymienione sekwencje sterujące. Wyjątkiem jest przypadek, gdy wykryto rozkaz graficzny. Cały następujący po nim blok bajtów jest wysyłany do bufora bez żadnej analizy. W bloku graficznym mogą bowiem wystąpić także ciągi bajtów identyczne z poszukiwanymi przez nas sekwencjami sterującymi, co spowodowałoby ich niepoprawną interpretację i w efekcie — awarię programu.

W chwili wykrycia sekwencji ⟨ESC⟩⟨J⟩ program zapamiętuje jego parametr, po czym zaczyna wysyłać bufor do drukarki. Następnie jest przesuwany o 1/216 cala i bufor zostanie wysłany ponownie. Jeżeli program znajdował się w trybie podwójnego powtórzenia, nastąpi jeszcze jeden ruch o 2/16 cala i trzeci wydruk zawartości bufora. Na koniec program odejmie od parametru pierwotnego rozkazu ⟨ESC⟩⟨J⟩ liczbę wykonanych, jednostkowych przesuwów (1 lub 2), po czym zrealizuje pozostałą część przesuwu wysyłając rozkaz ⟨ESC⟩⟨J⟩ ze skorygowanym parametrem.

Program MULTIDRUK został wyposażony w udogodnienia instalacyjne. Po pierwsze, ponowna próba wywołania programu nie prowadzi do jego ponownej instalacji — chyba że w międzyczasie zainstalowaliśmy inny program obsługujący przerwanie drukarki. Wykrywanie obecności programu odbywa się w ten sposób, że odczytujemy bieżący wektor przerwania drukarki i porównujemy wskazywany przezeń obszar pamięci z zawartością programu MULTIDRUK. Jeśli jest identyczna, znaczy to, że program rezyduje już w pamięci.

Po drugie, wywołując program możemy podać opcję **WY**, **ZA** lub **ST**. Zlecenie: **MULTIDRU WY** powoduje czasowe wyłączenie programu (program rezyduje nadal w pamięci, ale nie buforuje danych, tylko od razu przesyła je do drukarki). Po zleceniu: **MULTIDRU ZA** program jest ponownie uaktywniany, natomiast zlecenie: **MULTIDRU ST** podaje bieżący status programu: czy jest on aktywny, czy też nie. Jest jeszcze jedna opcja: 3, której można użyć przy pierwszym wywołaniu (**MULTIDRU 3**). Standardowo każda linia jest drukowana w dwóch przebiegach, a jeśli użyto opcji 3 — w trzech przebiegach.

Program MULTIDRUK jest przystosowany do pracy w formacie .COM. Należy więc najpierw poddać go asemblacji:

MASM MULTIDRU;

następnie konsolidacji, wytwarzając wersję .EXE:

LINK MULTIDRU;

i przetworzyć na format .COM:

EXE2BIN MULTIDRU. EXE MULTIDRU. COM

Na koniec chciałbym zaznaczyć szkieletowy charakter programu i jego pierwotne przeznaczenie do współpracy z systemem WINDOWS. Program w przedstawionej wersji nie będzie dobrze współpracować np. z tymi programami, które jednorazowo zmieniają standardowy odstęp między wierszami, a potem kończą każdą linię sekwencją ⟨CR⟩, ⟨LF⟩. Wprowadzenie tych modyfikacji w przedstawionym programie jest naprawę bardzo proste, ale nie chciałbym psuć przyjemności tym, którzy zdecydują się na samodzielną przeróbkę. Jeszcze tylko jedna wskazówka. Przygotowując program MULTIDRUK do współpracy z wybranym programem graficznym warto zacząć od wysłania próbnego wydruku do pliku dyskowego, a następnie przeprowadzić analizę zawartych w nim kodów sterujących programem PCTOOLS lub podobnym.



TURBO-DODATKI

Tworzenie profesjonalnych programów użytkowych często przypomina wyważanie otwartych drzwi — program musi zawierać wiele typowych funkcji, realizowanych zwykle od podstaw przy każdym nowym produkcie. Przykładami takich funkcji mogą być: edycja tekstu (każdy program wymaga wprowadzania danych…), tworzenie grafiki na ekranie czy też zarządzanie okienkami. Każda poważna firma software'owa posiada zwykle bibliotekę tego typu podprogramów — indywidualni programiści muszą jednak robić wszystko od początku.

Firma Borland International postawiła sobie za cel ułatwienie życia twórcom specjalistycznego oprogramowania, oferując pakiety procedur narzędziowych do kompilatorów **Turbo Pascala** i **Turbo BASIC-a**. Procedury zawarte w tych pakietach są dostarczane zarówno w postaci binarnej, jak i źródłowej. Można wbudować je bezpośrednio do tworzonych programów, uzyskując w prosty sposób wyrafinowane możliwości.

Oferowane obecnie pakiety biblioteczne dla Turbo Pascala to **Database Toolbox**, **Editor Toolbox**, **Graphix Toolbox**, **Game-Works** i **Numerical Method Toolbox**. Kompilator Turbo BASIC-a można natomiast wesprzeć pakietami **Telecom Toolbox**, **Database Toolbox** i **Editor Toolbox**.

Pakiet **Numerical Methods Toolbox**, adresowany do naukowców i inżynierów, pozwala wyznaczać miejsca zerowe funkcji, całkować, różniczkować, odwracać macierze i obliczać ich wartości własne, rozwiązywać równania różniczkowe i wyznaczać transformaty Fouriera. Wbudowane funkcje graficzne pozwalają w efektowny sposób zilustrować wyniki obliczeń.

Telecom Toolbox daje programom w Turbo BASIC-u szerokie możliwości telekomunikacyjne: możliwe jest sterowanie modemem (odpowiedniki większości komend XTalk), realizacja protokołu XMODEM, automatyczne wywoływanie według wbudowanej książki telefonicznej, transmisja z szybkościami 300, 1200 i 2400 bodów, automatyczny wydruk odebranego tekstu lub umieszczenie go na dysku i emulacja terminala VT52. Wszystko to w typowym dla Turbo BASIC-a środowisku okienek i rozwijanych menu.

Database Toolbox składa się z trzech modułów: **Trainer**, **Turbo Access** i **Turbo Sport**. Trainer służy do wprowadzenia w tematykę zarządzania zbiorami danych — trenuje tworzenie i działanie drzewiastej, indeksowanej struktury rekordów. Pozostałe dwa moduły uwalniają programistę od problemów związanych z wyszukiwaniem, dopisywaniem i kasowaniem rekordów z bazy danych, sortowaniem danych (według jednego lub wielu kluczy) i zarządzaniem pamięcią wirtualną przy operowaniu wielkimi zbiorami.

Funkcje zawarte w pakiecie **Editor Toolbox** pozwalają łatwo skonstruować własny edytor z pełnymi możliwościami obróbki tekstów, bezpośredniego dostępu do pamięci ekranu, zarządzania okienkami i rozwijanymi menu. Pakiet zawiera ponadto dwa przykładowe programy kompletnych edytorów: bloku edycji gotowego do wbudowania w program użytkowy i procesora tekstowego **MicroStar**.

Zasada wyposażania kompilatorów w pakiety procedur narzędziowych obowiązywała także w przypadku **Turbo Prologu**. Kompleksowe tworzenie baz wiedzy i systemów eksportowych w tym języku zostało wsparte pakietem **Turbo Prolog Toolbox** — najwszechstronniejszym z oferowanych przez Borland „dodatków". Pakiet umożliwia tworzenie popularnych form zobrazowania graficznego: wykresów słupkowych, kołowych, skalowanych wykresów. Wspiera komunikację według protokołu XMODEM, potrafi przetwarzać dane z pakietów Lotus 1-2-3, Symphony, dBASE III i Reflex, pozwala konstruować własne interfejsy graficzne z użytkownikiem i własny język zapytań (ang. *query language*). Pakiet, zawiera ponadto 40 przykładowych programów i liczy łącznie 8000 linii tekstu źródłowego.

Wszystkie produkty firmy Borland są oferowane w jednakowej cenie 99,95 dolarów za pakiet. Duże walory użytkowe sprawiają, że sam tylko Turbo Pascal i jego pakiety narzędziowe mają obecnie ponad 600 000 zarejestrowanych użytkowników.

Janusz Wrześniak

MYSZKA BEZ OGONA

Przy posługiwaniu się popularną myszką często zdarza się zaczepić jej przewodem o jakiś wystający przedmiot. Przewód myszki plącze się po blacie stołu, krępując ruchy. Kiedy indziej przewód myszki okazuje się za krótki do umieszczenia jej w najwygodniejszym miejscu. Kłopoty z przewodem są jednak nie znane użytkownikom myszki marki BMC, komunikującej się z komputerem bezprzewodowo, za pomocą promieni podczerwonych. Przewodem połączony jest natomiast z komputerem specjalny człon komunikacyjny, który trzeba umieścić w taki sposób, aby istniała łączność optyczna między nim a myszką. Niewyjaśniony pozostaje natomiast problem, co robić, gdy obok siebie pracuje kilka komputerów wyposażonych w bezprzewodowe myszy… Na zdjęciu: myszka BMC wraz z członem komunikacyjnym na podczerwień.

(R. W.)



KRZEMOWE SUPERMÓZGI

KRZYSZTOF WIŚNIEWSKI

Targi komputerowe Comdex-89 rozwiązały nareszcie języki przedstawicielom firmy Intel — pierwszy raz oficjalnie przedstawiono nowego „króla" w rodzinie procesorów wywodzących się od tak szacownego przodka, jakim był mikroprocesor Intel 8080. Kto by pomyślał, że to dopiero od kilkunastu lat znane są mikroprocesory — układy, które wzięły swój początek od nieudanego — albo raczej zbyt wolnego — sterownika obrazowego. Przeszły one potem przez fazę kalkulatorową i osiągnęły moc obliczeniową dużych komputerów z okresu swoich narodzin. Dzisiaj, dzięki ich popularności, wydaje się, że były one znane już od pradziejów.

Można powiedzieć spokojnie, że właściwie cała impreza w Chicago stała pod znakiem Intela. Jedną tylko niespodziankę sprawiła firma Tandon, wystawiając pomimo intelowskiego tzw. *Non-Disklosure* (czyli zakazu oficjalnego pokazu) nowy model komputera Tandon 386/33 z nową wersją procesora 80386/33 MHz na hanowerskich targach CeBIT-89. Firma sprytnie wybrnęła z opresji, tłumacząc się, że nowy produkt wyposażony jest nie w najnowszą odmianę procesora 386, ale w selekcjonowane egzemplarze wersji 25 MHz. Pomimo to stało się publiczną tajemnicą, że za miesiąc na targach Comdex taka wersja oficjalnie zostanie pokazana. Sprawiło to zresztą sporo „uciechy" przedstawicielom firm europejskich, trochę zaskoczonych takim traktowaniem ich rynku przez firmy amerykańskie. Dotychczas nowe opracowania trzymano zazdrośnie pod kluczem aż do targów w Chicago.

Właściwie w Europie „dano na pożarcie" tylko nowy supermikroprocesor o 64-bitowym słowie i architekturze RISC (ang. *Reduced Instruction Set Computer*) — Intel 80860. Jest to wspaniała maszyna, która przy częstotliwości zegara 40 MHz osiąga wydajność 40 MIPS, a przy pracy z koprocesorem arytmetycznym — aż 80 MFLOPS. Intel 80860 przeznaczony jest w zasadzie dla naukowych stanowisk pracy, gdzie głównym zadaniem komputera są bardzo skomplikowane obliczenia matematyczne oraz do stanowisk CAD i CAM.

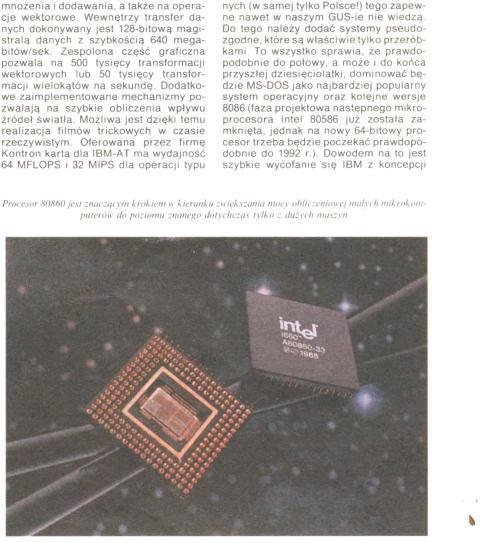
Jednocześnie z prezentacją samego mikroprocesora przedstawiono gotową kartę do komputerów osobistych zgodnych ze standardem IBM. We wnętrzu nowego mikroprocesora upakowano trzykrotnie więcej tranzystorów niż w 80386 i praktycznie tyle samo co w nowym 80486, czyli ponad 1,2 miliona. Wszystkie rozkazy wykonywane są, dzięki czteropoziomowemu „rurociągowi" (ang. *pipeline*), w jednym cyklu zegarowym. Część zmiennoprzecinkowa odpowiada normie IEEE-754. Operacje stałoprzecinkowe wykonywane są z dokładnością 32 lub 64 bity. Specjalne instrukcje pozwalają na równoległe wykonywanie mnożenia i dodawania, a także na operacje wektorowe. Wewnętrzny transfer danych dokonywany jest 128-bitową magistralą danych z szybkością 640 megabitów/sek. Zespolona część graficzna pozwala na 500 tysięcy transformacji wektorowych lub 50 tysięcy transformacji wielokątów na sekundę. Dodatkowe zaimplementowane mechanizmy pozwalają na szybkie obliczenia wpływu źródeł światła. Możliwa jest dzięki temu realizacja filmów trickowych w czasie rzeczywistym. Oferowana przez firmę Kontron karta dla IBM-AT ma wydajność 64 MFLOPS i 32 MIPS dla operacji typu *Integer* (tj. na liczbach całkowitych). Zainstalowanie dwóch równolegle pracujących kart pozwala na osiągnięcie wydajności 140 MFLOPS, czyli mocy całkiem dorosłego komputera (często nazywanego nawet superkomputerem) Cray-1. Cena proponowana przez Kontron dla wersji 32 MHz wynosi 20 000 DM, co przy 60-krotnym przyspieszeniu np. operacji graficznych w stosunku do 80386/80387 (20 MHz) nie jest wygórowanym żądaniem.

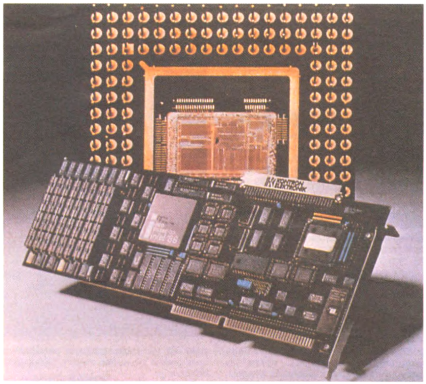
Pomimo tych i jeszcze innych zalet 80860 pozostanie jeszcze przez najbliższe lata w cieniu wielkiego brata z rodziny CISC i jego kolejnych wersji. Decyduje o tym popularność mikrokomputerów firmy IBM, a także oprogramowania. Nie możemy przecież zapominać o oficjalnie działających 10 milionach egzemplarzy systemu MS/PC-DOS (oraz podobnej liczbie systemów komputerowych). A ile jest bezprawnie kradzionych i powielanych (w samej tylko Polsce!) tego zapewne nawet w naszym GUS-ie nie wiedzą. Do tego należy dodać systemy pseudozgodne, które są właściwie tylko przeróbkami. To wszystko sprawia, że prawdopodobnie do końca przyszłej dziesięciolatki, dominować będzie MS-DOS jako najbardziej popularny system operacyjny oraz kolejne wersje 8086 (faza projektowa następnego mikroprocesora Intel 80586 już została zamknięta, jednak na nowy 64-bitowy procesor trzeba będzie poczekać prawdopodobnie do 1992 r.). Dowodem na to jest szybkie wycofanie się IBM z koncepcji

Nowa, 33 MHz wersja „starego" procesora Intel 80386



Procesor 80860 jest znaczącym krokiem w kierunku zwiększania mocy obliczeniowej małych mikrokomputerów do poziomu znanego dotychczas tylko z dużych maszyn





Równocześnie z prezentacją nowego procesora 80860 zaprezentowano gotową kartę do IBM AT — po wyposażeniu komputera w tę superszybką jednostkę centralną znacznie wzrasta wydajność maszyny

wprowadzenia razem z rodziną komputerów PS-2 systemu UNIX. Szybko zorientowano się, że pod sztandarami UNIX-a zbyt dużo się nie zdobędzie i powrócono do starego, ale popularnego DOS-a. Nawet IBM stał się częściowo niewolnikiem własnych standardów i, co nas nieco zaskoczyło, w przeciwieństwie do takich firm jak NCR, Compaq, AST, Hewlett Packard czy Olivetti, nie deklarował już na Comdexie swojej gotowości do podjęcia produkcji komputerów wyposażonych w nowy procesor. Chyba dopiero na jesiennym pokazie przedstawi już gotową wersję z rodziny PS-2 — model 100.

Na wiosennym Comdex-89 Intel w asyście szefów największych firm komputerowych: Cannavino (IBM), Parett (HP), Canion (Compaq), Cassoni (Olivetti), Gates (Microsoft), Kahn (Borland), King (Lotus), McGuinn (AT&T) ogłosił trzy ciekawe informacje. Pierwsza to ogłoszenie śmierci rynkowej procesora 80286 i zastąpienie go nowym 80SX386 zgodnym programowo z 80386, ale z 16-bitową magistralą (jak w 80286). W ten sposób Intel zamierza zdobyć nasycony w sporej części rynek komputerów poniżej 4—5 tys. DM, oraz zwiększy zapotrzebowanie na oprogramowanie dla mikroprocesorów 32-bitowych.

Drugą ciekawostką była oficjalna prezentacja 33 MHz wersji 80386DX wraz z wszystkimi elementami peryferyjnymi. Są to: koprocesor arytmetyczny 387DX, Cache-kontroler 82385, koprocesor LAN 82596DX, MCA-Chipset 82320 oraz kontroler magistrali EISA 82350.

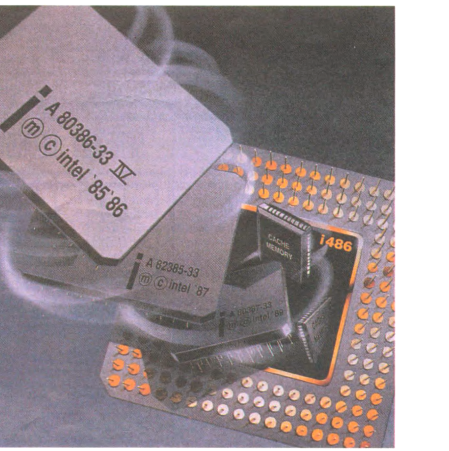
I wreszcie trzecia nowość — pierwszy pokaz całkiem nowego procesora Intel 80486. Z charakterystyką nowego procesora można się dokładnie zapoznać w liczącym aż 176 stronic materiale informacyjnym, zapełnionym drobnym maczkiem danych i parametrów Intela 80486. Podstawowe cechy procesora przedstawionego w dwa i pół roku po 80386 to: zgodna z 80386 część procesorowa, zgodna z 80387 część koprocesora, Cache--Controller, 8 KB pamięci Cache.

Niestety nie spełniły się oczekiwania umieszczenia w jednej strukturze również kontrolera DMA. Nie znaleziono już wolnego miejsca dla niego, na liczącej 2 cm^2 powierzchni zajmowanej przez pozostałe części systemu. Nowy procesor pracuje oczywiście w każdym ze znanych trybów pracy swoich poprzedników, czyli:

— Real Mode 8086/8088,
— Protected Mode 80286 z jego mechanizmami zabezpieczania danych przy pracy typu Multitasking,
— Native Mode i Virtual Mode 80386.

Zakres adresowania pamięci pozostał nie zmieniony i tak jak poprzednik może on bezpośrednio adresować 4 gigabity, a w trybie wirtualnym — 64 terabity przy nie zmienionej 32-bitowej szerokości magistrali adresowej. Ważną zmianą w stosunku do starych modeli jest nowy tryb pracy wieloprocesorowej. Pozwoli on na istotne zwiększenie wydajności pracy, tym bardziej że mechanizmy pracy wieloprocesorowej są już dosyć dobrze poznane i na rynku dostępne są odpowiednie wersje kompilatorów C, Fortranu czy Pascala. Od strony progra-

Kilka elementów z nowej, bardzo szybkiej serii Intela 803XX-33





A to nowy, bardzo wydajny mikroprocesor 80486

mowej dodano szereg nowych rozkazów. Trzy dotyczą operacji na rejestrach, a pozostałe operacji z pamięcią typu Cache.

Najistotniejsze zmiany dotyczą jednak nie właściwości natury programowej, ale zmian technologicznych i konstrukcyjnych. Podobnie jak w przypadku 64-bitowego 80860, zastosowano technologię 1-Mikron-CHMOS-IV, pozwalającą na upakowanie 1,2 miliona tranzystorów na powierzchni jednej struktury (powierzchnia około 2 cm^2). Zdecydowano się na umieszczenie koprocesora arytmetycznego razem z podstawowym procesorem prawdopodobnie pod naciskiem środowiska programistów. Dotychczas niewiele maszyn posiadało dodatkowo koprocesor, który jest bardzo atrakcyjny i interesujący dla programistów. Część koprocesorowa Intela 80486 potrafi wykonać te same operacje matematyczne 100 razy szybciej niż sam procesor wspomagany programowo. Jest jednocześnie od 4 do 6 razy szybszy od 80387 przy takiej samej częstotliwości zegarowej. W tabeli 1 mamy porównanie wymaganej ilości cykli zegarowych dla różnych rozkazów 80387 i 80486 (części koprocesorowej). W większości przypadków obydwa procesory Intela 80486 pracują jednocześnie w trybie równoległym, i sterowane są przez tę samą pamięć mikrokodu i sekwencer. Tylko jeden bit mikrokodu decyduje, czy wykonywana będzie operacja całkowitoliczbowa, czy zmiennoprzecinkowa. Przyspiesza to wymianę danych między procesorami i upraszcza konstrukcję.

Przyglądając się powierzchni układu scalonego, od razu można zauważyć, że spory obszar zajmowany jest przez kontroler Cache i statyczną, bardzo szybką pamięć SRAM-CACHE. Zastosowanie tego kontrolera pozwala efektywnie wykorzystać bardzo dużą szybkość jednostki centralnej i jednocześnie zwalnia programistę od myślenia o odpowiednim oprogramowaniu. Kontroler jest dla użytkownika „przezroczysty" i sam bezbłędnie modyfikuje kody programu, dostosowując je do własnych potrzeb (czyli optymalnego umieszczenia programu i danych w pamięci). Programista może tylko za pomocą dwóch rozkazów ingerować w pracę kontrolera Cache. Są to: INVD — unieważniający dane w pamięci Cache i WBINVD — wpisujący najpierw dane do pamięci operacyjnej i unieważniający następnie dane w pamięci Cache.

Jak możemy zauważyć w tabeli 2, sama część arytmetyczna i interfejsu wejścia-wyjścia ma istotnie większą wydajność. Większość operacji wymaga tylko jednego cyklu zegarowego, podobnie jak w procesorach RISC. Tak istotne przyspieszenie (prawie trzykrotne) uzyskano dzięki zastosowaniu kilku trików sprzętowych.

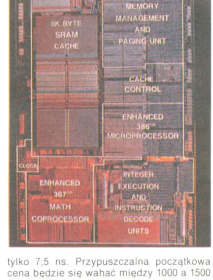
Pierwszym jest aż pięciostopniowy tzw. *pipelining* pozwalający jednocześnie przetwarzać pięć rozkazów w różnych stadiach interpretacji. Następnie zastosowanie techniki RISC w architekturze wewnętrznej przy jednoczesnej pewnej zgodności z zewnętrznym otoczeniem zgodnym z rodziną 8086. Szerokość wewnętrznej magistrali wynosi 128 bitów i pozwala na bardzo szybki transfer danych pomiędzy jednostką centralną, koprocesorem i pamięcią Cache. Na dodatek, jak już wspomniano, w bardzo wielu przypadkach procesor i jednostka arytmetyczna mogą działać w systemie wieloprocesorowym.

Zadbano też o łatwe rozpoznawanie przez system operacyjny i oprogramowanie, z którym procesorem mają do czynienia: otóż po sygnale RESET rejestr DH przyjmuje wartość 4. Wszystkie te zmiany pozwalają uzyskać przy częstotliwości zegara 33 MHz aż pięćdziesięciokrotne przyspieszenie pracy w stosunku do klasycznego IBM XT z procesorem 8088 i taktem zegarowym 4,77 MHz.

Wszystkie te nowości zostały zamknięte w obudowie z 168 wyprowadzeniami (poprzednik ma tylko 132 „nogi") i będą dostępne w normalnej sprzedaży w czwartym kwartale tego roku w wersji 25 MHz. W pierwszym kwartale 1990 r. sprzedawana będzie w ilościach komercyjnych wersja 33,33 MHz. Odmiana 40 MHz jest obecnie przygotowywana do produkcji.

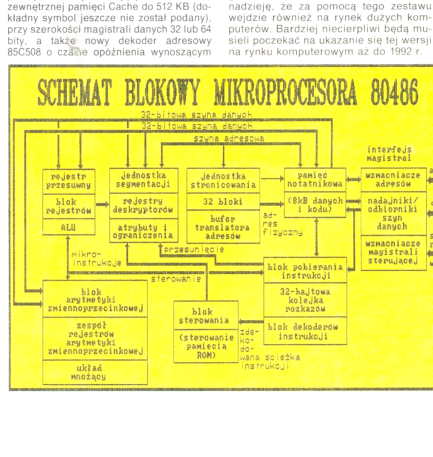
Jednocześnie z prezentacją nowego procesora Intel przedstawił gotową płytę główną do IBM AT oraz cały zestaw dodatkowych podzespołów. Do najważniejszych należą: kontroler DMA typu 82840, koprocesor sieci Ethernet typu 82596, zewnętrzny kontroler pamięci Cache pozwalający na rozbudowę zewnętrznej pamięci Cache do 512 KB (dokładny symbol jeszcze nie został podany), przy szerokości magistrali danych 32 lub 64 bity, a także nowy dekoder adresowy 85C508 o czasie opóźnienia wynoszącym

Wewnętrzna struktura nowego mikroprocesora pozwala na znacznie szybsze wykonywanie operacji niż modele poprzednie



tylko 7,5 ns. Przypuszczalna początkowa cena będzie się wahać między 1000 a 1500 dolarów.

Całkiem już futurystycznie dla normalnego użytkownika programów brzmi doniesienie o wersji 80486 wykonanej w technice ECL o wydajności 100 Mips. Intel dosłownie w ostatniej chwili przed targami w Chicago ukończył wspólnie z firmą Prime Computer opracowanie nowego modelu w najszybszej, jak dotąd, technice cyfrowej Emitter Coupled Logic (ECL). Całość mieści się na płytce o wymiarach 25 × 20 cm i sterowana jest zegarem o częstotliwości 150 MHz. Intel ma nadzieję, że za pomocą tego zestawu wejdzie również na rynek dużych komputerów. Bardziej niecierpliwi będą musieli poczekać na ukazanie się tej wersji na rynku komputerowym aż do 1992 r.



INTERFEJS — BUFOR DO ZX SPECTRUM

Dariusz Adam Przygoda

Mikrokomputer ZX Spectrum nie jest bogato wyposażony w urządzenia peryferyjne. Projektowany był jako urządzenie popularne, a co za tym idzie — tanie. Narzuciło to konstruktorom firmy Sinclair Research Ltd. wymóg stworzenia konstrukcji o możliwie małej ilości elementów. Konsekwencją tych założeń było zredukowanie do minimum wmontowanych w komputer urządzeń wejścia/wyjścia. Są one zintegrowane w specjalizowanym układzie scalonym ULA i realizują następujące funkcje:

- obsługę obszaru pamięci ekranu (treść) i koloru obrzeża,
- obsługę klawiatury,
- obsługę procesów komunikacji z układami zewnętrznej pamięci taśmowej (magnetofon),
- obsługę głośniczka komputera.

Wszystkie te funkcje (za wyjątkiem procesów obsługi pamięci ekranu) realizowane są przez dostęp do pól adresowych obszaru adresów urządzeń peryferyjnych (portów) mikroprocesora Z80. Mówiąc dokładniej, odwołanie się do jakiegokolwiek portu o adresie parzystym spowoduje:

- w przypadku użycia instrukcji IN — odczytanie danych o aktualnym stanie klawiatury i poziomu wejścia EAR,
- w przypadku użycia instrukcji OUT — w zależności od stanu poszczególnych bloków wysłanego bajtu: ustawienie koloru obrzeża ekranu i (z pewnymi uproszczeniami) ustawienie poziomów sygnałów sterujących głośniczek i wyjście MIC komputera.

Wykorzystanie połowy obszaru adresowego układów wejścia/wyjścia dla potrzeb, do pokrycia których w zupełności wystarczyłoby osiem adresów (przy sposobie adresowania portów przez mikroprocesor Z80 oznacza to 32760 „zmarnowanych" lokacji) spowodowane jest przyjęciem konwencji maksymalnych uproszczeń; do jednoznacznego uaktywnienia układu ULA wystarcza, aby bit A_0 szyny adresowej przyjął wartość zera logicznego.

Więcej szczegółów na temat organizacji wewnętrznych portów mikrokomputera ZX Spectrum można znaleźć w pozycji *ZX Spectrum BASIC Programming* autorstwa Stevena Vickersa (była ona dołączana jako instrukcja obsługi do pierwszych wersji komputera).

Jak już było powiedziane, przyjęcie takiego rozwiązania pozwoliło na znaczne obniżenie ceny urządzenia, ale stało się to kosztem prostoty dołączania urządzeń peryferyjnych. Niestety, każde urządzenie peryferyjne (drukarka, manipulatory, stacje dysków, urządzenia wykonawcze i pomiarowe) muszą być dołączane do ZX Spectrum poprzez interfejsy umożliwiające prawidłową współpracę, które separują wyprowadzone na złącze krawędziowe szyny: danych, adresową i sterującą mikroprocesora.

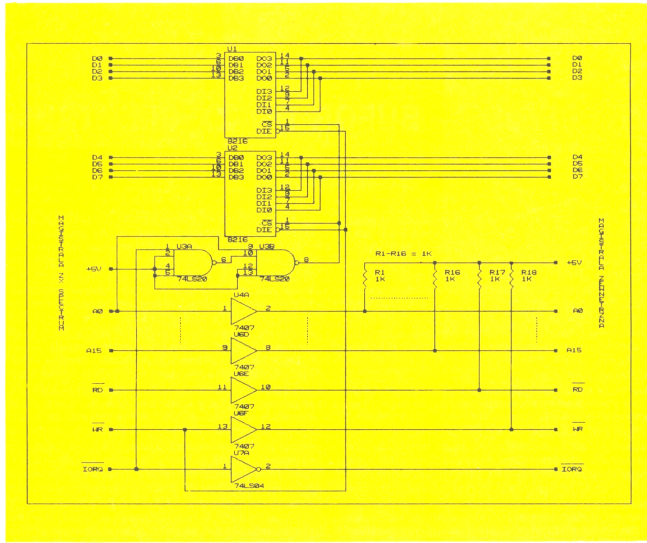
ZX Spectrum dni swojej świetności ma już dawno za sobą, jednak można oceniać, że liczba tych komputerów znajdująca się w rękach polskich użytkowników jest dosyć znaczna. Zabawka ta może w rękach zręcznego hobbisty stać się cennym narzędziem samodoskonalenia profesjonalnego, a zaprojektowane i wykonane układy rozszerzeń komputera — przynosić realną korzyść na polu jego działalności (nie tylko elektronicznej!).

Dołączenie urządzenia do komputera, wymaga — jak już było powiedziane — skonstruowania interfejsu. I nawet nie w tym problem, bo nie święci garnki lepią, ale w pewnych cechach konstrukcji ZX Spectrum. Otóż zastosowane w nim pamięci RAM to (m.in.) zasilane trzema napięciami elementy typu 4116. Charakteryzują się one ostrymi wymaganiami co do kolejności załączania napięć zasilających; w szczególności brak napięcia ujemnego powoduje bardzo szybkie (rzędu kilkudziesięciu milisekund) zniszczenie struktury elementu. Komputer zasilany jest napięciem +5 V (niestabilizowane +9 V), a pozostałe (−5 V i +12 V) wytwarza wbudowana przetwornica. Bywa ona niekiedy dosyć kapryśna, i chwilowy zanik napięcia +5 V może spowodować, że nie wzbudzi się ona ponownie, co oznaczać będzie zanik napięcia −5 V. Napięcie to wyprowadzone jest na złącze krawędziowe — także i przypadkowe zwarcie na tym złączu spowodować może jego zanik. Podobnie może się stać z napięciem zasilającym +12 V również wyprowadzonym na złącze. W ferworze uruchamiania nowej przystawki o taki przypadek nietrudno, szczególnie u tych mniej doświadczonych, a bardziej nerwowych. Potraktowane w ten sposób Spectrum żegna się z tym światem charkotem i rzężeniem głośniczka, a na ekranie pojawia się kolorowy wzór z kwadracików. Niektóre z nich zdają się pomrugiwać na nas ironicznie, jakby monitor chciał powiedzieć: jak nie potrafisz, nie pchaj się na afisz...

Z obłoków antropomorficznego natchnienia na ziemię sprowadza nas kwota wyszczególniona na rachunku za naprawę komputera.

Nieprzyjemnym skutkom chwili nieuwagi zapobiec można włączając pomiędzy komputer i przystawkę bufor-separator charakteryzujący się następującymi cechami:

- logicznymi: powinien zapewniać obustronny dostęp do wybranych obszarów pól adresowych mikroprocesora,
- sprzętowymi: w przypadku pojawienia się na jego



wyprowadzeniach napięć niezgodnych ze standardem TTL (leżących poza zakresem 0 V÷5 V) nie dopuścić do pojawienia się ich wewnątrz komputera nawet kosztem uszkodzenia własnej struktury.

W proponowanym rozwiązaniu pierwszy z tych punktów ograniczony został jedynie do dostępu do nieparzystych adresów obszaru adresowego urządzeń wejścia/wyjścia (powodem był planowany zakres wykorzystania komputera). W szczególności uniemożliwia to stosowanie mechanizmów dynamicznej wymiany zawartości obszaru pamięciowego o adresach 0—16383; jednakże nic nie stoi na przeszkodzie, aby po doprowadzeniu do interfejsu sygnałów /RD, /MREQ i /ROMCS i drobnej zmianie konstrukcji dekodera, zlikwidować to ograniczenie.

Schemat ideowy interfejsu przedstawiony jest na rys. 1. Do separacji szyny danych użyto w nim układów 8216 (UCY74S416). Zastosowanie tych właśnie (dziś może trochę przestarzałych już) elementów podyktowane zostało analizą schematów struktur analogicznych układów. Okazało się, że ten właśnie element charakteryzuje się największym prawdopodobieństwem zwarcia napięć spoza zakresu pracy, a tym samym niedopuszczenia do pojawienia się ich na szynie danych komputera. Tym też uwarunkowany był wybór kierunku włączenia układu w magistralę. Nie bez wpływu był także fakt wytwarzania układu przez przemysł rodzimy i jego dostępność na rynku.

Do separacji szyn adresowej i sterującej wykorzystano układy 74LS07; wybór ich podyktowany został zwiększonym zakresem napięć wyjściowych. Dodatkowo konstrukcja struktury powoduje, że w przypadkach awaryjnych wyprowadzenie otwartego kolektora „upala się", izolując szynę od reszty układu. Zastosowanie układów serii LS spowodowane zostało obciążalnością ZX Spectrum. Jeżeli komputer współpracować będzie jedynie z interfejsem (nic przecież nie stoi na przeszkodzie, aby równolegle do niego dołączyć do szyny np. stację dysków...) można je zastąpić układami 7407, jest to jednak ostateczność.

Zaprojektowanie konstrukcji mechanicznej pozostawione zostało inwencji wykonawcy; sugerowane jest jednak wykorzystanie podstawek pod układy scalone, bo niekiedy wylutowanie uszkodzonej kostki jest bardzo trudne...

Proponowane rozwiązanie zostało sprawdzone w praktyce i kilkakrotnie miało okazję wykazać swoją przydatność chroniąc komputer przed uszkodzeniem.

SEMINARIUM „INFORMIKA"

ZX-y — inaczej! (cz. II)

ARYTMETYKA ZMIENNOPOZYCYJNA MIKROKOMPUTERÓW ZX

W kolejnym odcinku cyklu poświęconego budowie i zastosowaniu mikrokomputerów ZX pragniemy przedstawić Czytelnikom najciekawszą (naszym zdaniem) część systemu operacyjnego mikrokomputerów ZX, a mianowicie kalkulator zmiennopozycyjny. Kalkulator ten umożliwia dokonywanie złożonych operacji arytmetyczno-logicznych na liczbach zmiennopozycyjnych. Zanim jednak przejdziemy do opisu właściwych procedur kalkulatora ZX zapoznamy Czytelników z podstawami arytmetyki komputerowej, to znaczy z systemami reprezentacji liczb.

W życiu codziennym człowiek posługuje się dziesiętnym systemem liczbowym. W systemie tym występuje dziesięć cyfr: 0, 1, ..., 8 i 9. Liczby większe od dziesięciu reprezentowane są jako wielokrotności podstawy systemu, i tak np. liczbę 235 należy rozumieć jako:

10^2 10^1 10^0

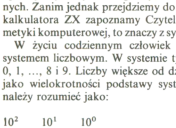
2 3 5

5* 1 = 5

3* 10 = 30

2*100 = 200

235



Podstawową jednostką informacji w mikrokomputerach jest bit, który przyjmuje wartości 0 i 1. Z tego też powodu mikrokomputery stosują do obliczeń system dwójkowy, w którym podstawą jest liczba dwa. Pozostawiamy Czytelnikom do sprawdzenia, że wspomnianą liczbę 235d („d" z ang. *decimal*) można przedstawić jako: 11101011b („b" z ang. *binary*):

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

1 1 1 0 1 0 1 1

1* 1 = 1

1* 2 = 2

0* 4 = 0

1* 8 = 8

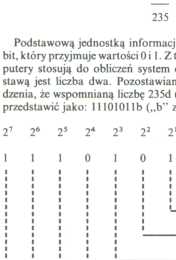
0* 16 = 16

1* 32 = 32

1* 64 = 64

1*128 = 128

235d



Ponieważ system zapisu dwójkowego jest bardzo „rozwlekły" powszechnie stosuje się system szesnastkowy. W systemie tym wyróżnia się cyfry 0, 1, . . ., 8, 9 oraz A, B, C, D, E, F. Cyfry A, B, . . ., F mają wartości dziesiętne odpowiednio: 10, 11, . . ., 15. Przejście z systemu szesnastkowego do dwójkowego (i odwrotnie) jest bardzo proste. Każdą z liczb systemu szesnastkowego wystarczy zapisać w postaci czteropozycyjnej liczby dwójkowej np. 1EBh („h" z ang. *hexadecimal*):

1	E	B
0001	1110	1011

Podobnie liczbę binarną wystarczy podzielić na „czwórki" i każdą tetradę zapisać w postaci szesnastkowej.

O ile zapis liczb całkowitych nieujemnych nie przedstawia większych kłopotów, o tyle zapis liczb ujemnych, a w szczególności ułamkowych jest bardziej skomplikowany.

Powszechnie do zapisu liczb rzeczywistych niecałkowitych stosuje się tak zwany system wykładniczy. W zapisie tym liczbę reprezentują dwa składniki: mantysa będąca ułamkiem z przedziału (0.5,1⟩ oraz wykładnik przy podstawie 2:

$N = m*2^e$ np. $0,1 = 0,8*2^{-3}$

W tak przyjętym zapisie pierwszy bit mantysy zawsze przyjmuje wartość 1. Właściwość ta w przypadku mikrokomputerów ZX używana jest do zapisywania w tym bicie znaku liczby i tak wartość 0 oznacza liczbę dodatnią, a 1 ujemną.

W systemie ZX liczba zmiennopozycyjna zapisywana jest w pięciu kolejnych bajtach. Najbardziej znaczący bajt reprezentuje 7-bitowy wykładnik powiększony o 128 ($= 2^7$). Mantysa z kolei zajmuje kolejne 32 bity. Zgodnie z wcześniejszą uwagą, jej najbardziej znaczący bit określa znak liczby, np.:

0,162d = (01111110)b.(00100101111000110101001111110111)b

7Eh 25h E3h 53h F7h

126d 0,148d

co po uwzględnieniu cech zapisu daje:

e = 126 − 128 = −2

m = 0,5 + 0,148 = 0,648 (mantysa ujemna)

$0,648*2^{-2} = 0,162$

oraz

−162d = (10001000)b.(10100010000000000000000000000000)b

88h A2h 00h 00h 00h

136d 0,6328125d

e = 136 − 128 = 8

m = −1*0,6328125 (mantysa ujemna)

$-0,6328125*2^8 = -162$

W przypadku mikrokomputera ZX81 wszystkie liczby (całkowite i ułamkowe) traktowane są jako zmiennopozycyjne. W przypadku mikrokomputera ZX Spectrum odmienny zapis obowiązuje dla liczb całkowitych z przedziału ⟨−65535d, 65535d⟩, czyli ⟨−FFFFh, FFFFh⟩. Liczby nieujemne {+C} z tego przedziału zapisywane są w dwóch bajtach mimo utrzymania 5-bajtowej konwencji reprezentowania liczb. Jeżeli bajty te ponumerujemy: b0, b1, b2, b3 i b4, to dla liczb {+C}:

b0 = b1 = b4 = 0

b2 = {+C} − 256*INT({+C}/256)

b3 = INT({+C}/256),

a dla liczb ujemnych {−C}:

b0 = b4 = 0

b1 = 255

b2 = 255 − ({ − C} − 256 * INT({ − C}/256))
b3 = 255 − INT({ − C}/256)

Na przykład

− 162d = (00h FFh 5Eh FFh 00h)

Liczby rzeczywiste nie będące liczbami całkowitymi i liczby całkowite wykraczające poza omawiany przedział zapisywane są dokładnie w myśl tych samych reguł co dla ZX81.

Należy tu dodać, że projektanci systemu ZX Spectrum dopuścili się pewnej niekonsekwencji; liczba zmiennoprzecinkowa zero może być niekiedy zinterpretowana przez system jako liczba stałoprzecinkowa o wartości — oczywiście — niezerowej. Może to stać się przyczyną trudnych do wykrycia błędów.

Jak łatwo można się przekonać, liczby, na których operuje kalkulator ZX, muszą być mniejsze co do wartości bezwzględnej niż $2^{128} \approx 3{,}403 * 10^{38}$, a wartości mniejsze co do wartości bezwzględnej niż $2^{-127} \approx 5{,}877 * 10^{-39}$ interpretowane są jako zero.

Kalkulator zmiennopozycyjny mikrokomputerów ZX jest w istocie rozbudowanym programem w języku asemblera mikroprocesora Z80; zajmuje on ok. 2,5 KB obszaru pamięci ROM.

Działanie kalkulatora oparte jest na Odwrotnej Notacji Polskiej (RPN z ang. *Reverse Polish Notation*). W systemie tym obowiązują następujące zasady:

- operacje dotyczą elementów struktury typu stos (LIFO — ang. *Last-In-First-Out*),
- operator wprowadzany jest po argumentach,
- operacja wykonywana jest po wprowadzeniu operatora.

Należy zauważyć, że RPN jest systemem, w którym do przeprowadzenia złożonych obliczeń jest wymagana najmniejsza liczba wykonywanych instrukcji i istnieje możliwość dostępu do wyników pośrednich. Cecha ta spowodowała, że system RPN jest stosowany w wielu kalkulatorach naukowych i programowalnych (np. firmy Hewlett-Packard), znalazł też zastosowanie w języku FORTH.

Stos LIFO, na którym operuje program kalkulatora — CALCULATE, określony jest przez zmienne systemowe STKEND i STKBOT. Każdorazowe przesłanie do kalkulatora liczby powoduje przesunięcie STKEND w górę, a pobranie — przesunięcie w dół. Operacje wykonywane są na argumencie (argumentach) znajdujących się na szczycie stosu. Niezależnie od stosu kalkulator ma sześć rejestrów roboczych (pamięci) stanowiących część obszaru zmiennych systemowych MEMBOT.

System operacyjny odwołuje się do kalkulatora poprzez jednobajtową instrukcję procesora RST 0028h (czyli RST FP__CALC). Następująca po tej instrukcji sekwencja bajtów określa rodzaj wykonywanej na zawartości stosu operacji. Kalkulator ma więc swój własny „mini-język" znacznie ułatwiający wykonywanie bardziej skomplikowanych programów numerycznych. Język ten oprócz podstawowych operacji ma również odpowiedniki struktur IF-THEN-ELSE oraz FOR--NEXT. Konstrukcja programu CALCULATE umożliwia również wykonywanie procedur rekurencyjnych. Kalkulator może wykonać ogółem 77 instrukcji w przypadku ZX81 (82 dla ZX Spectrum), z czego 67 (69 w ZX Spectrum) to operacje na zmiennych numerycznych, a pozostałe na zmiennych tekstowych.

Operacje, które są wykonywane na zmiennych tekstowych są w rzeczywistości wykonywane na 5-bajtowych parametrach określających te zmienne, a właściwe operacje są wykonywane w obszarze E__LINE.

Procedury pomocnicze, w które jest wyposażony kalkulator, umożliwiają przesyłanie liczb do i ze stosu oraz manipulacje na różnych ich reprezentacjach. Część procedur pomocniczych kalkulatora może być używana przez programistę.

Najbardziej użyteczne z nich to:

ALPHA (ZX81 14CEh, ZX Spectrum 2C8Dh). Procedura sprawdza, czy znak znajdujący się w rejestrze A reprezentuje literę alfabetu. Wynik testu sygnalizuje wskaźnik CY i tak CY = 1 gdy A jest kodem litery, CY = 0, gdy zawartość A nie jest kodem litery.

ALPHANUM (ZX81 14D2h, ZX Spectrum 2C88h). Procedura sprawdza, czy znak w rejestrze A jest cyfrą lub literą, a wynik testu sygnalizuje wskaźnik CY.

NUMERIC (ZX81 −, ZX Spectrum 2D1Bh). Procedura sprawdza, czy zawartość rejestru A jest kodem cyfry. Jeżeli tak, to CY = 0.

Procedura pełniąca tę samą rolę dla ZX81 ma postać:

```
        ; PROCEDURA NUMERIC (ZX81)
NUMERIC: CP   1CH  ; test kodu „0"
         RET  C    ; skończyć gdy mniejsze z CY = 1
         CP   26H  ; test kodu „A"
         CCF       ; gdy mniejsze to CY = 0
         RET       ; gdy większe/równe to CY = 1
```

STK__DIGIT (ZX81 1514h, ZX Spectrum 2D22h). Procedura sprawdza, czy zawartość rejestru A reprezentuje liczbę. Jeśli tak, to dokonuje jej zamiany na postać zmiennopozycyjną, która zostaje umieszczona na szczycie stosu kalkulatora.

STACK__A (ZX81 151Dh, ZX Spectrum 2D28h). Procedura zamienia liczbę dwójkową znajdującą się w rejestrze A na postać zmiennopozycyjną. Wynik konwersji znajduje się na szczycie stosu kalkulatora.

STACK__BC (ZX81 1520h, ZX Spectrum 2D2Bh). Procedura identyczna z poprzednią, przy czym zamianie ulega liczba dwójkowa znajdująca się w parze rejestrów BC.

STK__STORE (ZX81 12C3h, ZX Spectrum 2AB6h). Procedura przesyła na stos kalkulatora 5-bajtową reprezentację liczby, przy czym kolejne bajty (b0...b4) znajdują się w rejestrach A, E, D, C i B (jeżeli jest to postać zmiennopozycyjna, to w A znajduje się eksponenta, a w pozostałych rejestrach mantysa).

INT__TO__FP (ZX81 1548h, ZX Spectrum 2D3Bh). Procedura zamienia liczbę zapisaną w postaci liczby naturalnej na postać 5-bajtową, którą umieszcza na szczycie stosu kalkulatora. W chwili wywoływania procedury w rejestrze A musi znajdować się kod pierwszej cyfry, a jej adres musi wskazywać zmienna

Tablica 1. Operacje arytmetyczne

Kod		Mnemonik	Stos przed STKBOT–STKEND	Stos po STKBOT–STKEND
*	+			
0F	0F	addition	x,y	x+y
03	03	subtract	x,y	x−y
04	04	multiply	x,y	x*y
06	06	to_power	x,y	y^x
25	28	sqr	x	x^0.5
05	05	division	x,y	x/y
2E	32	n_mod_m	n,m	n−m*int(n/m), int(n/m)
1C	1F	sin	x	sin(x)
1D	20	cos	x	cos(x)
1E	21	tan	x	tg(x)
1F	22	asn	x	arcsin(x)
20	23	acs	x	arccos(x)
21	24	atn	x	arctan(x)
22	25	ln	x	ln(x)
23	26	exp	x	exp(x)
38	—	e_fp_fp	x,y	x*10^y
01	01	exchange	x,y	y,x
2D	31	duplicate	x	x,x
02	02	delete	x,y	x
18	1B	negate	x	−x
24	27	int	x	int(x)
26	29	sgn	x	1 gdy x>0 0 gdy x=0 −1 gdy x<0
27	2A	abs	x	abs(x)
A0	A0	stk_zero	–	0
A1	A1	stk_one	–	1
A2	A2	stk_half	–	1/2
A3	A3	stk_pi_2	–	pi/2
A4	A4	stk_ten	–	10
30	34	stk_data	–	data
35	39	get_argt	x	v
36	3A	truncate	x	sgn(x)*int(abs(x))
—	3D	re_stack	x	x

Tablica 2. Operacje logiczne.

Kod *	Kod +	Mnemonik	Stos przed STKBOT–STKEND	Stos po STKBOT–STKEND
07	07	or	x,y	x gdy y=0 1 gdy y<>0
08	08	no_&_no	x,y	x gdy y<>0 0 gdy y=0
2C	30	not	x	1 gdy x=0 0 gdy x<>0
32	36	less_0	x	1 gdy x<0 0 gdy x>=0
33	37	greater_	x	1 gdy x>0 0 gdy x<0
09	09	no_l_eql	x,y	1 gdy x<=y 0 gdy x>y
0A	0A	no_gr_eq	x,y	1 gdy x>=y 0 gdy x<y
0B	0B	nos_neql	x,y	1 gdy x<>y 0 gdy x=y
0C	0C	no_grtr	x,y	1 gdy x>y 0 gdy x<=y
0D	0D	no_less	x,y	1 gdy x<y 0 gdy x>=y
0E	0E	nos_eql	x,y	1 gdy x=y 0 gdy x<>y

Tablica 3. Operacje na zmiennych tekstowych.

Kod *	Kod +	Mnemonik	Stos przed STKBOT–STKEND	Stos po STKBOT–STKEND
10	10	str_&_no	p(a$),y	p(a$) gdy y<>0 p(o$) gdy y=o
11	11	str_l_eql	p(a$),p(b$)	1 gdy a$<=b$ 0 gdy a$>b$
12	12	str_gr_eq	p(a$),p(b$)	1 gdy a$>=b$ 0 gdy a$<b$
13	13	strs_neql	p(a$),p(b$)	1 gdy a$<>b$ 0 gdy a$=b$
14	14	str_grtr	p(a$),p(b$)	1 gdy a$>b$ 0 gdy a$<=b$
15	15	str_less	p(a$),p(b$)	1 gdy a$<b$ 0 gdy a$>=b$
16	16	strs_eql	p(a$),p(b$)	1 gdy a$=b$ 0 gdy a$<>b$
17	17	strs_add	p(a$),p(b$)	p(a$+b$)
19	1C	code	p(a$)	code(a$(1))
1B	1E	len	p(a$)	len(a$)
1A	1D	val	p(a$)	val(a$)
—	18	val$	p(a$)	p(val$(a$))
2A	—	str$	x	p(str$(x))
—	2E	str$	x	p(str$(x))
2B	—	chr$	x	p(chr$(x))
—	2F	chr$	x	p(chr$(x))

systemowa CH__ADD. Jeżeli wartość znajdująca się w rejestrze A nie będzie dziesiętnym cyfry, to na stosie kalkulatora zostanie odłożona wartość 00h.

DEC__TO__FP (ZX81 14D9h, ZX Spectrum 2C9Bh). Jedna z ważniejszych procedur powodująca zamianę liczby zapisanej w postaci dziesiętnej na odpowiadającą jej postać zmiennopozycyjną. Adres początku reprezentacji dziesiętnej musi znajdować się w zmiennej systemowej CH__ADD oraz pierwszy bajt tej liczby w rejestrze A. Wynik zamiany zostaje umieszczony na szczycie stosu kalkulatora.

STK__FETCH (ZX81 13F8h, ZX Spectrum 2BF1h). Procedura pobiera ze stosu kalkulatora liczbę zmiennopozycyjną do rejestrów B, C, D, E i A. Efekt jej działania jest przeciwny do efektu działania procedury STK__STORE.

FP__TO__BC (ZX81 158Ah, ZX Spectrum 2DA2h). Procedura pobiera liczbę zmiennopozycyjną ze stosu kalkulatora, po czym zamienia ją na liczbę dwójkową (całkowitą) i przesyła ją do pary rejestrów BC. Jeśli wartość bezwzględna pobranej liczby jest większa niż FFFFh, to wskaźnik CY = 1; natomiast gdy liczba jest mniejsza od zera to wskaźnik Z = 0.

FIND__INT (ZX81 0EA7h, ZX Spectrum –). Procedura działająca podobnie do FP__TO__BC z tą różnicą, że jeśli CY = 1 lub Z = 0 następuje powrót do systemu z raportem B.

FIND__INT2 (ZX81 –, ZX Spectrum 1E99h). Procedura identyczna z FIND__INT.

FP__TO__A (ZX81 15CDh, ZX Spectrum 2DD5h). Procedura analogiczna do FP__TO__BC z tą różnicą, że pobrana liczba zostaje umieszczona w rejestrze A; wskaźnik CY = 1, gdy wartość bezwzględna pobranej liczby jest większa niż FFh, a wskaźnik Z = 0, gdy jest to liczba ujemna.

FIND__INT1 (ZX81 –, ZX Spectrum 1E94h). Procedura podobna do poprzedniej, lecz wystąpienie CY = 1 lub Z = 0 będzie sygnalizowane błędem B.

Procedura o podobnym działaniu dla ZX81 ma postać:

```
; PROCEDURA FIND__INT1 (ZX81)
    CALL  15CDH     ; FP__TO__A
    JR C, REP__B    ; jeżeli CY = 1 to błąd B
    RET Z           ; koniec gdy CY = 0 i Z = 1
REP__B: JP  0EADH   ; w przeciwnym razie błąd B
```

STK__TO__A (ZX81 0C02h, ZX Spectrum 2314h). Procedura pobiera liczbę ze stosu kalkulatora, zamienia ją na liczbę całkowitą dwójkową i umieszcza wynik w rejestrze A. Jeżeli wartość bezwzględna liczby była większa niż FFh następuje powrót do systemu z raportem B. Informacja o znaku jest umieszczana w rejestrze C (01h dla liczb nieujemnych, FFh dla pozostałych).

STK__TO__BC (ZX81 0BF5h, ZX Spectrum 2307h). Procedura pobiera dwie liczby ze stosu kalkulatora, zamienia je na liczby całkowite dwójkowe i umieszcza wynik w rejestrach B i C. Jeśli wartość bezwzględna którejś z liczb jest większa niż FFh następuje powrót do systemu z raportem błędu B. Informacje o znakach (jak w STK__TO__A) zostają umieszczone odpowiednio w rejestrach E i D.

PRINT__FP (ZX81 15DBh, ZX Spectrum 2DE3h). Procedura drukuje w obszarze D__FILE w miejscu określonym przez DF__CC wartość znajdującą się na szczycie stosu kalkulatora. W zależności od zakresu liczba może być wydrukowana w jednej z następujących postaci:

- wykładniczej (dla wartości bezwzględnych mniejszych niż 10^{-5} lub nie mniejszych niż 10^{13},
- ułamka dziesiętnego (dla liczb z przedziału $\langle 10^{5}, -1)$),
- całkowitej (dla liczb całkowitych mniejszych niż 10^{13}, ale dla liczb większych niż 10^{8} jest drukowanie jedynie 8 bardziej znaczących cyfr, a pozostałe zostają zastąpione zerami).

SET__STK__B (ZX81 14A6h, ZX Spectrum –). Procedura ustala położenie stosu kalkulatora i zmienne STKBOT, STKEND na adres zawarty w parze rejestrów HL.

Dla ZX Spectrum identyczna w działaniu będzie procedura

```
; SET__STK__B (ZX SPECTRUM)
    JP   16C2H      ; w SET__WORK
```

CLEAR (ZX81 149Ah, ZX Spectrum –). Procedura wykonuje instrukcję języka BASIC — CLEAR.



CLEAR_RUN (ZX81 —, ZX Spectrum 1EAFh). Procedura identyczna z CLEAR dla zawartości pary rejestrów BC = 0000h, dla BC ⟨⟩ 0000h oprócz wykonania CLEAR nastąpi przesunięcie RAMTOP do wskazanego przez BC adresu wraz z instalacją stosu maszynowego.

X_TEMP (ZX81 14A3h, ZX Spectrum —). Procedura czyści obszar roboczy interpretera i stos kalkulatora.

SET_MIN (ZX81 —, ZX Spectrum 16B0h). Procedura identyczna w skutkach z X_TEMP.

SET_MEM (ZX81 14BCh, ZX Spectrum —). Procedura przenosi wskaźnik obszaru roboczego kalkulatora (przechowywany w zmiennej MEM) do obszaru MEMBOT i czyści obszar stosu kalkulatora powyżej obszaru roboczego interpretera.

SET_STK (ZX81 —, ZX Spectrum 16C5h). Procedura podobna w skutkach do SET_MEM.

Wszystkie operacje, które mogą być wykonywane w kalkulatorze zmiennopozycyjnym, można podzielić na 7 grup:

- operacje arytmetyczne,
- operacje logiczne,
- operacje na zmiennych tekstowych,
- operacje na komórkach pamięci kalkulatora,
- skoki,
- generator wielomianów Czebyszewa,
- inne.

W zależności od rodzaju wykonywanej operacji parametr lub parametry muszą znajdować się na stosie kalkulatora lub bezpośrednio za bajtem kodu operacji, podobnie jak w przypadku skoków. Jeśli w czasie wykonywania operacji występuje błąd, to jest on sygnalizowany na monitorze z jednoczesnym zawieszeniem wykonywania programu. W tabelach 1, 2 i 3 zestawiono poszczególne operacje.

SKOKI

Kalkulator zmiennopozycyjny oprócz możliwości wykonywania operacji arytmetyczno-logiczno-tekstowych ma również możliwość wykonywania skoków i pętli. Do tego celu służą następujące operacje (w nawiasach podano odpowiednie kody):

jump (ZX81 2Fh, ZX Spectrum 33h). Skok bezwarunkowy. Wartość skoku jest zapisana w kodzie uzupełnień do dwóch bezpośrednio za bajtem instrukcji i określa, ile bajtów (instrukcji) należy przeskoczyć.

jump-true (00h). Skok warunkowy; skok wykonywany w zależności od wartości znajdującej się na szczycie stosu kalkulatora. Jeśli wartość ta wynosi 0, to skok nie jest wykonany.

dec-jr-nz (ZX 81 31h, ZX Spectrum 35h). Skok w pętli, instrukcja podobna w swym działaniu do instrukcji asemblera DJNZ, przy czym funkcję licznika pętli spełnia zmienna BERG (ZX81, BREG w ZX Spectrum).

GENERATOR WIELOMIANÓW CZEBYSZEWA

series_06 (86h), **series_08** (88h), **series_0C** (8Ch). Generatory wielomianów Czebyszewa aproksymują funkcje sin, atn, ln i exp. Pozostałe funkcje są obliczane jako kombinacje wyżej podanych. Wielomiany te są generowane według wzoru:

$T_n(z) = 2*z*T_{n-1}(z) - T_{n-2}(z)$

$T_0(z) = 1, \quad T_1(z) = z$

przy czym $T_n(z)$ jest n-tym wielomianem.

Dla funkcji sin n = 6, atn n = 8, ln i exp n = 12. Wartości odpowiednich współczynników muszą być zapisane w postaci zmiennopozycyjnej i znajdować się bezpośrednio za bajtami kodu generacji.

OPERACJE NA KOMÓRKACH PAMIĘCI

Kalkulator może wykonywać działania na sześciu komórkach swojej pamięci operacyjnej (nie mylić z obszarem pamięci maszynowej mikroprocesora; komórki te są to specjalnie zarezerwowane pięciobajtowe obszary tej pamięci).

(Uwaga: niektóre rozkazy kalkulatora, np. generacja wielomianów Czebyszewa, wykorzystują te obszary dla swoich potrzeb niszcząc ich zawartość).

st__mem__0 (C0h), **st__mem__1** (C1h), **st__mem__2** (C2h), **st__mem__3** (C3h), **st__mem__4** (C4h), **st__mem__5** (C5h). Kopiowanie wartości znajdującej się na szczycie stosu kalkulatora do wskazanego rejestru (mem__0...mem__5).

get__mem__0 (E0h), **get__mem__1** (E1h), **get__mem__2** (E2h), **get__mem__3** (E3h), **get__mem__4** (E4h), **get__mem__5** (E5h). Kopiowanie zawartości wskazanego rejestru (mem__0... ...mem__5) na szczyt stosu.

INNE OPERACJE

end__calc (ZX81 34h, ZX Spectrum 38h). Koniec operacji. Program kalkulatora po zdekodowaniu instrukcji end__calc zawiesza operacje i wykonuje RET do głównego programu.

fp__calc__2 (ZX81 37h, ZX Spectrum 3Bh). Wykonanie operacji o kodzie zawartym w rejestrze B mikroprocesora.

peek (ZX81 28h, ZX Spectrum 2Bh). Odpowiednik funkcji PEEK języka BASIC. Argument (adres) musi znajdować się na stosie kalkulatora; tam też umieszczany jest wynik działania operacji.

usr (ZX81 29h, ZX Spectrum −). Odpowiednik funkcji USR języka BASIC. Argument musi znajdować się na stosie kalkulatora.

usr__no (ZX81 −, ZX Spectrum 2Dh). Operacja identyczna jak usr dla ZX81.

Sposób operowania kalkulatorem zmiennopozycyjnym przedstawiony zostanie na następujących przykładach.

Przykład 1

Obliczyć zasięg, na jaki doleci ciało wyrzucone pod kątem α do poziomu z prędkością v, przy czym wartości α i v znajdują się na szczycie stosu kalkulatora. Kąt α jest zapisany w radianach. Wynik pozostawić w kalkulatorze. Zasięg obliczamy według wzoru: $x = v^2 * \sin(2*\alpha)/g$

Uwaga. W poniższym i w kolejnych programach linie sygnowane „*" dotyczą tylko ZX81, zaś „+" tylko ZX Spectrum.

```
RST       FP_CALC     ; wywołanie kalkulatora
                      ; stos: α, v
DB        duplicate   ; α, v, v
DB        multiply    ; α, v²
DB        stk_data    ; α, v², 9.81
DB        F4H         ; 5 kolejnych bajtów
DB        1CH         ; reprezentuje
DB        F5H         ; data=9.81
DB        C2H
DB        90H
DB        division    ; α, v²/9.81
DB        exchange    ; v²/9.81, α
DB        duplicate   ; v²/9.81, α, α
DB        addition    ; (v²)/9.81, 2*α
DB        sin         ; (v²)/9.81, sin(2*α)
DB        multiply    ; x=(v²)*sin(2*α)/9.81
DB        end_calc    ; koniec, powrót do
RE                    ; procedury nadrzędnej.
```

Przykład 2

Obliczyć za pomocą kalkulatora zmiennopozycyjnego wartość wyrażenia n!, a wynik wydrukować na ekranie w miejscu o współrzędnych 10, 12. Liczba n jest liczbą całkowitą dodatnią, mniejszą niż 38 i jest zapisana w postaci dziesiętnej w linii programu 1 REM.

```
          LD BC,    (CH_ADD)  ; zachować oryginalne CH_ADD
          PUSH      BC
*         LD HL,    4079H     ; (PROG) ZX81, czyli początek
+         LD HL,    (PROG)    ; programu w języku Basic
          LD BC,    0005H     ; w HL adres pierwszego bajtu
          ADD HL,   BC        ; w linii 1 REM
          LD (CH_ADD),HL      ; adres do zmiennej CH_ADDR
          LD A,     (HL)      ; kod pierwszego znaku do A
          CALL      DEC_TO_FP ; postać dwójkowa na stos
          CALL      FP_TO_A   ; i do A
          JR NZ,    CONT
          INC       A
CONT:     LD B,     A
          RST       FP_CALC
          DB        stk_one   ; 0!=1
          DB        duplicate
          DB        st_mem_0
LOOP:     DB        multiply  ; rekurencyjne obliczenie n!
          DB        get_mem_0
          DB        stk_one
          DB        addition
          DB        st_mem_0
          DB        dec_jr_nz ; B:=B-1
          DB        FAM       ; pętla do LOOP, gdy B>0
          DB        delete
          DB        end_calc  ; n!
+         LD A,     02H
+         CALL      CHAN_OPEN
*         LD B,     0AH       ; pozycja X=10
*         LD C,     0CH       ; pozycja Y=12 w D_FILE
*         CALL      PRINT_AT  ; ustalić współrzędne
+         LD A,     16H       ; kod AT
+         LD C,     0AH       ; pozycja X
+         LD B,     0CH       ; pozycja Y w D_FILE
+         CALL      PR_AT_TAB ; ustalić współrzędne
          CALL      PRINT_FP  ; wydrukować liczbę
          POP       BC        ; odtworzyć CH_ADD
          LD (CH_ADD), BC
          RET                 ; koniec.
```

W następnym odcinku cyklu przedstawimy możliwości rozbudowy standartowych funkcji języka Basic mikrokomputerów ZX oraz inne zastosowania kalkulatora zmiennopozycyjnego.

JANUSZ J. MŁODZIANOWSKI
TADEUSZ A. ZALESKI

KOMPUTER W SZKOLE

ALGORYTM WYZNACZANIA WSPÓŁRZĘDNYCH SŁOŃCA, KSIĘŻYCA I PLANET

(CZĘŚĆ III)

W poprzednich odcinkach zamieściliśmy algorytm pozwalający wyznaczyć współrzędne równikowe Słońca, Księżyca, Merkurego, Wenus i Marsa. Teraz przyszła kolej na Jowisza. Kolejne kroki postępowania prowadzące do wyznaczenia DL, DB i DR są analogiczne jak poprzednio. Korzysta się również z tych samych współczynników J1, ..., J33.

Poprawność konstrukcji tych wyrażeń można sprawdzić porównując wartości liczbowe. Dla daty 28 czerwca 1969 roku (tej samej, dla której wyznaczyliśmy uprzednio współczynniki J1, ..., J33) powinny one wynosić:

	DL	DB	DR
Jowisz	8353,9	4699,8	5,45233

Wartości te posłużą do dalszych obliczeń. Podobnie jak i poprzednio DR jest odległością planety od Słońca wyrażoną w jednostkach astronomicznych.

Dalsze postępowanie, prowadzące do wyznaczenia rektascensji i deklinacji, jest takie samo, jak w przypadku Merkurego, Wenus i Marsa, z tym, że dla Jowisza wartość LR wynosi:

$$LR = (DL/3600)*S + 2*PI*J18 + DLR,$$

gdzie $DLR = -17*SIN(J5*2*PI)/3600*S$, S podobnie jak poprzednio jest równe PI/180.

Pamiętamy, że jeśli LR jest ujemne, to dodajemy do niego $2 \times PI$ (LR jest wyrażone w radianach), procedurę powtarzamy aż uzyskamy LR dodatnie.



Porównanie obliczonych współrzędnych (TEST) z podanymi w Roczniku Astronomicznym (ROCZNIK)

Jowisz

	TEST	ROCZNIK
LR/S	188° 34' 10"	188° 34' 03"
DDR/S	1° 18' 20"	1° 18' 20"
DR	5,45233	5,45265
α	$11^h 54^m 27^s$	$11^h 54^m 27^s$
δ	+2° 00' 51"	+2° 00' 39"

Zgodnie z zapowiedzią podamy teraz sposoby wykorzystania obliczonych współrzędnych równikowych planet do wyznaczenia momentów wschodów i zachodów tych ciał niebieskich.

Wschodem ciała niebieskiego nazywamy moment jego przecięcia z horyzontem astronomicznym, kiedy wyłania się ono z niewidocznej części strefy niebieskiej, a zachodem, kiedy zanurza się w widocznej części strefy niebieskiej w niewidoczną.

Obliczamy wpierw wartości kąta godzinnego. Kąt godzinny jest to, ogólnie biorąc, kątowa odległość danego ciała niebieskiego od południka. Obliczanie kąta godzinnego t przeprowadzamy dla momentu wschodu i zachodu. Sposób obliczania tego kąta zależy od ciała niebieskiego:

dla Słońca

$$\cos t = \frac{-\sin 51' - \sin\varphi \times \sin\delta}{\cos\varphi \times \cos\delta},$$

dla Księżyca

$$\cos t = \frac{\sin 7' - \sin\varphi \times \sin\delta}{\cos\varphi \times \cos\delta},$$

dla planet

$$\cos t = \frac{-\sin 35' - \sin\varphi \times \sin\delta}{\cos\varphi \times \cos\delta},$$

gdzie φ — szerokość geograficzna miejsca obserwacji,

δ — deklinacja ciała niebieskiego w momencie jego wschodu lub zachodu, natomiast t jest szukanym kątem godzinnym.

W powyższych wzorach uwzględniono kilka zjawisk, m.in. refrakcję, paralaksę oraz (w przypadku Słońca i Księżyca) górny bieg ich tarcz. Następnie obliczamy miejscowy czas gwiazdowy wschodu sw i miejscowy czas gwiazdowy zachodu sz danego ciała ze wzorów:

$$sw = \alpha w + 24 - t,$$
$$sz = \alpha z + t,$$

gdzie αw i αz są to rektascensje ciała niebieskiego w momencie wschodu i zachodu, a wyrażone są, podobnie jak t, w mierze czasowej. Wcześniej t obliczyliśmy w stopniach, a zamiany na miarę czasową dokonujemy dzieląc te wartości przez 15. Jeśli sw lub sz jest większe niż 24, to odejmujemy 24, tj. wartości sw i sz muszą być z przedziału $\langle 0, 24 \rangle$. Następnie przeliczamy miejscowy czas gwiazdowy wschodu sw, lub zachodu sz, na czas środkowoeuropejski (zimowy) wschodu **csew**, lub zachodu **csez** ze wzorów:

$csew = (sw - L - tg) * 0{,}997262 + 1^h$, gdzie

L — długość geograficzna obserwatora wyrażona w godzinach i liczona jest od południka Greenwich na wschód,

tg — czas gwiazdowy Greenwich o północy czasu UT (uniwersalnego), obliczany ze wzoru:

$tg = 6^h 38^m 45{,}836 + 8640184{,}542 * T + 0{,}0929 * T^2$, gdzie

$$T = \frac{JD - 2415020}{36525}$$

JD — dzień juliański o północy danego dnia.

Podobne obliczenia przeprowadzamy w celu wyznaczenia czasu środkowoeuropejskiego zachodu **csez**,

csez = (sz − L − tg)∗0,997262 + 1^h

Gdy sw − L − tg lub sz − L − tg jest większe od 24, to odejmujemy 24, a gdy mniejsze od 0, to dodajemy 24. W obu przypadkach, wartości **t**, **tg**, **csew**, **csez** redukujemy do przedziału ⟨0, 24⟩ poprzez wielokrotne dodawanie lub odejmowanie liczby 24.

We wzorach na obliczanie czasu gwiazdowego miejscowego wschodu **sw** lub zachodu **sz** danego ciała niebieskiego, potrzebna jest znajomość rektascensji i kąta godzinnego (tj. również deklinacji) w momencie wschodu czy zachodu tego ciała. Tego oczywiście nie znamy. Zwykle postępujemy w ten sposób, że początkowo przyjmujemy wartości rektascensji i deklinacji na 12 godzinę danego dnia, w którym chcemy wyliczyć moment wschodu czy zachodu. Następnie obliczamy **t**, a dalej **sw** i **sz**, oraz **csew** i **csez**. Otrzymamy w ten sposób przybliżony moment wschodu **csew** czy zachodu **csez**, dla którego obliczamy ponownie **zw**, **zz**, **sw**, **sz**, **t** oraz interesujący nas końcowy rezultat, tj. czas środkowoeuropejski wschodu **csew** i zachodu **csez**. Aby obliczyć w czy z zaczynamy obliczenia od samego początku poczynając od obliczenia nowych wartości J1, ..., J33. Liczba kroków iteracji zależy od zakładanej dokładności.

Powyższa procedura spowodowana jest tym, że Słońce, Księżyc i planety (głównie Merkury i Wenus) szybko zmieniają swoje położenie wśród gwiazd, a więc zmieniają swoje współrzędne α i δ w ciągu doby.

Przykład

Obliczamy moment wschodu i zachodu Księżyca dnia 28 czerwca 1989 r. w Warszawie (φ = 52° 13′ 21″,0, L = $1^h 24^m 02^s$,36).

Dla 28 czerwca 2969 r. 12^h otrzymujemy:

α = $17^h 03^m 23^s$
δ = −27° 48′ 16″
cos t = 0,68414936
t = 46^s,831254 = 3^h,1220836
sw = $24 + \alpha - t$
sw = 13^h,93421
sz = α + t
sz = 20^h,178378
T = 0,69488022
tg = 1674^s,3942 = 18^h,394224
csew = 19^h,089665 = $19^h 05^m$
csez = 25^s,3824479 = 1^h,3824479

obliczenie momentu wschodu

dla 28 czerwca 1969 r. 18^h,08966 TU
α = $17^h 20^m 42^s$
δ = −28° 09′ 10″
cos t = 0,69422403
t = 46^s,034585 = 3^h,0689723
sw = 14^h,276092
csew = 19,43061 = $19^h 26^m$

i po wyliczeniach dla tego momentu dostajemy

csew = 19^h,449765 = $19^h 27^m$

obliczenie momentu zachodu.

Otrzymaliśmy powyżej, że csez = 1^h,3824479, ale już 28 czerwca.

Dla tego momentu
α = $16^h 31^m 00^s$
δ = −26° 46′ 51″
t = 3^h,2723817
sz = 19^h,788921
csez = 0^h,99405777 = $0^h 59^m$,6

Według Rocznika Astronomicznego na 1969 r. wschód Księżyca nastąpił o $19^h 27^m$ CSE, a zachód o $0^h 59^m$ CSE. Uwaga! W 1969 roku nie obowiązywał czas letni (CWE). Przy obliczaniu momentu wschodu i zachodu szukaliśmy α i δ na przybliżony moment wschodu i zachodu w czasie uniwersalnym, a algorytm oblicza α i δ dla zadanego momentu w czasie efemerydalnym. Uproszczono algorytm z tego powodu, że w 1969 roku i w latach obecnych różnica między czasem UT i ET nie przekracza 1 minuty.

Możemy również obliczyć azymut ciała niebieskiego przy wschodzie lub zachodzie ze wzorów:
dla Słońca

$$\sin A = \frac{\cos\delta \times \sin t}{\cos 51'}$$

dla Księżyca

$$\sin A = \frac{\cos\delta \times \sin t}{\cos 07'}$$

dla planet

$$\sin A = \frac{\cos\delta \times \sin t}{\cos 35'}$$

oczywiście wartości δ i t bierzemy dla momentów wschodu czy zachodu. Azymut liczymy od południka na zachód, np. punkt zachodu ma azymut równy 90°.

Warto również policzyć maksymalną wysokość h ciała niebieskiego nad horyzontem, jaką osiągnie danego dnia:

— gdy ciało znajduje się na południe od zenitu

$$h = 90^\circ - \varphi + \delta$$

— gdy ciało znajduje się na północ od zenitu

$$h = 90^\circ + \varphi - \delta$$

Są to wysokości ciała niebieskiego nad horyzontem w momencie tzw. kulminacji górnej, tj. wtedy, gdy przecina ono południk niebieski w jego górnej części. Zachodzi również zjawisko kulminacji dolnej, tj. gdy ciało niebieskie przecina południk niebieski w jego dolnej części. Wtedy

$$h = \varphi + \delta - 90$$

W momencie kulminacji górnej ciała niebieskiego miejscowy czas gwiazdowy jest równy rektascensji ciała górującego, czyli

$$s = \alpha$$

Przykładowo, jeśli w danym momencie góruje gwiazda o α = 10^h, to wtedy miejscowy czas gwiazdowy wynosi 10^h. W momencie kulminacji dolnej

$$s = \alpha + 12^h.$$

Znając deklinację, szerokość geograficzną i kąt godzinny można obliczyć azymut i wysokość ciała niebieskiego z układu wzorów

$$\sin z \times \sin A = \cos\delta \times \sin t$$
$$\sin z \times \cos A = -\cos\varphi \times \sin\delta + \sin\varphi \times \cos\delta \times \cos t$$
$$\cos z = \sin\varphi \times \sin\delta + \cos\varphi \times \cos\delta \times \cos t,$$

gdzie z = 90° − h jest tzw. odległością zenitalną.

Również możemy obliczyć momenty zmierzchu cywilnego i astronomicznego. Gdy Słońce znajduje się 6° pod horyzontem, to mamy wtedy koniec zmierzchu (początek świtu) cywilnego, a gdy znajdzie się 18° pod horyzontem, to jest koniec zmierzchu (początek świtu) astronomicznego. Czas trwania zmierzchu tau zależy od szerokości geograficznej φ i od deklinacji Słońca δ i obliczany jest ze wzoru

$$\cos(t + tau) = \frac{\sin h - \sin\varphi \times \sin\delta}{\cos\varphi \times \cos\delta},$$

gdzie h — wysokość Słońca (h = −6° dla zmierzchu cywilnego oraz h = −18° dla zmierzchu astronomicznego),

t — kąt godzinny wschodu lub zachodu Słońca obliczany wg wzoru na cos t dla Słońca podany na początku tego odcinka.

Jak można obliczyć, zmierzch cywilny może trwać od zachodu do wschodu Słońca dla szerokości geograficznej powyżej 60°. Natomiast zmrok astronomiczny może trwać od zachodu do wschodu Słońca dla szerokości geograficznych powyżej 48°. Przykładowo, dla $\varphi = 54°$ w dniu 1 stycznia czas trwania zmierzchu cywilnego wynosi 44^m, a czas trwania zmierzchu astronomicznego wynosi $2^h 14^m$.

Opracował: IRENEUSZ WŁODARCZYK

Liczba	T	Funkcja	J18	J19	J22	J25
DL						
19934	0	SIN	0	1	0	0
5023	1	COS	0	0	0	0
2511	0	COS	0	0	0	0
1093	0	COS	0	2	-5	0
601	0	SIN	0	2	0	0
-479	0	SIN	0	2	-5	0
-185	0	SIN	0	2	-2	0
137	0	SIN	0	3	-5	0
-131	0	SIN	0	1	-2	0
79	0	COS	0	1	-1	0
-76	0	COS	0	2	-2	0
-74	1	COS	0	1	0	0
68	1	SIN	0	1	0	0
66	0	COS	0	2	-3	0
63	0	COS	0	3	-5	0
53	0	COS	0	1	-5	0
49	0	SIN	0	2	-3	0
-43	1	SIN	0	2	-5	0
-37	0	COS	0	1	0	0
25	0	SIN	2	0	0	0
25	0	SIN	0	3	0	0
-23	0	SIN	0	1	-5	0
-19	1	COS	0	2	-5	0
17	0	COS	0	2	-4	0
17	0	COS	0	3	-3	0
-14	0	SIN	0	1	-1	0
-13	0	SIN	0	3	-4	0
-9	0	COS	2	0	0	0
9	0	COS	0	0	1	0
-9	0	SIN	0	0	1	0
-9	0	SIN	0	3	-2	0
9	0	SIN	0	4	-5	0
9	0	SIN	0	2	-6	3
-8	0	COS	0	4	-10	0
7	0	COS	0	3	-4	0
-7	0	COS	0	1	-3	0
-7	0	SIN	0	4	-10	0
-7	0	SIN	0	1	-3	0
6	0	COS	0	4	-5	0
-6	0	SIN	0	3	-3	0
5	0	COS	0	0	2	0
-4	0	SIN	0	4	-4	0
-4	0	COS	0	0	3	0
4	0	COS	0	2	-1	0
-4	0	COS	0	3	-2	0
-4	1	COS	0	2	0	0
3	1	SIN	0	2	0	0
3	0	COS	0	0	5	0
3	0	COS	0	5	-10	0
3	0	SIN	0	0	2	0
-2	0	SIN	2	-1	0	0
2	0	SIN	2	1	0	0
-2	1	SIN	0	3	-5	0
-2	1	SIN	0	1	-5	0
DB						
-4692	0	COS	0	1	0	0
259	0	SIN	0	1	0	0
227	0	COS	0	0	0	0
-227	0	COS	0	2	0	0
30	1	SIN	0	1	0	0
21	1	COS	0	1	0	0
16	0	SIN	0	3	-5	0
-13	0	SIN	0	1	-5	0
-12	0	COS	0	3	0	0
12	0	SIN	0	2	0	0
7	0	COS	0	3	-5	0
-5	0	COS	0	1	-5	0
DR						
5.20883	0	COS	0	0	0	0
-0.25122	0	COS	0	1	0	0
-0.00604	0	COS	0	2	0	0
0.00260	0	COS	0	2	-2	0
-0.00170	0	COS	0	3	-5	0
-0.00106	0	SIN	0	2	-2	0
-0.00091	1	SIN	0	1	0	0
-0.00084	1	COS	0	1	0	0
0.00069	0	SIN	0	2	-3	0
-0.00067	0	SIN	0	1	-5	0
0.00066	0	SIN	0	3	-5	0
0.00063	0	SIN	0	1	-1	0
-0.00051	0	COS	0	2	-3	0
-0.00046	0	SIN	0	1	0	0
-0.00029	0	COS	0	1	-5	0
0.00027	0	COS	0	1	-2	0
-0.00022	0	COS	0	3	0	0
-0.00021	0	SIN	0	2	-5	0

Liczba	I	Funkcja	J18	J19	J22	J25

Z IBM PS/2 W „MIKROKANAŁACH"

W połowie roku 1987 system PS/2 był na ustach wszystkich interesujących się komputerami osobistymi. Dzisiaj jak gdyby zainteresowanie nieco opadło, a rozwój i ekspansja rodziny PS/2 wydaje się mniejsza, nic oczekiwano. Tym niemniej PS/2 zdobył sobie już na rynku ustabilizowaną pozycję i to mimo wysiłków groźnej konkurencji, forsującej komputery tzw. standardu EISA. Co zatem decyduje o atrakcyjności systemu PS/2, oczywiście poza faktem, że narodził się on w firmie IBM?

System PS/2 obejmuje formalnie modele 30, 50, 60, 70 i 80, ale model 30 jest tutaj wyraźnym outsiderem, wyraźnie nie pasującym do reszty rodziny i nie posiadającym podstawowego elementu jej architektury, czyli tzw. mikrokanału, o którym za chwilę. Twórcom rodziny PS/2 przyświecała idea stworzenia jednolitej struktury w obrębie całej palety produkcyjnej IBM oraz ugruntowanie solidnej bazy zarówno pod przyszły rozwój komputerów osobistych, jak i integrację tych „drobnoustrojów" z większymi systemami tej samej marki.

Modele 50 i 60 są wyposażone w procesor 80286, modele 70 i 80 — w procesor 80386 (oczekiwane jest pojawienie się modelu z nowym procesorem 80486). Zapewnia im to możliwość wykorzystania wirtualnej adresacji pamięci i otwiera drogę nowym systemom operacyjnym, jak OS/2 i UNIX. Wszystkie te modele mają też architekturę opartą o inteligentne „mikrokanały" (ang. *microchannel*). Mikrokanały zapewniają znacznie szybszą wymianę danych pomiędzy poszczególnymi podsystemami niż dotychczasowa tradycyjna, nieinteligentna magistrala. Wadą magistrali jest możliwość równoczesnej transmisji danych tylko między dwoma obiektami: procesorem i pamięcią operacyjną, pamięcią operacyjną i sterownikiem dysków, itd. Jeżeli dwa podsystemy próbują transmitować równocześnie, urządzenie o niższym priorytecie musi po prostu zaczekać.

Mikrokanały pozwalają znacznie poprawić ten stan rzeczy, umożliwiając praktycznie równoległą transmisję pomiędzy różnymi podsystemami. Zalety mikrokanałów mogą się jednak wyraźnie ujawnić w warunkach, w których opisane okoliczności zachodzą, a więc np. w systemach wieloprocesorowych lub np. w serwerze sieciowym, w którym intensywnie i równocześnie może transmitować dane oprócz procesora dwa lub więcej sterowników dyskow sztywnych, które w końcu także można rozpatrywać jako specjalizowane procesory. Architektura mikrokanału przewiduje obok procesora głównego do 15 koprocesorów, realizujących transmisję danych w najprzeróżniejszej formie. Natomiast w systemach jednoprocesorowych (a zaliczają się do nich typowe konfiguracje komputerów „biurkowych") mikrokanał nie daje w zasadzie nic istotnego.

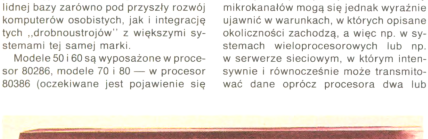
Architektura mikrokanału jest chroniona patentami, a zatem ewentualni chętni do produkcji uzupełniających kart itd. muszą zacząć od zakupu licencji w firmie IBM. Oprócz tego konieczne są specjalizowane układy scalone, obsługujące mikrokanały.

W systemie PS/2 stosowane są wyłącznie stacje dysków elastycznych o średnicy 3,5 cala i pojemnościach 720 KB i 1,44 MB. Nowy format dyskietek charakteryzuje się licznymi zaletami, w szczególności jest znacznie bardziej poręczny i odporny na udary mechaniczne.

Bardzo istotną nowość stanowią w systemie PS/2 także sterowniki graficzne, zwłaszcza karta VGA. W odróżnieniu od mikrokanału karta VGA stała się powszechnie uznanym, przyszłościowym standardem. Oprócz dość wysokiej rozdzielczości (standardowo 640 × 480 punktów) oferuje ona niezwykłe możliwości operowania barwą, w czym zasługa m.in. analogowego systemu sterowania monitora, pozwalającego na bezstopniowy wybór barw.

Komputery rodziny PS/2 wyróżniają się bardzo nowoczesnym rozwiązaniem technologicznym. Oprócz specjalizowanych układów wielkiej skali integracji zastosowano montaż powierzchniowy, znacznie dogodniejszy dla automatycznych linii produkcyjnych. Zmniejszono też o połowę rozstaw zestyków w gniazdach dla kart dodatkowych, co w sumie pozwoliło na znaczną miniaturyzację komputerów rodziny PS/2.

ROLAND WACŁAWEK



„Młody Technik — InforMik" wydaje Instytut Wydawniczy „Nasza Księgarnia"

Rada Redakcyjna: doc. dr Zygmunt Dąbrowski, inż. Jerzy Jasiuk, dr Zygmunt Kalisz, mgr Zbigniew Słowiński, mgr inż. Jerzy Siek, dr Zbigniew Płochocki, Piotr Postawka, mgr inż. Roland Wacławek, prof. dr hab. Andrzej K. Wróblewski (przewodniczący), mgr inż. Grzegorz Zalot.

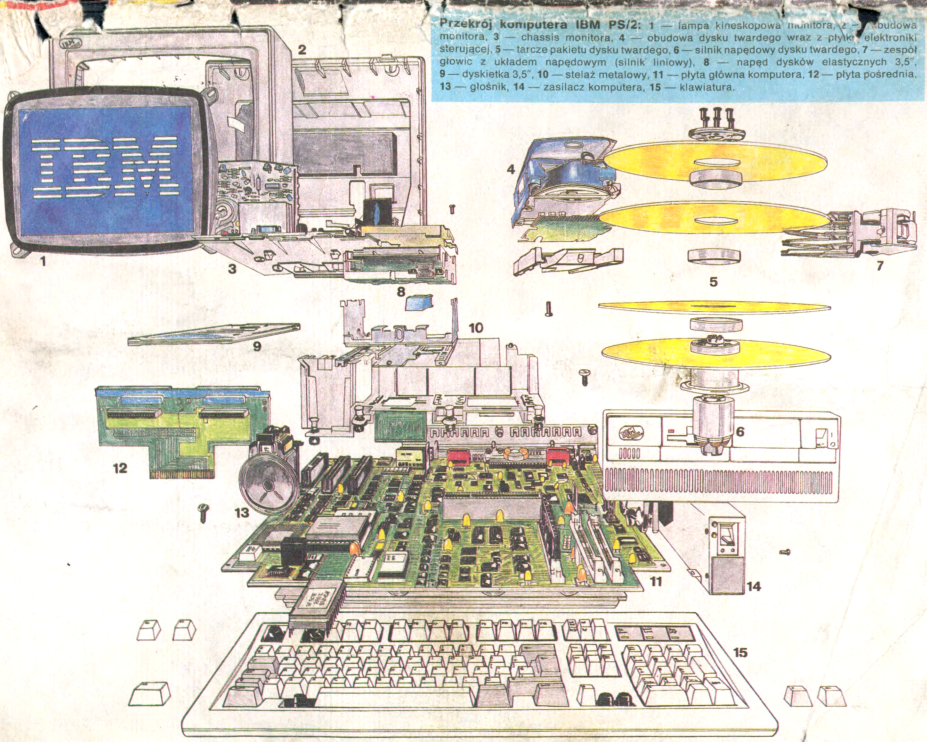
Zespół redakcyjny: *„InforMik"* redaguje zespół *„Młodego Technika"* Jerzy Klawiński (sekretarz red.), Jacek Nowicki (red.), Dariusz A. Przygoda (red.), Lidia Sadowska-Szlaga (korekta), Józef Trzionka (redaktor naczelny), Roland Wacławek (software), Grzegorz Zalot (hardware), Izabela Żur (red. tech.).

Stali współpracownicy: Wojciech Apel, Tadeusz Basista, Jacek Jędrzejowski, Piotr Postawka, Marek Szczepański, Krzysztof Wiśniewski.

Adres redakcji: ul. Spasowskiego 4, 00-389 Warszawa, lub skr. poczt. 380, 00-950 Warszawa. **Telefony:** centrala: 26 24 31 do 36. Dział Łączności z Czytelnikami — wewn. 60, pozostałe działy: wewn. 42 i 47. Redaktor naczelny: 26 26 27 lub wewn. 87.

Redakcja zastrzega sobie prawo adiustacji i skracania nadesłanych materiałów. Artykułów nie zamówionych redakcja nie zwraca.

Druk: Łódzkie Zakłady Graficzne. Zam. 98/13/90
Nakład 10.300 egz.



Przekrój komputera IBM PS/2: 1 — lampa kineskopowa monitora, 2 — obudowa monitora, 3 — chassis monitora, 4 — obudowa dysku twardego wraz z płytą elektroniki sterującej, 5 — tarcze pakietu dysku twardego, 6 — silnik napędowy dysku twardego, 7 — zespół głowic z układem napędowym (silnik liniowy), 8 — napęd dysków elastycznych 3,5", 9 — dyskietka 3,5", 10 — stelaż metalowy, 11 — płyta główna komputera, 12 — płyta pośrednia, 13 — głośnik, 14 — zasilacz komputera, 15 — klawiatura.

Cena zł 3000,–
Indeks nr 366013
PL ISSN 0860-5696