TOWARZYSTWO
NAUKOWE ORGANIZACJI I KIEROWNICTWA
Oddział w Szczecinie

Antoni Nowakowski, Zdzisław Szyjewski
Waldemar Wolski

PROGRAMOWANIE MIKROKOMPUTERÓW

(pod kontrolą systemu operacyjnego CP/M)

SZCZECIN 1985

TOWARZYSTWO NAUKOWE ORGANIZACJI I KIEROWNICTWA
Oddział w Szczecinie

Antoni Nowakowski, Zdzisław Szyjewski
Waldemar Wolski

PROGRAMOWANIE MIKROKOMPUTERÓW
/pod kontrolą systemu operacyjnego CP/M/

SZCZECIN 1985

Recenzent:

Wojciech Olejniczak

621.32 : 621.3.06

S p i s t r e ś c i

W S T Ę P

Coraz szybszy rozwój zastosowań mikrokomputerów w wielu dziedzinach życia gospodarczego i społecznego, a także stały wzrost produkcji mikrokomputerów w kraju i tworzące się w związku z tym zainteresowanie problemami szybkiego "opanowania" sprzętu, powodują potrzebę opracowania różnego rodzaju podręczników prezentujących zasady, sposoby i doświadczenia w zakresie wykorzystania mikrokomputerów.

Prezentowany podręcznik pt. "Programowanie mikrokomputerów pod kontrolą systemu operacyjnego CP/M" ma na celu zapoznanie bezpośredniego użytkownika mikrokomputerem z:

- programowaniem w języku BASIC,
- możliwościami wykorzystania systemu operacyjnego CP/M we własnych programach użytkowych,

na tle ogólnych problemów budowy, klasyfikacji i zastosowań mikrokomputerów, uzupełnionych przeglądem dostępnego w kraju sprzętu i słownikiem terminów mikrokomputerowych.

Przyjęty układ podręcznika umożliwia opanowanie podstawowego języka wyższego rzędu dla mikrokomputerów /BASIC/, niezbędnego dla oprogramowania problemów użytkownika, a jednocześnie użytkownikowi bardziej zaawansowanemu pozwala na poszerzenie możliwości wykorzystania sprzętu poprzez realizację specjalistycznych funkcji systemu operacyjnego.

Od wydawnictw producentów sprzętu /dokumentacji dostarczanej wraz z mikrokomputerem/ podręcznik ten różni się ponadto tym, że przyjęty sposób wykładu wynika z własnych doświadczeń autorów w zakresie wykorzystania mikrokomputerów. Zawiera zatem wyniki i wnioski doświadczeń praktycznych.

Najdogodniejszą formą zdobywania umiejętności praktycznych w zakresie programowania mikrokomputerów jest forma kursu /organizatorem jest TNOiK Szczecin/ w ramach którego słuchacz ma do dyspozycji wykładowcę, podręcznik i sprzęt.

1. BUDOWA I KLASYFIKACJA MIKROKOMPUTERÓW

1.1. Budowa i zasady działania mikrokomputerów

Mikrokomputer zbudowany jest z trzech podstawowych bloków:
- mikroprocesora z układem synchronizacji wewnętrznej,
- pamięci,
- układów służących do sprzęgania mikroprocesora z urządzeniami wejścia-wyjścia,

połączonych za pomocą szyn /por. rys. 1/.

Mikroprocesor koordynuje wszelką działalność systemu, a przede wszystkim pobiera, dekoduje i wykonuje rozkazy umieszczone w pamięci, kontroluje za pośrednictwem szyny sterowania dostęp pozostałych elementów systemu do szyn: danych i adresowej.

Zegar wytwarza ciąg sygnałów wykorzystywanych do synchronizowania działania mikroprocesora. Pojawienie się sygnału z układu zegara inicjuje przejście procesora z jednego stanu do drugiego. Podczas zmiany stanów mikroprocesor wykonuje określone elementarne czynności.

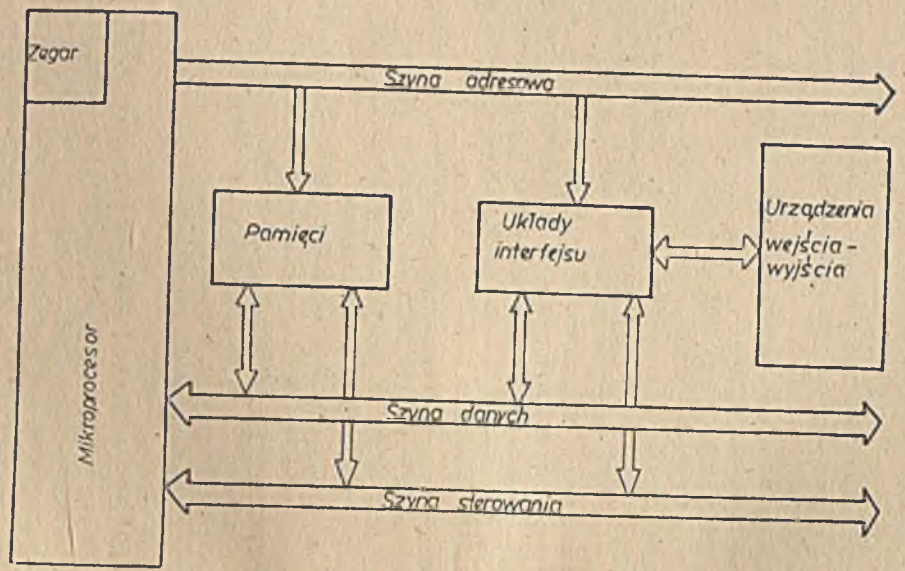
Pamięci służą do przechowywania rozkazów wykonywanych przez mikroprocesor oraz danych, których rozkazy te dotyczą. Pamięć dzielona jest zwykle na stałą tylko odczytywalną /ROM/ oraz pamięć umożliwiającą zapis i odczyt /RAM/.

Układy interfejsu służą do dołączenia do mikrokomputera różnych typów urządzeń wejścia-wyjścia. Najczęściej spotykanymi urządzeniami wejścia-wyjścia w systemach mikrokomputerowych są: monitory ekranowe, pamięci na dyskach elastycznych i kasetach magnetycznych, klawiatury oraz drukarki.

Szyny służą do łączenia wymienionych elementów w system mikrokomputerowy.

Podstawowymi elementami mikrokomputera jest mikroprocesor. Jest to układ scalony o bardzo dużym stopniu integracji zawierający od kilkunastu do kilkuset tysięcy tranzystorów umieszczonych na niewielkiej powierzchni i stanowiących jeden element /całość/. Mikroprocesor w systemie mikrokomputerowym spełnia funkcję jednostki centralnej.

Typowy mikroprocesor zbudowany jest z czterech bloków funkcjonalnych:
- sterowania /interpretacja rozkazów pobieranych z pamięci



Rys.1 Architektura mikrokomputera

- a następnie kontrolowanie na tej podstawie wewnętrznego działania mikroprocesora oraz reakcja na sygnały z zewnątrz, jak również wytwarzanie sygnałów umożliwiających organizowanie współpracy innych elementów systemu z mikroprocesorem/,
- jednostki arytmetyczno-logicznej /ALU/,
- rejestrów służących do zapisu i odczytu danych,
- wewnętrznych szyn przesyłowych wiążących wymienione bloki.

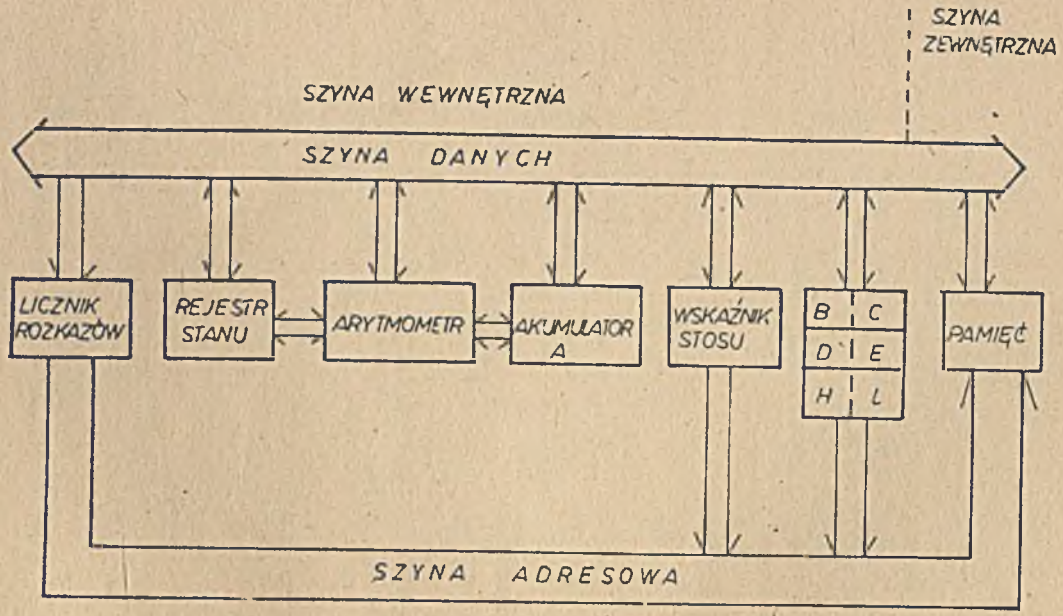
Strukturę funkcjonalną mikroprocesora Intel 8080 ilustruje rys. 2 i na jego przykładzie zostaną przedstawione zasady działania mikroprocesora.

Mikroprocesor Intel 8080 jest mikroprocesorem 8-mio bitowym, co oznacza ogólnie, że operacje wykonywane są na danych obejmujących 8 bitów. Polecenie wykonania operacji określa komórkę pamięci podlegającą modyfikacji oraz rodzaj modyfikacji. Polecenie to przesyłane jest 8 liniami szyny danych. Można ją podzielić na wewnętrzną i zewnętrzną, przy czym szyną wewnętrzna przesyła się dane między elementami mikroprocesora, natomiast szyną zewnętrzną między mikroprocesorem a układami interfesu. Szyny są dwukierunkowe, możliwe jest zatem wysyłanie, jak i przyjmowanie danych przez mikroprocesor.

W każdym mikroprocesorze wyróżnić można szereg rejestrów /urządzeń przechowywania informacji/ spełniających różne funkcje. Zestaw rejestrów roboczych mikroprocesora obejmuje akumulator /A/ oraz 6 rejestrów pomocniczych /B, C, D, E, H i L/. Akumulator jest łącznikiem między mikroprocesorem a pamięcią, umożliwia przesyłanie danych do i z pamięci, jak również na danych znajdujących się w akumulatorze można wykonywać pewne operacje. Adres wybranej komórki pamięci przesyłany jest 16-to bitową szyną adresową, co oznacza możliwość wybrania jednej z 65536 /64 K/ komórek. Jest to górna granica rozmiarów pamięci, w której może być przechowywany program i dane.

Adres 16-to bitowy można uzyskać za pomoc rejestrów pomocniczych, które można wykorzystywać jako poszczególne rejestry 8-mio bitowe i na ich zawartościach wykonywać określone działania oraz jako rejestry 16-to bitowe /poprzez łączenie rejestrów w pary DE, HL/.

Arytmometr, wykonujący obliczenia, jest układem 8-mio bitowym sprzężony z akumulatorem i szyną danych. Bezpośrednie poła-



Rys.2 Struktura funkcjonalna mikroprocesora Intel 8080

czenie arytmometru z akumulatorem oznacza, że ten ostatni uczestniczy w większości operacji arytmetycznych i logicznych, przechowywując zarówno argumenty, jak i wyniki obliczeń pośrednich. Rozkazy logiczne mikroprocesora pozwalają obliczyć sumę logiczną, iloczyn logiczny i różnicę symetryczną. Rozkazy arytmetyczne obejmują dodawanie, dodawanie z przeniesieniem, odejmowanie i odejmowanie z pożyczką. Rozkazy dodawania z przeniesieniem i odejmowanie z pożyczką umożliwiają wykonywanie tzw. działań wielokrotnej precyzji, w których argumenty są zawarte w więcej niż jednym słowie. W każdym przypadku jednym z argumentów lub argumentem wyłącznym jest akumulator, drugim natomiast może być jakikolwiek z rejestrów B, C, D, E, H lub L, komórka pamięci o adresie wskazanym przez zawartość pary HL lub też bezpośredni argument 8-mio bitowy.

Rejestr stanu odzwierciedla wyniki operacji wykonanych przez mikroprocesor jest rejestrem 8-mio bitowym. Praktyczne znaczenie bitów stanu procesora związane jest z możliwością zmiany kolejności wykonywania rozkazów programu.

W przypadku gdy z aktualnego stanu mikroprocesora wynika potrzeba pobrania kolejnego rozkazu lub kolejnego bajtu rozkazu szyna adresowa powinna wskazywać adres odpowiedniej komórki pamięci. Umożliwia to specjalny rejestr zwany licznikiem rozkazów. Jego wartość zwiększa się sekwencyjnie w miarę wykonywania przez mikroprocesor kolejnych rozkazów. Zmiana sekwencji związana jest ze wskaźnikami tworzonymi przez bity rejestru stanu. Każda skokowa zmiana sekwencji wykonywanych rozkazów programu wymaga zmiany zawartości licznika rozkazów. Wartości bitów warunków rejestru stanu, sprawdzane przez mikroprocesor powodują skok /jeżeli warunek jest spełniony/ lub sekwencyjne wykonywanie programu /jeżeli warunek nie został spełniony/.

Wykonując program, który korzysta z podprogramów, występuje sytuacja, w której sterowanie zostaje przekazane do podprogramu. Po jego wykonaniu sterowanie wraca do programu głównego. Wymaga to przechowania informacji o adresie powrotu do programu głównego. Do tego celu służy odrębny rejestr adresowy zwany wskaźnikiem stosu.

W ten sposób przedstawione zostały ideowo zasady funkcjonowania mikroprocesora /na przykładzie mikroprocesora Intel 8080/, które są bardzo zbliżone do zasad działania jednostki centralnej

komputera wykonanego w tradycyjnej technologii. Do rozważań wybrano niektóre elementy mikroprocesora, natomiast na rys. 3 przedstawiono budowę wewnętrzną tego mikroprocesora. Kostka mikroprocesora zawiera 40 końcówek /ich opis zawiera tablica 1/ i jest wykonana w technologii NMOS.

Tak zbudowany i funkcjonujący mikroprocesor komunikuje się ze swoim otoczeniem, które obejmuje program i sprzęt /zgodnie z rys. 1 - zegar, pamięci, układy interfejsu/. Sygnały wychodzące z mikroprocesora powinny przekazywać informacje o stanie i czynnościach wykonywanych aktualnie przez mikroprocesor. Natomiast sygnały wejściowe powinny umożliwiać zmianę sekwencji sterowania i wymuszenia pewnych stanów mikroprocesora.

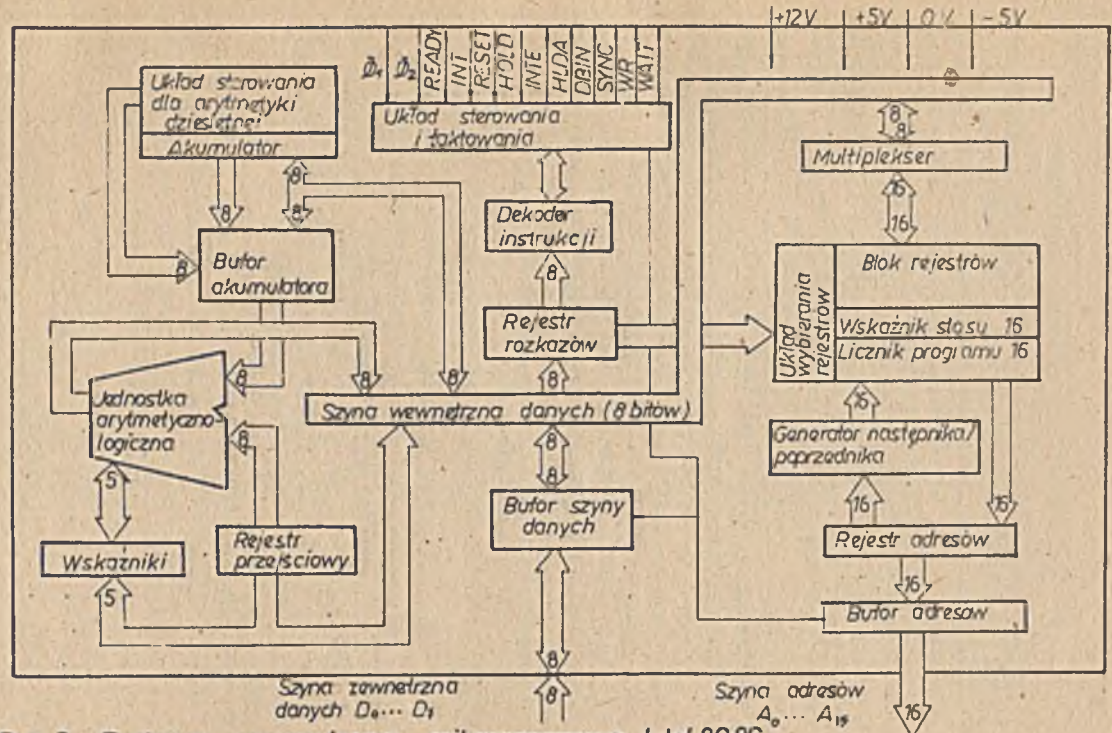
W mikroprocesorze Intel 8080 wyróżnia się następujące sygnały sterujące:

- sygnał kasowania RESET,
- sygnały zegarowe Φ_1 i Φ_2,
- sygnał synchronizacji SYNC,
- sygnały stanu oczekiwania WAIT i READY,
- sygnały blokady HOLD i HLDA,
- sygnały przerwania INT i INTE,
- sygnały sterowania szyny danych - DBIN,
- sygnał zapisu - WR.

Stan tych sygnałów jest początkowy w wewnętrznym rejestrze sumowany logicznie tak aby na wyjściu rejestru pojawiła się informacja:

- sygnał odczytu danych z pamięci MEMR,
- sygnał zapisu danych do pamięci MEMW,
- sygnał odczytu układu wejście/wyjście I/OR,
- sygnał zapisu do układu wejście/wyjście I/OW,
- sygnał potwierdzenia przyjęcia przerwania INTA.

Wymienione sygnały tworzą szyny sterowania mikroprocesora. Wykonanie dowolnego rozkazu składa się z dwóch etapów - pobranie rozkazu i wykonanie rozkazu /realizacja funkcji wskazanej w kodzie operacji/. Kod operacji mieści się w pierwszym bajcie każdego rozkazu, pozostałe bajty zawierają adresy operandów bądź operandy. Każdorazowe odwołanie się do pamięci zewnętrznej, wymagające wysłanie adresu oraz zapisu lub odczytu jednego słowa pamięć, jest wykonywana w tak zwanym cyklu maszynowym. Wyróżnić można kilka typów cykli maszynowych ze względu na kierunek,



Rys.3 Budowa wewnętrzna mikroprocesora Intel 8080

Tablica 1

Opis końcówek mikroprocesora Intel 8080

Oznaczenie i kierunek przesyłania	Opis końcówek
$A_0 \dots A_{15}$ /wyjścia/	Końcówki wyjściowe szyny adresów. Po pojawieniu się sygnału wejściowego HOLD mikroprocesor może przejść w stan odcięcia, tzn. wprowadza szynę adresów $A_0 \dots A_{15}$ i szynę danych $D_0 \dots D_7$ w stan dużej impedancji i jednocześnie wyprowadza sygnał HLDA informujący, że sygnał HOLD został przyjęty
$D_0 \dots D_7$ /wejścia/wyjścia/	Końcówki dwukierunkowej, zewnętrznej szyny danych. Można je wprowadzić w stan dużej impedancji po przyjęciu przez mikroprocesor sygnału HOLD. Przy wyprowadzaniu danych przez tę szynę pojawia się jednocześnie sygnał na końcówce $\overline{WR}$, natomiast przy wprowadzaniu danych pojawia się sygnał na końcówce DBIN
1, 2	Końcówki wejściowe zegara
DBIN /wyjście/	Pojawienie się sygnału na tej końcówce oznacza, że mikroprocesor znajduje się w stanie czytania informacji z szyny danych $D_0 \dots D_7$
$\overline{WR}$ /wyjście/	Pojawienie się sygnału na tej końcówce oznacza, że mikroprocesor wyprowadza informacje przez szynę danych $D_0 \dots D_7$ do pamięci lub urządzeń zewnętrznych
HOLD /wejście/	Wejście sygnału wstrzymania. Sygnał na tej końcówce powinien trwać co najmniej pięć cykli zegarowych, aby mikroprocesor mógł przyjąć żądanie wstrzymania. Po przyjęciu żądania, które jest możliwe po zakończeniu przesyłania z /do/ pamięci /czyli sygnał READY jest aktywny/, końcówki szyny danych i adresów wprowadzane są w stan dużej impedancji

Oznaczenie i kierunek przesyłania	Opis końcówek
HIDA /wyjście/	Sygnał wyjściowy oznaczający przyjęcie sygnałów wstrzymania
WAIT /wyjście/	Sygnał wyjściowy oznaczający czekanie na zakończenie przesyłania z /do/ pamięci
READY /wejście/	Sygnał wejściowy oznaczający zakończenie przesyłania z /do/ pamięci
RESET /wejście/	Sygnał wejściowy używany do zapoczątkowania pracy mikroprocesora. Powinien trwać co najmniej trzy cykle zegara. Po pojawieniu się sygnału na końcówce RESET licznik programu jest ustawiany na zero i rozpoczyna się normalny cykl maszynowy. Stany pozostałych rejestrów mikroprocesora nie ulegają zmianie
INT /wejście/	Sygnał wejściowy oznaczający żądanie przerwania. Mikroprocesor może przyjąć to żądanie po zakończeniu wykonywania instrukcji
INTE /wyjście/	Końcówka wyjściowa, na którą wysyła się sygnał zezwalający na przerwanie. Sygnał ten pojawia się na końcu każdego cyklu rozkazowego mikroprocesora. Jeśli w czasie, gdy sygnał na końcówce INTE jest aktywny pojawi się sygnał na końcówce INT oznaczający żądanie przerwania, to mikroprocesor rozpocznie wykonywanie sekwencji operacji należących do obsługi tego przerwania
SYNC /wyjście/	Sygnał na tej końcówce wyprowadzany przez mikroprocesor na początku każdego cyklu maszynowego. Pojawienie się sygnału oznacza, że na zewnętrznej szynie danych $D_0 \ldots D_7$ jest wyprowadzana informacja identyfikująca bieżący cykl maszynowy
+12V, -5V, +5V, DV /GND/ /wejścia/	Zasilanie

miejsce i rodzaj przesłania danych:

- cykl pobrania kodu operacji,
- cykl odczytu z pamięci,
- cykl zapisu do pamięci,
- cykl odczytu ze stosu,
- cykl zapisu na stos,
- cykl wejścia,
- cykl wyjścia,
- cykl przerwania,
- cykl zatrzymania.

Każdy cykl maszynowy składa się z kilku etapów realizacji zwanych stanami /taktami/. W poszczególnych stanach wykonuje się elementarne przesłanie i operacje arytmetyczno-logiczne. Każdy stan odpowiada jednemu cyklowi dwufazowego zegara taktującego pracę mikroprocesora.

Układy wejścia-wyjścia zawierają wiele rejestrów, poprzez które mikroprocesor komunikuje się z urządzeniami wejścia-wyjścia. Rejestry te nazywamy portami wejścia-wyjścia. Mikroprocesor ma możliwość realizacji funkcji adresowania, zapisywania oraz odczytywania portów, podobnie jak komórek pamięci. Porty buforują dane wysyłane i odbierane pomiędzy urządzeniami wejścia -wyjścia a mikroprocesorem.

Mikroprocesor może współpracować z urządzeniami wejścia-wyjścia na zasadzie:

- bezpośredniej /prowadzona tylko ze specyficznymi urządzeniami wejścia-wyjścia, które nie wymagają wymiany informacji sterujących procesem przesyłania danych/,
- potwierdzeń wzajemnych /oprócz danych przesyłane są również informacje na podstawie których określa się gotowość urządzenia wejścia-wyjścia i mikroprocesora do przesłania danych/,
- przerwań /współpraca ta przebiega podobnie jak praca z potwierdzeniami wzajemnymi, wsparta jest dodatkowo sygnałem przerwania. Praca z przerwaniami przebiega z reguły w dwóch fazach: inicjacji i podtrzymywania transmisji za pomocą sygnałów przerwań/,
- bezpośredniego dostępu do pamięci /mikroprocesor jedynie inicjuje przesłanie danych między urządzeniem wejścia-wyjścia a pamięcią, nie pośrednicząc dalej w tej operacji, a

następnie otrzymuje informację o zakończonej transmisji na zasadzie przerwania/.

W praktyce trudno wyróżnić wymienione sposoby w czystej formie /w pracy z przerwaniami korzysta się z sygnałów potwierdzeń, a w pracy bezpośredniego dostępu z sygnału przerwania/.

System przerwań został wprowadzony w komputerach w celu zapewnienia możliwości przekazania sterowania do programu obsługi urządzenia wejścia/wyjścia w chwili ustawienia stanu gotowości.

System przerwań oprócz identyfikacji źródła przerwania powinien umożliwiać:

- zapamiętanie bieżącego stanu przerwanego programu i odtworzenie go po zakończeniu obsługi przerwania,
- wznowienie przerwanego programu po zakończeniu obsługi przerwania,
- obsługę wielu przerwań powodowanych przez różnorodne przyczyny.

Stosowanie przerwań powoduje, że:

- dane mogą być wysyłane i przyjmowane podczas wykonywania innych zadań,
- możliwa jest obsługa wielu urządzeń wejścia/wyjścia i zdarzeń zewnętrznych,
- program pracuje niezależnie od powolnej i asynchronicznej pracy urządzenia.

Przedstawione ogólne zasady działania mikroprocesorów mogą odbiegać od spotykanych w konkretnych mikroprocesorach /można ich naliczyć obecnie kilkadziesiąt/, które różnią się między sobą między innymi: długością słowa, sposobami adresowania, możliwością wykonywania operacji wejścia/wyjścia, repertuarem rozkazów, szybkością ich realizacji, strukturą wewnętrzną i technologią wykonania.

1.2. Klasyfikacja mikrokomputerów

Mikrokomputery zbudowane na bazie mikroprocesorów kojarzą się przede wszystkim z pojęciem angielskim personal computer i polskim komputer osobisty. Jednak dokładniejsza klasyfikacja sprzętu mieszczącego się w tych pojęciach nie jest jeszcze precyzyjna.

Komputery osobiste - jest to kierunek zastosowań informatyki

związany z mikrokomputerami, a co ważniejsze z obsługą indywidualnych potrzeb pojedynczego człowieka /tzn. odbiorcy masowego i nie związanego z informatyką/. Oznacza to, że jako sprzęt muszą być proste w obsłudze przez bezpośredniego użytkownika i łatwe w oprogramowaniu również przez bezpośredniego użytkownika. W obsłudze i programowaniu nie występują pomiędzy sprzętem a użytkownikiem odpowiednie służby informatyczne tak charakterystyczne dla dotychczasowych, tradycyjnych zastosowań informatyki. Od strony oprogramowania systemowego i narzędziowego powinny umożliwiać użytkownikowi zaspokojenie jego różnorodnych potrzeb związanych zarówno w życiu domowym jak i zawodowym.

Według tej ostatniej przesłanki mikrokomputery podzielić można na:

- nieprofesjonalne, wspomagające różne czynności życia domowego użytkownika /obliczenia, planowanie budżetu, gry telewizyjne itp./, mające charakter wyposażenia mieszkania,
- profesjonalne, wspomagające obliczenia i czynności zawodowe.

Wśród mikrokomputerów nieprofesjonalnych wyróżnia się zazwyczaj:

- mikrokomputery kieszonkowe, spełniające funkcje praktycznych kalkulatorów programowanych,
- mikrokomputery walizkowe /portable computer/, których podstawową cechą jest możliwość łatwego przenoszenia dzięki integracji podstawowego zestawu /mikroprocesor, klawiatura, wyświetlacz, pamięć zewnętrzna/,
- mikrokomputery domowe /home computer/ charakteryzujące się wykorzystaniem urządzeń domowych /odbiornik telewizyjny i magnetofon kasetowy jako urządzenie wejścia-wyjścia/, pełnym zestawem urządzeń zewnętrznych, możliwościami realizacji obliczeń zawodowych /ale w domu/.

Mikrokomputery profesjonalne można podzielić według kryterium długości słowa na:

- mikrokomputery profesjonalne 8-mio bitowe,
- mikrokomputery profesjonalne 16-to bitowe,
- mikrokomputery profesjonalne 32-u bitowe.

Charakterystykę mikrokomputerów według przedstawionej klasyfikacji ilustruje tablica 2. Wskazuje ona, że kolejne grupy mikrokomputerów charakteryzują się wzrastającymi zasobami sprzęto-

Tablica 2

Klasyfikacja mikrokomputerów

Nazwa	Hardware	Software	Możliwości rozbudowy	Przykłady
I Mikrokomputery kieszonkowe	1. PaO: 0,5 - 10 KB 2. Wyświetlacz: 1 x 40 znaków 3. Klawiatura mała	BASIC /ROM/	podłączenie kasety	PC 1500 Sharp
II Mikrokomputery małoobce /portable computer/	1. PaO: 16 KB 2. Wyświetlacz: /4-8/ x 40 zn. 3. Klawiatura: mała	BASIC /ROM/	podłączenie kasety - mikrodysk elastyczny	FUTURE 1 RADIO SHACK 100 HYSON PX-8
III Mikrokomputery domowe /home computer/	1. PaO: 16-64 KB 2. Wyświetlacz: monitor TV /16-24/x /40-64/ znaków 3. Semigrafika 160 x 96 punktów 4. Klawiatura: standardowa 5. Pamięć masowa: kaseta	BASIC, LOGO, FORTH /ROM/	dysk elastyczny mała drukarka	IX Sinclair
IV Mikrokomputery profesjonalne 8 bitowe	1. PaO: 64-128 KB 2. Monitor CRT /16-24/x/40-80/ 3. Grafika 192x280 punktów 4. Klawiatura - standardowa 5. Pamięć masowa - dyski elastyczne 6. Drukarka: standardowa	Dyskowy system jednozadaniowy BASIC, PASCAL	dysk twardy specjalizowane układy wejścia-wyjścia	APPLE II CompAS-8
V Mikrokomputery profesjonalne 16 bitowe	1. PaO: 64 KB - 1 MB 2. Monitor CRT /25-27/ x /80-132/ znaki 3. Grafika 640 x 240 4. Klawiatura - standardowa + klawisze programowe 5. Pamięć masowa - dyski elastyczne i twarde 6. Drukarka - standardowa	Dyskowy system operacyjny wielozadaniowy BASIC, FORTRAN PASCAL itp. Oprogramowanie dialogowe	Adaptery sieci lokalnych specjalizowane układy wejścia -wyjścia	LISA, IBM PC/XT IP 150
VI Mikrokomputery profesjonalne 32 bitowe	1. PaO do 4 GB 2. Monitor CRT grafika 3. Klawiatura - standardowa 4. Pamięć masowa - dyski twarde 5. Drukarka standardowa	Dyskowy operacyjny Duży zestaw języków oprogramowania Oprogramowanie dialogowe	specjalizowane układy wejścia -wyjścia	MELMAC - 32 IP 9000

wymi i programowymi.

W podziale mikrokomputerów według długości słowa mikroprocesora wyróżnić jeszcze należy rozwiązania pośrednie tzn.

- mikroprocesory pseudo 16-to bitowe, posiadające 8-mio bitowe szyny przesyłania danych lecz 16-to bitowe rejestry i arytmometr,
- mikroprocesory pseudo 32-u bitowe posiadające 16-to bitowe szyny przesyłania danych i 32-u bitowe rejestry.

Przedstawiona na rys. 1 architektura mikrokomputera jest architekturą logiczną, pokazuje bowiem bloki funkcjonalne mikrokomputera i powiązania między nimi. Architektura fizyczna mikroprocesorów może być podstawą klasyfikacji mikrokomputerów. Przyjmując to kryterium wyróżnić można dwie grupy mikrokomputerów:

- monolityczne /jednopłytkowe/, które w jednym układzie scalonym zawierają kilka lub wszystkie elementy architektury logicznej mikrokomputera. Przykładem takiego rozwiązania może być mikrokomputer Intel 8048, który zawiera na jednym płatku krzemu: mikroprocesor, pamięć RAM /o pojemności 64x 8/, pamięć ROM /o pojemności 1K x 8/, układ wejścia-wyjścia zorganizowany w 3 porty 8-mio bitowe oraz 3 testowane programowo linie wejścia/wyjścia, oscylator i 8-mio bitowy czasomierz/licznik. Mikrokomputery monolityczne są układami o właściwościach ściśle określonych przez producenta, a użytkownik może wybierać ten czy inny mikrokomputer spełniający żądane funkcje;
- segmentowe /modułowe/, w których realizacja funkcji jednego elementu struktury logicznej przebiega w wielu układach scalonych, tzn. jako części mikrokomputera występują elementy pokazane na rys. 1, a dodatkowo jeszcze pewne układy towarzyszące. Dla mikroprocesora Intel 8080 układami towarzyszącymi są układ zegarowy 8224 i sterownik szyny sterującej 8228.

Przyjmując za kryterium podziału typ zastosowania mikrokomputera wyróżnić wśród nich można:

- mikrokomputery uniwersalne,
- mikrokomputery przeznaczone do specjalnych zastosowań.

Wśród uniwersalnych zastosowań /przez analogię do dziedzin zastosowania tradycyjnych komputerów/ wyróżnić można:

- obliczenia naukowo-techniczne,
- przetwarzanie danych gospodarczych,

o ile dotyczą one powszechnie stosowanych algorytmów i powszechnie występujących dziedzin przetwarzania. Jako przykład mikrokomputera uniwersalnego w zastosowaniach gospodarczych przytoczyć można ELWRO 523, którego konfiguracja wskazuje, że jest to uniwersalny mikrokomputer biurowy.

Wśród specjalnych zastosowań mikrokomputerów wyróżnić można:
- sterowanie procesami przemysłowymi,
- wspomagane mikrokomputerem wytwarzanie,
- wspomagane mikrokomputerem projektowanie,
- dydaktyka,
- systemy telekomunikacyjne,
- systemy techniki pomiarowej itp.

Podobnie jak w przypadku tradycyjnych systemów komputerowych odniesieniu do mikrokomputerów wyróżnić można:
- systemy jednomikroprocesorowe,
- systemy wielomikroprocesorowe,

System wielomikroprocesorowy jest to system, w którym występują:
- co najmniej dwa mikroprocesory,
- wspólna pamięć,
- wspólne układy interfejsowe wejścia-wyjścia.

Systemy wielomikroprocesorowe z dzieloną szyną lokalną umożliwiają współpracę kilku mikroprocesorów /w przypadku mikroprocesora Intel 8086 trzech mikroprocesorów/. Jeden z nich pełni rolę nadrzędną i nadzoruje dostęp do szyny lokalnej dwóch pozostałych, z których jeden ma wyższy priorytet niż drugi. Konieczność synchronizacji wszystkich mikroprocesorów powoduje, że w systemie znajduje się wspólny układ zegarowy /8284/. Głównym przeznaczeniem omawianej architektury jest łączenie mikroprocesorów specjalizowanych /np. mikroprocesora centralnego Intel 8086 ze specjalizowanym mikroprocesorem wejścia-wyjścia Intel 8089 lub mikroprocesora centralnego Intel 8086 z koprocesorem arytmetycznym Intel 8087, który rozszerza jego listę rozkazów o złożone operacje arytmetyczne.

Systemy wielomikroprocesorowe z dzieloną /w czasie/ szyną systemową umożliwiają połączenie w systemie kilku mikroprocesorów centralnych. Zasoby systemu są podzielone na prywatne i sy-

stemowe /współdzielone/. Do zasobów prywatnych dostęp ma tylko jeden mikroprocesor centralny. Poszczególne mikroprocesory wykonują zadania autonomiczne wykorzystując swoje zasoby prywatne. Komunikacja między mikroprocesorami wymaga wykorzystania zasobów systemowych. Po stwierdzeniu, do której przestrzeni adresowej - prywatnej czy systemowej należy dany adres, uaktywniane są odpowiednie układy sterowania. W przypadku szyny prywatnej dokonuje się przesłanie danej. W przypadku szyny systemowej drugim warunkiem przesłania, jest stwierdzenie że szyna jest wolna.

Przyjmując za kryterium podziału możliwość łączenia kilku mikrokomputerów w system, wyróżnić można:

- systemy jednomikrokomputerowe,
- systemy wielomikrokomputerowe /lokalne sieci mikrokomputerowe/.

System mikrokomputerowy obsługuje jeden, autonomiczny strumień danych wejściowych. Mikrokomputer dysponuje własnym oprogramowaniem i wykonuje własne, określone zadania. W systemach wielomikrokomputerowych występuje komunikacja pomiędzy poszczególnymi mikrokomputerami na poziomie danych lub rozkazów. Systemy wielomikrokomputerowe obsługują wiele zadań słabo ze sobą powiązanych a dotyczących określonej organizacji gospodarczej na ograniczonym obszarze /lokalne sieci danych/.

Ponadto mikrokomputery mogą współpracować z dużymi komputerami lub sieciami komputerowymi.

1.3. Urządzenia zewnętrzne systemów mikrokomputerowych

Do podstawowych urządzeń zewnętrznych mikrokomputerów zaliczyć należy:

a/ monitory ekranowe,
b/ pamięci zewnętrzne na dyskach i kasetach magnetycznych,
c/ drukarki.

Ad a/ Terminal ekranowy umożliwia:

- dialog użytkownika z systemem,
- prezentację informacji,
- wstępną obróbkę tekstów.

Terminale te podzielić można na:

- monitory ekranowe zależne, przeznaczone do pracy w systemach monitorowych,

- monitory ekranowe niezależne, wyposażone w interfejsy szeregowe umożliwiające bezpośrednią wymianę danych z komputerem i przeznaczone do pracy jako końcówki operatorskie,
- monitory ekranowe graficzne przeznaczone do graficznego prezentowania informacji,
- terminale ekranowe inteligentne wyposażone w rozbudowaną pamięć operacyjną oraz zestaw urządzeń wejścia-wyjścia, pozwalające na wykonanie przez monitor ekranowy wielu czynności realizowanych uprzednio przez komputer.

Aktualnie w systemach mikrokomputerowych można wykorzystać następujące monitory ekranowe produkcji krajowej:

- monitor ekranowy 7952N, przeznaczony do pracy jako konsola operatora w systemach mikrokomputerowych, pojemność ekranu 1920 znaków,
- monitor ekranowy CM 7209/7953N/ przeznaczony do pracy w systemach mikrokomputerowych SM3, MERA-60, PDP-11 i inne, pełna emulacja terminalu ekranowego VT-52, pojemność ekranu 1920 znaków.

Ad b/ Charakterystykę wybranych pamięci taśmowych i dyskowych przedstawiają tabele 3 i 4.

Produkowana przez MERA-KFAP pamięć SP60M na dysku elastycznym /może ona współpracować z PDP-11/03, PDP-11V03 i innymi posiadającymi analogiczny interfejs/ posiada następującą charakterystykę.

Pamięć zawiera: jednostkę pamięci PL x 45D, selektor, formator /kontroler/ połączone kablem z adapterem interfejsu umieszczonym w mikrokomputerze.

Jednostka pamięci PL x 45D umożliwia zapis i odczyt szeregowej informacji na dysku elastycznym. Jednostka pamięci może obsługiwać dwa dyski elastyczne. Uruchomienie jednostki pamięci, wymianę dysków, wybór żądanej strony danego dysku umożliwiają przełączniki.

Przetwarzanie informacji odczytanej lub przeznaczonej do zapisu, a także sterowanie ruchem głowicy odbywa się z formatora i selektora. Formator przekształca dane wysyłane z mikrokomputera ma postać dogodną do zapisu na dysku, a także przygotowuje dane odczytane z dysku do wysłania do mikrokomputera. Ponadto zadaniem formatora jest

w przypadku odczytu odszukanie danych na dysku lub w przypadku zapisu odszukanie żądanego sektora, w którym dane należy umieścić. Selektor steruje ruchem wybranej głowicy jednostki pamięci na ścieżkę o żądanym adresie, ustawia głowicę i informuje formator o gotowości PL x 45D do poprawnej pracy - odczytu lub zapisu oraz wybiera dane.

Tablica 3

Charakterystyka magnetycznych pamięci taśmowych

Lp.	Typ urządzenia	Pojemność /bajtów/	Szybkość przesyłania danych /bajtów/sek	Poziom cen*
1.	Mikrokasety	5K - 20K	100 - 300	30 - 80 %
2.	Pełnowymiarowe kasety	30K - 150K	300 - 4800	50 - 300 %
3.	Mikrotaśmy /typ cartridge/	60K - 150K	500 - 2000	100 - 300 %
4.	Taśmy typ cartridge /standard/	3M - 65M	1K - 12K	600 - 2000 %

Tablica 4

Charakterystyka pamięci na dyskach magnetycznych

Lp.	Typ urządzenia	Pojemność /bajtów/	Średni czas dostępu	Poziom cen*
1.	Mikrodyskietka - 3"	8 - 20K	800 milisek	75 - 1500 %
2.	Minidyskietka - 5,25"	40 - 200K	800 milisek	250 - 500 %
3.	Dyskietka 8"	150 - 3MB	400 milisek	400 - 1200 %
4.	Mini dysk 4,25"	3 - 6MB	100 milisek	800 - 2000 %
5.	Twardy dysk 8"	10 - 40MB	80 milisek	1800 - 4000 %
6.	Twardy dysk 14"	20 - 80MB	60 milisek	1500 - 6000 %
7.	Twardy dysk stały	2 - 40MB	10 milisek	5000 - 20000 %
8.	Dysk 8" - typ Winchester	5 - 80MB	70 milisek	1000 - 8000 %
9.	Dysk 5,25" typ Winchester	5 - 60MB	65 milisek	800 - 600 %

* cena łącznie z jednostką sterującą

Nośnikiem informacji w pamięci SP60 M są dyski elastyczne o pojemności na jednym jednostronnie zapisanym dysku 2,05 M bita, średni czas dostępu 210 ms, ilość dysków - 2.

Pamięć kasetowa PK-3 /SM-5214/, produkowana przez Warszawskie Zakłady Urządzeń Informatyki MERAMAT, posiada następującą charakterystykę: kaseta typu "Compact", gęstość zapisu 32 bity/mm, zapis jednoznakowy, ilość ścieżek - 2, nominalna przerwa międzyblokowa 20,3 mm, szybkość transmisji 8000 b/sek, odczyt zapisanej informacji w obu kierunkach.

Ad c/ Charakterystykę wybranych drukarek mikrokomputerowych przedstawia tablica 5.

Zakłady Mechaniczno-Precyzyjne MERA-BŁONIE, główny producent drukarek, oferuje 3 typy drukarek możliwych do wykorzystania w systemach mikrokomputerowych D-180, D-200 i D-100.

Drukarka D-200 - jest uderzeniową drukarką mozaikową średniej szybkości, z 9-igłową głowicą drukującą. Wyposażona jest w pamięć buforową do zapisu pełnego wiersza znaków /co umożliwia wydruk znaków podczas ruchu głowicy w obie strony/. Podstawowe dane techniczne drukarki D-200:

- prędkość liniowa druku 180 zn/s,
- zestaw znaków - dowolny do 256 znaków,
- gęstość pozioma druku 10,12 lub 16 zn/cal,
- gęstość pionowa druku 6,8 lub 10 w/cal,
- ilość znaków w wierszu 132 - 210,
- papier - obrzeżnie perforowany lub bez perforacji o szerokości od 4 do 17 cali,
- ilość kopii 4,
- rodzaj druku - normalny, szeroki, pochyły, szeroko-pochyły, wysoki, wysokoszeroki, o normalnej i podwójnej intensywności oraz wyrazisty, druk semigraficzny i plotowanie.

Nieco mniejsze parametry techniczne posiada drukarka D-100.

Tablica 5

Charakterystyka drukarek mikrokomputerowych

Lp.	Producent Model	Cena	Szybkość druku zn/sek	Ilość znaków w linii	Uwagi i komentarze
1.	EPSON AMERICA				
	MX 80	645 $	80	80	Interfejs szereg. i równoleg.
	MX 100	995 $	80	233	-"-
2.	OLIVETTI				
	PRAXIS 35	600 $	80	132	-"-
3.	Radio Shack				
	V	1860 $	160	132-163	-"-
	VI	300 $	30	40-80	-"-
	Daisywheel - Z	1960 $	43	163	-"-
4.	OKIDATA				
	Microline 82A	840 $	120	80-132	-"-
5.	FUJITSU				
	SP 830	2120 $	80	130	-"-

Przedstawione urządzenia zewnętrzne systemów mikrokomputerowych należą do urządzeń tradycyjnych, chociaż w każdej z wymienionych grup można zauważyć szybki rozwój konstrukcji i technologii np. dyski elastyczne o podwójnej gęstości zapisu, monitory graficzne o dużej rozdzielczości obrazu itp. Pojawiają się ponadto urządzenia niekonwencjonalne, które obecnie wykorzystywane są na zasadzie eksperymentu, ale które mogą wprowadzić nową jakość w zastosowaniach mikrokomputerów np.: urządzenia rozpoznawania i syntezy mowy.

1.4. Tendencje rozwoju mikrokomputerów

Rozwój mikrokomputerów odbywa się w zawrotnym tempie w ciągu kilkunastu ostatnich lat /Intel 4004 - 1971 r./, wyróżnia się już po kilka generacji technologicznych, czy też architektonicznych mikrokomputerów. Rozwój ten dokonuje się pod wpływem dążenia do realizacji następujących celów:

a/ obniżenie kosztów produkcji i eksploatacji środków technicznych, przy jednoczesnym wzroście niezawodności sprzętu,

b/ wzrostu wydajności i zasobów sprzętu,

c/ potrzeby dostosowania środków informatyki do specyfiki bezpośrednich zastosowań /użytkowników/,

d/ zapewnienia łatwości i prostoty obsługi.

Ad a/ O tendencji tej znanej i w tradycyjnej informatyce świadczyć mogą następujące informacje.

Mikroprocesor 4-bitowy, stosowany w kalkulatorach kieszonkowych kosztował w 1977 r. - 6 dol, obecnie kosztuje mniej niż 2 dol., a przewiduje się, że w latach 1987 - 97 jego cena spadnie do 1 - 0,35 dol.

Mikroprocesor 8-bitowy: 1977 r. - kilkanaście dol., 1982 r. - 3 dol., 1997 r. - 1 dol.

Mikroprocesor 16-bitowy: 1977 r. - około 100 dol., obecnie 10 dol., 1997 r. 1-2 dol.

Podobne tendencje można zauważyć w pamięciach półprzewodnikowych. Wolne pamięci w 1977 r. kosztowały 0,08 centa/bit, w 1987 r. przewiduje się cenę 0,001-0,002 centa/bit, a w roku 1992 - 0,0001-0,0005 centa/bit. Ceny pamięci o średniej prędkości wynosiły w 1977 r. 0,2 centa/bit, a

przewiduje się w 1987 r. 0,003-0,004 centa/bit i w 1992 r. - 0,0005-0,001 centa/bit. Ceny pamięci szybkich kształtowały się w 1977 r. 0,7 centa/bit, w 1987 r. powinny kosztować 0,015-0,025 centa/bit, a w 1992 - 0,0007-0,0035 centa/bit.

Przewidywane obniżki cen wynikają ze stosowania układów scalonych o coraz większej skali integracji, nowych technologii wytwarzania oraz nowych materiałów /np. zamiast krzemu - arsenek galu/.

Ad b/ W tym zakresie wyróżnić można szereg symptomów.

Pierwszy objaw to wzrost ilości mikroprocesorów o coraz dłuższym słowie. Powszechne zastosowanie znajduje obecnie mikroprocesory 16-bitowe, a w fazie wprowadzania do powszechnego użytku znajdują się mikroprocesory 32-bitowe, tworząc coraz większe zasoby i możliwości budowanych w oparciu o te mikroprocesory systemy.

Ilość układów logicznych i pamięciowych w jednej kostce układu scalonego zwiększa się średnio czterokrotnie w ciągu 2 lat. Powstała nowa technologia umożliwiając budowę "trójwymiarowej" kostki poprzez nakładanie układów scalonych na siebie. Doświadczalne kostki z arsenku glinu są kilkakrotnie szybsze /5-10 razy/ od układów krzemowych.

Pojemności pamięci zewnętrznych osiągają następujące wielkości: dysk o średnicy 3 1/4" - 0,4 Mbajty, dyski stałe Winchester o średnicy 14" - 600 Mbajtów, dyski optyczne o średnicy 14" - 4 G bajtów. Podobne tendencje zauważyć można we wszystkich grupach urządzeń wejścia-wyjścia.

Ad c/ Zastosowania mikrokomputerów powstają zwykle w wyniku wyraźnego zapotrzebowania użytkownika: inżyniera, projektanta, księgowego, magazyniera, dydaktyka itp. i spełniają jego specyficzne wymagania. Stąd dające się zaobserwować różnorodność zastosowań i funkcjonujących w nich mikrokomputerów, jako autonomicznych stanowisk pracy. Kolejne fazy rozwoju tych zastosowań to łączenie mikrokomputerów w sieci oraz podłączanie mikrokomputerów do sieci jako inteligentnych terminali.

Według założeń opracowanych przez Japończyków mikrokomputer w najbliższym czasie powinien wykonywać 2 miliony rozkazów na sekundę, posiadać od 0,5 do 5 M bajtów pamięci operacyjnej oraz

pamięć dyskową o pojemności ponad 100 M bajtów z czasem dostępu około 1 milisekundy. Potrzeby użytkowników zmierzające do tworzenia systemów umożliwiających "inteligentną" konwersację z mikrokomputerem stwarzają jeszcze większe wymagania trudno wyobrażalne dzisiaj. Przez analogię do rozwoju tradycyjnej informatyki zaczyna mówić się o supermikrokomputerach, a połączenie dwóch słów super i mikro w jednym słowie wydaje się paradoksem.

2. STRUKTURA SYSTEMU OPERACYJNEGO CP/M

2.1. Znaczenie systemu operacyjnego

Sprzęt nowoczesnego mikrokomputera jest rozbudowany i można go stosować do wielu celów, ale sprzęt ten jest połączony z otoczeniem w sposób niewygodny lub trudny do wykorzystania. W celu poskromienia tej "gołej maszyny" stworzone zostały systemy operacyjne, które potrafią zarządzać podstawowymi zasobami sprzętowymi i zapewniają lepsze łącze z użytkownikami oraz ich programami. Systemy operacyjne mają tak istotny wpływ na efektywność działania mikrokomputera, że wiele osób uważa je za nieodłączne od sprzętu. Aby wykonać swe zadanie, użytkownik musi współdziałać z systemem operacyjnym, ponieważ jest to jego podstawowe łącze z mikrokomputerem. Termin system operacyjny oznacza te moduły programowe w obrębie systemu mikroprocesorowego, które rządzą sterowaniem zasobami sprzętowymi, takimi jak procesor /procesory/, pamięć operacyjna, pamięć zewnętrzna, urządzenia wejścia/wyjścia oraz plikami. Moduły te rozstrzygają konflikty, upraszczają efektywne wykorzystanie systemu oraz tworzą łącze między programem użytkownika a sprzętem fizycznym mikrokomputera. Natomiast moduły użytkowe, translatory języków, programy biblioteczne i środki programowe do usuwania błędów nie wchodzą do systemu operacyjnego. Moduły te po prostu wykorzystują system operacyjny w czasie ich wykonywania się w pamięci mikrokomputera.
Na pytanie, co steruje mikrokomputerem i działa jako pośrednik między nim a użytkownikiem? To właśnie system operacyjny jest takim programem, który gdy jest wykonywany, steruje pracą mikrokomputera.

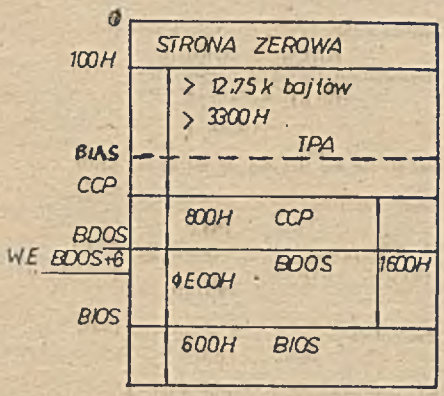
2.2. Budowa systemu operacyjnego CP/M

System operacyjny CP/M może być eksploatowany na mikrokomputerach, które są oparte na mikroprocesorach INTEL 8080 i 8085 lub Z-80, posiadają pamięć typu RAM o minimalnej pojemności 20k bajtów. Pamięć operacyjna typu RAM system CP/M dzieli na pięć obszarów /rys. 4/, określonych przez rezydujące w nich programy:

- strona zerowa
- obszar tymczasowego przechowywania programu TPA /ang.

Transient Program Area/

- obszar komunikacji z operatorem CCP /ang. Console Command Processing/
- obszar zajęty przez podstawowy system dyskowy BDOS /ang. Basic Disk Operating System/
- obszar podstawowego systemu wejścia-wyjścia BIOS /ang. Basic I/O System/

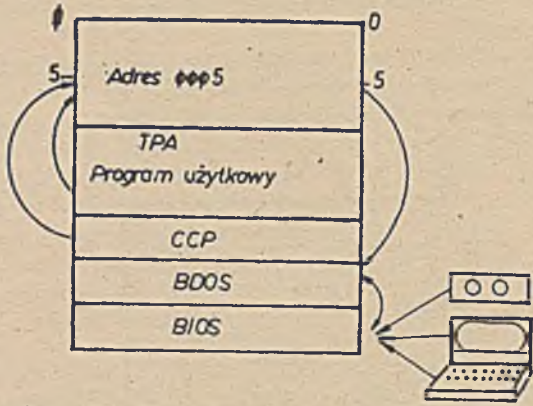


Rys.4 Obraz pamięci w systemie CP/M

Na rys. 5 przedstawiono podział pamięci wraz z zaznaczonymi obszarami adresowymi poszczególnych części.
Adresy modułów oblicza się według następujących wzorów /MSIZE odpowiada rozmiarowi pamięci, choć nie musi pokrywać się z jej fizycznym rozmiarem/

BIAS = /MSIZE-20/ x 1024
CCP = BIAS + 3400H
BDOS = CCP + 806H
BIOS = CCP + 1600H

gdzie:
MSIZE przyjmuje wartości od 20 do 64



Rys.5 Powiązania funkcjonalne modułów w systemie OP/M

Strona zerowa jest pierwszym obszarem pamięci, który zaczyna się od adresu ØØØØ do adresu ØFFH tzn. zajmuje wielkość 256 bajtów i jest tworzona dynamicznie podczas inicjowania pracy systemu przez program ładujący. Zapewnia ona łączność między programami CCP, BDOS i programami użytkownika. Deklaracja poszczególnych pól strony zerowej przedstawia się następująco:

```
0000 =          RAM       EQU      0          ;POCZATEK PAMIECI RAM
0000                      ORG      RAM        ;USTAWIENIE LICZNIKA PROG.
0000            WARMBOOT  DS       3          ;FUNK WEJSCIA DO WARM BOOT
0002 =          BIOSPAGE  EQU      AM+2       ;ADRES SKOKU DO BIOS'U +3
0003            IOBYTE    DS       1          ;BAJT WEJSCIA/WYJSCIA
0004            CURSER    DS       1          ;AKTUALNY UZYTKOWNIK(BIT 7-4)
0004 =          CURDISK   EQU      CURSER     ;OPCJONALNE PRZYDZIAL DYSKOW/R
0005            BDOSE     DS       3          ;FUNKT WEJSCIA DO BDOS
                                               ;
005C                      ORG      RAM+5CH    ;OBSZAR NIEUZYWANY
005C            FCB1      DS       16         ;FILE CONTROL BLOCK NR1
006C            FCB2      DS       1          ;FILE CONTROL BLOCK NR 2
0050                      ORG      RAM+80     ;OBSZAR NIEUZYWANY
                COMTAIL
0050            COMTAIL$COUNT DS 1            ;LICZNIK ZNAKOW W KOMENDZIE
0051            COMTAIL$CHARS DS 127          ;TRESC KOMENDY OPERATORSKIEJ
                ;DLA OPERACJI CZYTANIA/PISANIA BUFOR MA POSTAC
                ;
0080                      ORG      RAM+80H    ;REDEFINICJA BUFORA
0080            DMABUFFER DS 128              ;OPCJONALNY ADRES "DMA"
0100                      ORG RAM+100H        ;OBSZAR NIEUZYWANY
                TPA                           ;POCZATEK OBSZARU TPA
```

warmboot - trzybajtowe pole zawiera instrukcję skoku do podprogramu reinicjacji systemu, którego adres znajduje się w tablicy wektorów modułu BIOS

IOBYTE - zawiera cztery dwubitowe pola, każde z nich przyjmować może następujące wartości: 00, 01, 10 i 11.

nr bitu	7	6	5	4	3	2	1	0
urządzenia logiczne	List /drukarka/		Punch /perforator/		Reader /czytnik/		Console /konsola/	

Wartość poszczególnych pól interpretowane są przez BIOS, jako urządzenia wejścia/wyjścia /fizyczne/ wg następującego zestawienia:

Tablica 6

Wartości komórki IOBYTE

Urządzenia logiczne	Urządzenia fizyczne			
	00	01	10	11
Console /CON:/	TTY:	CRT:	BAT:	UC1:
Reader /RDR:/	TTY:	PTR:	UR1:	UR2:
Punch /PUN:/	TTY:	PTP:	UP1:	UP2:
List /LST:/	TTY:	CRT:	LPT:	UL1:

Komendą STAT można zmienić zawartość bajtu IOBYTE a tym samym ustalić inny przydział urządzeń. Aktualny przydział zachowywany jest do momentu ponownego przeładowania systemu przez operatora.

CURDISK - czterobitowe pole /bity 0-3/ określa logiczny przydział dysku w systemie. Domyślnie przyjmowana jest wartość Ø, co oznacza dysk A. Jeżeli wartość jest 1, to przydział jest dla dysku B itd. Pole można zmieniać wywołując odpowiednią funkcję modułu BDOS lub w programie użytkowym.

BDOSE - trzybajtowe pole zawierające instrukcję skoku do modułu BDOS. Wykorzystywane może być przez program użytkownika do realizacji jednej z funkcji modułu BDOS.

FCB1 i FCB2 - obszary do tworzenia tablic FCB /File Control Block/ dla programów wywoływanych przez moduł CCP. Każdy z bloków przechowuje tylko po 16 bajtów tablicy FCB. Po otwarciu zbioru pozostałe pola bloku

FCB uzupełniane są danymi ze słownika opisu zbioru znajdującego się na dysku. Powoduje to przykrycie danych bloku FCB2, jeżeli w tym czasie wypełniały ten blok.

Zawartość pierwszych 16 bajtów strony zerowej przedstawiono poniżej.

```
0000 C3 03 F2 94 01 C3 00 D4 C3 1F F6 FF 00 FF 00 FF .........
     JMP Wboot Iobyte JMP Bdos
```

Moduł CCP - zapewnia łączność operatora z systemem operacyjnym, umożliwia czytanie, interpretowanie i wykonywanie komend systemu oraz ładowanie i wykonywanie programów użytkowych. Komendy mogą występować w dwóch postaciach: ośmioznakowa nazwa zbioru typu COM lub nazwa jednej z 5 wbudowanych w moduł CCP komend. Jeżeli wprowadzona nazwa jest jedną z wbudowanych komend, to CCP wykonuje ją wewnątrz modułu CCP. Gdy nazwa nie jest komendą modułu CCP, to następuje przeszukiwanie katalogu przydzielonego dysku w celu znalezienia nazwy zbioru rozszerzonej o ".COM". i gdy zostanie znaleziona, to po wczytaniu jego zawartości do obszaru TPA następuje wykonanie załadowanego programu. Nazwę komendy pisanej na konsoli operatorskiej moduł CCP umieszcza w obszarze strony zerowej /bufor o długości 128 bajtów/ do czasu naciśnięcia klawisza RETURN. W buforze operator może dokonywać modyfikacji niekompletnej komendy przez użycie klawisza CONTROL łącznie z innym klawiszem konsoli. Poniżej podano różne możliwości użycia klawiszy funkcyjnych obsługiwanych przez moduł CCP:

CONTROL-C: Warm Boot Jeżeli pierwszym znakiem wprowadzonym z konsoli będzie CONTROL-c, to moduł CCP wykona reinicjacje systemu.

CONTROL-E: Fizyczny koniec linii /ang. Physical End-of-line/ Użycie klawisza CONTROL-E powoduje wysłanie na konsolę znaku CR/LF bez wykonania wprowadzonej komendy przez moduł CCP.

CONTROL-H: Cofacz znaku /ang. Backspace/ - ostatnio wprowadzony znak z konsoli zostanie usunięty z bufora i ekranu.

CONTROL-J: Nowa linia /ang. Line Feed/ CONTROL-M: CR /ang. Carriage Return/ - nastąpi przejście do nowej linii i cofnięcie

kursora do początku linii.

CONTROL-P: Echo /ang. Printer Echo/ - pierwsze użycie klawisza CONTROL-P powoduje przesłanie każdego znaku z konsoli na drukarkę. Powtórne użycie komendy zawiesza wysyłanie znaków na drukarkę.

CONTROL-R: Powtórz linie /ang. Repeat Command Line/ - użycie klawisza zaznaczone jest na ekranie znakiem # po ostatnio wprowadzonym znaku z jednoczesnym wymazaniem zawartości w buforze i przejściem do nowej linii, i powrotem kursora do początku linii.

Control-S: Stop /ang. Stop Screen Output/ - powoduje czasowe zatrzymanie wyświetlania na ekranie do czasu wprowadzenia dowolnego znaku z konsoli.

Control-U lub Control-X: Unieważnij linię /ang. Undo Command Line/ - wykonują te same funkcje, kasując wprowadzoną linię na ekranie i buforze.

Rub lub DEL: kasuj ostatni znak /ang. Delete Last Character/ - kasuje znaki w buforze potwierdzając kasowanie wyświetleniem kasowanego znaku na ekranie konsoli.

Moduł BDOS - moduł zarządza /logicznie/ całym systemem wejścia - wyjścia za pomocą wbudowanych funkcji, które można podzielić na dwa typy:

- funkcje wejścia/wyjścia pojedynczego znaku, które realizują wysyłanie i odbieranie danych pomiędzy systemem mikrokomputera a logicznymi urządzeniami - tzn. konsola, "czytnikiem", "perforatorem" i drukarką /funkcje 1 + 12/
- funkcje zarządzania zasobami dyskowymi, tzn. tworzenie nowych zbiorów, kasowanie istniejących, otwieranie, zamykanie, czytanie i pisanie 128 bajtowych rekordów z/do tych zbiorów /funkcje 13 + 40/

Funkcje modułu BDOS przedstawiono w kolejności ich kodów w tabeli 7.

Tablica 7

Funkcje modułu BDOS

Kod funkcji	Działanie funkcji
1	2
0	Inicjuje system
1	Czyta znak z konsoli /klawiatury/
2	Pisze znak na ekran

1	2
3	Czyta znak z logicznego urządzenia typu czytnik taśmy
4	Pisze znak na logiczne urządzenia typu perforator
5	Pisze znak na drukarkę
6	Bezpośrednio pisze lub czyta znak bez kontroli BDOS na konsole /ekran/
7	Czyta aktualny bajt IOBYTE
8	Ustala nową wartość IOBYTE
9	Wysyła na konsolę ciąg znaków zakończony "$"
10	Czyta ciąg znaków z konsoli do bufora
11	Sprawdza stan klawiatury
12	Podaje wersję systemu
13	Nadaje dyskom stan R/W
14	Określa domyślny nr dysku
15	Otwiera zbiór do czytania/pisania
16	Zamyka zbiór
17	Szuka w słowniku pierwszego opisu zbioru
18	Szuka następnego opisu zbioru
19	Kasuje zbiór /opis w słowniku/
20	Czyta sekwencyjnie "rekord"
21	Pisze sekwencyjnie "rekord"
22	Tworzy nowy plik
23	Zmienia nazwę pliku
24	Wskazuje, który logiczny dysk jest aktywny
25	Określa domyślny nr dysku
26	Ustala adres DMA /bufor do czytania/pisania/
27	Podaje adres wektora allokacji bloków zbioru
28	Nadaje stan R/o dyskowi o nr domyślnym
29	Wskazuje który z dysków ma stan R/O
30	Nadaje status zbiorowi
31	Podaje adres bloku DPB w module BIOS
32	Ustala/czyta nr użytkownika zbioru
33	Czyta "rekord" o dostępie bezpośrednim
34	Pisze "rekord" o dostępie bezpośrednim
35	Określa rozmiar zbioru o dostępie bezpośrednim
36	Umieszcza liczbę rekordów dla następnego czytania/pisania bezpośredniego

1	2
37	Zmienia nr dysków /logicznie/
40	Pisze bezpośrednio "rekord" i zeruje go

Wszystkie funkcje modułu BDOS wywoływane są przez użycie instrukcji CALL ØØØ5H. Kod funkcji modułu BDOS umieszczony jest w rejestrze C, zaś w rejestrze E dodatkowe parametry wejściowe. Jeżeli przekazywaną wartością parametru jest adres bufora lub blok opisu zbioru /FCB/, to należy użyć pary rejestrów DE. Po wykonaniu funkcji system przekazuje informacje do rejestru A lub pary rejestrów HL.

Schemat wywołania funkcji /nr 0 - 12/ modułu BDOS dla pojedynczego bajtu danych w języku INTEL 8080 wygląda następująco:

```
MVI  C, KODØFUNKCJI
MVI  E, BAJTDANYCH
CALL BDOS
```

Dla pozostałych funkcji /nr 13 - 40/ schemat ten wygląda następująco:

```
MVI  C, KODØFUNKCJI
LXI  D, Adres
CALL BDOS
```

Funkcja Ø: Inicjacja systemu

Kod funkcji: C = ØØH

Parametr We: brak

Wyjście: brak powrotu

```
0000 =          B$SYSRESET EQU  0   ;REINICJACJA SYSTEMU
0005 =          BDOS       EQU '5   ;WEJSCIE DO BDOS'U
0000 0E00                  MVI C,B$SYSRESET    ;FOBIRZ KOD FUN.
0002 C30500                JMP BDOS   ;MOZNA UZYC JMP
                                      ;ZAMIAST CALL,ALE
                                      ;BEZ POWROTU
```

Cel funkcji - reinicjuje działanie systemu poprzez wywołanie funkcji Warm boot. Użycie tej funkcji w programie użytkowym powoduje zakończenie programu i powrót do CP/M.

Funkcja 1: Czyta znak z konsoli

Kod funkcji: C = Ø1H

Parametr We: brak

Wyjście: A = wczytany znak z konsoli

Przykład

```
0001 =          B$CONIN   EQU  1  ;FUNKCJA 1
0005 =          BDOS      EQU  5  ;WEJSCIE BDOS
0000 0E01               MVI C,B$CONIN  ;POBIERZ KOD FUN.
0002 CD0500             CALL BDOS ;WYWOLAJ FUN.
```

Cel funkcji - funkcja czyta bajt danych z klawiatury konsoli i umieszcza go w rejestrze A. Jeżeli wprowadzony jest znak alfanumeryczny, to będzie jednocześnie wyświetlony na ekranie. Wszystkie znaki sterujące, włączając CONTROL-C, są wczytywane lecz nie wyświetlane na ekranie.

Funkcja 2: Pisz znak na ekran

Kod funkcji: C = Ø2H

Parametr we: E = znak do wysłania

Wyjście: brak

Przykład

```
0002 =          B$CONOUT    EQU 2  ;FUNKCJA 2
0005 =          BDOS        EQU 5  ;WEJSCIE BDOS
0000 0E02            MVI C,B$CONOUT ;LADUJ KOD FUN.
0002 1E2A            MVI E,'*'      ;E=ZNAK DO WYSLANIA
0004 CD0500          CALL BDOS      ;WYWOLAJ FUNKCJE
```

Cel funkcji - funkcja wysyła znak danych z rejestru E na konsolę. Jak dla funkcji 1, jeżeli dane jest znakiem TAB, to będzie przez BDOS przesunięta do następnej kolumny, która będzie wielokrotnością ośmiu.

W poniższym przykładzie podprogram wysyła komunikat na konsolę, wykorzystując omawianą funkcję. Tekst komunikatu zakończony jest pustym znakiem /zero/.

```
                ;PODPROGRAM WYSYLA KOMUNIKAT
                ;NA KONSOLE.SEPARATOREM KONCA 0.
                ;
                ;WYKONAC NASTEPUJACA SEKWENCJE
                ;  KOMUNIKAT DB 'TRESC KOMUNIKATU',0
                ;        LXI H,KOMUNIKAT
                ;         CALL MSOUT
                ;
0002 =          B$CONOUT     EQU  2   ;FUNKCJA 2
0005 =          BDOS         EQU  5   ;WE DO BDOSU
                ;
                MSOUT:
0000 7E                 MOV  A,M ;POBIERZ BAJT
0001 B7                 ORA  A
0002 C8                 RZ       ;POWROT J.ZERO
0003 23                 INX  H   ;ADR. NASTEP.ZN.
00 4 E5                 PUSH H   ;PAM. ADR.
0005 5F                 MOV  E,A ;BAJT DO WYSL.
0006 0E02               MVI C,B$CONOUT
0008 CD0500             CALL BDOS
000B E1                 POP  H   ;PRZYWROC ADR.KOM.
000C C30000             JMP  MSOUT ;N STEPNY ZNAK
```

Funkcja 3: Czytaj znak z czytnika

Kod funkcji: C = Ø3H

Parametr we: brak

Wyjście: A = wprowadzony znak

```
0003 =          B4READIN  EQU   3 ;FUNKCJA 3
0005 =          BDOS      EQU   5 ;WEJSCIE DO BDOS
0000 0E03          MVI C,B4READIN ;POD.KOD FUNKCJI
0002 CD0500        CALL BDOS
```

Cel funkcji - funkcja czyta znak z logicznego "czytnika" do rejestru A. W praktyce może to być dostępne urządzenie wejścia zdefiniowane w module BIOS.

Funkcja 4: Pisz znak do perforatora

Kod funkcji: C = Ø4H

Parametr we: E = wysyłany znak

Wyjście: brak

Przykład

```
0004 =          B4PUNOUT  EQU   4 ;KOD FUNKCJI 4
0005 =          BDOS      EQU   5
0000 0E04          MVI C,B4PUNOUT ;POBIERZ K.FUN.
0002 1E2A          MVI E,'*'      ;BAJT DO WYSLANIA
0004 CD0500        CALL BDOS
```

Cel funkcji - funkcja ta wykonuje działanie odwrotne do funkcji poprzedniej. Urządzenie wyjścia musi być zdefiniowane w module BIOS.

Funkcja 5: Wyślij znak na drukarkę

Kod funkcji: C = Ø5H

Parametr we: E = wysyłany znak

Wyjście: brak

Przykład

```
0005 =          B4LSTOUT  EQU   5 ;KO  FUNKCJA 5
0005 =          BDOS      EQU   5
0000 0E05          MVI C,B4LSTOUT ;KOD FUN.
0002 CD0500        CALL BDOS
```

Cel funkcji - funkcja wysyła wyspecyfikowany bajt z rejestru E na drukarkę.

W dołączonym podprogramie drukującym komunikat na drukarkę wykorzystano omawianą funkcję.

```
                ;PISZ NA DRUKARKE Z BUFORA
                ;KONIEC OZNACZONY 0
                ;
                ;WYKONAJ SEKWENCJE:
                ;   LXI H,BUFOR
                ; CALL WL
                ;
                ;.
0005 =          B$LSTOUT  EQU  5 ;FUNKCJA 5
0005 =          BDOS      EQU  5
000D =          CR      EQU  ODH  ; R
000A =          LF        EQU  OAH   ;LF
                WL:
0000 E5                 PUSH  H     ;ZACHOW.ADR.BUF.
0001 7E                 MOV   A,M   ;POBIERZ ZNAK
0002 B7                 ORA   A     ;SPRAW. CZY 0
0003 CA2000             JZ    WL$LSTX ;TAK WYJSCIE
0006 FE0A               CPI   LF    ;CZY LF
0008 CC1600             CZ    WL$LSTLF ;TAK POWR.POCR
000B 5F                 MOV   E,A   ;ZNAK DO WYSLANIA
000C 0E05               MVI C,B$LSTOUT
000E CD0500             CALL  BDOS
0011 E1                 POP   H     ;PRZYWR.ADR. BUFORA
0012 23                 INX  H      ;NAST.ZNAK
0013 C30000             JMP   WL
                WL$LSTLF:
0016 0E05               MVI   C,B$LSTOUT  ;WYSLANIE CR
0018 1E0D               MVI   E,CR  ;WYS IJ CR
001A CD0500             CALL  BDOS
001D 3E0A               MVI   A,LF  ;NOWA LINIA LF
001F C9                 RET
                WL$LSTX:
0020 E1                 POP   H
0021 C9                 RET
```

Funkcja 6: Bezpośredni dostęp do konsoli

Kod funkcji: C = Ø6H

Parametr we: E = ØFFH dla wejścia

E = Inny niż ØFFH dla wyjścia

Wyjście: A = bajt wejściowy lub status

Przykład

```
0006 =          B$DIRCONIO EQU 6 ;FUNKCJA 6
0005 =          BDOS       EQU 5
                ;PRZYKLAD NA WPROWADZE IE ZNAKU
0000 0E06           MVI C,D$DIRCONIO ;KOD FUN.
0002 1EFF           MVI E,0FFH  ;OFFH MOZNA WPROWADZAC
                                ;A=00 JEZELI BRAK ZNAKU
                                ;A=NZ JEZELI JEST ZNAK
0004 CD0500         CALL BDOS
                ;PRZYKLAD NA WYPROWADZENIE Z KONSOLI
0007 0E06           MVI C,B$DIRCONIO ;KOD FUNK.
0009 1E2A           MVI E,'*' ;NIE OFF TO WYSLIJ
000B CD0500         CALL BDOS
```

Cel funkcji - funkcja wykonuje podwójną rolę: czyta i pisze znaki z konsoli. Jeżeli wartość w rejestrze E nie jest ØFFH, wtedy zawartość rejestru wysłana jest do konsoli, gdy E = ØFFH, to może być czytany znak z konsoli. Ustawiany jest status konsoli w rejestrze A. Wartość A = ØØ oznacza brak znaku do wczytania, w przeciwnym wypadku oznacza, że znak wczytano.

Funkcja 7: Pobiera IOBYTE

Kod funkcji: C = Ø7H

Parametr we: brak

Wyjście: A = aktualna wartość IOBYTE

Przykład

```
0007 =          B$GETIO  EQU 7  ;POBIRZ IOBYTE
0005 =          BDOS     EQU 5
0000 0E07            MVI C,B$GETIO  ;KOD FUN W C
0002 CD0500          CALL BDOS   ;A=IOBYTE
```

Cel funkcji: funkcja pobiera aktualną wartość IOBYTE ze strony zerowej i umieszcza ją w rejestrze A. Każde dwa bity pola mogą przyjmować jedną z czterech wartości: 00, 01, 10 i 11. Wartości te są interpretowane przez BIOS dla przydziału fizycznych urządzeń

Funkcja 8: Ustaw nową wartość IOBYTE

Kod funkcji: C = Ø8H

Parametr we: E = nowa wartość IOBYTE

Wyjście: brak

Przykład:

List podprogramu pokazuje w jaki sposób przydzielić logiczne urządzenie typu "czytnik" w BIOS'ie jako konsole.

```
                ;POKAZANO JAK MOZNA PRZYDZIELIC
                ;LOGICZNY "CZYTNIK" DO      JAKO UR1
                ;
0007 =          B$GETIO  EQU 7 ;POBIERZ IOBYTE
0008 =          B$SETIO  EQU 8 ;USTAW IOBYTE
0005 =          BDOS     EQU 5
000C =          IO$RDRM  EQU 0000$1100B ;MASKA "CZYTNIKA"
0008 =          IO$RUR1  EQU 2 SHL 2    ;PRZYDZIAŁ UR1
                ;
0100                ORG 100H   ;START
0100 0E07           MVI C,B$GETIO ;POBIERZ AKTUAL.
                                  ;IOBYTE
0102 CD0500         CALL BDOS
                ;A=AKTUALNY IOBYTE
0105 E6F3           ANI (NOT IO$RDRM) AND 0FFH;
                                ;CHRON BITY "CZYTNIKA"
0107 F608           ORI IO$RUR1  ;NOWA WAR.IOBYTE
0109 5F             MOV E,A
010A 0E08           MVI C,B$SETIO
010C CD0500         CALL BDOS  ;USTAW NOWY IOBYTE
```

Funkcja 9: Pisz ciąg znaków zakończony znakiem "$"

Kod funkcji: C = Ø9H

Parametr we: DE = adres pierwszego bajtu ciągu

Wyjście: brak

Przykład

```
0009 =          B$PRINTS        EQU 9 ;FUNKCJA 9
0005 =          BDOS            EQU 5 ;WEJSCIE DO BDOS'U
000D =          CR              EQU 0DH ;ZNAK CR
000A =          LF      EQU 0AH ;ZNAK LF
0009 =          TAB     EQU 09H ;ZNAK TAB
0000 0D0A09544EKOMUN    DB CR,LF,TAB,'TEXT KOMUNIK..','$'
                ;TEXT MUSI KONCZYC SIE ZNAKIEM DOLARA $
0012 0E09               MVI C,B$PRINTS  ;KOD FUNKCJI
0014 110000             LXI D,KOMUN     ;ADRES KOMUNIKATU
0017 CD0500             CALL BDOS
```

Cel funkcji: funkcja wyprowadza ciąg znaków na konsole. Adres ciągu podany jest w parze rejestrów DE. Ostatnim znakiem ciągu musi być znak dolara "$", który traktowany jest przez BDOS jako koniec ciągu.

Funkcja 10: Czytaj z konsoli ciąg znaków

Kod funkcji: C = ØAH

Parametr we: DE = adres bufora /ciągu/

Wyjście: wprowadzane znaki do bufora wyświetlane na konsoli

Przykład

```
000A =          B$READCONS      EQU 10 ;FUNKCJA 10
0005 =          BDOS    EQU 5  ;PUNKT WE DO BDOS'U
0050 =          BUFLEN  EQU 80 ;WIELKOSC BUFORA
                BUFFER
0000 50         BUFMAXCH        DB BUFLEN  ;MAX.ILOSC ZNAKOW
                                           ;W BUFORZE
0001 00         BUFACTCH        DB 0 ;AKTUALNA ILOSC ZNAKOW
0002            BUFCH   DS BUFLEN  ;OBSZAR BUFORA
0052 0E0A               MVI C,B$READCONS ;KOD FUNKCJI
0054 110000             LXI D,BUFFER    ;ADRES BUFORA W DE
0057 CD0500             CALL BDOS
```

Cel funkcji: funkcja czyta ciąg znaków z konsoli i przesyła je do bufora zdefiniowanego przez użytkownika. Bufor musi mieć następującą strukturę:

bufor	długość max	ilość znak.	znak 1	znak 2	...	
bajt	1	2	3	4		

Bajt pierwszy zawiera wielkość bufora, bajt drugi wypełniany jest przez BDOS i zawiera aktualną liczbę wprowadzonych znaków. Znak CR /Carriage Return/ nie jest wprowadzany do bufora. Jeżeli liczba znaków przekroczy max. długość bufora, to następne znaki nie będą wpisywane do ciągu.

Funkcja 11: Czytaj status konsoli

Kod funkcji: C = ØBH

Parametr we: brak

Wyjście: A = ØØH jeżeli nie wprowadzono znaku

A = ØFFH jeżeli wprowadzono znak

Cel funkcji: funkcja określa czy wprowadzany znak do konsoli czeka na wejście. Wartość rejestru A po wykonaniu funkcji określa ten stan.

Przykład

```
000B =          B$CONST EQU 11  ;FUNKCJA 11
0005 =          BDOS    EQU 5
0000 0E0B               MVI C,B$CONST  ;KOD FUNKCJI
0002 CD0500             CALL BDOS      ;A=00 JESLI NIE MA ZNAKU
                                       ;A=0FFH JESLI JEST ZNAK
```

Funkcja 12: Pobierz wersje CP/M

Kod funkcji: C = ØCH

Paramter we: brak

Wyjście: HL = nr wersji CP/M

Przykład

```
000C =          B$GETVER        EQU 12 ;FUNKCJA 12
0005 =          BDOS    EQU 5
0000 0E0C               MVI C,B$GETVER ;KOD FUNKCJI
0002 CD0500             CALL BDOS      ;H=00 DLA CP/M
                                       ;L=VERSJA CP/M
```

Cel funkcji - funkcja informuje o numerze wersji CP/M, i tak:

H = ØØH dla CP/M, H = Ø1H dla MP/M

L = ØØH dla wersji 1.x

L = 2ØH dla CP/M 2.Ø, 21H dla 2.2 itd.

Funkcja 13: Ustaw dysk systemowy

Kod funkcji: C = ØDH

Parametr we: brak

Wyjście: brak

Przykład

```
000D =          R$DSKRESET EQU 13 ;FUNKCJA 13
0005 =          BDOS       EQU  5
0000 0E00       MVI C,R$DSKRESET ;KOD FUNKCJI
0002 CD0500            .CALL BDOS
```

Cel funkcji - funkcja przydziela dysk A jako dysk systemowy, przywraca ponownie adres DMA, na ØØ8ØH /adres bufora używanego przez BDOS to czytania/pisania na dysk/ i nadaje wszystkim logicznym dyskom status R/W.

Funkcja 14: Przydziel logicznie dysk

Kod funkcji: ØEH

Parametr we: E = kod dysku /kieszeni dyskowej/

ØØH = dysk A

Ø1H = dysk B, itd.

Wyjście: brak

Przykład

```
000E =          B$SELDSK  EQU 14 ;FUNKCJA 14
0005 =          BDOS      EQU  5
0000 0E0E                MVI C,R$SELDSK ;KOD FUNKCJI
0002 1E00                MVI E,0   ;E=0 DLA A: ,1 DLA B: ,ITD
0004 CD0500              CALL BDOS
```

Cel funkcji - funkcja przydziela logicznie dysk wskazany w rejestrze E jako aktualnie przydzielony do systemu.

Funkcja 15: Otwórz zbiór

Kod funkcji: C = ØFH

Paramter we: DE = adres bloku FCB

Wyjście: A = kod słownika /powrotu/

Przykład

```
000F =          R$OPEN  EQU 15 ;FUNKCJA 15
0005 =          BDOS    EQU  5
                FCB              ;FILE CONTROL BLOCK (FCB)
0000 00         FCB$DISK     DB 0 ;ZNAJDZ DYSK AKT.
0001 46494C454EFCB$NAME      DB 'FILENAME' ;NAZWA ZBIORU
0009 545950     FCB$TYP      DB 'TYP' ;TYP ZBIORU
000C 00         FCB$EXTENT   DB 0 ;NR ROZSZERZENIA
000D 0000       FCB$RESV     DB 0,0 ;DLA CP/M
000F 00         FCB$RECCUSED DB 0 ;ILOSC REKORDOW
0010 0000000000FCB$ARUSED    DB 0,0,0,0,0,0,0,0
0018 0000000000                 DB 0,0,0,0,0,0,0,0
                    ;ALLOKACJA BLOKOW (NR BLOKU)
0020 00         FCB$SEQREC   DB 0 ;SEKW.REKORD DO R/W
0021 0000       FCB$RANREC DW 0 ;BEZPOSREDNI REKORD
                            ;DO CZYTANIA/PISANIA
0023 00         FCB$RANRECO DB 0 ;BAJT NADMIARU
0024 0E0F                MVI C,B$OPEN  ;KOD FUNKCJI
0026 110000              LXI D,FCB ;ADRES BLOKU FCB
0029 CD0500              CALL BDOS
```

Cel funkcji - funkcja otwiera specyfikowany zbiór do czytania lub pisania. Adres bloku FCB musi być podany w rejestrze DE /para rejestrów/. Blok FCB powinien zawierać następujące dane: nr użytkownika, nr logicznego dysku, nazwę zbioru, typ zbioru. Wszystkie pozostałe pola bloku mogą ustawione być na zero. Kod powrotu ustawiony jest w rejestrze A. Jeżeli zawartość wynosi ØFFH, to brak opisu zbioru w słowniku, gdy A = 0, 1, 2 lub 3 to zbiór został otwarty.

Funkcja 16: Zamknij zbiór

Kod funkcji: C = 1ØH

Parametr we: DE = adres bloku FCB

Wyjście: A = kod słownika /powrotu/

Przykład

```
0010 =          B$CLOSE  EQU 16 ;FUNKCJA 16
0005 =          BDOS:    EQU  5
0000            FCB  DS 36  ;BLOK FCB
0024 0E10                MVI C,B$CLOSE  KOD FUNKCJI
0026 110000              LXI D,FCB ;ADRES BLOKU FCB
0029 CD0500              CALL BDOS ;A=1,2,3 JESLI POPRAWNIE
                               ;A=0FFH JSLI BRAK ZBIORU
```

Cel funkcji - funkcja zamyka zbiór danych, podobnie jak funkcja 15. Kod powrotu znajduje się w rejestrze A. Jeżeli jest to wartość 0, 1, 2 lub 3, to zamknięcie zbioru zakończyło się sukcesem. Zamknięcie zbioru polega na uaktualnieniu słownika dysku. Informacje te pobierane są z bloku FCB. Jeżeli zamykany zbiór typu TEX. był pisany, to użytkownik musi przed wykonaniem operacji zamknięcia przesłać do zbioru rekord danych zawierający znak 1AH /koniec danych/.

Funkcja 17: Szukaj pierwszego opisu zbioru

Kod funkcji: C = 11H

Parametr we: DE = adres bloku FCB

Wyjście: A = kod powrotu

Przykład

```
0011 =          B$SEARCHF EQU 17 ;FUNKCJA 17
0005 =          BDOS      EQU  5
                FCB:
                ;BLOK FCB
0000 00         FCB$DISK DB 0 ;PRZYDZIAŁ SYSTEMOWY
0001 4E415A5741FCB$NAME DB 'NAZWA???' ;NAZWA ZBIORU
0009 543F50     FCB$T P  DB 'T?P'  ;TYP ZB.
000C 00         FCB$EXTENT DB 0 ;NR ROZSZE.
000D 0000       FCB$RESV DB 0,0 ;DLA CP/M
000F 00         FCB$RECUSED DB 0 ;ILOSC REK.
0010 0000000000FCB$ABUSED DB 0,0,0,0,0,0,0,0
0018 0000000000           DB 0,0,0,0,0,0,0,0
                ;ALLOKACJA BLOKOW 1024 BAJTOWYCH
0020 00         FCB$SEQREC DB 0 ;SEKW. REK. DO R/W
0021 0000       FCB$RANREC DW 0 ;BEZPOSR. REK.
0023 00         FCB$RANRECO DB 0 ;BAJT NADMIARU
0024 0E11                MVI C,B$SEARCHF ;KOD FUNK.
0026 110 00              LXI D,FCB ;ADRES FCB
0029 CD0500              CALL BDOS ;A=0,1,2,3 TO
                       ;POPRAWNIE WYKONALA SIE F.
                           ;A=0FFH NIE ZNALEZ.ZB.
```

Cel funkcji - funkcja szuka pierwszego opisu zbioru w słowniku dysku. Jeżeli zbiór znaleziono, to w rejestrze A umieszczana jest wartość Ø, 1, 2 lub 3, która pomnożona przez 32 i dodana do adresu DMA określa adres, gdzie pamiętany będzie pierwszy blok FCB.

Funkcja 18: Szukaj następnego opisu zbioru

Kod funkcji: C = 12H

Parametr we: Brak /przyjmuje poprzednią wartość z funkcji 17/

Wyjście: A = kod powrotu

Przykład

```
0012 =          B$SEARCHN EQU 18 ;FUNKCJA 18
0005 =          BDOS      EQU  5
0000 0E12                MVI C,B$SEARCHN ;KOD FUNKCJI
                ;BLOK FCB ZBEDNY SPRAWDZ PO WYKONANIU
                ;FUNKCJI 17 ZAWARTOSC A
0002 CD0500              CALL BDOS
```

Cel funkcji: funkcja szuka następnego opisu zbioru, dla FCB wyspecyfikowanego dla funkcji 17.

Funkcja 18 musi być użyta łącznie z funkcją 17. Kod powrotu ustawiany jest w rejestrze A, jeżeli A = ØFFH oznacza to, że następnego opisu zbioru nie znaleziono, gdy wartość ta wynosi 0, 1, 2 lub 3, to wskazuje względny numer wejścia dla słownika.

/rekord zawiera 4 opisy słownika/

Funkcja 19: Kasuj zbiór

Kod funkcji: C = 13H

Parametr we: adres bloku FCB

Wyjście: A = status zbioru

Przykład

```
0013 =          B$ERASE EQU 19 ;FUNKCJA 19
0005 =          BDOS    EQU  5

                FCB:            ;BLOK FCB
0000 00         FCB$DISK DB 0 ;DOMYSLA WAR.
0001 3F3F4E414DFCB$NAME DB '??NAME' ;NAZWA MASKOWANA
0007 545950     FCB$TYP  DB 'TYP' ;TP ZB.
000A 00         FCB$EXTENT DB 0 ;NR EXTENTU
000B 0000       FCB$R SV DB 0,0 ;DLA CP/M
000D 00         FCB$RECUSED DB 0 ;ILOSC REK.
000E 0000000000FCB$ABUSED DB 0,0,0,0,0,0,0,0
0016 0000000000           DB 0,0,0,0,0,0,0,0
                ;NR BLOKOW ZBIORU
001E 00         FCB$SEQREC DB 0 ;NR REK.SEKW.
001F 0000       FCB$RANREC DB 0,0 ;NR REK.BEZP.
0021 00         FCB$RANRECO DB 0 ;RAJT NADMIARU
0022 0E13                MVI C,B$ERASE ;KOD FUNKCJI
0024 110000              LXI D,FCB ;ADRES BLOKU FCB
0027 CD0500              CALL BDOS ;A=0FFH BRAK ZBIORU
```

Cel funkcji - funkcja logicznie usuwa opis zbioru ze słownika dysku. Jeżeli nie znaleziono zbioru, to wartość rejestru A = ØFFH.

Funkcja 20: Czytaj sekwencyjnie

Kod funkcji: C = 14H

Parametr we: DE = adres bloku FCB

Wyjście: A = status

Przykład

```
0014 =          B$READSEQ EQU 20 ;FUNKCJA 20
0005 =          BDOS      EQU  5
                FCB:                ;BLOK FCB
0000 00         FCB$DISK  DB 0
0001 4E414D4520FCB$NAME   DB 'NAME    ' ;NAZWA ZB.
0009 545950     FCB$TYP   DB 'TYP'   ;TYP
000C                      DS 24   ;USTAWI FUNK OPEN
0024 0E14                MVI C,B$READSEQ ;KOD FUNKCJI
0026 11 000              LXI D,FCB
0029 CD0500              CALL BDOS
```

Cel funkcji - funkcja czyta rekord /128-bajtowy sektor/ ze zbioru dyskowego do bufora pamięci, którego adres określony jest przez funkcję DMA /kod 26/. Domyślnie adres przyjęty jest 8ØH. Poprawnie wykonana operacja czytania zaznaczona jest wartością ØØH w rejestrze A.

Wartość 1AH oznacza koniec danych a wartość ØFFH koniec zbioru.

Funkcja 21: Pisz sekwencyjnie

Kod funkcji: C = 15H

Parametr we: DE = adres bloku FCB

Wyjście: A = status

Przykład

```
0015 =          B$WRITESEQ EQU 21 ;FUNKCJA 21
0005 =          BDOS       EQU  5
                FCB:                ;BLOK FCB
0000 00         FCB$DISK   DB 0
0001 46494C454EFCB$NAME    DB 'FILENAME'
0009 545950     FCB$TYP    DB 'TYP'
000C                       DS 24 ;WYPEŁNI OPEN
0024 0E15                  MVI C,B$WRITESEQ ;KOD FUNKCJI
0026 110000                LXI D,FCB   ;BLOK FCB
0029 CD0500                CALL BDOS
```

Cel funkcji - funkcja pisze rekord, którego adres zdefiniowany w DMA /kod 26/ do zbioru określonego blokiem FCB. Działanie jest dokładnie odwrotne do poprzedniej funkcji.

Funkcja 22: Utwórz nowy plik

Kod funkcji: C = 16 H

Parametr we: DE = adres bloku FCB

Wyjście: A = status zbioru

Przykład

```
0016 =          B$CREATE EQU 22 ;FUNKCJA 22
0005 =          BDOS     EQU  5
                FCB:              ;BLOK FCB
0000 00         FCB$DISK DB 0
0001 46494C454EFCB$NAME  DB 'FILENAME' ;NAZWA ZB.
0009 545950     FCB$TYP  DB 'TYP'
000C 00         FCB$EXTENT DB 0
000D 0000       FCB$RESV DB 0,0
000F 00         FCB$RECUSED DB 0
0010 0000000000FCB$ABUSED DW 0,0,0,0
0018 00000000 0           DW 0,0,0,0
0020 00         FCB$SEQREC DB 0
0021 0000       FCB$RANREC DW 0
0023 00         FCB$RANRECO DB 0
0024 0E16                 MVI C,B$CREATE ;KOD FUNKCJI
0026 110000               LXI D,FCB ;ADRES FCB
0029 CD0500               CALL BDO
```

Cel funkcji - funkcja tworzy nowy zbiór o wyspecyfikowanej nazwie i typie. Po wykonaniu funkcji w rejestrze A zapisany jest status, gdy A = 0, 1, 2 lub 3 to operacja wykonała się poprawnie. Wartość A = ØFFH oznacza, że słownik dysku jest pełny.

Funkcja 23: Zmieć nazwę zbioru

Kod funkcji: C = 17H

Parametr we: DE = adres bloku FCB

Wyjście: A = status zbioru

Przykład

```
0017 =          B$RENAME EQU 23 ;FUNKCJA 23
0005 =          BDOS     EQU  5
                FCB:              ;BLOK FCB
0000 00               DB 0
0001 4F4C444E41       DB 'OLDNAME' ;STARA NAZWA
0008 545950           DB 'TYP'
 00B 00000000         DB 0,0,0,0
000F 00               DB 0 ;FCB +16
0010 4E45574E41       DB 'NEWNAME' ;NOWA NAZWA
0017 545950           DB 'TYP'
001A 00000000         DB 0,0,0,0
001E 0E17                MVI C,B$RENAME
0020 110000              LXI D,FCB
0 23 CD0500              CALL BDOS ;A=00H POPRAWNIE
                                   ;A=0FFH BRAK ZBIORU
```

Cel funkcji - funkcja zmienia nazwę istniejącego zbioru i typ na nową nazwę i typ. Do zmiany nazwy wykorzystywany jest jeden blok FCB, z tym, że pierwsze 16 bajtów bloku FCB dotyczy starej nazwy zbioru a 16 następnych bajtów nowej nazwy. Kod powrotu ustawiony jest w rejestrze A, jeżeli operacja wykonała się poprawnie, to A = ØØ. W przypadku braku nazwy zbioru w słowniku wartość A = ØFFH.

Funkcja 24: Określ aktywne dyski w systemie

Kod funkcji: C = 18H

Parametr we: brak

Wyjście: HL = mapą aktywnych dysków

Przykład

```
0018 =          B$GETACTDSK  EQU 24 ;FUNKCJA 24
0005 =          BDOS EQU 5
0000 0E18               MVI C,B$GETACTDSK ;KOD FUNKCJI
0002 CD0500             CALL BDOS         ;HL=MAPA AKTYW.DYSKOW
                ;BIT=1 DYSK AKTYWNY
                ;BITY 15 14 13 ...2 1 0
                ;DYSK  P  O  N ...C B A
```

Cel funkcji - funkcja podaje w parze rejestrów HL nazwę aktywnych dysków znanych w systemie. Każdy ustawiony bit pary rejestrów HL oznacza numer dysku. W rejestrze L określane są dyski od A - P.

Funkcja 25: Podaj aktualny domyślny dysk w systemie
Kod funkcji: C = 19H
Parametr we: brak
Wyjście: A = aktualny dysk
/Ø = A, 1 = B, ..., F = P

Przykład

```
0019 =          B$GETCURDSK EQU 25 ;FUNKCJA 25
0005 =          BDOS EQU 5
0000 0E19               MVI C,B$GETCURDSK ;KOD FUNKCJI
0002 CD0500             CALL BDOS  ;A=0 JESLI A: ,1JESLI B:...
```

Cel funkcji - po wykonaniu funkcji rejestr A zawiera wartość, która wskazuje nr dysku, jeżeli A = Ø, to dysk A jest aktualnie przydzielony do systemu, 1 to dysk B itd.

Funkcja 26: DMA /czytaj/pisz/ - adres bufora
Kod funkcji: C = 1AH
Parametr we: DE = DMA /czytaj/pisz/ adres bufora
Wyjście: brak

Przykład

```
001A =          B$SETDMA  EQU 26 ;FUNKCJA 26
0005 =          BDOS      EQU  5
0000            SECBUFF  DS 128 ;WIELKOSC BUFORA
0080 0E1A               MVI C,B$SETDMA ;KOD FUN.
0082 110000             LXI D,SECBUFF  ;ADRES BUFORA
0085 CD0500             CALL BDOS
```

Cel funkcji - ustalony jest nowy adres bufora DMA dla operacji czytania/pisania z dysku. W momencie pierwszego startowania systemu CP/M adres DMA ustawiany jest opcjonalnie jako 0080H.

Funkcja 27: Podaj adres wektora allokacji
Kod funkcji: C = 1 BH
Parametr we: brak
Wyjście: HL = adres wektora allokacji

Przykład

```
001B =          B$GETALVEC EQU 27;FUNKCJA 27
0005 =          BDOS       EQU  5
0000 0E1B               MVI C,B$GETALVEC
0002 CD0500             CALL BDOS
                ;HL=ADRES WEKTORA AKTUALNIE
                ;PRZYDZIELONYCH DYSKOW
```

Cel funkcji - funkcja podaje w parze rejestrów HL adres podstawowy lub początek adresu wektora allokacji dla aktualnie przy-

dzielonego dysku. Informacja ta może być użyta w programie dla określenia wolnego obszaru na dysku.

Funkcja 28: Ustaw status dysku na R/O

Kod funkcji: C = 1CH

Parametr we: brak

Wyjście: brak

Przykład

```
001C =          B$SETDSKRO  EQU  28;FUNKCJA 28
                ;USTAW DYSK NA R/O
0005 =          BDOS        EQU   5
0000 0E1C               MVI C,B$SETDSKRO ;KOD FUNKCJI
0002 CD0500             CALL BDOS
```

Cel funkcji - funkcja ustawia aktualnie przydzielony dysk tylko do czytania. Operacja pisania na tym dysku jest zabroniona.

Funkcja 29: Podaj status R/O dysków

Kod funkcji: C = 1DH

Parametr we: brak

Wyjście: HL = mapa dysków R/O

Przykład

```
001D =          B$GETTRODSKS EQU 29 ;FUNKCJA 29
0005 =          BDOS         EQU  5
0000 0E1D               MVI C,B$GETTRODSKS ;KOD FUN.
0002 CD0500             CALL BDOS
```

Cel funkcji - para rejestrów HL pokazuje, które z logicznych dysków w systemie mają status R/O. Ustawione bity w rejestrze L wskazują numery dysków.

Funkcja 30: Ustaw status zbioru

Kod funkcji: C = 1EH

Parametr we: DE = adres bloku FCB

Wyjście: A = status

Przykład

```
001E =          B$SETFAT EQU  30 ;FUNKCJA 30
0005 =          BDOS     EQU   5
                FCB:
0000 00         FCB$DISK DB 0
0001 46494C454EFCB$N ME DB 'FILENAME'
0009 D4         FCB$TYP  DB 'T'+80H
                ;TYP Z ATRYB R/O
000A 5950                DB 'YP'
000C 0000000000          DW 0,0,0,0,0,0,0,0
001C 0E1E               MVI C,B$SETFAT
001E 110000             LXI D,FCB
0021 CD0500             CALL BDOS
```

Cel funkcji - nazwa i typ zbioru wykorzystuje tylko siedem bitów w każdym znaku. Najbardziej znaczący bit znaku wykorzystywany jest do określania atrybutu zbioru /łącznie mamy 11 bitów/. Każdy atrybut zaznaczony jest pojedynczym bitem, i tak

f1 - f4 - wykorzystywane przez użytkownika
f5 - f8 - wykorzystywane przez CP/M
t1 - atrybut R/O zbioru
t2 - atrybut SYS zbioru
t3 - wykorzystywany przez CP/M

f - oznacza nazwę zbioru /1 - 8 znaki nazwy/
t - oznacza typ zbioru /1 - 3 znaki typu/

Funkcja 31: Pobierz adres bloku DPB
Kod funkcji: C = 1FH
Parametr we: brak
Wyjście: HL = adres bloku DPB

Przykład

```
001F =          R$GETDPB  EQU  31 ;FUNKCJA 31
0005 =          BDOS      EQU  5
0000 0E1F                 MVI C,R$GETDPB;KOD FUN.
0002 CD0500               CALL BDOS
              ;HL=ADRES BLOKU DPB
```

Cel funkcji - funkcja podaje w parze rejestrów HL adres początku tablicy DPB /Disk Parametr Block/, w której opisana jest fizyczna charakterystyka dysku. Dla wszystkich dysków /tego samego typu/ występuje tylko jeden blok DPB.

Funkcja 32: Ustawia/pobiera numer użytkownika zbioru
Kod funkcji: C = 20H
Parametr we: E = 0FFH - pobierz nr użytkownika
E = Ø - 15 - ustaw nr użytkownika
Wyjście: A = aktualny nr użytkownika, jeżeli E było ustawione na 0FFH

Przykład

```
0020 =          R$SETGETUN  EQU  32 ;FUNKCJA 32
0005 =          BDOS        EQU  5
0000 0E20                   MVI C,R$SETGETUN ;KOD FUNKCJI
0002 1E0F                   MVI E,15 ;NR UZYTKOWNIKA
0004 CD0500                 CALL BDOS ;POB.NR UZYTK.
0007 0E20                   MVI C,R$SETGETUN ;KOD FUN.
0009 1EFF                   MVI E,0FFH ;WSKAZ UZYT.
000B CD0500                 CALL BDOS  ;A=AKTUAL UZYT.NR(0_15)
```

Cel funkcji - funkcja służy do zmiany numeru użytkownika zbioru. W prosty sposób można pracować na zbiorze innego użytkownika.

Funkcja 33: Czytaj rekord bezpośrednio

Kod funkcji: C = 21H

Parametr we: DE = adres bloku FCB

Wyjście: A = kod powrotu

Przykład

```
0021 =          B$READRAN  EQU 33;FUNKCJA 33
0005 =          BDOS       EQU  5
                FCB:                ;BLOK FCB
0000 00         FCB$DISK DB 0
0001 46494C454EFCB$NAME DB 'FILENAME' ;NAZWA ZB.
0009 545950     FCB$TYP  DB 'TYP'     ;TYP ZB.
000C 00         FCB$EXTENT DB 0
000D 0000       FCB$RESV DB 0,0
000F 00         FCB$RECUSED DB 0 ;ILSC REK.
0010 0000000000FCB$ARUSED DB 0,0,0,0,0,0,0,0
0018 0000000000           DB 0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC DB 0
0021 0000       FCB$RANREC DW 0
0023 00         FCB$RANRECO DB 0
                ;
0024 D204       RANRECO DW 1234 ;PRZYKLADOWY NR REK.
                ;REKORD BEDZIE CZYTANY DO  BUFORA
                ;O ADR. 80H LUB USTAW. FUN. SETDMA
                ;
0026 2A2400             LHLD RANRECO ;POB.NR REK.
0029 222100             SHLD FCB$RANREC
002C 0E21               MVI C,B$READRAN
002E 110000             LXI D,FCB
0031 CD0500             CALL BDOS
                ;A=0 OPERACJA POPRAWNA
                ;A=01 PROBA CZYT.NIE Z PIS. REK.
                ; =03 CP/M NIE MOZE ZAMK. ZB.
                ;=04 PROBA CZYT.NIEIST. ZB.
                ; =06 KONIEC ZB.
```

Cel funkcji - czyta bezpośrednio rekord zbioru określonego blokiem FCB, którego numer podany jest w 33, 34 bajcie tablicy FCB. Bajt 35 używany jest przez system CP/M.

Kod powrotu w rejestrze A może przyjmować następujące wartości:

A = 00 /operacja poprawna/

A = 01 /próba czytania niezapisanego rekordu/

A = 04 /niemożliwe czytanie niezapisanego bloku/

A = 03 /CP/M nie mógł zamknąć aktualnego bloku/

A = 06 /koniec zbioru na dysku/

Funkcja 34: Pisz rekord bezpośrednio

Kod funkcji: C = 22H

Parametr we: DE = adres bloku FCB

Wyjście: A = kod powrotu

Przykład

```
0022 =          B$WRITERAN  EQU 34 ;FUNKCJA 34
0005 =          BDOS        EQU  5
                FCB:                ;BLOK FCB
0000 00         FCB$DISK DB 0
0001 46494C454EFCB$NAME DB 'FILENAME'
 009 545950     FCB$TYP  DB 'TYP'
000C 00         FCB$EXTENT DB 0
000D 0000       FCB$RESV DB 0,0
000F 00         FCB$RECUSED DB 0
0010 0000000000FCB$ABUSED DB 0,0,0,0,0,0,0,0
0018 0000000000           DB 0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC DB 0
0021 0000       FCB$RANREC DW 0 ;NR REK. BEZPOSR.
0023 00         FCB$RANRECO DB 0;BAJT NADMIARU
                ;
0024 0204       RANRECNO DW 1234;REKORD NR 1234
                ;WYWOŁANIE /ZAPIS REK.NR 1234/
0026 2A2400             LHLD RANRECN ;W HL 1234
0029 222100             SHLD FCB$RANREC;PAMIETA W
                              ;POLU FCB$RANREC BLOKU FCB
002C 0E22               MVI C,B$WRITERAN ;KOD FUNKCJI
002E 110000             LXI D,FCB
0031 CD0500             CALL BDOS;  A=0 POPR WNIE ZAPISANO
                                   ;A=03 BL.ZAMK.
                                  ;A=05 BL.PELNY SLOW.
                                    ;A=06 KON.ZBIORU
```

Cel funkcji - funkcja pisze 128 rekord danych bezpośrednio do zbioru. Działanie jest analogiczne do poprzedniej funkcji tylko odwrotne.

Kod powrotu ustawiany jest w rejestrze A:

A = ØØ /operacja poprawna/

A = 03 /nie zamknięto bloku/

A = 05 /zapełniony słownik dysku/

A = 06 /koniec dysku dla zbioru/

Funkcja 35: Podaj rozmiar zbioru o dostępie bezpośrednim

Kod funkcji: C = 23H

Parametr we: DE = adres bloku FCB

Wyjście: rozmiar zbioru /ilość rekordów/

Przykład

```
0023 =          B$GETFSIZ  EQU 35;FUNKCJA 35
0005 =          BDOS       EQU  5
                FCB:                ;BLOK FCB
0000 00         FCB$DISK   DB 0
0001 46494C454E CB$NAME    DB 'FILENAME'
0009 545950     FCB$TYP    DB 'TYP'
000C 00         FCB$EXTENT DB 0;NREXTENTU
000D 0000       FCB$RESV   DB 0,0
000F 00         FCB$RECUSED DB 0
0010 0000000000FCB$ABUSED DB 0,0,0,0,0,0,0,0
0018 0000000000           DB 0,0,0,0,0 0,0,0
0020 00         FCB$SEQREC DB 0
0021 0000       FCB$RANREC DW 0;NR REKDO BEZ.R/W
0023 00         FCB$RANRECO DB 0
0024 0E23               MVI C,B$GETFSIZ;KOD FUNKCJI
0026 110000             LXI D,FCB ;ADRES BLOKU FCB
0029 CD0500             CA L BDOS
002C 2A2100             LHLD FCB$RANREC;W HL NR OSTATNIEGO
                ;REKORDU W ZBIORZE(LOGICZNA WIELKOSC ZB.)
```

Cel funkcji - funkcja określa w polach 33, 34 bloku FCB ilość rekordów zbioru o dostępie bezpośrednim /rekordy 128 bajtowe/ Jeżeli bajt 35 równy jest 01, to zbiór osiągnąć liczbę 65.536 rekordów.

Funkcja 36: Ustaw numer rekordu dla zbioru o dostępie bezpośrednim

Kod funkcji: C = 24H

Parametr we: adres bloku FCB

Wyjście: nr rekordu w polu FCB

Przykład

```
0024 =          B$SETRANREC  EQU  36;FUNKCJA 36
0005 =          BDOS         EQU  5
                FCB:                      ;BL.FCB
0000 00         FCB$DISK     DB 0
0001 46494C454EFCB$NAME      DB 'FILENAME
0009 545950     FCB$TYP      DB 'TYP'
000C 00         FCB$EXTENT DB 0
000D 0000       FCB$RESV     DB 0,0
000F 0000000000FCB$ABUSED DB 0,0,0,0,0,0,0,0
0017 0000000000           DB 0,0,0,0,0,0,0,0
001F 00         FCB$SEQREC DB 0
0020 0000       FCB$R NREC DW 0
0022 00         FCB$RANRECO DB 0
                ;...ZBIOR OTWARTO DO CZYT.
                ;LUB PISANIA SEKWENCYJNIE
0023 0E24               MVI C,B$SETRANREC;KOD FUNKCJI
0025 110000             LXI D,FCB ;ADRES BLOKU FCB
0028 CD0500             CALL BDOS
002B 2A2000             LHLD FCB$RANREC;W HL NR REK.
                ;W DOSTEPIE DEZFOS.
```

Cel funkcji - funkcja wstawia w pole 33, 34 bloku FCB liczbę rekordów zbioru czytanego lub pisanego sekwencyjnie.

Funkcja 37: Zmień przydział dysków w systemie

Kod funkcji: C = 25H

Parametr we: DE = bitowa mapa przydziału

Wyjście: A = COH

Przykład

```
0025 =          B$RESETD  EQU 37;FUNKCJA 37
0005 =          BDOS      EQU  5
                  ;DE=MAPA PRZYDZIELONYCH DY .
                  ;BIT JEST=1 DYSK DO ZMIANY
                  ;BITY 15 14...2 1 0
                  ;DYSK  P  O...C B A
                ;
0000 010200             LXI D,0000$0000$0000$0010B
                ;ZMANA DLA DYSKU B:
0003 0E25               MVI C,B$RESETD;KOD FUNKCJI
0005 CD0500             CALL BDOS
```

Cel funkcji - funkcja zmienia logiczny przydział dysków wg indywidualnego ustawienia wartości w parze rejestrów DE. Ustawiony bit rejestru E oznacza dyski od A do P /kolejny bit oznacza kolejny numer dysku/

Funkcja 40: Pisz bezpośrednio z zerowaniem bloku

Kod funkcji: C = 28H

Parametr we: DE = adres bloku FCB

Wyjście: A = kod powrotu

```
0028 =          B$WRITERANZ  EQU 40;FUNKCJA 40
0005 =          BDOS       EQU 5
                FCB:
0000 00         FCB$DISK DB 0
0001 46494C454EFCB$NAME DB 'FILENAME'
0009 545950     FCB$TYP  DB 'TYP'
000C 00         FCB$EXTENT DB 0
000D 0000       FCB$RESV DB 0,0
000F 00         FCB$RECUSED DB 0
0010 0000000000FCB$ABUSED DB 0,0,0,0,0,0,0,0
0018 0000000000           DB 0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC DB 0
0021 0000       FCB$RANREC  W 0
0023 00         FCB$RANRECO DB 0
0024 D204       RANRECNO    DW 1234;NR REK. 1234
                ;
0026 2A2400             LHLD RANRECNO;POB.NR REK.
0029 222100             SHLD FCB$RANREC;PAMIETAJ
002C 0E28               MVI C,B$WRITERANZ
002  110000             LXI D,FCB  ;ADRES BL. FCB
0031 CD0500             CALL BDOS
                ;  A=00 POPRAWNIE
                ;  A=03 BLAD ZAMKNIECIA
                ;  A=05 PELNY SLOWNIK
                ;  A=06 KONIC ZBIORU
```

Cel funkcji - funkcja pisze bezpośrednio rekord do zbioru wypełniając go wartością ØØH.
Kod powrotu ustawiany jak dla funkcji 34.

MODUŁ BIOS BIOS jest jedynym modułem systemu CP/M zależnym od urządzeń wejścia/wyjścia. Użytkownik musi sam opisać ten moduł definiując fizyczny dostęp do sprzętu mikrokomputera. BIOS utrzymuje programową łączność pomiędzy modułem CCP /Console Command Procesor/, BDOS'em /Basic Operating System/ i fizycznymi urządzeniami mikrokomputera. Moduły CCP i BDOS "widzą" urządzenia mikrokomputera tylko jako logiczne otoczenie. Dlatego też mogą pozostawać niezmienione w kolejnych instalacjach mikrokomputera. Wzorcowy moduł BIOS'u zawiera podprogramy dostępu do fizycznych urządzeń najniższego poziomu, w których zdefiniowane są cztery typy urządzeń:

- konsola: CP/M komunikuje się z otoczeniem via konsola
- "czytnik" i "perforator": urządzenia te są normalnie wykorzystywane jako urządzenia wejścia/wyjścia
- List: jest to drukarka, używana jako hard-copy
- dysk: mogą to być dowolne standardowe pojedyńczej lub podwójnej gęstości flopy dyski lub twarde /hard disk/ dyski o wielomilionowej pojemności.

Moduł BIOS rozpoczyna się tablicą, w której zdefiniowano 17 instrukcji skoków do podprogramów w BIOS'ie. Moduły CCP i BDOS kiedy komunikują się z urządzeniami mikrokomputera wykorzystują tę tablicę do realizacji swoich funkcji.
Strona zerowa pamięci /adres 0000H do 0002H/ zawiera instrukcję skoku do drugiego podprogramu w tablicy skoków modułu BIOS.

Tablica 8

Tablica skoków w module BIOS

Nr kodu	Adres funkcji	Nazwa funkcji
1	2	3
00H	XX 00H	JMP BOOT ; ładowanie systemu /pierwszy raz/
03H	XX 03H	JMP WBOOT ; reinicjacja systemu
06H	XX 06H	JMP CONST ; status konsoli
09H	XX 09H	JMP CONIN ; czytaj znak z konsoli
0CH	XX 0CH	JMP CONOUT ; drukuj znak na konsoli
0FH	XX 0FH	JMP LIST ; wysyła znak na drukarkę
12H	XX 12H	JMP PUNCH ; wyślij znak na perforator

1	2	3
15H	XX 15H	JMP READER ; czytaj znak z czytnika
18H	XX 18H	JMP HOME ; ustaw głowice na ścieżce zerowej
1BH	XX 1BH	JMP SELDSK ; wybierz jednostkę dyskową
1EH	XX 1EH	JMP SETTRK ; wybierz ścieżkę
21H	XX 21H	JMP SETSEC ; wybierz sektor
24H	XX 24H	JMP SETDMA ; ustaw adres DMA
27H	XX 27H	JMP READ ; czytaj /128-bajtów/ sektor
2AH	XX 2AH	JMP WRITE ; pisz /128-bajtów/ sektor
2DH	XX 2DH	JMP LISTST ; określ status drukarki
30H	XX 30H	JMP SECTRAN ; określ sektor fizyczny

Wielkość XX oznacza wielokrotność 256-bajtowego obszaru, np. dla pamięci mikrokomputera wynoszącej 64k bajtów XX = F2H.
Bardziej szczegółowe funkcje wymienianych podprogramów przedstawiono poniżej:

- BOOT - ładuj system CP/M do pamięci RAM, czytając z dysku A binarne moduły CCP, BDOS i BIOS i rozmieszcza ich na odpowiednich adresach. Następnie inicjuje wszystkie wymagane obszary strony zerowej i przekazuje sterowanie do modułu CCP. WBOOT - reinicjuje działanie systemu po wciśnięciu przez operatora klawiszy CONTROL-C /musi to być pierwszy znak w nowej linii/
- CONST - ustala stan klawiatury. Informacja przekazywana jest w rejestrze akumulatora.

  A = OFFH oznacza, że znak jest gotowy do wczytania

  A = 00H oznacza, że znak nie oczekuje na wczytanie

  Moduł BDOS wywołuje podprogram jako funkcję 11.
- CONIN - czyta znak z konsoli i umieszcza go w rejestrze A. Normalnie CONIN będzie wywoływał podprogram CONST aż zawartość rejestru A = ØFFH. Podprogram CONIN wywoływany jest przez CCP i przez BDOS jako funkcja 1.
- CONOUT - wysyła znak na konsole /z rejestru C/. Podprogram musi najpierw sprawdzić czy konsola jest gotowa do przyjęcia znaku, opóźniając jeżeli to konieczne wysłanie znaku.
- LIST - wysyła znak z rejestru C na drukarkę. Podprogram jest wywoływany też przez moduł CCP w odpowiedzi na wciśnięcie klawisza CONTROL-P.
- PUNCH - wysyła znak do urządzenia zdefiniowanego jako

"perforator".

- READER - wczytuje znak z urządzenia "czytnik tp" do rejestru A
- HOME - wysyła do sterownika dysku polecenie umieszczenia głowicy na ścieżce i sektorze zerowym
- SELDSK - wybiera jednostkę dyskową wg numeru zadanego w rejestrze c /Ø = A, 1 = B, .../. Pamięta również ten numer dla późniejszego wykorzystania przez podprogramy READ i WRITE. Para rejestrów HL zawiera adres bloku DPH /Disk Parametr Header/
- SETTRK - wysyła do sterownika dysku polecenie umieszczenie głowicy na ścieżce o numerze zadanym w parze rejestrów BC. Numer ten wykorzystywany jest następnie przez podprogramy READ lub WRITE modułu BIOS. Moduł BDOS wywołuje ten podprogram wtedy gdy będzie wykonywana operacja czytania lub pisania 128-bajtowego "rekordu". Numer ścieżki jest wielkością absolutną i może być z przedziału 0-OFFFFH /65.535/
- SETSEC - przekazuje sterownikowi dysku numer sektora zadanego w rejestrze A /wartość z przedziału 0-OFFH /255/ /. Wartość sektora jest logicznym numerem, na podstawie którego podprogramu wyliczy numer fizyczny sektora.
- SETDMA - podprogram zachowuje w parze rejestrów BC adres obszaru /bufora/ pamięci używanego do operacji wejścia/ wyjścia. Wielkość bufora wynosi 128 bajtów, którego adres jest C80H. Użytkownik może ustalić położenie bufora w dowolnym miejscu pamięci RAM.
- READ - wczytuje rekord /długości 128 bajtów/ spod adresu określonego przez podprogramy SETTRK, SETSEC pod adres określony przez podprogram SETDMA. Jeżeli operacja wykonała się poprawnie, to rejestr A zawiera ØØH, w przypadku błędu ustawiona jest wartość Ø1H i wysłana jest na konsolę informacja: BDOS Errr on X: Bad Sector
  W takim przypadku /błędu/ można zignorować ten fakt wciskając klawisz RETURN lub przeładować /reinicjować/ system wciskając klawisze CONTROL-C przekazując sterowanie do modułu CCP. WRITE - działa tak jak podprogram READ, lecz w kierunku przeciwnym - z pamięci na dysk.
- LISTST - określa stan gotowości drukarki podając jej status w rejestrze A. Jeżeli A = ØFFH to drukarka może przy-

jąć znak, gdy A = ØØ to oznacza brak gotowości do przyjęcia znaku.

- SECTRAN - określa numer sektora fizycznego w parze rejestrów HL dla zadanego numeru logicznego w parze rejestrów BC. Adres tablicy translacji sektorów umieszczany jest w parze rejestrów DE.

Użytkownik we własnym programie może wywołać dowolny podprogram modułu BIOS wykorzystując adres punktu wejścia do tablicy skoków, który znajduje się w bajcie 1 i 2 strony zerowej.
Najbardziej znacząca wartość adresu znajduje się w bajcie 2 strony zerowej. Przykładowy podprogram wywołujący podprogram modułu BIOS /nie wykorzystujący wejścia do modułu BDOS/ wygląda następująco:

```
                ;PODPROGRAM BEZPOSREDNIO WYWOLUJE
                ;FUNKCE BIOS'U
                ;***NALEZY WYKONAC SEKWENCJE:
                ;

                ;       MVI L,NUMER$KODU
                ;       CALL BIOS
                ;.....TU POWROT
0000 F5         BIOS:       PUSH PSW ;ZACHOWAJ REJ. A
0001 3A0200                 LDA 0002H;POB.SPOD ADR. 2
                                     ;ADR.WEK.POD.
0004 67                     MOV H,A;W HL ADR. POD.
0005 F1                     POP PSW ;PRZYWROC ZAW. A
0006 E9                     PCHL    ;WYKONAJ PODP.
```

2.3. Zbiory dyskowe w module BIOS

System CP/M dzieli dyskietkę na dwie części. Pierwsza część składa się z dwóch pierwszych ścieżek i przeznaczona jest dla systemu. Rozmieszczenie modułów systemu jest następujące:

ścieżki	Sektory
Ø	01 ← CCP → 17 18 ← BDOS → 26
1	01 — BDOS → 19 20 ← BIOS → 26

Pozostałe ścieżki wykorzystywane są przez moduł BDOS dla zbiorów danych. Dla 8 calowej dyskietki wielkość ta wygląda następująco:

charakterystyka fizyczna dyskietki:	obliczenia	
77 ścieżek/dyskietce	77	ścieżek w ogółem
	- 2	ścieżki w CP/M
26 sektorów/ścieżka	75	ścieżek dla zbiorów
	x 26	ilość sektorów/ścieżce
128 bajtów/sektor	1950	il. sektorów dla zbiorów
	x 128	bajtów/sektor
2 ścieżki dla CP/M	249 600	bajtów dla zbiorów
	- 1024	wielkość bloku allok.
1024 bajtów/blok allokacji	243,75	ilość bloków allokacji
	242	numer ostatniego bloku allokacji /licząc od zera/

Zarządzanie wszystkimi plikami w systemie odbywa się poprzez moduł BDOS, który zakłada nowe zbiory, czyta i pisze do/z nich, zamyka i otwiera je. Zbiór opisany jest w bloku FCB /File Control Block/, z którego system korzysta w trakcie operacji na rekordach zbioru /patrz tablica 9/.

Tablica 9

Opis bloku FCB

Bajty	Zawartość
0	U - numer jednostki dyskowej
1 - 8	f - nazwa pliku
9 - 11	t - typ pliku i oznaczenie stanu
12	ex - numer wersji rozszerzenia
13 - 14	S - zarezerwowane przez system
15	rc - liczba rekordów
16 - 31	d - adresy używanych bloków /jednostek allokacji/
32	cr - bieżący numer rekordu czytanego lub pisanego sekwencyjnie
33 - 35	r - numer rekordu R/W o dostępie bezpośrednim w segmencie

gdzie:

ex = rb/128, r = rb mod 128

rb - numer bezwzględny w zbiorze

System czyta/pisze 128-bajtowe "rekordy" do sektorów, które

ponumerowane logicznie od 1 do 26. Każdemu numerowi logicznemu odpowiada jeden numer fizyczny /por. tablica 10/ - /w ramach jednej ścieżki/.

Tablica 10

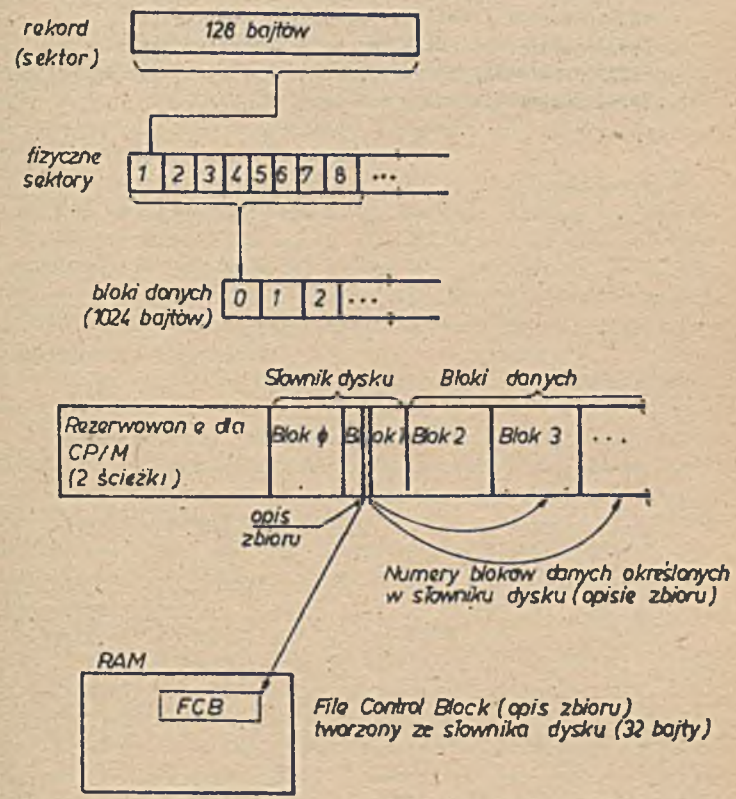
Powiązanie sektorów fizycznych i logicznych

Sektory fizyczne	Sektory logiczne
01, 07, 13, 19	Ø, 1, 2, 3
25, 05, 11, 17	4, 5, 6, 7
23, 03, 09, 15	8, 9, 10, 11
21, 02, 08, 14	12, 13, 14, 15
20, 26, 06, 12	16, 17, 18, 19
18, 24, 04, 10	20, 21, 22, 23
16, 22	24, 25

Między dwoma kolejnymi sektorami logicznymi występuje 6 rekordów fizycznych, które traktowane są przez system jako obszary "czasowe". W czasie gdy moduł czyta i ładuje sektor n do pamięci /bufora/ dyskietka obróci się dokładnie o dalsze 6 sektorów i ustawi się na logicznym sektorze n + 1. Rozwiązanie takie wiąże się z fizycznymi własnościami napędu dyskietek. Czas potrzebny na przeczytanie jednej pełnej ścieżki wynosi 4 1/2 sekundy. Każde osiem logicznych sektorów stanowi jednostkę allokacji zbioru, /wielkość ta zdefiniowana jest w module BIOS/. Zbiór musi składać się z całkowitej liczby jednostek allokacji /bloków/. Bloki numerowane są przez system /od zera/ i pamiętane w opisie zbioru /FCB/ w polu d.

Jeden blok FCB opisuje zbiór o wielkości 16 k bajtów. Jeśli plik jest większy, to tworzony jest nowy blok FCB w katalogu dysku z tą samą nazwą i typem zbioru. Katalog dysku /słownik/ zajmuje dwie pierwsze jednostki allokacji obszaru plików i zawiera opisy tych plików /blok Ø i 1 drugiej ścieżki/. Jeden opis w słowniku pliku zajmuje 32 bajtów /jest to pierwsze 32 bajtów bloku FCB/. Katalog dysku może zawierać max. 64 opisów plików danych. Powiązania pomiędzy sektorami, blokami allokacji, słownikiem dysku i blokiem opisu zbioru /FCB/ przedstawia rys. 6.

Moduł BDOS w trakcie dostępu do zbiorów korzysta z modułu BIOS, w którym zdefiniowane są tablice dyskowe opisujące fizyczną organizację dyskietek i zbiorów.



Rys.6 Powiązania pomiędzy sektorami, blokami danych, słownikiem dysku i blokiem FCB

Tablica DPH /ang. Disk Parametr Header/ tworzona jest oddzielnie dla każdej jednostki dyskowej, w której znajdują się adresy innych struktur /tablic/, opisujących dyskietki.
Tablica DPB /ang. Disk Parametr Block/ związana jest z logicznym typem dyskietki /jeden DPB dla każdego typu dyskietki/. Adres tablicy znajduje się w tablicy DPH.

3. ZBIORY DYSKOWE W SYSTEMIE OPERACYJNYM CP/M

3.1. Logiczna struktura dysku

Jedną z głównych funkcji każdego systemu operacyjnego jest zarządzanie zbiorami na nośnikach magnetycznych. System operacyjny zajmuje się alokacją zbioru na nośniku, gospodarką obszaru użytkowego, identyfikacją zbiorów, metodami dostępu do zbiorów, ochroną danych itp.

System operacyjny CP/M również realizuje te funkcje, chociaż z uwagi na swoją prostotę nie w tak szerokim zakresie jak systemy operacyjne dużych komputerów. Szczególnie uciążliwy dla użytkownika systemu CP/M może się okazać brak skutecznych mechanizmów ochrony danych. Inne typowe funkcje zarządzania zbiorami na nośnikach magnetycznych realizowane są przez system bądź poprzez oprogramowanie specjalne.

W dalszej części skoncentrujemy się na zbiorach przechowywanych na dyskach elastycznych /dyskietkach/.

Z punktu widzenia użytkownika systemu operacyjnego, szczególnie istotna jest gospodarka użytkowym obszarem dyskowym oraz sposób identyfikacji zbioru w systemie. Fizyczna struktura dyskietki została omówiona wcześniej i nie będziemy do niej wracać, skoncentrujemy się na logicznej strukturze dyskietki oraz sposobach identyfikowania zbiorów danych przez użytkownika systemu operacyjnego.

Użytkownik tworzy nowe zbiory na dyskietce nie troszcząc się o ich fizyczne rozmieszczenie. W zależności od wielkości zapisywanych zbiorów na jednym woluminie może być zapisany tylko jeden zbiór lub wiele zbiorów aż do wyczerpania wolnego obszaru. System operacyjny CP/M nie obsługuje natomiast zbiorów wielowoluminowych. Dla celów gospodarki obszarem dyskowym system operacyjny prowadzi katalog dysku, gdzie zapisane są informacje o wszystkich zbiorach zapisanych na dysku; wolnych obszarach użytkowych. Poprzez katalog system operacyjny identyfikuje zbiór na dyskietce i znajduje go fizycznie zgodnie z zapisanymi w katalogu numerami sektorów. Opis każdego zbioru w katalogu ma postać standardową i w zasadzie tworzony jest i wykorzystywany przez oprogramowanie systemu operacyjnego. Użytkownik mikrokomputera nie wykorzystuje katalogu dysku w sposób bezpośredni.

3.2. Identyfikacja zbiorów na dyskach

Użytkownik systemu operacyjnego powinien posiadać łatwe mechanizmy identyfikowania zbioru na dysku i dostępu do zapisanych w nim danych.

Zbiory na dyskach identyfikuje się poprzez nazwę. Nazwa może występować w postaci jawnej /identyfikuje wówczas najczęściej jeden konkretny zbiór/ lub w postaci maskowanej /identyfikuje wówczas najczęściej grupę zbiorów/.

Nazwa zbioru w systemie operacyjnym CP/M składa się z dwóch zasadniczych części:

- podstawowej,
- rozszerzenia.

Dla pełnej identyfikacji nazwę należy poprzedzić prefiksem określającym nazwę jednostki napędowej dysków elastycznych. Gdy nazwa jednostki dyskowej nie występuje, przez domniemanie przyjmowana jest nazwa aktualnie aktywnej jednostki dyskowej.

Nazwa zbioru przyjmuje postać ogólną:

X: n n n n n n n n . r r r

gdzie:

X - określa symbol jednostki dyskowej

n n n n n n n n - określa podstawową część nazwy zbioru o długości nie przekraczającej 8 znaków,

r r r - określa rozszerzenie nazwy, złożone z nie więcej niż 3 znaków

Obowiązkową częścią nazwy jest tylko część podstawowa, rozszerzenie i symbol jednostki są opcjonalne. Trzy części nazwy rozdzielone są reparatorami, co umożliwia pisanie tych części nie w pełnej długości ale tylko znaki znaczące. W takim przypadku pozostałe znaki są uzupełnione, do maksymalnej wielkości danej części nazwy, spacjami. Separatorem oddzielającym numer jednostki od części podstawowej nazwy jest znak dwukropka /:/, natomiast część podstawową od rozszerzenia oddziela znak kropki /./.

Pomijając symbol jednostki pomijamy również znak dwukropka, a pominięcie rozszerzenia wymaga pominięcia również znaku kropki. Pominięcie rozszerzenia oznacza, że chcemy identyfikować zbiór, który posiada podaną nazwę i rozszerzenie równe trzem spacjom.

Nazwa

B:ZBIOR.ABC

w pełni identyfikuje zbiór o nazwie ZBIOR i rozszerzeniu ABC znajdujący się w jednostce B dysków elastycznych.

Natomiast nazwa

ABCDEF

identyfikuje zbiór o nazwie ABCDEF z rozszerzeniem równym 3 spacje znajdującym się w jednostce dysków elastycznych, z której aktualnie wprowadzony jest system.

Nazwa jawnie identyfikująca zbiór nie może zawierać następujących znaków specjalnych:

. , ; : ? * [ ] < >

przy czym dopuszczalne są wszystkie znaki alfabetyczne, numeryczne oraz pozostałe znaki specjalne.

Trzyznakowe rozszerzenie nazwy zbioru oddzielone od części podstawowej kropką, pozwala na identyfikacje zbiorów zawierających podobne informacje w różnych reprezentacjach.

Rozszerzenie nazwy zbioru może być również wykorzystywane do oznaczenia generacji zbiorów stałych wykorzystywanych w systemach przetwarzania danych. Obsługa tego pola w pełni wykonana jest przez użytkownika systemu. System operacyjny nie przetwarza tego pola w sposób specjalny a jedynie interpretuje zapisane tam znaki poprzez niektóre swoje procedury nierezydentne.

Nośnik dyskowy wykorzystywany jest do zapisywania zbiorów danych wymagających zapamiętania przez pewien okres czasu. Dane te to najczęściej informacje stałe i robocze wykorzystywane w systemach obliczeniowych wykonywanych na mikrokomputerze. Ponadto na dysku przechowywane są zbiory zawierające inne informacje, takie jak różne postacie programów, zbiory tymczasowe, biblioteki wspomagające pracę programisty, czy też nierezydentne procedury systemu operacyjnego. Dla częściowej identyfikacji zawartości zbioru dyskowego system operacyjny wykorzystuje pole rozszerzenia nazwy zbioru, gdzie pewne standardowe znaki mają określoną interpretację.

System operacyjny CP/M poprzez standardowe rozszerzenia nazwy zbiorów określa typ danego zbioru, co jest wykorzystywane w nierezydentnych procedurach systemu.

System CP/M przyjmuje następujące standardowe rozszerzenia nazw zbiorów:

. ASM - dla zbiorów zawierających program źródłowy w języku Assembler,
. BAS - dla zbiorów zawierających program źródłowy w języku BASIC,
. FOR - dla zbiorów zawierających program źródłowy w języku FORTRAN,
. COM - dla zbiorów zawierających program w postaci wykonywalnej /tzn. użycie nazwy zbioru z rozszerzeniem COM powoduje ładowanie tego programu do obszaru TPA systemu i jego automatyczne uruchomienie/,
. PRN - dla zbiorów zawierających wydruk pokompilacyjny powstały w wyniku kompilacji programu procedurami nierezydentnymi,
. REL - dla zbiorów zawierających postać półskomplikowaną, nieskonsolidowaną powstałą w wyniku kompilacji programu procedurami nierezydentnymi /zbiór taki jest zbiorem wejściowym dla programu łączącego/
. LIB - dla zbiorów zawierających biblioteki podprogramów w postaci półskomplikowanej, nieskonsolidowanej /zbiór taki jest wykorzystywany przez program łączący dla dołączenia modułów na etapie łączenia/,
. TEX - dla zbiorów zawierających tekst,
. HEX - dla zbiorów w formacie hexadecymalnym /zbiory takie powstają w wyniku kompilacji języka Assembler/,
. BAK - dla zbiorów starej generacji wykorzystywanych przez EDYTOR,
. [illegible] - dla zbiorów roboczych w systemie.

Standardowe rozszerzenia nazw zbiorów, jak widać z podanych znaczeń, dotyczą głównie zbiorów zawierających programy w różnych postaciach. Od postaci źródłowej poczynając, poprzez pośrednie postacie procesu kompilacji, aż do postaci wynikowej.

Przechowywanie programów na dysku w systemie operacyjnym CP/M jest nieco odmienne od stosowanych w dużych systemach operacyjnych rozwiązań. System operacyjny CP/M nie obsługuje zbioru dyskowego zawierającego bibliotekę programów, to znaczy zbioru o specjalnej organizacji, którego elementami są kolejne programy w różnej postaci przechowania. Zbiór w systemie operacyjnym CP/M może zawierać tylko jeden program, a rozszerzenie wskazuje na postać tego zbioru. Rozszerzenia .ASM .BAS i .FOR są wykorzysty-

wane dla przechowania postaci źródłowej trzech głównych języków programowania. Rozszerzenie .COM jest wykorzystywane dla przechowywania wszelkich programów w postaci ładowalnej. Wypisanie na klawiaturze nazwy zbioru /tylko części podstawowej/, powoduje wyszukanie na dysku zbioru z wypisaną nazwą i przyjętym przez domniemanie rozszerzeniem .COM, załadowanie zawartości tego zbioru do pamięci i automatyczne wykonanie. W zbiorach z rozszerzeniem .COM oprócz programów ładowalnych użytkownika przechowywane są również procedury nierezydentne systemu operacyjnego. Inne rozszerzenia nadawane są automatycznie podczas procesu kompilacji, dla zbiorów zawierających postacie pośrednie kompilacji.

W związku z takim sposobem wykorzystania rozszerzenia nazwy zbioru, zalecane jest, aby część podstawowa nazwy zbiorów, zawierających różne postacie programu, była niezmienna a poprzez rozszerzenie można identyfikować jaka postać przechowania znajduje się w konkretnym zbiorze dyskowym.

I tak zbiory o nazwach

PROG.BAS

PROG.PRN

PROG.REL

PROG.COM

zawierają program o nazwie PROG w różnych postaciach, od źródłowej w języku programowania BASIC, poprzez pośrednie postacie kompilacji, do postaci ładowalnej.

Rozszerzenie jest traktowane jako integralna część nazwy zbioru. Niektóre procedury nierezydentne przyjmują przez domniemanie określone rozszerzenia. Dla przykładu procedura BASC umożliwiająca kompilację programu w języku BASIC przyjmuje i akceptuje jedynie zbiory z rozszerzeniem .BAS jako zbiory wejściowe do kompilacji, natomiast dla zbiorów wynikowych dodaje rozszerzenia .PRN i .REL odpowiednio do ich zawartości.

Postać jawna nazwy zbioru wykorzystywana jest do przeszukiwania katalogu dysku lub w fazie tworzenia nowego zbioru. Gdy chcemy operować grupą zbiorów możemy wykorzystywać nazwy maskowane. Nazwa będzie maskowana jeśli występują w niej znaki "?" lub "*".

Użycie znaku "?" w nazwie oznacza maskowanie w miejscu wystąpienia "?" dowolnego znaku w danej pozycji. Znak zapytania

maskuje tylko jeden znak na określonej pozycji. Gdy chcemy maskować kilka znaków należy powtórzyć znak zapytania na kolejnych maskowanych pozycjach nazwy.

Nazwa maskowana

ABC???.COM

jest równoważna nazwom

ABCDEF.COM

ABCKLA.COM

ABCXYZ.COM

i innym nazwom zaczynającym się od liter ABC i posiadającym rozszerzenie .COM.

Użycie natomiast znaku "*" w nazwie zbioru oznacza maskowanie całej części nazwy zbioru /podczas gdy znak "?" maskuje tylko 1 znak nazwy/.

Nazwa maskowana

*.COM

oznacza wszystkie zbiory z rozszerzeniem .COM i dowolną częścią podstawową, nazwa maskowana

ABC.*

oznacza wszystkie zbiory o nazwie podstawowej ABC z dowolnym rozszerzeniem, a nazwa maskowana

.

oznacza wszystkie zbiory na aktualnie aktywnej dyskietce.

Przykłady nazw zbiorów

PROG1.BAS

PROG1.PRN

PROG1.COM

DANE

DANE.1

B:DANE

powyższe nazwy są nazwami jawnymi dla różnych zbiorów, jedynie zbiory identyfikowane przez DANE i B:DANE mogą dotyczyć tego samego zbioru, gdy aktualnie aktywną jednostką napędową jest jednostka B.

Nazwa maskowana

PROG1.*

pozwala na odwołanie się do trzech pierwszych zbiorów z powyższego przykładu

A:NAZ.:

B,A.ROZ

KLM,B

Powyższe nazwy są przykładem użycia niedopuszczalnych znaków, czyli nie mogą być nazwami zbiorów.

4. KOMENDY REZYDENTNE

4.1. Komendy systemu operacyjnego CP/M

Pisząc o strukturze dyskietki mówiliśmy, że dwie pierwsze ścieżki każdej dyskietki zajęte są przez system operacyjny. Przed rozpoczęciem pracy system operacyjny z dyskietki musi być wprowadzony do pamięci mikrokomputera. Oczywiście nie cały system operacyjny znajduje się w pamięci a jedynie jego część rezydentna, czyli zawierająca podstawowe, najczęściej wykorzystywane procedury systemu operacyjnego.

Dla komunikowania się z systemem operacyjnym użytkownik wykorzystuje komendy, które interpretowane powodują wykonanie odpowiednich procedur systemu operacyjnego. Procedury znajdujące się w części rezydentnej systemu mogą być uruchamiane poprzez wypisanie ich nazwy na klawiaturze i wciśnięcie klawisza RETURN. Inne wymagają kontaktu ze zbiorem dyskowym.

W związku z tym w systemie operacyjnym CP/M wyróżnia się dwa typy komend:

- rezydentne, tzn. znajdujące się w rezydentnej części systemu operacyjnego, możliwe do wykonania bez odwołania się do jednostki dyskowej,
- nierezydentne, tzn. znajdujące się w nierezydentnej części systemu czyli w zbiorach na dysku. Wykonanie tych komend wymaga odwołania się do dyskietki, gdzie w zbiorze z rozszerzeniem .COM znajduje się dana komenda.

Komendy rezydentne i nierezydentne wywołuje się przez napisanie ich nazwy na klawiaturze i wciśnięcie klawisza RETURN. W przypadku komend nierezydentnych wymagane jest aby na dyskietce, do której następuje odwołanie występował zbiór zawierający procedurę obsługi tej komendy. Gdy zbiór taki jest niedostępny, komenda nie jest wykonywana a na monitorze wypisywany jest odpowiedni komunikat.

W tym rozdziale zajmiemy się komendami rezydentnymi a w następnym omówimy najważniejsze dla użytkownika komendy nierezydentne.

4.2. Komenda DIR

Komenda DIR /directory/ wyprowadza na konsolę systemową nazwy wszystkich zbiorów z katalogu dysku w postaci część podstawowa i rozszerzenie. Podanie komendy DIR /bez operandu/ powoduje wyprowadzenie nazw wszystkich zbiorów znajdujących się na aktualnie przydzielonym dysku. Komenda w tej postaci jest równoważna komendzie DIR *.*. Formalna postać komendy jest następująca:

DIR nazwa zbioru

Jeśli nazwa zbioru w komendzie nie występuje to drukowana jest pełna zawartość katalogu dysku, gdy nazwa zbioru występuje w postaci jawnej lub maskowanej, to drukowane są tylko informacje o zbiorach, których nazwa dotyczy.

Przykłady użycia komendy:

```
DIR X.Y
DIR X?Y.C?M
DIR ??.Y
```

Podobnie do innych komend CCP, nazwa maskowana może być poprzedzona przez nazwę jednostki napędowej dysków elastycznych. W podanych niżej przykładach użycia komendy DIR przed wyprowadzeniem nazw zbiorów nastąpi wybranie odpowiedniej jednostki dyskowej:

```
DIR B:
DIR B:X.Y
DIR B:*.A?M
```

Jeżeli na wskazanej jednostce napędowej dysków elastycznych nie zostaną znalezione zbiory, to na konsolę systemową wyprowadzony będzie komunikat:

NOT FOUND

Komenda DIR bez operanda jest najczęściej pierwszą komendą odnoszącą się do nowo zainstalowanej dyskietki, gdyż pozwala na zorientowanie się w jej zawartości.

4.3. Komenda ERA

Komenda ERA /erase/ usuwa, kasuje zbiór, którego nazwa występuje po komendzie, z aktualnie przydzielonej jednostki dyskowej.

Postać komendy

ERA nazwa zbioru

W komendzie można wykorzystywać nazwy jawne oraz nazwy maskowane.

Kasowaniu nie podlegają fizycznie dane zapisane w zbiorze lub zbiorach kasowanych, a jedynie aktualizowane są zapisy w katalogu dysku. Z katalogu wymazywana jest informacja o kasowanym zbiorze a zajmowany przez niego obszar, traktowany jest przez system operacyjny jako obszar wolny. Fizyczne zniszczenie zapisanych danych nastąpi w momencie przydzielenia tego obszaru dla innego zbioru zapisywanego na dysku.

System operacyjny CP/M nie posiada żadnych mechanizmów ochrony danych przed przypadkowym zniszczeniem. Wypisanie komendy ERA i jej wykonanie powoduje wymazanie danych z systemu operacyjnego. W związku z tym należy bardzo ostrożnie korzystać z komendy ERA, szczególnie w przypadku użycia nazw maskowanych zbiorów kasowanych.

Przykłady użycia komendy:

ERA X.Y	zbiór o nazwie X.Y umieszczony na aktualnie przydzielonym dysku jest usunięty a dotychczas zajmowany obszar może być wykorzystany.
ERA X.*	wszystkie zbiory o nazwie X /część podstawowa nazwy/ są usuwane z aktualnie przydzielonego dysku.
ERA *.ASM	wszystkie zbiory o rozszerzeniu nazwy ASM są usuwane z aktualnie przydzielonego dysku.
ERA X?Y.C?M	wszystkie zbiory odpowiadające nazwie maskowanej X?Y.C?M są usuwane z aktualnie przydzielonego dysku
ERA B:ZB1.A	zbiór o nazwie ZB1 i rozszerzeniu A jest usunięty z dyskietki w jednostce B.

4.4. Komenda REN

Komenda REN /rename/ umożliwia użytkownikowi zmianę nazw zbiorów dyskowych. Postać komendy jest następująca:

REN nowa nazwa zb = stara nazwa zbioru

Zbiór o nazwie stara nazwa otrzymuje nazwę nowa nazwa.

Przykłady użycia komendy:

REN X.Y = Q.R	Zbiór Q.R otrzymuje nową nazwę X.Y
REN XYZ.COM = XYZ.A	Zbiór XYZ.A otrzymuje nową nazwę XYZ.COM

Nazwy zbiorów mogą być poprzedzone nazwą jednostek napędowych dysków elastycznych. Przyjmuje się, że jeżeli podana zostanie

nazwa jednostki napędowej dysków dla starej nazwy, to zbiór nowa nazwa związany jest z tą samą jednostką napędową. Jeżeli obie nazwy zbiorów poprzedzone są nazwami jednostek napędowych, to należy użyć w obu przypadkach tych samych nazw, gdyż operacja zmiany nazwy wykonywana jest tylko w katalogu dysku i nie powoduje żadnych operacji na danych. Jeżeli w komendzie użyjemy jako nazwy nowej nazwy uprzednio już występującej, to wyprowadzony zostanie komunikat o błędzie:

FILE EXISTS

a zmiana nazwy zbioru nie będzie wykonana.
Podobnie użycie jako starej nazwy, nazwy zbioru nieistniejącego spowoduje wyprowadzenie komunikatu:

NOT FOUND

W komendzie REN można używać tylko nazw jawnych.

4.5. Komenda SAVE

Komenda SAVE powoduje wyskładowanie n stron /256-bajtowe bloki pamięci/ obszaru TPA do zbioru dyskowego o podanej nazwie. W systemie CP/M obszar TPA rozpoczyna się od drugiej strony, tzn. od adresu 100H. Jeżeli program użytkownika zajmuje obszar o adresach od 100H do 2FFH, to komenda SAVE musi odwoływać się do 2 stron pamięci.
Postać komendy jest następująca:

SAVE n nazwa zbioru

Przykłady użycia komendy:

SAVE 3 X.COM Kopiuje pamięć od adresu 100H do 3FFH do zbioru X.COM

SAVE 40 Q Kopiuje pamięć od adresu 100H do 28FFH do zbioru o nazwie Q

SAVE 4 X.Y Kopiuje pamięć od adresu 100H do 4FFH do zbioru o nazwie X.Y

Komenda SAVE może również odwoływać się do nazw jednostek napędowych dysków elastycznych.

SAVE 10 B:ZT.COM Kopiuje pamięć od adresu 100H do DAFFH do zbioru ZT.COM znajdującego się na jednostce napędowej dysków elastycznych B.

4.6. Komenda TYPE

Komenda TYPE wyprowadza zawartość wskazanego zbioru źródłowego w kodzie ASCII z aktualnie przydzielonego dysku. Zawartość zbioru wyprowadzana jest na konsolę systemową. Komenda TYPE może odwoływać się do nazw jednostek napędowych dysków elastycznych. Postać komendy jest następująca:

TYPE nazwa zbioru

Przykłady użycia komendy:

TYPE X.PLM

TYPE XXX

TYPE B:C.PRN

W komendzie TYPE można używać tylko nazw jawnych.

Komendy tej używa się dla wylistowania zawartości zbioru dyskowego. Należy pamiętać, że wydruk odbywa się w kodzie ASCII i w związku z tym może dotyczyć tylko zbiorów, których zawartość posiada interpretację graficzną w tym kodzie. W szczególności komendy TYPE nie należy używać do zbiorów z rozszerzeniem .COM.

5. KOMENDY NIEREZYDENTNE

5.1. Informacje ogólne o komendach nierezydentnych

Komenda nierezydentna, to program systemowy ładowany do obszaru TPA wykonywany pod kontrolą modułu CCP. Komendy nierezydentne ładowane są z odpowiedniego zbioru dyskowego o nazwie odpowiadającej nazwie komendy i rozszerzeniu typu COM. System CP/M obsługuje następujące, podstawowe komendy nierezydentne:

STAT
ASM
ED
LOAD
DDT
PIP
DUMP
MOVCPM
SYSGEN

Niektóre z nich zostaną omówione w tym rozdziale a inne w innym miejscu.

5.2. Komenda STAT

Komenda STAT może w zależności od podanej postaci realizować różne funkcje. Podstawową funkcją komendy jest dostarczenie użytkownikowi informacji statystycznej o zasobach mikrokomputera, w szczególności o zbiorach na dysku i zagospodarowaniu urządzeń. Informacja statystyczna może mieć różny stopień szczegółowości, co użytkownik osiąga poprzez podanie odpowiednich dyspozycji w komendzie. Przy pomocy komendy STAT można również dokonać czasowej zmiany przydziału urządzeń zewnętrznych.
Postać komendy jest następująca:

STAT

lub

STAT dyspozycja

Pierwsza postać komendy inicjuje obliczanie i wyprowadzanie informacji o wolnych obszarach na poszczególnych jednostkach dyskowych. Informacja wyprowadzana jest w następującej formie:

X:R/W,SPACE:nnnK

lub

X:R/O,SPACE:nnnK

gdzie:

dla każdej jednostki dyskowej X:

R/W - wskazuje zbiór, z którego dane mogą być czytane lub pisane,

R/O - wskazuje zbiór, z którego dane mogą być tylko czytane,

nnn - wielkość dostępnej pamięci na dysku podawana w kbajtach.

Druga postać komendy STAT może zawierać dyspozycje, które realizują następujące funkcje:

STAT X:

X: - nazwa jednostki dyskowej

Podawana jest wówczas wielkość wolnej pamięci na danej jednostce dyskowej. Postać wyprowadzanej informacji jest następująca: BYTES REMAINING ON X: nnnK

Inna postać komendy

STAT nazwa zbioru

gdzie:

nazwa zbioru - nazwa zbioru dyskowego

powoduje wyprowadzenie na konsolę informacji o zbiorze wskazanym przez nazwę umieszczoną w komendzie.

Informacja wyprowadzana jest w następującej formie:

RECS BYTS EX D:FILENAME.TYP

rrr bbbK ee d:pppppppp.sss

gdzie:

rrr - liczba 128-bajtowych rekordów,

bbb - liczba kbajtów zajmowanych przez zbiór na dysku /bbb = rrr x 128/1024/,

ee - liczba obszarów zajmowanych przez zbiór, przy czym każdy obszar jest 16 kbajtowym segmentem pamięci dyskowej /ee = bbb/16/,

d: - symbol /nazwa/ jednostki dyskowej,

pppppppp - część podstawowa nazwy zbioru,

sss - rozszerzenie nazwy zbioru.

STAT X: nazwa zbioru

Podana postać komendy powoduje wykonanie poprzednio omówionej funkcji dla zbioru znajdującego się na wskazanej przez X: jednostce dyskowej.

STAT X: nazwa zbioru. typ $S

gdzie:

X: nazwa jednostki dyskowej

nazwa zbioru.typ - część podstawowa i rozszerzenie nazwy zbioru

Podana postać komendy powoduje wyprowadzenie informacji w przykładowo przedstawionej formie:

Size	Recs	Bytes	Ext	Acc	
48	48	6K	1	R/O	A:ED.COM
55	55	12K	1	R/O	A:PIP.COM
65536	128	2K	2	R/W	A:X.DAT

Parametr $S powoduje wyprowadzenie informacji "Size".

Pole "Size" przedstawia rozmiar zbioru pozornego w rekordach, natomiast "Recs" podaje liczbę rekordów pozornych w obszarze 16k bajtowym zbioru. Dla zbiorów o organizacji sekwencyjnej obie te wartości są jednakowe. Pole "Acc" wskazuje status zbioru. Status ten określa tryb dostępu do zbioru /odczyt, zapis/. Status dostępu może być zmieniany przez podanie komendy w jednej z poniższych postaci:

STAT d:nazwa zbioru.typ $R/O
STAT d:nazwa zbioru.typ $R/W
STAT d:nazwa zbioru.typ $SYS
STAT d:nazwa zbioru.typ $DIR

gdzie:

R/O - zbiór przeznaczony tylko do odczytu,

R/W - zbiór przeznaczony do zapisu i odczytu,

SYS - zbiór systemowy,

DIR - usunięcie statusu zbioru.

O ile zbiór ma nadany status R/O, to próba wykonania operacji zapisu do zbioru lub jego usunięcia spowoduje wyprowadzenie komunikatu:

BDOS Err on d: File R/O

Komenda STAT umożliwia również uzyskanie informacji i sterowanie przydziałem fizycznych urządzeń zewnętrznych do jednostek logicznych obsługiwanych na poziomie systemu operacyjnego. System wykorzystuje cztery jednostki logiczne, do których mogą być przypisane różne urządzenia zewnętrzne. CP/M używa następujących jednostek logicznych:

CON: konsola systemowa /wykorzystywana przez CCP do komunikacji z operatorem/,

RDR: urządzenie wejściowe,

PUN: urządzenie wyjściowe,
LST: drukarka.
Urządzenia fizyczne przyłączone do systemu sterowane są podprogramami modułu BIOS, gdzie przypisania mają charakter stały dla danej implementacji systemu operacyjnego.
Fizycznymi urządzeniami systemu CP/M mogą być:
CRT: - monitor ekranowy,
TTY: - konsola dalekopisowa,
BAT: - urządzenie wejście znakowe, na przykład czytnik taśmy papierowej,
UC1: - konsola systemowa zdefiniowana przez użytkownika,
PTR: - szybki czytnik taśmy papierowej,
UR1: - czytnik zdefiniowany przez użytkownika /nr 1/,
UR2: - czytnik zdefiniowany przez użytkownika /nr 2/,
PTP: - perforator taśmy papierowej,
UP1: - perforator nr 1 zdefiniowany przez użytkownika,
UP2: - perforator nr 2 zdefiniowany przez użytkownika,
LPT: - drukarka,
UL1: - urządzenie drukujące zdefiniowane przez użytkownika.
U w a g a : Dwukropek jest częścią składową nazwy.
Wymienione nazwy urządzeń fizycznych nie zawsze trafnie charakteryzują urządzenia faktycznie możliwe do przyłączenia w systemie. Przykładowo PTP: może symbolizować pamięć zewnętrzną na kasecie magnetycznej. Dokładny opis przyłączonych urządzeń fizycznych określony jest przy generacji systemu CP/M w module BIOS.
Komenda zapisana w postaci
STAT VAL: - drukuje listę wszystkich możliwych przypisań urządzeń fizycznych do jednostek logicznych.
Informacja wyprowadzana jest w następującej formie:

CON:=TTY:CRT:BAT:UC1:
RDR:=TTY:PTR:UR1:UR2:
PUN:=TTY:PTP:UP1:UP2:
LST:=TTY:CRT:LPT:UL1:

Każde z wymienionych czterech urządzeń fizycznych może być przypisane do jednostki logicznej podanej po lewej stronie znaku równości.
Komenda
STAT DEV: - wyprowadza aktualny przydział urządzeń fizycznych do jednostek logicznych w postaci listy, której przy-

kład podano poniżej:

CON:=CRT:

RDR:=UR1:

PUN:=PTP:

LST:=TTY:

Przy pomocy komendy STAT można czasowo zmienić przypisanie urządzeń fizycznych do urządzeń logicznych zgodnie z aktualnymi potrzebami użytkownika.

STAT id1,id2=pd2,...idn=pdn

gdzie:

id_i - nazwa jednostki logicznej,

pd_i - nazwa urządzenia fizycznego.

Przy pomocy przedstawionej wersji komendy STAT można aktualizować przypisania urządzeń zewnętrznych. Przykładowa aktualizacja przypisań może być następująca:

STAT LST:=LPT:,RDR:=TTY:

W porównaniu z przykładowo podanym wcześniej przypisaniem zostały zmienione przydziały fizycznych urządzeń dla drukarki i urządzenia wejścia. W miejsce dotychczasowej konsoli dalekopisowej jako urządzenie drukujące została przypisana drukarka, a w miejsce zdefiniowanego przez użytkownika czytnika jako urządzenie wejściowe została przypisana konsola dalekopisowa.

Przydział taki obowiązuje do ponownego użycia komendy STAT zmieniającej ten przydział lub do ponownego ładowania systemu operacyjnego.

Komenda STAT umożliwia też uzyskanie szczegółowej informacji o jednostce dyskowej.

STAT d:DSK: - komenda w tej postaci umożliwia wyprowadzenie informacji stanowiących charakterystykę jednostki dyskowej o nazwie d:, przy czym nazwa jest z zakresu A: , B: ,..., P:.

Charakterystyka dysku wyprowadzana jest w następującej formie:

d: charakterystyka dysku

65536: 128 bajtów rozmiar rekordu, łączna pojemność rekordów

8192: kbajtów pojemność dysku, łączna pojemność w kbajtach

128: 32 bajty katalogu } wielkość katalogu reprezentowana

0: obszar katalogu } przez ilość wprowadzonych zbiorów

1024: rekordy/obszary

128: rekordy/blok

58: sektory/ścieżki
2: ścieżki dodatkowe

Podanie komendy w postaci:

STAT DSK:

spowoduje wyprowadzenie informacji charakteryzujących wszystkie aktywne dyski.

5.3. Komenda PIP

Komenda PIP /Peripheral Interchange Program/ jest programem realizującym przesłanie zbiorów danych między różnymi nośnikami wykorzystywanymi w konfiguracji mikrokomputera pracującego pod kontrolą systemu operacyjnego CP/M.

Inicjowanie pracy odbywa się poprzez podanie komendy w jednej z poniższych postaci:

PIP

lub

PIP dyspozycja

gdzie:

dyspozycja - parametry podane w następującej postaci:

d=s1,s2, ..., sn

gdzie:

d - zbiór lub jednostka logiczna odbierająca dane,

s_i - zbiór lub urządzenie, z którego następuje przesłanie danych.

Wywołanie komendy w pierwszej postaci umożliwia podawanie dyspozycji z konsoli systemowej. Program zgłasza gotowość przyjmowania dyspozycji wyprowadzając znak "*" na konsolę. Koniec dyspozycji dla programu wskazywany jest podaniem znaku "cr".

Druga postać komendy przyjmuje dyspozycję podaną bezpośrednio jako operand. Po wykonaniu polecenia program kończy obsługę dyspozycji, przenosząc sterowanie na poziom komend systemu CP/M. Dla wykonania jednej operacji wygodniejsza jest druga postać komendy, natomiast wykonanie większej liczby operacji wykonujemy przy pomocy pierwszej postaci komendy, która powoduje jednorazowe ładowanie procedury PIP do pamięci. Kolejne operacje wydawane po znaku zaproszenie do pisania * wykonywane są tak długo dopóki użytkownik nie wciśnie klawisza RETURN z pustą linią dyspozycji. Powoduje to powrót do systemu.

Zakłada się, że każdy ze zbiorów źródłowych zawiera znaki kodu ASCII i zakończony jest standardowym znakiem końca zbioru 1AH /za wyjątkiem przedstawionym przy omawianiu parametru O/. Maksymalna długość dyspozycji w komendzie nie może przekraczać 255 znaków.

Zbiór określony jako odbierający dane może być jednocześnie jednym ze zbiorów źródłowych. W takim przypadku dopiero bezbłędne wykonanie całej dyspozycji powoduje zmianę w zapisie tak użytego zbioru odbierającego dane. Pominięcie nazwy zbioru odbierającego powoduje, że zbiór wyjściowy przyjmuje nazwę zbioru wejściowego. Błąd w realizacji programu spowoduje, że zbiór odbierający dane nie będzie zmieniony.

Przykład użycia:

PIP ZBIOR.COM=B:ZBIOR.COM - kopiuje z dysku B: zbiór o nazwie ZBIOR.COM, do zbioru o takiej samej nazwie na dysku przydzielonym do systemu. Zbiór na dysku B: pozostaje bez zmian.

```
PIP
*X=Y,Z
*B:A.BAS=B:B.BAS,A:C.BAS.D.BAS
* cr
```

Realizacja przedstawionego ciągu dyspozycji spowoduje połączenie zbiorów Y oraz Z z zapisaniem do zbioru X i połączenie zbiorów B, C, D o rozszerzeniu typu BAS z zapisaniem do zbioru A.BAS na dysku B:

Program PIP akceptuje także skrócony format parametrów. Reguły skracania są następujące:

PIP X: = nazwa zbioru - kopiuje wszystkie zbiory o podanej nazwie z dysku aktualnie przydzielonego do systemu na dysk X:. Nazwa /lub nazwy/ zbiorów nie ulega zmianie.

PIP X: = Y:nazwa zbioru - kopiuje wszystkie zbiory o podanej nazwie z dysku o nazwie Y:

PIP nazwa zbioru = Y: - komenda ta jest równoważna wywołaniu PIP nazwa zbioru = Y: nazwa zbioru i powoduje kopiowanie zbioru o wskazanej nazwie z jednostki dyskowej Y: na dysk aktualnie przydzielony do systemu /bez zmiany nazwy zbioru/.

PIP X:nazwa zbioru=Y: - wywołanie to jest równoważne poprzedniemu z tym, że dyskiem przeznaczenie jest jednostka X:.

Dla wszystkich form skróconych zbiory muszą znajdować się na różnych jednostkach. Jeżeli na dysku przeznaczenia /odbierającym dane/ istnieje już zbiór o podanej nazwie, wtedy zawartość zbioru źródłowego jest kopiowana do istniejącego już zbioru, powodując zniszczenie poprzedniej zawartości.

W komendzie PIP można posługiwać się również nazwami maskowanymi zbiorów.

Przykłady użycia:

PIP B:=*.COM - kopiuje wszystkie zbiory o rozszerzeniu nazwy typu COM z jednostki aktualnie przydzielonej do systemu na dysk B:

PIP A:=B:ZAP.* - kopiuje z dysku B: na dysk A: wszystkie zbiory o nazwie ZAP, bez względu na to jakie posiadają rozszerzenie

PIP A:=B:*.* - kopiuje wszystkie zbiory z dysku B: na dysk A:, zachowując ich nazwy.

Komenda PIP pozwala również odwoływać się do fizycznych i logicznych jednostek, które przyłączone są do systemu. Nazwy urządzeń są analogiczne z nazwami podanymi przy omawianiu komendy STAT, z tym że ich lista jest dodatkowo uzupełniona kilkoma urządzeniami specjalnymi. Dodatkowe urządzenia, to:

NUL - przesłanie 40 pustych znaków zer kodu ASCII /np. koniec perforacji/,

EOF - przesłanie znaku końca zbioru /1AH = ctl-Z w kodzie ASCII/ do urządzenia odbierającego. Znak ten jest także automatycznie wstawiany na końcu każdej transmisji kodu ASCII.

PRN - urządzenie spełniające funkcje LST /patrz opis komendy STAT/, za wyjątkiem tego, że znaki tabulacji działają na każdą ósmą kolumnę tekstu, wiersze są numerowane, a nowa strona rozpoczyna się co 60 wierszy.

Podając w dyspozycji typ urządzenia należy pamiętać, że urządzenie przeznaczenia musi być zdolne do przyjmowania danych, a urządzenie źródłowe musi mieć możliwość generowania danych /np. LST: nie może być źródłem danych, a PRN: - przeznaczeniem/. Jeżeli zbiorem odbierającym dane jest zbiór dyskowy, to podczas przebiegu programu PIP tworzony jest zbiór tymczasowy o rozszer-

rzeniu typu ZZ, który przyjmuje właściwą nazwę po pozytywnym zakończeniu kopiowania.
Kopiowanie można przerwać w każdym momencie, używając do tego celu dowolnego klawisza mikrokomputera.
Kopiowanie z wyprowadzaniem na urządzenia fizyczne ilustruje kilka poniższych przykładów:

PIP LST:=PROG.BAS - drukuje zbiór o nazwie PROG.BAS

PIP CON:=X.ASM,Y.ASM,Z.ASM - łączy zbiory o rozszerzeniu ASM oraz wyprowadza zawartość zbiorów na konsolę

PIP X.HEX=CON:,Y.HEX,PTR: - otwiera zbiór o nazwie X.HEX oraz wpisuje do niego dane czytane z konsoli /aż do wciśnięcia klawisza funkcyjnego ctl-Z/, po czym dołącza dane ze zbioru Y,HEX i przeczytane z urządzenia PTR: /aż do napotkania znaku ctl-Z/

PIP PUN:=NUL,X.ASM,POF:,NUL: - przesyła na perforator 40 znaków pustych, następnie przesyła zawartość zbioru X.ASM oraz znak końca zbioru i 40 znaków pustych.

Program PIP akceptuje także dodatkowe parametry modyfikujące jego działanie. Parametry te /w liczbie co najmniej 1/ muszą być zamknięte w nawiasach kwadratowych, mogą /lecz nie obowiązkowo/ być oddzielone od siebie spacjami i dotyczą tego elementu wiersza dyspozycji, po którym bezpośrednio następują.
Parametry programu PIP mają następującą postać:

B - przesyłanie blokowe. Dane są buforowane przed przesłaniem tak długo, dopóki z urządzenia będącego źródłem danych nie nadejdzie znak ctl-S. Pozwala to przesłać dane na dysk z urządzenia ciągłego czytania /np. czytnik taśmy magnetycznej w kasecie/. Po otrzymaniu znaku ctl-S bufory są opróżnione, a program PIP oczekuje na dalsze dane wejściowe. Wielkość bufora zależy od rozmiaru pamięci mikrokomputera.

Dm - podczas przesyłania usuwane będą te dane, które znajdują się poza kolumną m. Parametr ten używany jest najczęściej do redagowania wierszy przesyłanych na wąską drukarkę lub konsolę.

E - odwzorowuje na konsoli wszystkie operacje przesłania

F - eliminuje z przesyłanego zbioru wszystkie znaki wysuwu do nowej strony

Gn - udostępnia użytkownikowi zbiór z numerem n. /n z przedziału 1-15/

H - nadzoruje transmisję danych heksadecymalnych

I - pomija rekordy typu ":00" w czasie transmisji zbioru heksadecymalnego. Parametr ten automatycznie włącza parametr H

L - zmienia wielkie litery na małe

N - dodaje do każdego wyprowadzanego wiersza numer /poczynając od numeru 1 z przyrostem 1/. Prowadzące zera są pomijane. Po numerze wiersza umieszczany jest znak średnika. Jeżeli parametr ma postać N2, to po numerze wiersza bez pomijania zer wstawiany jest znak tabulacji. Tekst jest rozszerzany zgodnie ze znakiem tabulacji, gdy zostanie podany także parametr T

O - przesyła standardowy zbiór typu COM /nie w kodzie ASCII/, ignorując znak końca zbioru

Qsctl-Z - kopiowanie danych ze zbioru, aż do napotkania w przesyłanym łańcuchu znaków ctl-Z

Pn - włączenie stronicowania. Gdy n = 1 lub gdy n nie występuje, to wysuw realizowany jest do 60 wierszy

R - udostępnia zbiory o statusie zbiorów systemowych

Ssctl-Z - rozpoczyna kopiowanie zbioru od łańcucha znaków przedstawionego jako s, a zakończonego znakiem ctl-Z. Parametry S i Q mogą być użyte do rozdzielenia zbioru na elementy. Łańcuch znaków s jest włączony w operację kopiowania. Łańcuchy znaków występujące po parametrach S i Q są automatycznie tłumaczone na wielkie litery

TN - włącza funkcję tabulacji dla każdej n-tej kolumny przesyłanych danych

U - przekodowuje małe litery na wielkie w trakcie operacji transmisji danych

V - jeżeli zbiorem odbierającym jest zbiór dyskowy, to sprawdzana jest poprawność kopiowania przez powtórny odczyt zapisanych danych

W - ignoruje ochronę zapisu dla zbiorów o statusie R/O bez podawania informacji na konsolę

Z - pominięcie bitu parzystości dla każdego znaku kodu ASCII

Przykłady użycia:

PIP PGM.BAS=B:[V] - kopiuje zbiór PGM.BAS z jednostki dyskowej B: na dysk aktualnie przydzielony do systemu ze sprawdzaniem poprawności zapisu

PIP LPT:=PGM.ASM[NTBU] - drukuje zbiór PGM.ASM numerując każdy wiersz, z włączeniem tabulacji i tłumacząc małe litery na wielkie

PIP PROG.BAS=PRG.BAS - SGOSUB 1200 ctl-ZQCLOSE≠≠2xtl-Z - kopiuje zbiór PRG.BAS do zbioru o nazwie PROG.BAS rozpoczynając kopiowanie po napotkaniu łańcucha znaków "GOSUB 1200", a przerywając kopiowanie po przesłaniu łańcucha "CLOSE≠≠2".

W ramach standardowego działania programu PIP nie zapisuje danych w zbiorach ze statusem ochrony zapisu R/O. Jeżeli dokonana byłaby próba zapisu w zbiorze z ustawioną ochroną informacji, to program wyprowadzi następujący komunikat:

DESTINATION FILE IS R/O, DELETE Y/N ?

Jeżeli użytkownik odpowie na komunikat "y", to zbiór zostanie udostępniony do zapisu. Przy odpowiedzi negatywnej zapis do zbioru nie będzie wykonany, a na konsolę wyprowadzony zostanie komunikat:

** NOT DELETED **

Transmisja do zbioru chronionego jest pominięta, a program PIP kontynuuje działanie zgodnie z następnymi dyspozycjami.

Przykład użycia:

PIP A:=B:*.COM[W] - kopiuje wszystkie zbiory o statusie różnym od systemowego. Kopiowanie dokonywane jest z dysku B: na A:, przy czym w trakcie kopiowania niszczone będą zbiory chronione na dysku A:

5.4. Komenda DUMP

Komenda DUMP umożliwia wyprowadzenie zawartości zbioru dyskowego w postaci kodu heksadecymalnego. Zawartość zbioru wypro-

wadzana jest na konsolę systemową.
Postać komendy jest następująca:

DUMP nazwa zbioru

Informacje wyprowadzane są w układzie szesnastu bajtów w każdej linii, przy czym każda linia poprzedzana jest heksadecymalnym adresem bezwzględnym. Wyprowadzanie informacji może być przerwane poprzez wciśnięcie klawisza "rubout".

5.5. Komenda SYSGEN

Komenda SYSGEN dokonuje kopiowania systemu CP/M między sformatowanymi dyskami. Kopiowanie systemu odbywa się w trybie następujących poleceń konwersacyjnych:

SYSGEN	zainicjowanie działania programu
SYSGEN VERSION m.m	przedstawienie wersji programu SYSGEN
SOURCE DRIVE NAME /OR RETURN TO SKIP/	Wprowadza nazwę dysku źródłowego, zawierającego system lub też można pominąć, gdy system jest już w pamięci po działaniu programu MOVCPM
SOURCE ON d THEN TYPE RETURN	Podaje nazwę jednostki dyskowej z systemem
FUNCTION COMPLETE	System został skopiowany do pamięci
DESTINATION DRIVE NAME /OR RETURN TO REBOOT/	Podaje nazwę dyskietki, na którą będzie kopiowany
DESTINATION ON d THEN	Odpowiedź systemu, określająca nazwę jednostki, na którą będzie wykonywana kopia
FUNCTION COMPLETE	Zakończenie kopiowania na dysk wskazany w poprzednim komunikacie

Po zakończeniu kopiowania systemu nowy dysk zawiera skopiowaną wersję systemu, przy czym dostępne są wyłącznie komendy rezydentne. Dla osiągnięcia pełnej zgodności między dyskami kopiowanymi należy dokonać kopiowania zbiorów. Kopiowanie zbiorów można zrealizować w następujący sposób:

PIP B:=A:*.* [V]

Ostatnio podana komenda spowoduje przepisanie wszystkich zbiorów z dysku A: na dysk B:, przy czym w trakcie kopiowania będzie realizowana kontrola poprawności zapisu.
Należy zauważyć, że kopiowanie systemu nie powoduje niszczenia

zbiorów zapisanych na dysku, na który następuje kopiowanie systemu.

5.6. Komenda ED - edytor

Tworzenie i modyfikacja zbiorów źródłowych zapisanych w kodzie ASCII odbywa się programem ED.COM. Wywołanie programu jest następujące:

A > ED < nazwa zbioru > . < typ >

lub

A > ED d: < nazwa zbioru > d1:

gdzie:

nazwa zbioru - jawna nazwa zbioru

d: - nazwa jednostki dyskowej

d1: - nazwa jednostki dyskowej, na której tworzona będzie nowa generacja zbioru edytowanego.

Pierwsza postać komendy powoduje utworzenie nowej generacji zbioru edytowanego na tej samej jednostce dyskowej co zbiór pierwotny. Również zbiory robocze będą założone na tej samej jednostce.

Wywołanie drugiej postaci tworzy zbiór nowej generacji oraz zbiory robocze na jednostce dyskowej oznaczonej nazwą d1:. Po zakończeniu edycji zbioru, dyskiem przydzielonym aktualnie do systemu staje się dysk d1:. Edytor operuje na zbiorze źródłowym oznaczonym na rys. 7 jako zbiór X.y, przenosząc tekst ze zbioru do bufora pamięci, w którym to buforze tekst może być przeglądany i/lub zmieniany. Wielkość bufora dla edytora zależy od wielkości pamięci RAM i wynosi około 50 000 znaków dla 64K bajtów pamięci. Jeżeli zbiór określony w operandzie komendy już istnieje, to będzie jego zawartość wprowadzana do bufora pamięci, w której może być przetwarzana komendami edytora. Gdy zbiór o podanej nazwie nie istnieje, to zostanie on utworzony i gotowy będzie do przyjmowania danych. Zasada działania edytora przedstawiona została na rys. 7, a sprowadza się ona do następującej logiki pracy. Zbiór zawierający tekst do edycji należy wprowadzić przy pomocy komendy A do bufora pamięci. Po zakończeniu edycji dla części zbioru /np. gdy zbiór przekracza wielkość bufora/ należy przy pomocy komendy "W", która przepisuje zedytowaną część do zbioru tymczasowego pod nazwą "nazwa zb.$$$". Po

całkowitym zakończeniu edycji komendę "E" przepisujemy pozostałość zbioru źródłowego. W przypadku zakończenia edycji w jednym etapie używamy tylko komendy "E". Po wykonaniu komendy "E" zbiór /nazwa zb. $$$"/ ulega zmianie na końcowy. Poprzednio istniejący zbiór typu .BAK zostanie zniszczony, a na jego miejsce utworzony zostanie zbiór pierwotny. Po kilku edycjach zawsze więc istnieją dwie najbardziej aktualne wersje poprawianego zbioru.
Organizacja bufora pamięci przedstawiona została na rys. 8.
Program ED akceptuje wprowadzanie zarówno wielkich jak i małych liter kodu ASCII. Można jednak użyć komendy edytora /"u"/, która zdekoduje znaki w buforze pamięci na wielkie litery. Odwołanie się tego polecenia jest użycie komendy "-U". Inicjowany program ED startuje w trybie "U".
Na rys. 7 przedstawiono komendy, które pełnią następujące funkcje:

nA - kopiuje n wierszy ze zbioru źródłowego do bufora pamięci, wskaźnik wierszy w buforze ustawia się na początku bufora

nW - przepisuje n wierszy z bufora /od początku bufora/ pamięci do tymczasowego zbioru roboczego. Pozostałe wiersze w buforze przesuwa do początku bufora

E - kończy edycję, przepisując całą zawartość bufora pamięci do zbioru tymczasowego typu .$$$. Następnie zmienia nazwę na taką jaką posiadał zbiór źródłowy. Zbiór źródłowy otrzymuje rozszerzenie .BAK, a poprzedni zbiór typu .BAK zostaje skasowany

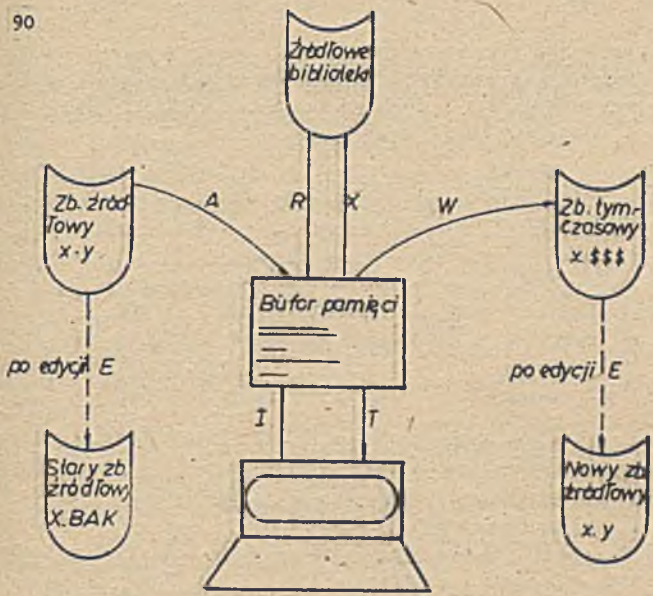
H - wykonuje czynności analogiczne jak poprzednia komenda oraz inicjuje ponownie program ED, przy czym aktywnym zbiorem źródłowym staje się nowo utworzony zbiór /tzn. wykonuje komendy: nE, ED nazwa zb. typ/

O - komenda ta realizuje powrót do sytuacji przed edycją. Bufor pamięci zostaje wymazany, tymczasowy zbiór skasowany, a wskaźnik wierszy ustawiony na 1

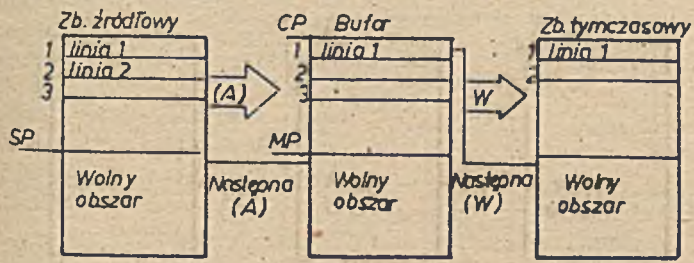
Q - realizuje operacje jak komenda nO z jednoczesnym powrotem do CP/M.

U w a g i :

n - oznacza liczbę z przedziału o-65535. Jeżeli liczbę przed komendą opuścimy, to przyjmowana jest wartość 1. Zastąpienie n przez znak #, oznacza n = 65535.
Użycie komendy "OA i OW" powoduje wprowadzenie lub wyprowadzenie co najmniej połowy pojemności bufora.



Rys.7 Zasada działania programu ED



Rys. 8 Organizacja bufora pamięci

Bufor pamięci wypełniany jest sekwencją wierszy wprowadzanych przy pomocy komendy A. Posiada on własny wskaźnik znakowy, zwany kursorem ⟨CP⟩. Wskaźnik ten zmienia swoje położenie zgodnie z poleceniami użytkownika.

```
pierwszy 1 _ _ _ _ _ _ _ _ _ _ _ ⟨CR ⟩⟨ Lf ⟩
wiersz   2 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ⟨cr ⟩⟨ Lf ⟩

                  CP

aktualny 3 _ _ _ _ _ _ _ _ _ _ _ ⟨cr ⟩⟨ Lf ⟩
wiersz ⟨CL⟩
         4 _ _ _ _ _ _ _ _ _ _ _ _ ⟨cr ⟩⟨ Lf ⟩

ostatni  5 _ _ _ _ _ ⟨cr ⟩⟨ Lf ⟩
wiersz
```

Kursor może znajdować się przed pierwszym znakiem pierwszego wiersza, za ostatnim znakiem ostatniego wiersza lub między dwoma dowolnymi znakami. Wierszem aktualnym ⟨CL⟩ nazywa się wiersz zawierający kursor ⟨CP⟩.

Program ED zgłasza użytkownikowi swoją gotowość do przyjmowania komend znakiem gwiazdki * .

Bufor pamięci po inicjacji jest pusty i oczekuje na wprowadzenie danych ze zbioru źródłowego /komenda A/ lub klawiatury /komenda I/. Przy wprowadzaniu wierszy z konsoli /klawiatury/, koniec każdego wiersza kończymy wciśnięciem klawisza ⟨RETURN⟩, co powoduje dopisanie na końcu linii znaku "cr" i "Lf".

Wciśnięcie jednoczesne klawiszy "CTL" i "Z" kończy tryb wprowadzania z konsoli i program znowu przechodzi w tryb komend wyświetlając znak *. /kursor znajduje się za znakiem ⟨Lf⟩ ostatniego wiersza/.

Po wprowadzeniu tekstu do bufora pamięci można modyfikować wprowadzony tekst lub wyprowadzić na ekran.

Komendy operujące na buforze pamięci mogą być poprzedzone znakiem "≠≠", "+" lub "-". Znak "+" jest opcjonalny, gdy nie jest podany, to przyjmuje się, że występuje przed komendą.

Następujące komendy sterują położeniem kursora /CP/:

± B ⟨cr⟩ - przesuwa CP do początku bufora, jeżeli + i do końca, jeżeli -

± nC ⟨cr⟩ - przesuwa CP o ± n znaków /w kierunku przodu CP, jeżeli "+"/ licząc znaki cr i lf jako dwa oddzielne znaki

± nD ⟨cr⟩ - kasuje n znaków /z przodu CP jeżeli + i z tyłu CP, jeżeli -

± nK ⟨cr⟩ - kasuje n wierszy źródłowego tekstu od punktu położenia kursora CP. Jeżeli kursor CP nie znajduje się na początku wiersza, to kasowana jest część wiersza z przodu CP /gdy +/ lub z tyłu CP /gdy -/.

± nL ⟨cr⟩ - przesuwa kursor CP o ±n wierszy. Gdy n = o, to CP przesuwany jest na początek aktualnego wiersza CL. Jeżeli n ≠ o, to kursor CP przesuwany jest najpierw na początek wiersza CL, a następnie przesuwany jest n wierszy w dół /gdy +/ lub w górę /gdy -/. W przypadku zbyt dużej wartości n, kursor CP zatrzymuje się na początku lub końcu bufora /w zależności od użytego znaku/.

± nT ⟨cr⟩ - listuje na ekranie n wierszy. Jeżeli n = ø, to wypisywany jest wiersz CL od pierwszego znaku do CP. Jeżeli n = 1, to wypisywany jest wiersz CL od CP do ostatniego znaku wiersza. Jeżeli n > 1, to wyprowadzany jest aktualny wiersz i n-1 wierszy następnych /gdy +/ i n wierszy poprzedzających CP /gdy -/

± n ⟨cr⟩ - wykonuje sekwencję komend ±nLT, która przesuwa CP i listuje wiersz CL.

Wszystkie komendy edytora ED mogą być podawane pojedyńczo lub grupowo, aż do wypełnienia 128-znakowego bufora konsoli. Przy grupowym wprowadzaniu komend, są one wykonywane kolejno aż do napotkania znaku cr, który kończy łańcuch. Operator może użyć komend funkcyjnych w trakcie pracy edytora /tzn. klawiszy funkcyjnych/.

DEL - kasuje ostatni znak

CTL-U - kasuje wiersz

CTL-C - ponownie inicjuje CP/M

CTL-E - specjalny powrót karetki przy długich łańcuchach /umożliwia pisanie długich łańcuchów znaków bez używania klawisza CR .

Edytor CP/M wyposażony jest w komendy pozwalające w buforze pamięci przeszukiwać tekst i go zmieniać.

Postać komend jest następująca:

$$n\ F\ C_1\ C_2 \ldots C_k \left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$$

gdzie $C_1 \ldots C_k$ jest ciągiem znaków w buforze. Komenda wyszukuje w buforze zadany ciąg znaków n razy, a po pozytywnym jego zakończeniu kursor CP przesuwany jest za znak C_k. Jeżeli próby poszukiwania zakończą się negatywnie, to CP nie zmienia swojego położenia. Łańcuch znaków może zawierać znak CTL-L, który jest reprezentowany przez parę znaków ⟨cr⟩⟨Lf⟩.

$$I\ C_1 \ldots C_k \langle cr \rangle$$

lub

$$I\ C_1 \ldots C_k \langle CTL\text{-}Z \rangle$$

Komenda umożliwia wstawienie zadanego łańcucha znaków w dowolne miejsce tekstu. Gdy komenda zakończona jest znakiem CTL-Z, to znaki łańcucha wstawiane są bezpośrednio po kursorze CP, a CP umieszcza się za ostatnim znakiem łańcucha. Gdy komenda kończy się wciśnięciem klawisza ⟨cr⟩, wtedy za ostatnim znakiem C_k dopisywane są znaki ⟨cr⟩⟨Lf⟩

$$n\ SC_1C_2 \ldots C_n \langle CTL\text{-}Z \rangle d_1\ d_2 \ldots d_m \left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$$

Komenda podstawia n razy w miejsce łańcucha znaków $C_1 \ldots C_n$ łańcuch znaków $d_1 \ldots d_m$. Łańcuchy nie muszą się pokrywać co do długości.

Taki sam efekt można uzyskać ciągiem komend

$$F\ C_1\ C_2 \ldots C_3, \langle CTL\text{-}Z \rangle - n\ D\ I\ d_1\ d_2 \ldots d_m \left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$$

$$n\ N\ C_1 \ldots C_k \left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$$

Komenda ta realizuje funkcje jak komenda F, z tym że jednocześnie z przeszukiwaniem następuje przepisanie zbioru źródłowego do zbioru tymczasowego. Przepisywanie zbioru realizowane jest aż do n wystąpień łańcucha znaków C_k lub do końca bufora

$$n\ C_1\ C_2 \ldots C_k \langle CTL\text{-}Z \rangle d_1\ d_2 \ldots d_m \langle CTL\text{-}Z \rangle e_1\ e_2 \ldots e_q$$

$$\left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$$

Komenda ta powoduje n-krotne wykonanie następujących czynności: od aktualnej pozycji CP poszukiwany jest łańcuch znaków C_k, po jego znalezieniu za znak C_k wstawiany jest łańcuch d_m. Następnie wszystkie znaki po d_m do początku łańcucha c_1 są kasowane. Kursor CP ustawia się pod d_m. Jeżeli łańcuch znaków e_q nie zostanie

znaleziony, wtedy kasowanie nie będzie wykonane.
Edytor dysponuje także komendami obsługi biblioteki programów źródłowych i wielokrotnego wykonywania zadanych sekwencji komend.

R $f_1 f_2 \ldots f_n \left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$ - komenda przepisuje do bufora pamięci zawartość zbiorów $f_1 \ldots f_n$ z typem . LIB. Przepisywanie rozpoczyna się po aktualnym położeniu CP.

np. RMACRO ⟨ CR ⟩

V ⟨cr⟩ - komenda powoduje włączenie numeracji bezwzględnej przed każdym wierszem. Numer ten nie jest zapisywany w zbiorze końcowym. Do nadanego w ten sposób wiersza można się odwołać, np.: nr wiersza: ⟨ cr ⟩

ØV ⟨cr⟩ - powoduje wyprowadzenie statystyki bufora w postaci w/c, gdzie:
W - oznacza liczbę dostępnych bajtów pamięci bufora
C - oznacza rozmiar bufora w bajtach

nX ⟨cr⟩ - komenda przenosi najbliższe n wierszy /od wiersza aktualnego/ do zbioru tymczasowego o nazwie X$$$$$$$$. LIB, który jest tworzony tylko w czasie edycji. Przeniesione wiersze mogą być pobrane komendą R.

ØX ⟨cr⟩ - komenda służy do kasowania zbioru X$$$$$$$$ .LIB Zbiór ten jest kasowany także po zakończeniu edycji komendą E lub Q.

n M $C_1 C_2 \ldots C_n \left\{ \begin{matrix} cr \\ CTL\text{-}Z \end{matrix} \right\}$ - gdy, n > 1 komenda wykonuje n razy wskazany ciąg. Gdy n = Ø lub 1 łańcuch komend wykonywany jest aż do wystąpienia błędu lub końca bufora. Błędy sygnalizowane przez ED.

W sytuacjach błędnych program wyprowadza komunikat:

BREAK X AT C

gdzie
X jest kodem błędu
? - nierozpoznana komenda
- przepełniony bufor pamięci /użyć jedną z komend D, K, N, S lub W do zmniejszenia zajętości bufora/

≠≠ - komenda nie może być wykonana wskazaną liczbą powtórzeń
0 - zbiór o rozszerzeniu .LIB nie może być udostępniony komendzie R

Podczas edycji wystąpienie błędu dyskowego sygnalizowane jest komunikatem

BDOS ERR on d: BAD SECTOR

Sprawdzany jest również status zbioru /tryb dostępności/ dla zbiorów tylko do czytania lub systemowych

FILE IS READ/ONLY
'SYSTEM' FILE NOT ACCESSIBLE

Po wystąpieniu takich komunikatów, jeżeli to konieczne, to można zmienić status zbioru komendą STAT.

Przykłady użycia komend ED.

1. B2T <cr> przesuń CP do początku bufora i drukuj dwie linie
2. 5CØT <cr> przesuń CP o 5 znaków i drukuj od początku linii
3. 2L-T <cr> przesuń o dwie linie w dół i drukuj poprzednią linię
4. -L≠K <cr> przesuń w górę o jedną linię i skasuj 65535 linii następnych
5. I <cr> wprowadzono 2 linie tekstu i przejście do komend ED WPROWADZAMY <cr>
   TERAZ TEKST <cr>
   <CTL-Z>
6. <cr> przesuń w dół CP o jedną linię i wydrukuj ten tekst
7. B≠≠T <cr> przesuń CP do początku bufora i wydrukuj cały bufor
8. FRAZ <cr> znajdź koniec tekstu "RAZ"
9. BINOWY <CTL-Z> <cr> wstaw "nowy" od początku bufora
10. FTERAZ <CTL-Z> - 5DINOW <CTL-Z> <CR> - znajdź ciąg "TERAZ" i skasuj go w to miejsce wstaw ciąg "NOW"
11. SGAMA <CTL-Z> DELTA <CR> - wymień wyraz GAMA na DELTA
12. MSGAMA <CTL-Z> DELTA <CR> - wymień wszystkie wyrazy GAMA na DELTA

Komenda	Funkcja
nA	dołącz linie do bufora
± B	początek kursora w buforze
± nC	przesuń kursor o n znaków

± nD	skasuj znaki
E	zakończ edycję i zamknij zbiory /normalne zakończenie/
nF	odszukaj ciąg znaków
H	zakończ edycję i otwórz ponownie zbiory do edycji
I	wprowadź tekst
nJ	umieść łańcuchy znaków obok siebie
± nK	skasuj linie
± nL	przesuń w dół/górę linie
nM	wykonaj ciąg funkcji wielokrotnie
nN	znajdź następne wystąpienie z automatycznym przepisywaniem
O	powrót do poprzedniego stanu zbioru
± nP	przesuń i drukuj strony
Q	zakończyć edycję bez zmiany zbiorów
R	czytaj zbiory biblioteczne
nS	wymień ciągi znaków n razy
± nT	drukuj linie
± U	tłumacz małe litery na duże, jeżeli "U", nie tłumacz, jeżeli "-u"
nW	poślij n linii do zbioru typu .$$$
nZ	zawiesić działanie programu na n sekund
± n <cr>	przesuń i drukuj /± nLT/
nnnn: <cr>	ustaw CP przed wskazaną linią
:nnnnT <cr>	drukuj do wskazanej linii

```
B>ED COMP.BAS

NEW FILE
*VI
    1:  10 'PROGRAM PRROWNUJE DWIE LICZBY
    2:  20 REM MNIEJSZA DRUKUJE NA EKRANIE
    3:  30 INPUT "WPROWADZ DWIE LICZBY";X,Y
    4:  40 IF X <= Y
    5:          THEN
    6:            IF X < Y
    7:            THEN
    8:               PRINT X;
    9:            ELSE
   10:               PRINT "NIE MA"
   11:          ELSE
   12:           PRINT Y;
   13:  50 PRINT "JEST MNIESZA"
   14:  60 GOTO 20
   15:  40 END
   16:
    : *B3T
    1:  10 'PROGRAM PRROWNUJE DWIE LICZBY
    2:  20 REM MNIEJSZA DRUKUJE NA EKRANIE
    3:  30 INPUT "WPROWADZ DWIE LICZBY";X,Y
    1: *3L
    4: *14:T
   14:  60 GOTO 20
   14: *S20^Z30
   14: *14:T
   14:  60 GOTO 30
   14: *L
   15: *T
   15:  40 END
   15: *S4^Z7
   15: *0T
   15:  7*T
0 END
   15: *15:T
   15:  70 END
   15: *-5LT
   10:                PRINT "NIE MA"
   10: *E

B> Przyklad sesji edytora ED.COM^C

B>Oznaczenie w liscie: ^Z=control=z
```

ED

```
B>ED COMP.BAS
*$A/
    1: *15T
    1:  10 'PROGRAM PRROWNUJE DWIE LICZBY
    2:  20 REM MNIEJSZA DRUKUJE NA EKRANIE
    3:  30 INPUT "WPROWADZ DWIE LICZBY";X,Y
    4:  40 IF X <= Y
    5:          THEN
    6:            IF X < Y
    7:            THEN
    8:              PRINT X;
    9:            ELSE
   10:              PRINT "NIE MA"
   11:          ELSE
   12:           PRINT Y;
   13:  50 PRINT "JEST MNIESZA"
   14:  60 GOTO 30
   15:  70 END
    1: *LT
    2:  20 REM MNIEJSZA DRUKUJE NA EKRANIE
    2: *FNA¬ZI MONITORZE¬Z
    2: *2:T
    2:  20 REM MNIEJSZA DRUKUJE NA MONITORZE EKRANIE
    2: *FORZE¬ZBD
    2: *2:T
    2:  20 REM MNIEJSZA DRUKUJE NA MONITORZE
    2: *B
    1: *E
Oznaczenia:
B>  ¬Z=CONTROL-Z
```

B> Przyklad sesji edytora dla istniejacego zbioru.

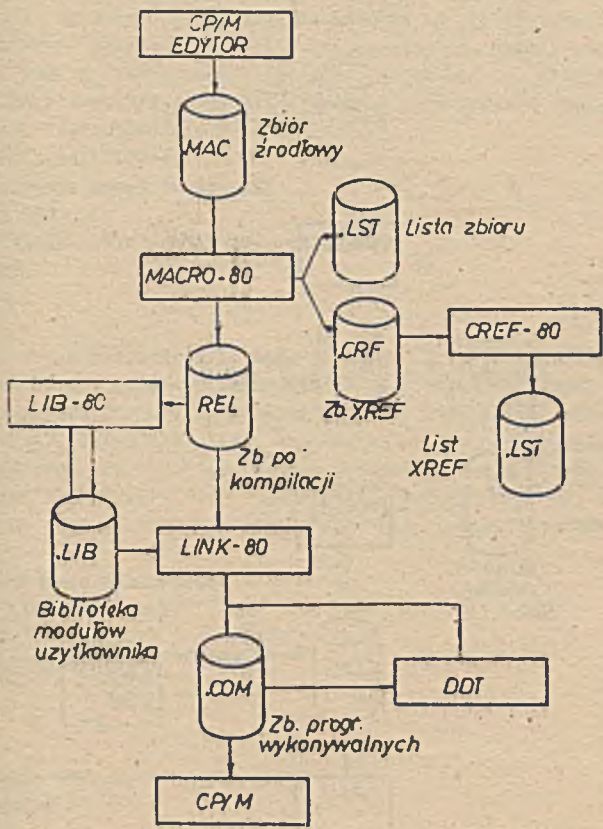
5.7. Inne komendy nierezydentne

Obejmują między innymi pakiet programów, składający się z następujących programów:

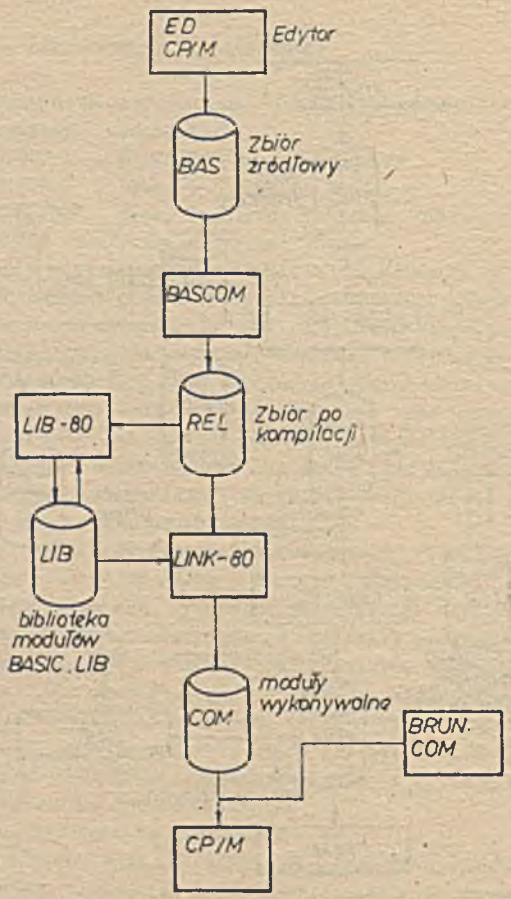
M80.COM - MACRO-80 Macro assembler
L80.COM - LINK.80 Linking Loader
CREF80.COM - Cross-Reference Facility -CREF
LIB.COM - Library Manager

Powiązania między programami pakietu przedstawia rys. 9.
MACRO-80 posiada niemal wszystkie własności assemblerów dużych systemów komputerowych. Pozwala on na realizację pełnego zestawu makroinstrukcji, zgodnego ze standardem INTEL'a. Liczba dołączonych makroinstrukcji jest limitowana jedynie wielkością dostępnej pamięci RAM. Program źródłowy może być napisany w języku assembler 8080 lub Z80, tłumaczony jest na moduł /moduły/ typu REL z kodem wynikowym relokowalnym lub jako moduł absolutny /stałe adresy w pamięci/. Moduły te można łączyć i ładować do pamięci przy pomocy programu LINIK-80.
LINK-80 jest programem łącząco-ładującym. Umożliwia łączenie i ładowanie pod zadany adres w pamięci RAM dowolnej liczby modułów /programów/ typu REL, wykonując jednocześnie realizację odwołań zewnętrznych, które występują w modułach programowych poprzez przeszukiwanie ich w zbiorze bibliotecznym. Wynikiem pracy programu jest utworzenie zbioru typu COM, który może być zapisany na dysku lub wykonany bezpośrednio jako program użytkowy.
CREF80.COM jest programem, który z pliku typu CRF /utworzonego przez program MACRO-80/ generuje listing z listą odwołań w programie wynikowym /po kompilacji/.
LIB-80 zarządzanie bibliotekami.
Program LIB.COM może być użyty do tworzenia własnych bibliotek podprogramów napisanych w językach Assemblera, Basic'a, Fortranu, Pascala i innych. Podprogramy te mogą być specjalizowanymi modułami tworzonymi przez programistę lub też modułami z istniejącej biblioteki np. BASLIB.LIB rys. 10.
Zawarte w bibliotece moduły mogą być dołączane do programu głównego za pomocą programu LINK-80 z parametrem /S, który musi wystąpić za nazwą biblioteki, np.

```
A > L80 PROG, BASLIB/S, PROG1/N/E
```



Rys. 9 Powiązania pomiędzy programami pakietu



Rys.10 Tworzenie biblioteki modułów BASIC

Zbiór PROG.REL jest programem głównym /po kompilacji/, w którym występują odwołania zewnętrzne do modułów w bibliotece BASLIB. LIB, LINK-80 przeszukuje bibliotekę, dołączając moduły do programu PROG i tworzy moduł /program ładowalny/ o nazwie PROG1. COM, który zostanie zapisany na dysku A jako program użytkowy. Parametr "/E" spowoduje wyjście do CP/M.

Przykład sesji LIB-80.

1/ Tworzenie nowej biblioteki

```
A > LIB
*TRANLIB = SIN, COS, TAN, ATN, ACOG
*EXP
*/E
A >
```

W tym przykładzie, LIB wywołuje program LIB-80, który zgłasza się gwiazdką /*/. TRANLIB jest nazwą tworzonej biblioteki. SIN, COS ... są nazwami zbiorów, które będą włączone do TRANLIB. Moduł EXP jest inną nazwą zbioru, który będzie też dołączony do TRANLIB. Użycie parametru /E dla LIB-80 zmieni nazwę TRANLIB.LIB na TRANLIB.REL przed wyjściem do systemu CP/M. /Użycie CTRL-C zamiast "/E" nie zmieni typu biblioteki/.

2/ Listowanie biblioteki

```
A > LIB   < cr >
*TRANLIB.LIB/U
*TRANLIB.LIB/L
      .
      .
      .
/lista symboli w TRANLIB.LIB/
*CTRL-C
A >
```

UWAGA: Użycie klawisza CTRL-C w komendzie LIB powoduje wyjście do CP/M bez zmiany jakichkolwiek zbiorów.

Komendy LIB-80

Wywołanie LIB-80 następuje po wprowadzeniu następującej komendy

A > LIB < cr >

Format komendy LIB-80 jest następujący

przeznaczenie = źródło/parametr

Wszystkie pola komendy są opcjonalne, pominięcie ich powoduje przyjęcie wartości domyślnych.
Pole "przeznaczenie" ma następującą postać:

< nazwa zbioru > =

Jeżeli pole to jest pominięte, to domyślnie przyjmowana jest nazwa FORLIB. z typem zbioru REL.
Pole "źródło" może mieć następującą postać:

1/ nazwa-zbioru <moduł1, moduł2, ... >

2/ nazwa-zbioru, nazwa zbioru 2, ...

LIB-80 tworzy bibliotekę użytkownika tylko ze zbiorów typu .REL i modułów biblioteki typu .REL.
Pole "/parametr" ma następującą postać:

/E - powoduje wyjście do CP/M ze zmianą typu biblioteki na .REL. Jeżeli tworzona jest nowa biblioteka lub nie zmieniamy jej typu, to należy użyć klawisza CNTL-C zamiast parametru /E.

/R - zmienia nazwę biblioteki typu .LIB na typ .REL.
Użycie parametru /R wywołuje te same funkcje co parametr E/, z tą różnicą, że program zgłasza się do wykonania następnej funkcji i nie wychodzi do CP/M.

/L - listuje nazwy w bibliotece

/U - listuje symbole niezdefiniowane

/C - usuwa komendy LIB-80, następuje powrót do stanu przed sesją

/O - list w postaci aktualnej

/H - list w postaci hex.

OSTRZEŻENIE !!!

Zawsze wykonaj kopię swojej dyskietki przed użyciem programu LIB-80.

6. OPIS JĘZYKA ASEMBLER INTEL 8080

6.1. Wprowadzenie

Stosowanie instrukcji bezpośrednio odpowiadających rozkazom procesora, wymaga od programisty znajomości struktury funkcjonalnej mikroprocesora.
Rys. 2 przedstawia taki schemat funkcjonalny mikroprocesora Intel 8080. Poniżej opisano bardziej szczegółowo funkcje modułów mikroprocesora, i tak:
Pamięć - stosowane są dwa rodzaje pamięci operacyjnej: pamięć tylko do czytania ROM /ang. Read only memory/ i pamięć do pisania i czytania RAM /ang. Random access memory/. Każde osiem bitów pamięci stanowi jeden bajt, który ma swój adres. Licznik programowy /PC/ stanowi 16 bitowy rejestr umożliwiający adresowanie pamięci w zakresie od zera do 65535 /64 K bajtów/. W mikroprocesorze Intel 8080 adres pamięci można określić też za pomocą rejestrów roboczych. Rejestry robocze - zestaw rejestrów roboczych obejmuje akumulator oznaczony symbolem A, oraz 6 rejestrów roboczych oznaczonych literami B, C, D, E, H i L.
Rejestr stanu /PSW/ - określa /odzwierciedla/ wyniki operacji wykonywanych w mikrokomputerze. Przyporządkowanie poszczególnych bitów w PSW przedstawia się następująco:

b7	b6	b5	b4	b3	b2	b1	b0	nr bitu
S	Z	0	0	A_x	P	1	C	
bit znaku	bit zera	nie wyk.		bit przeniesienia połówkowego	bit parzystości		bit carry /przeniesienia/	

- bit znaku S jest ustawiany /ma wartość 1/ gdy po operacji bit b7 wyniku równa się 1
- bit zera Z jest ustawiany gdy wynik operacji równa się zero
- bit przeniesienia połówkowego A_x jest ustawiany gdy w wyniku operacji nastąpiło przeniesienie z pozycji b3 na pozycję b4 wyniku
- bit parzystości P - jest ustawiany, jeżeli liczba jedynek

w wyniku jest parzysta

- bit przeniesienia C /carry/ jest ustawiany gdy nastąpiło przepełnienie najbardziej znaczącej pozycji wyniku /nadmiar/.

Stos i licznik stosu /SP/ - szesnastobitowy rejestr SP, w którym pamiętany jest adres ostatniej pozycji na stosie. Programista jest odpowiedzialny za zainicjowanie początku stosu.

Zbiór instrukcji mikroprocesora 8080 można podzielić wg ich metod adresowania wewnętrznego i/lub pamięci. Adresowanie rejestrowe - odbywa się poprzez adresowanie zawartością rejestrów lub pary rejestrów roboczych, np. instrukcja "CMP E" jest interpretowana jako porównanie zawartości rejestru E z zawartością akumulatora, natomiast instrukcja PCHL działa na parze rejestrów /adresowanie 16 bitowe/ wymieniając zawartość pary rejestrów HL z licznikiem programowym PC.

Adresowanie natychmiastowe - instrukcje te zawierają bezpośrednio za rozkazem daną, która jest pobierana przez procesor w momencie wykonywania się instrukcji. Dana może być stałą jednobajtową lub dwubajtową, wynika to z bezpośrednio z samej instrukcji.

Adresowanie bezpośrednie - wykonują instrukcje, które zajmują w pamięci 3 bajty: jeden dla kodu instrukcji i dwa dla 16-bitowego adresu. Adres może być stałą lub zmienną.

Adresowanie łączone - w którym instrukcje używają np. dwóch typów adresowania np. instrukcja "CALL adres" wykonuje skok pod wskazany adres wysyłając jednocześnie aktualną wartość licznika programowego na stos.

Adresowanie pośrednie pamięci wykonywane jest via parą rejestrów np. MOV M, C, para rejestrów HL zawiera adres pamięci.

- prześlij dane pomiędzy rejestrem lub pamięcią i rejestrem

  MOV prześlij

  MVI prześlij natychmiastowo

  LDI ładuj akumulator bezpośrednio z pamięci

  STA zapamiętaj akumulator bezpośrednio w pamięci

  LHLD ładuj H i L bezpośrednio z pamięci

  SHLD zapamiętaj H i L bezpośrednio w pamięci

gdy w nazwie instrukcji występuje "X", to oznacza, że mamy do czynienia z parą rejestrów

  LXI ładuj parę rejestrów daną bezpośrednią

LDAX ładuj akumulator spod adresu w rejestrach
STAX pamiętaj akumulator pod adresem w rejestrach
XCHG wymień zawartości rej. HL z DE
XTHL wymień zawartość licznika SP z HL

- arytmetyczne instrukcje obejmują dodawanie, odejmowanie, dodanie jeden lub zmniejszenie o 1 zawartości rejestrów lub pamięci

ADD dodaj do akumulatora
ADI dodaj bezpośrednio do akumulatora
ADC dodaj do akumulatora z carry
ACI dodaj bezpośrednio do akumulatora z carry
SUB odejmij od akumulatora
SUI odejmij bezpośrednio od akumulatora
SBB odejmij od akumulatora z pożyczką /carry/
SBI odejmij od akumulatora bezpośrednio z pożyczką
INR zwiększa wartość specyfikowanego bajtu o 1
DCR zmniejsza wartość specyfikowanego bajtu o 1
INX zwiększa zawartość pary rejestrów o 1
DCX zmniejsza zawartość pary rejestrów o 1
DAD dodaj pary rejestrów: zawartość pary rejestrów dodaj do zawartości pary rej. H i L

- logiczne instrukcje wykonują operacje /Boole'a/ na danych w rejestrach i pamięci ustawiając jednocześnie kody warunków w PSW

ANA logiczne AND z akumulatorem
ANI logiczne AND z akumulatorem i bajtem danych
ORA logiczne OR z akumulatorem
ORI logiczne OR z akumulatorem i bajtem danych
XRA różnica symetryczna OR z akumulatorem
XRI różnica symetryczna OR z bajtem danych

- instrukcje porównujące zawartość 8-bitowej wartości z zawartością akumulatora:

CMP porównaj
CPI porównaj z daną bezpośrednią

- instrukcje przesuwające zawartości akumulatora o jeden bit w lewo lub prawo:

RLC przesunięcie akumulatora w lewo
RRC przesunięcie akumulatora w prawo
RAL przesunięcie w lewo poprzez carry

RAR przesunięcie w prawo poprzez carry

- -

CMA negacja akumulatora
CMC negacja bitu carry
STC wstaw bit carry

- instrukcje skoku bezwarunkowe

JMP skok
CALL wywołaj
RET powrót

badając jeden z czterech kodów warunków w PSW można wykorzystać je w instrukcjach skoku

NZ nie zero /Z = 0/
Z zero /Z = 1/
NC nie carry /C = 0/
C carry /C = 1/
PO nieparzystość /P = 0/
PE parzystość /P = 1/
P plus /S = 0/
M minus /S = 1/

- instrukcja skoku zależna od kodów warunku w PSW, przedstawia tablica 11.

Tablica 11

Skok	wywołanie podprogramu	powrót z podprogramu	Warunek w PSW
JC	CC	RC	carry
JNC	CNC	RNC	nie carry
JZ	CZ	RZ	zero
JNZ	CNZ	RNZ	nie zero
JP	CP	RP	plus
JM	CM	RM	minus
JPE	CPE	RPE	parzystość
JPO	CPO	RPO	nieparzystość

- instrukcje skoku zmieniające zawartość licznik programu /PC/:

PCHL prześlij zawartość H i L do PC
RST specjalny restart dla przerwań

- stos, licznik stosu:

PUSH załaduj dwa bajty danych z pary rejestrów na stos
POP pobierz dwa bajty ze stosu
XTHL wymień zawartość wierzchołka stosu z parą rejestrów H i L
SPHL prześlij zawartość H i L do licznika stosu SP

- instrukcje wejścia/wyjścia
  IN pobierz daną do akumulatora
  OUT wyślij daną z akumulatora
- instrukcje maszynowe:
  EI odblokuj przerwania systemu
  DI zablokuj przerwania systemu
  HLT stop, czekaj na przerwanie
  NOP nic nie rób.

6.2. Język asemblera 8080

Instrukcje i dyrektywy asemblera 8080 mogą zawierać maksymalnie cztery następujące pola:

{ etykieta: / nazwa } kod operacji Operand ; komentarz

Pola mogą być separowane dowolną ilością spacji, lecz instrukcja lub dyrektywa muszą zajmować jedną linię zakończoną znakiem <Cr> i <Lf>.
Tworząc program za pomocą edytora /ED.COM/ znaki <Cr> i <Lf> dopisywane są na końcu lini przez edytor po naciśnięciu klawisza <RETURN>.
Zbiór znaków języka jest następujący:

- litery alfabetu A-Z /małe i duże/
- cyfry od 0 do 9
- znaki specjalne: +, -, *, /, (,), ', &, :, $, @, ?, =, <, > %, !, spacja, ;, ., CR, FF, HT

Etykieta /nazwa jest opcjonalna, lecz użyta może występować /ta sama/ tylko raz w programie.
W etykiecie mogą występować znaki dolara, które nie są wliczane do długości /max. długość 16 znaków/.
Kod operacji zawierać może:

- dyrektywę asemblera
- pseudoinstrukcję

- kod rozkazu maszynowego procesora

Operand zawiera argument lub argumenty operacji i stanowi zazwyczaj wyrażenia utworzone ze stałych i nazw z operacjami arytmetycznymi bądź logicznymi. Zasady tworzenia wyrażeń omówione będą w dalszej części.

Komentarz: zawiera dowolne znaki występujące po średniku /;/ do końca wiersza. Dodatkowo jako komentarz traktowane są linie programu rozpoczynające się znakiem gwiazdki /*/.

etykieta	kod operacji	Operand	Komentarz
ABC	MVI	C, *	; prześlij do C znak *

Następujące symbole w polu operandów są zastrzeżone:

- $ adres /aktualny/ odwołania w programie
- A akumulator
- B rejestr B lub para rejestrów BC
- C rejestr C
- D rejestr D lub para rejestrów DE
- E rejestr E
- H rejestr H lub para rejestrów HL
- L rejestr L
- SP licznik stosu
- PSW rejestr stanu /zawiera A i bajt stanu/
- M odwołanie do pamięci poprzez rejestry H i L

Stałe traktowane są jako wartości numeryczne 16 bitowe oznaczane są literą bezpośrednio zawartością stałej;

- B - stała binarna
- O,Q - stała oktalna
- D - stała dziesiętna
- H - stała heksadecymalna

Dopuszcza się używania w stałych znaku dolara $ na zasadach jak w nazwach.

Łańcuchy znaków stanowią sekwencje znaków ujęte w znak apostrofu i nie mogą przekraczać 64 znaków.

Operatory arytmetyczne:

- + dodawanie
- - odejmowanie
- * mnożenie
- / dzielenie całkowite
- MOD modulo /7 MOD 3 = 1/

Operatory przesunięcia:

Y SHR X przesuń operand 'Y' w prawo 'X' bitów
Y SHL X przesuń operand 'Y' w lewo o 'X' bitów

Operatory logiczne:

NOT logiczna negacja
AND logiczne mnożenie
OR logiczne dodawanie
XOR różnica symetryczna

Operatory porównania:

EQ równe
NE nie równe
LT mniejsze niż
LE mniejsze lub równe
GT większe niż
GE większe lub równe

Dyrektywy asemblera zapisywane są w polu operacji i nie mają maszynowych kodów operacji. Służą tylko kompilatorowi do sterowania procesem tłumaczenia programu źródłowego. Znaczenie ich jest następujące

ORG - oznacza początek programu i/lub danych
END - oznacza koniec programu
EQU - wiąże etykietę ze stałą
SET - przydziela etykiecie nową wartość
DB - definiuje bajty danych
DW - definiuje kolejne dwa bajty danych.
DS - definiuje obszar pamięci w bajtach.

Dyrektywa ORG ma następującą postać:

< etykieta > ORG < wyrażenie >

Wartość wyrażenia musi być znana /wartość 16-bitowa/ przed instrukcją ORG.

W programie może być użyta kilka dyrektyw ORG. Kompilator przyjmuje wartość wyrażenia za początek licznika adresów w programie w momencie napotkania dyrektywy ORG. Użycie dyrektywy ORG bez operandu oznacza rozpoczęcie adresacji programu od zera. Znak dolara użyty w operandzie dyrektywy ORG oznacza aktualną wartość licznika adresów. Zawartość licznika adresów jest podawana względem początku programu, gdyż podczas translacji nie są znane absolutne adresy miejsc pamięci.

Dyrektywa END jest opcjonalna w programie, lecz gdy występuje, musi być ostatnią instrukcją programu. Format dyrektywy:

⟨etykieta⟩ END

⟨etykieta⟩ END ⟨stała⟩ lub ⟨wyrażenie⟩

Wartość wyrażenia lub stałej jest punktem startu programu, dyrektywa bez operandu oznacza start od zera. Dyrektywa EQU jest używana dla zdefiniowania jednego symbolu wartością innego symbolu

np. BDOS EQU 5H

Dyrektywa SET ma znaczenie podobne jak EQU z tym, że SET definiuje wartość etykiety do chwili pojawienia się nowej SET /dla tej samej etykiety/.

Dyrektywa IF i ENDIF określają zakres instrukcji programu źródłowego, które będą włączone lub wyłączone przez kompilator do modułu wynikowego. Przykład użycia dyrektyw, ilustruje poniższy przykład:

```
0040 =          MSIZE   EQU 64    ;DEKLARACJE
B000 =          BIAS    EQU (MSIZE-20) * 1024
E400 =          CCP     EQU (MSIZE-20)*1024 + 3400H
EC00 =          BDOS    EQU CCP + 800H
FA00 =          BIOS    EQU BDOS+0E00H
0200                    ORG 200H
0200            P1      DS  3          ;MIESCE W PAO 3 BAJTY
0203 0000       P2      DW  0          ;DWA BAJTY ZER
0205 01020304   P3      DB  1,2,3,4    ;WART. 1 2 3 4
```

Instrukcje 8080.

Lista instrukcji /por. tablice 12 i 13/ przedstawiona będzie w kolejności alfabetycznej, umożliwi to szybkie znalezienie jej opisu.

ACI dodaj bezpośrednio z carry

ACI dodaje zawartość drugiego bajtu instrukcji wraz z bitem "carry" do zawartości akumulatora

operacja	operand
ACI	dana
flagi PSW:	Z, S, P, Ax, C

ADC dodaj z carry

dodaje jeden bajt danych łącznie z przeniesieniem "carry" do zawartości akumulatora i wynik pamięta w akumulatorze

operacja	operand
ADC	rejestr
flagi PSW:	Z, S, P, C, Ax

ADD dodaj

00 NOP	2B DCX H	56 MOV D,M	81 ADD C	AC XRA H	D7 RST 2
01 LXI B,D16	2C INR L	57 MOV D,A	82 ADD D	AD XRA L	D8 RC
02 STAX B	2D DCR L	58 MOV E,B	83 ADD E	AE XRA M	D9 ...
03 INX B	2E MVI L,D8	59 MOV E,C	84 ADD H	AF XRA A	DA JC Adr
04 INR B	2F CMA	5A MOV E,D	85 ADD L	B0 ORA B	DB IN D8
05 DCR B	30 ...	5B MOV E,E	86 ADD M	B1 ORA C	DC CC Adr
06 MVI B,D8	31 LXI SP,D16	5C MOV E,H	87 ADD A	B2 ORA D	DD ...
07 RLC	32 STA Adr	5D MOV E,L	88 ADC B	B3 ORA E	DE SBI D8
08 ...	33 INX SP	5E MOV E,M	89 ADC C	B4 ORA H	DF RST 3
09 DAD B	34 INR M	5F MOV E,A	8A ADC D	B5 ORA L	E0 RPO
0A LDAX B	35 DCR M	60 MOV H,B	8B ADC E	B6 ORA M	E1 POP H
0B DCX B	36 MVI M,D8	61 MOV H,C	8C ADC H	B7 ORA A	E2 JPO Adr
0C INR C	37 STC	62 MOV H,D	8D ADC L	B8 CMP B	E3 XTHL
0D DCR C	38 ...	63 MOV H,E	8E ADC M	B9 CMP C	E4 CPO Adr
0E MVI C,D8	39 DAD SP	64 MOV H,H	8F ADC A	BA CMP D	E5 PUSH H
0F RRC	3A LDA Adr	65 MOV H,L	90 SUB B	BB CMP E	E6 ANI D8
10 ...	3B DCX SP	66 MOV H,M	91 SUB C	BC CMP H	E7 RST 4
11 LXI D,D16	3C INR A	67 MOV H,A	92 SUB D	BD CMP L	E8 RPE
12 STAX D	3D DCR A	68 MOV L,B	93 SUB E	BE CMP M	E9 PCHL
13 INX D	3E MVI A,D8	69 MOV L,C	94 SUB H	BF CMP A	EA JPE Adr
14 INR D	3F CMC	6A MOV L,D	95 SUB L	C0 RNZ	EB XCHG
15 DCR D	40 MOV B,B	6B MOV L,E	96 SUB M	C1 POP B	EC CPE Adr
16 MVI D,D8	41 MOV B,C	6C MOV L,H	97 SUB A	C2 JNZ Adr	ED ...
17 RAL	42 MOV B,D	6D MOV L,L	98 SBB B	C3 JNP Adr	EE XRI D8
18 ...	43 MOV B,E	6E MOV L,M	99 SBB C	C4 CNZ Adr	EF RST 5
19 DAD D	44 MOV B,H	6F MOV L,A	9A SBB D	C5 PUSH B	F0 RP
1A LDAX D	45 MOV B,L	70 MOV M,B	9B SBB E	C6 ADI D8	F1 POP PSW
1B DCX D	46 MOV B,M	71 MOV M,C	9C SBB H	C7 RST 0	F2 JP Adr
1C INR E	47 MOV B,A	72 MOV M,D	9D SBB L	C8 RZ	F3 DI
1D DCR E	48 MOV C,B	73 MOV M,E	9E SBB M	C9 RET Adr	F4 CP Adr
1E MVI E,D8	49 MOV C,C	74 MOV M,H	9F SBB A	CA JZ	F5 PUSH PSW
1F RAR	4A MOV C,D	75 MOV M,L	A0 ANA B	CB ...	F6 ORI D8
20 ...	4B MOV C,E	76 HLT	A1 ANA C	CC CZ Adr	F7 RST 6
21 LXI H,D16	4C MOV C,H	77 MOV M,A	A2 ANA D	CD CALL Adr	F8 RM
22 SHLD Adr	4D MOV C,L	78 MOV A,B	A3 ANA E	CE ACI D8	F9 SPHL
23 INX H	4E MOV C,M	79 MOV A,C	A4 ANA H	CF RST 1	FA JM Adr
24 INR H	4F MOV C,A	7A MOV A,D	A5 ANA L	D0 RNC	FB EI
25 DCR H	50 MOV D,B	7B MOV A,E	A6 ANA M	D1 POP D	FC CM Adr
26 MVI H,D8	51 MOV D,C	7C MOV A,H	A7 ANA A	D2 JNC Adr	FD ...
27 DAA	52 MOV D,D	7D MOV A,L	A8 XRA B	D3 OUT D8	FE CPI D8
28 ...	53 MOV D,E	7E MOV A,M	A9 XRA C	D4 CNC Adr	FF RST 7
29 DAD H	54 MOV D,H	7F MOV A,A	AA XRA D	D5 PUSH D	
2A LHLD Adr	55 MOV D,L	80 ADD B	AB XRA E	D6 SUI D8	

D8 – stała lub wyrażenie 8-bitowe
D16 – stała lub wyrażenie 16-bitowe
Adr – 16 bitowy adres

JUMP

C3	JMP	
C2	JNZ	
CA	JZ	
D2	JNC	JMP
DA	JC	
E2	JPO	
EA	JPE	
F2	JP	
FA	JM	
E9	PCHL	

CALL

CD	CALL	
C4	CNZ	
CC	CZ	
D4	CNC	CALL
DC	CC	
E4	CPO	
EC	CPE	
F4	CP	
FC	CM	

RETURN

C9	RET
C0	RNZ
C8	RZ
D0	RNC
D8	RC
E0	RPO
E8	RPE
F0	RP
F8	RM

RESTART

C7	RST 0
CF	RST 1
D7	RST 2
DF	RST 3
E7	RST 4
EF	RST 5
F7	RST 6
FF	RST 7

ROTATE

07	RLC
0F	RRC
17	RAL
1F	RAR

CONTROL

00	NOP
76	HLT
F3	DI
FB	EI

MOVE (cont.)

58	MOV	E,B	
59	MOV	E,C	
5A	MOV	E,D	
5B	MOV	E,E	
5C	MOV	E,H	
5D	MOV	E,L	
5E	MOV	E,M	
5F	MOV	E,A	
60	MOV	H,B	
61	MOV	H,C	
62	MOV	H,D	
63	MOV	H,E	
64	MOV	H,H	
65	MOV	H,L	
66	MOV	H,M	
67	MOV	H,A	
68	MOV	L,B	
69	MOV	L,C	
6A	MOV	L,D	
6B	MOV	L,E	
6C	MOV	L,H	
6D	MOV	L,L	
6E	MOV	L,M	
6F	MOV	L,A	
70	MOV	M,B	
71	MOV	M,C	
72	MOV	M,D	
73	MOV	M,E	
74	MOV	M,H	
75	MOV	M,L	
77	MOV	M,A	
78	MOV	A,B	
79	MOV	A,C	
7A	MOV	A,D	
7B	MOV	A,E	
7C	MOV	A,H	
7D	MOV	A,L	
7E	MOV	A,M	
7F	MOV	A,A	

ACCUMULATOR

80	ADD	B
81	ADD	C
82	ADD	D
83	ADD	E
84	ADD	H
85	ADD	L
86	ADD	M
87	ADD	A
88	ADC	B
89	ADC	C
8A	ADC	D
8B	ADC	E
8C	ADC	H
8D	ADC	L
8E	ADC	M
8F	ADC	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	M
97	SUB	A
98	SBB	B
99	SBB	C
9A	SBB	D
9B	SBB	E
9C	SBB	H
9D	SBB	L
9E	SBB	M
9F	SBB	A
A0	ANA	B
A1	ANA	C
A2	ANA	D
A3	ANA	E
A4	ANA	H
A5	ANA	L
A6	ANA	M
A7	ANA	A
A8	XRA	B
A9	XRA	C
AA	XRA	D
AB	XRA	E
AC	XRA	H
AD	XRA	L
AE	XRA	M
AF	XRA	A
B0	ORA	B
B1	ORA	C
B2	ORA	D
B3	ORA	E
B4	ORA	H
B5	ORA	L
B6	ORA	M
B7	ORA	A
B8	CMP	B
B9	CMP	C
BA	CMP	D
BB	CMP	E
BC	CMP	H
BD	CMP	L
BE	CMP	M
BF	CMP	A

CONSTANT DEFINITION

0B0H, 1AH	Hex
1050, 105	Decimal
720, 720	Octal
110 118, 001108	Binary

OPERATORS

( )
*, MOD SHL, SHR
+, –
NOT
AND
OR, XOR

MOVE IMMEDIATE

06	MVI B,	08
0E	MVI C,	
16	MVI D,	
1E	MVI E,	
26	MVI H,	
2E	MVI L,	
36	MVI M,	
3E	MVI A,	

Acc IMMEDIATE

C6	ADI	08
CE	ACI	
D6	SUI	
DE	SBI	
E6	ANI	
EE	XRI	
F6	ORI	
FE	CPI	

LOAD IMMEDIATE

01	LXI B,	08
11	LXI D,	
21	LXI H,	
31	LXI SP,	

STACK OPS

C5	PUSH B
D5	PUSH D
E5	PUSH H
F5	PUSH PSW
C1	POP B
D1	POP D
E1	POP H
F1	POP PSW

DOUBLE ADD

09	DAD B
19	DAD D
29	DAD H
39	DAD SP

MOVE

40	MOV B,B	
41	MOV B,C	
42	MOV B,D	
43	MOV B,E	
44	MOV B,H	
45	MOV B,L	
46	MOV B,M	
47	MOV B,A	
48	MOV C,B	
49	MOV C,C	
4A	MOV C,D	
4B	MOV C,E	
4C	MOV C,H	
4D	MOV C,L	
4E	MOV C,M	
4F	MOV C,A	
50	MOV D,B	
51	MOV D,C	
52	MOV D,D	
53	MOV D,E	
54	MOV D,H	
55	MOV D,L	
56	MOV D,M	
57	MOV D,A	

SPECIALS

EB	XCHG
27	DAA
2F	CMA
37	STC
3F	CMC

INCREMENT

04	INR B	
0C	INR C	
14	INR D	
1C	INR E	
24	INR H	
2C	INR L	
34	INR M	
3C	INR A	
03	INX B	
13	INX D	
23	INX H	
33	INX SP	

DECREMENT

05	DCR B	
0D	DCR C	
15	DCR D	
1D	DCR E	
25	DCR H	
2D	DCR L	
35	DCR M	
3A	DCR A	
0B	DCX B	
1B	DCX D	
2B	DCX H	
3B	DCX SP	

LOAD/STORE

0A	LDAX B	
1A	LDAX D	
2A	LHLD Adr	
3A	LDA Adr	
02	STAX B	
12	STAX D	
22	SHLD Adr	
32	STA Adr	

INPUT/OUTPUT

D3	OUT	08
DB	IN	

PSEUDO INSTRUCTION

ORG	Adr
EQU	[Adr]
EQU	D16
SET	D16
DS	D16
DB	D8 [ ]
DW	D16 [ ]
IF	D16
ENDIF	
MACRO	[ ]
ENDM	

STANDARD SETS

A	SET	7
B	SET	0
C	SET	1
D	SET	2
E	SET	3
H	SET	4
L	SET	5
M	SET	6
SP	SET	6
PSW	SET	6

FLAG BYTE STACK FORMAT

7	6	5	4	3	2	1	0
S	Z	0	AC	0	P	1	CY

dodaje jeden bajt danych do zawartości akumulatora, wynik pamiętany jest w akumulatorze

operacja	operand
ADD	rejestr
flagi PSW:	Z, S, P, C, Ax

ADI dodaj bezpośrednio

dodaje zawartość drugiego bajtu instrukcji do zawartości akumulatora i wynik pamięta w akumulatorze

operacje	operand
ADI	dana
flagi PSW:	Z, S, P, C, Ax

Operand może być w formie: stałej, etykiety /jako zdefiniowana wcześniej wartość/ i wyrażenia

ANA logiczny AND z akumulatorem

wykonuje logiczny AND używając zawartości specyfikowanej danej i akumulatora, wynik umieszcza w akumulatorze

operacja	operand
ANA	rejestr
flagi PSW:	Z, S, P, C, Ax

ANI logiczna koniunkcja /AND/ bezpośrednio z akumulatorem

wykonuje AND używając zawartości drugiego bajtu instrukcji i akumulatora, wynik umieszcza w akumulatorze

operacja	operand
ANI	dana
flagi PSW:	Z, S, P, C, Ax

CALL wywołaj

jest kombinacją funkcji PUSH i JMP. Instrukcja CALL przenosi zawartość licznika programu /PC/ /adres następnej instrukcji/ na stos i wykonuje skok pod adres wyspecyfikowany w operandzie

operacja	operand
CALL	adres
flagi PSW:	brak

CC wywołaj jeśli carry

jest kombinacja funkcji instrukcji JC i PUSH. Instrukcja CC testuje bit "carry" w PSW. Jeżeli jest ustawiony na jeden, CC przenosi zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja	operand

CC adres
flagi PSW: brak

CM wywołaj jeżeli minus

jest kombinacją funkcji instrukcji JM i PUSH. Instrukcja testuje bit "znaku", jeżeli jest ustawiony na jeden, CM przenosi zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja operand
CM adres
flagi PSW: brak

CMA neguj akumulator

neguje każdy bit akumulatora. Wszystkie flagi PSW pozostają niezmienione

operacja operand
CMA -

CMC neguj carry

neguje aktualną wartość bitu carry. Wszystkie pozostałe flagi PSW pozostawia niezmienione

operacja operand
CMC -

CMP porównaj z akumulatorem

porównuje specyfikowany bajt z zawartością akumulatora. Wartości porównywane pozostają niezmienione, ustawiane są tylko flagi w PSW. Bit "zera" wskazuje na równość, nie "carry" wskazuje, że akumulator jest większy niż specyfikowany bajt; "carry" wskazuje, że akumulator jest mniejszy niż specyfikowany operand

operacja operand
CMP rejestr
flagi PSW: Z, S, P, C, Ax

CNC wywołaj jeżeli nie carry

jest kombinacją funkcji instrukcji JNC i PUSH. Instrukcja testuje bit "carry", jeżeli jest ustawiony na zero, to CNC przesyła zawartość licznika programu /CP/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja operand
CNC adres
flagi PSW: brak

CNZ wywołaj jeżeli nie zero

jest kombinacją funkcji instrukcji JNZ i PUSH. Instrukcja testuje bit "zero" w PSW, jeżeli bit "zera" jest "zgaszony", to CNZ przenosi zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja	operand
CNZ	adres
flagi PSW:	brak

CP wywołaj jeżeli plus

jest kombinacją funkcji instrukcji JP i PUSH. Instrukcja testuje bit "znaku" w PSW, jeżeli bit jest ustawiony na zero, CP przenosi zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja	operand
CP	adres
flagi PSW:	brak

CPE wywołaj jeżeli parzystość

jest kombinacja funkcji instrukcji JPE i PUSH. Instrukcja testuje bit parzystości w PSW, jeżeli jest ustawiony /ma wartość jeden/, to CPE przesyła zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja	operand
CPE	adres

CPI porównaj bezpośrednio

porównuje zawartość drugiego bajtu instrukcji z zawartością akumulatora ustawiając bit "carry" i bit "zera" w PSW. Wartości porównywane pozostają niezmienione.
Bit "zera" ustawiony jest gdy równe; nie "carry" wskazuje, że zawartość akumulatora jest większa niż bezpośrednia dana; carry wskazuje, że akumulator jest mniejszy niż bezpośrednia dana

operacja	operand
CPI	dana

CPO wywołaj jeżeli nieparzystość

jest kombinacja funkcji instrukcji JPO i PUSH. Instrukcja testuje bit "parzystości" w PSW, jeżeli ustawiony jest na zero, to CPO przesyła zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja	operand

CPO adres
flagi PSW: brak

CZ wywołaj jeżeli zero

jest kombinacją funkcji instrukcji JZ i PUSH. Instrukcja testuje bit "zero" w PSW, jeżeli flaga jest ustawiona /ma wartość jeden/, to CZ przesyła zawartość licznika programu /PC/ na stos i wykonuje skok pod adres specyfikowany w operandzie

operacja operand
CZ adres

DAA dziesiętne uzupełnienie akumulatora

wyrównuje /adjustuje/ osiem bitów zawartości akumulatora do postaci dwu czterobitowych kodów cyfr dziesiętnych

operacja operand
DAA -
flagi PSW: Z, S, P, C, Ax

DAD dodaj pary rejestrów

dodaje 16-bitowe wartości specyfikowane w parze rejestrów do zawartości w parze HiL

operacja operand
DAD { B, D, H, SP }

Instrukcja DAD może dodawać tylko zawartości rejestrów BiC, DiE, HiL lub SP /licznik stosu/

flagi PSW: C

DCR zmniejsz o 1

odejmuje od zawartości specyfikowanego bajtu /rejestru/ jeden, wynik umieszcza w operandzie

operacja operand
DCR rejestr
flagi PSW: Z, S, P, Ax

DCX zmniejsz o 1 zawartość pary rejestrów

działa jak DCR, tylko, że na parze rejestrów

operacja operand
DCR { B, D, H, SP }
flagi PSW: brak

DI blokuj przerwania

przerwania systemu są blokowane kiedy procesor obsługuje przerwanie lub wykonuje instrukcję DI

operacja	operand
DI	-
flagi PSW:	brak

EI odblokuj przerwanie

wykonuje odblokowanie przerwania systemu, zezwalając na wykonywanie się instrukcji programu zawieszonego przerwaniem

operacja	operand
EI	-

HLT stop procesor

zatrzymuje pracę procesora, czekając na przerwanie. Licznik programowy zawiera adres następnej instrukcji do wykonania

IN pobierz daną z portu

czyta 8-bitów danych specyfikowanych w porcie i ładuje je do akumulatora

operacja	operand
IN	adres
flagi PSW:	brak

INR zwiększ o jeden

dodaje jeden do specyfikowanego bajtu

operacja	operand
INR	rejestr
flagi PSW:	Z, S, P, Ax

INX zwiększ o jeden zawartość pary rejestrów

operacja	operand
INX	B D H SP
flagi PSW:	brak

JC skok jeżeli carry

testuje bit "carry" w PSW, jeżeli ustawiony jest na jeden, następuje skok pod adres specyfikowany w operandzie

operacja	operand
JC	adres
flagi PSW:	brak

JM skok jeżeli minus

testuje bit "znaku" w PSW, jeżeli ustawiony jest na jeden, następuje skok pod adres specyfikowany w operandzie

operacja operand
JM adres
flagi PSW: brak

JMP skok

wykonuje skok bezwarunkowy pod adres /umieszczony w drugim i trzecim bajcie instrukcji/ specyfikowany w operandzie

operacja operand
JMP adres
flagi PSW: brak

JNC skok jeśli nie carry

testuje bit "carry" w PSW, jeżeli ustawiony jest na zero, następuje skok pod adres specyfikowany w operandzie

operacja operand
JNC adres
flagi PSW: brak

JNZ skok jeśli nie zero

testuje bit "zera" w PSW, jeżeli ustawiony jest na zero, następuje skok pod adres specyfikowany w operandzie

operacja operand
JNZ adres
flagi PSW: brak

JP skok jeśli plus

testuje bit "znaku" w PSW, jeśli ustawiony jest na zero, następuje skok pod adres specyfikowany w operandzie

operacja operand
JP adres
flagi PSW: brak

JPE skok jeżeli parzystość

sprawdza w akumulatorze ilość jedynek, jeśli jest parzysta następuje skok pod adres specyfikowany w operandzie

operacja operand
JPE adres
flagi PSW: brak

JPO skok jeśli nieparzystość

sprawdza w akumulatorze ilość jedynek, jeśli jest nieparzysta następuje skok pod adres specyfikowany w operandzie

operacja operand
JPO adres
flagi PSW: brak

JZ skok jeśli zero

testuje bit "zera" w PSW, jeśli ustawiony jest na jeden, następuje skok pod adres specyfikowany w operandzie

operacja	operand
JZ	adres
flagi PSW:	brak

LDA ładuj akumulator z pamięci bezpośrednio

ładuje akumulator zawartością bajtu pamięci o adresie zadanym w drugim i trzecim bajcie instrukcji

operacja	operand
LDA	adres
flagi PSW:	brak

LDAX ładuj akumulator pośrednio

ładuje akumulator zawartością bajtu pamięci o adresie zawartym w parze rejestrów BC lub DE

operacja	operand
LDAX	{ B, D }
flagi PSW:	brak

LHLD ładuj Hi L bezpośrednio

ładuje najpierw rejestr L zawartością bajtu pamięci o adresie zadanym w drugim i trzecim bajcie instrukcji. Następuje ładuje rejestr H o zawartością pamięci o adresie o jeden bajt większym

U w a g a !

Wartość dwubajtowa np. 804EH zapisana jest w pamięci następująco: na adresie niższym 4E, na adresie wyższym 80

operacja	operand
LHLD	adres
flagi PSW:	brak

LXI ładuj parę rejestrów daną bezpośrednią

jest trzybajtową instrukcją; drugi i trzeci bajt zawiera "daną źródłową", która ładowana jest do pary rejestrów specyfikowanych w operandzie

operacja	operand
LXI	{ B, D, H, SP }, dana
flagi PSW:	brak

MOV prześlij

umieszcza w rejestrze R1 zawartość rejestru R2. Zawartość rejestru R2 nie ulega zmianie

operacja	operand
MOV	R1, R2
flagi PSW:	brak

Instrukcja może umieszczać zawartość rejestru w pamięci o adresie zawartym w parze rejestrów HL /parę HL można oznaczać jedną literą M/

operacja	operand
MOV	M, rejestr

Operacja odwrotna do powyższej pozwala w rejestrze umieścić daną z pamięci o adresie zawartym w HL

operacja	operand
MOV	rejestr, M
flagi PSW:	brak

MVI prześlij natychmiastowo

umieszcza w rejestrze drugi bajt instrukcji

operacja	operand
MVI	rej, dana
flagi PSW:	brak

instrukcja może umieścić daną bezpośrednio w pamięci o adresie zawartym w parze rejestrów HL

MVI	M, dana

NOP nic nie rób

operacja	operand
NP	-

ORA suma logiczna z akumulatorem

wykonuje sumę logiczną akumulatora z zawartością rejestru specyfikowanego w operandzie /wynik w akumulatorze/

operacja	operand
ORA	rejestr
flagi PSW:	Z, S, P, C, Ax

operacja może być wykonywana z zawartością bajtu pamięci

ORA	M

ORI suma logiczna akumulatora z bajtem danych

wykonuje sumę logiczną akumulatora z zawartością drugiego bajtu instrukcji. Wynik operacji umieszcza w akumulatorze

operacja	operand

ORI dana
flagi PSW: Z, S, P, C, Ax

OUT wyślij do portu
wysyła zawartość akumulatora pod adres specyfikowany w operandzie

operacja operand
OUT adres
flagi PSW: brak

PCHL prześlij H i L do licznika programu
umieszcza zawartość pary rejestrów HL do licznika programu /PC/. Efekt instrukcji jest taki jak instrukcji JMP

operacja operand
PCHL -
flagi PSW: brak

POP przenieś
przenosi dwa bajty danych z wierzchołku stosu do pary rejestrów specyfikowanych w operandzie

operacja operand
POP { B, D, H, PSW }
flagi PSW: brak

PUSH wyślij
wysyła zawartość pary rejestrów na stos.

operacja operand
PUSH { B, D, H, PSW }
flagi PSW: brak

RAL rotacja w lewo poprzez carry
wykonuje rotację zawartości akumulatora i bitu carry o jedną pozycję w lewo

operacja operand
RAL
flagi PSW: C

RAR rotacja w prawo poprzez carry
wykonuje rotację zawartości akumulatora i bitu carry o jedną pozycję w prawo

operacja operand

RAR

flagi PSW: C

RC powrót jeśli carry

testuje bit carry w PSW, jeśli jest ustawiony na jeden, to instrukcja pobiera dwa bajty z wierzchołka stosu i umieszcza je w liczniku programowym /PC/

operacja	operand
RC	-
flagi PSW:	brak

RET powrót z podprogramu

pobiera dwa bajty z wierzchołku stosu i umieszcza je w liczniku programowym /PC/

operacja	operand
RET	-
flagi PSW:	brak

RLC rotacja w lewo /logicznie/ akumulatora

wykonuje rotację zawartości akumulatora o jedną pozycję w lewo przenosząc bit b_7 na b_0, bit "carry" na b_7

operacja	operand
RLC	-
flagi PSW:	C

RM powrót jeśli minus

testuje bit "znaku" w PSW, jeśli ustawiony jest na jeden, to instrukcja pobiera dwa bajty z wierzchołka stosu i umieszcza je w liczniku programowym /PC/

operacja	operand
RM	-
flagi PSW:	brak

RNC powrót jeśli nie carry

testuje bit "carry" w PSW, jeśli flaga ustawiona jest na zero, to RNC pobiera z wierzchołka stosu dwa bajty i umieszcza je w liczniku programowym /PC/

operacja	operand
RNC	-
flagi PSW:	brak

RNZ powrót jeśli nie zero

testuje bit "zera" w PSW, jeśli ustawiony jest na zero, to RNZ pobiera z wierzchołka stosu dwa bajty i umieszcza je w liczniku programowym /PC/

operacja operand
RNZ -
flagi PSW: brak

RP powrót jeśli plus
testuje bit "znaku" w PSW, jeśli ustawiony jest na zero, to RP pobiera z wierzchołka stosu dwa bajty i umieszcza je w liczniku programowym

operacja operand
RP -
flagi PSW: brak

RPE powrót jeśli parzystość
sprawdza w akumulatorze ilość jedynek, jeśli jest parzysta, to pobiera z wierzchołka stosu dwa bajty i umieszcza je w liczniku programowym /PC/

operacja operand
RPE -
flagi PSW: brak

RPO powrót jeśli nieparzystość
sprawdza w akumulatorze ilość jedynek, jeśli nieparzysta, to pobiera z wierzchołka stosu dwa bajty i umieszcza je w liczniku programowym /PC/.

operacja operand
RPO -
flagi PSW: brak

RRC rotacja akumulatora w prawo /logicznie/
ustawia bit "carry" równy bitowi b0 akumulatora, następnie wykonuje rotację w prawo, przenosząc /jeden bit/ bit z pozycji najniższej do pozycji najwyższej pozostałe przesuwając w prawo

operacja operand
RRC -
flagi PSW: C

RST restart /programowy/
jest odmianą instrukcji CALL przeznaczoną przede wszystkim dla restartu programu i ponownego powrotu do programu głównego. Procesor przenosi wartość operandu instrukcji na 3, 4 i 5 pozycję bitu dwubajtowego pola /bity pozostałe to zera/ i umieszcza go w liczniku programowym /PC/

operacja operand

RST kod

U w a g a ! wartość kodu Ø-7

flagi PSW: brak

RZ powrót jeśli zero

testuje bit "zera" w PSW, jeśli ustawiony jest na jeden, to RZ pobiera z wierzchołka stosu dwa bajty i umieszcza je w liczniku programowym

operacja operand

RZ -

SBB odejmij z pożyczką

odejmuje bit "carry" i zawartość specyfikowaną w operandzie od zawartości akumulatora. Wynik operacji umieszcza w akumulatorze

operacja operand

SBB { rejestr / M }

flagi PSW: Z, S, P, C, Ax

SBI odejmij bezpośrednio daną z pożyczką

odejmuje bit "carry" i zawartość drugiego bajtu instrukcji od zawartości akumulatora. Wynik operacji umieszcza w akumulatorze

operacja operand

SBI dana

flagi PSW: Z, S, P, C, Ax

SHLD pamiętaj H i L bezpośrednio

pamięta zawartość pary rejestrów HL w pamięci o adresie specyfikowanym w operandzie

operacja operand

SHLD adres

flagi PSW: brak

SPHL przenieś H i L do SP

ładuje zawartość rejestrów H i L do licznika stosu SP

operacja operand

SPHL -

flagi PSW: brak

STA pamiętaj akumulator bezpośrednio

pamięta zawartość akumulatora w pamięci o adresie specyfikowanym w operandzie

operacja operand

STA adres
flagi PSW: brak

STAX pamiętaj akumulator pośrednio

pamięta zawartość akumulatora w pamięci o adresie specyfikowanym parą rejestrów w operandzie

operacja	operand
STAX	{ B, D }

flagi PSW: brak

STC ustaw carry

ustawia bit carry na jeden

operacja	operand
STC	-

flagi PSW: brak

SUB odejmij

odejmuje od zawartości akumulatora bajt danych specyfikowany w operandzie. Wynik umieszcza w akumulatorze.

operacja	operand
SUB	{ rejestr M }

flagi PSW: Z, S, P, C, Ax

SUI odejmij bezpośrednią daną

odejmuje od zawartości akumulatora zawartość drugiego bajtu instrukcji i wynik pamięta w akumulatorze

operacja	operand
SUI	dana

flagi PSW: Z, S, P, C, Ax

XCHG wymień H i L z D i E

wymienia zawartości rejestrów HL z rejestrami DE

operacja	operand
XCHG	-

flagi PSW: brak

XRA różnica symetryczna z akumulatorem

wykonuje różnicę symetryczną /logicznie/ zawartości akumulatora z zawartością specyfikowaną w operandzie. Wynik w A.

operacja	operand
XRA	{ rejestr M }

flagi PSW: Z, S, P, C, Ax

XRI różnica symetryczna akum. z daną bezpośrednią

wykonuje różnicę symetryczną /logicznie/ zawartości akumulatora z zawartością drugiego bajtu instrukcji. Wynik umieszcza w A

operacja	operand
XRI	dana
flagi PSW:	Z, S, P, C, Ax

XTHL wymień H i L z wierzchołkiem stosu

wymienia dwa bajty wierzchołka stosu z dwoma bajtami /zawartością/ rejestrów H i L

operacja	operand
XTHL	-
flagi PSW:	brak

6.3. Kompilacja zbiorów źródłowych, łączenie i wykonywanie programów

Program jako zbiór źródłowy tworzony może być za pomocą edytora ED.COM. Nazwa zbioru, programu w języku asembler może być dowolna 8 znakowa, typ zbioru musi być ASM. Wymagane jest to przez kompilator asemblera, którego wywołujemy w następujący sposób:

ASM ⟨ nazwa zbioru ⟩ [P1P2P3]

gdzie:

⟨ nazwa zbioru ⟩ jest 8 znakową nazwą /bez typu zbioru/

P1 - oznacza jednostkę dyskową ze zbiorem źródłowym,

P2 - oznacza jednostkę dyskową na której będzie plik wynikowy typu .HEX. /jeśli P2 = Z, to asembler nie tworzy pliku wynikowego/

P3 - oznacza jednostkę dyskową, na której ma być umieszczony tekst programu typu .PRN /jeżeli P3 = X, to tekst jest wyświetlany na monitorze, gdy P3 = Z kompilator nie tworzy pliku/.

Wykaz komunikatów o błędach kompilatora zawiera tablica 14.
Utworzony przez kompilator zbiór typu .HEX można uruchomić programem DDT.COM /szczegółowe omówienie w następnym punkcie/ lub utworzyć programem LOAD.COM moduł wykonywalny typu .COM.
Wywołanie programu LOAD wygląda następująco:

A ⟩ LOAD ⟨ nazwa zbioru ⟩ /typu HEX/

Utworzony zbiór typu .COM zapisywany jest na dysku aktualnie przydzielonym do systemu. Uruchomienie programu następuje po napisaniu na monitorze nazwy zbioru.
W załączonym poniżej przykładzie przedstawiono praktyczny program, który wymazuje ekran monitora i ustawia kursor w lewym górnym rogu ekranu. Zbiór źródłowy założono edytorem pod nazwą C.ASM. W wyniku użycia kompilatora ASM.COM i Loadera LOAD.COM otrzymano postać wykonywalną programu o nazwie C.COM. Operator pisząc C na klawiaturze wywołuje program, który wykonuje się w obszarze TPA.

Tablica 14

Komunikaty o błędach kompilatora

Błędy są sygnalizowane na lewym marginecie listu programu. Kody błędów:

D - błąd danych
E - błąd wyrażenia
L - błąd etykiety
N - błędna instrukcja
O - błąd nadmiaru
P - błąd fazy: etykieta zmienia wartość w czasie translacji
R - błąd rejestru
V - błąd operandu
S - błąd syntaktyczny

NO SOURCE FILE PRESENT - zbiór nie istnieje na dysku lub typ zbioru nie jest ASM
NO DIRECTORY SPACE - pełny katalog dysku
SOURCE FILE READ ERROR - błąd w trakcie czytania zbioru
OUTPUT FILE WRITE ERROR - błąd w zbiorze wynikowym, powtórz translację
CANNOT CLOSE FILE - błąd zamknięcia zbioru. Sprawdzić czy dysk nie jest chroniony przed zapisem

```
;
;     "C.COM" DLA IMP85
;      W.WOLSKI
;     PROGRAM WYMAZUJE EKRAN
;     I USTAWIA KURSOR W LEWYM GORNYM ROGU
;
;===== DEKLARACJE ==========================
;
BDOS    EQU 0005H    ;WEJSCIE DO BDOS'U
KODF    EQU 2        ;FUNKCJA CONOUT
GROG    EQU 14       ;KOD DLA PRZESUNIECIA W GORNY ROG KURSORA
WEKRAN  EQU 12       ;KOD DLA WYMAZANIA EKRANU
;
        ORG 100H     ;START PROGRAMU
        LXI H,0      ;AKTUALNY SP W CCP
        DAD SP
        SHLD SPSTARY ;WARTOSC PAMIETAJ
;LOKALNA WARTOSC SP
        LXI SP,SPNOWY
;===== PROGRAM GLOWNY ======================
;
N1      MVI E,GROG   ;PRZESLIJ DO E ZNAK
        CALL WYSLIJ  ;WYSLIJ ZNAK NA KONSOLE
N2      MVI E,WEKRAN ;PRZESLIJ DO E ZNAK
        CALL WYSLIJ  ;WYSLIJ ZNAK NA KONSILE
        JMP WYJSCIE  ;WROC DO CCP
;
;=======PODPROGRAMY ========================
;
WYSLIJ  MVI C,KODF   ;POBIERZ DO E KOD FUNKCJI BDOS'U
        JMP BDOS     ;SKOCZ DO BDOS'U
;
WYJSCIE LHLD SPSTARY
;(MOZNA WYKONAC JMP 0000H )
        SPHL
        RET          ;WROC DO CCP
;=======ZMIENNE============================
SPSTARY DS 2         ;WARTOSC SP Z CCP
;       OBSZAR STOSU
        DS 32        ;REZERWACJA 16 POZIOMOW
SPNOWY:
        END
```

A) Program przykladowy C.ASM

```
ASM C BBB
HIFS ASSEMBLER - VER 1.0
0143
0C0H USE FACTOR
END OF ASSEMBLY

                ;      C.COM" DLA IMP85
                ;      W.WOLSKI
                ;     PROGRAM WYMAZUJE EKRAN
                ;     I USTAWIA KURSOR W LEWYM GORNYM ROGU
                ;
                ;======DEKLARACJE ==========================
 0005 =         ;BDOS    EQU 0005H    ;WEJSCIE DO BDOS'U
0002            KODF     EQU 2        ;FUNKCJA CONOUT
000E =          GROG     EQU 14       ;KOD DLA PRZESUNIECIA W GORNY ROG E
000C =          WEKRAN   EQU 12       ;KOD DLA WYMAZANIA EKRANU
                ;
0100                     ORG 100H     ;START PROGRAMU
0100 210000             LXI H,0      ;AKTUALNY SP W CCP
0103 39                 DAD SP
0104 222101             SHLD SPSTARY ;WARTOSC PAMIETAJ
                ;LOKALNA WARTOSC SP
0107 3 4301             LXI SP,SPNOWY
                ;====== PROGRAM GLOWNY ==========================
                ;
0106 1E0E       N1      MVI E,GROG   ;PRZESLIJ DO E ZNAK
010C CD1701             CALL WYSLIJ ;WYSLIJ ZNAK NA KONSOLE
010F 1E0C       N2      MVI E,WEKRAN ;PRZESLIJ DO E ZNAK
0111 CD1701             CALL WYSLIJ ;WYSLIJ ZNAK NA KONSOLE
0114 C31C01             JMP WYJSCIE ;WROC DO CCP
                ;
                ;=======PODPROGRAMY ==========================
                ;
0117 0E 2       WYSLIJ  MVI C,KODF   ;PODLICZ DO E KOD FUNKCJI BDOS'U
0119 C30500             JMP BDOS     ;SKOCZ DO BDOS'U
                ;
011C 2A2101     WYJSCIE LHLD SPSTARY
                ;(MOZNA WYKONAC JMP 0000H )
011F F9                 SPHL
0120 C9                 RET              ;WROC DO CCP
                ;=========ZMIENNE;==========================
0121            SPSTARY DS 2         ;WARTOSC SP Z CCP
                ;       OBSZAR STOSU
0123                    DS 32        ;REZERWACJA 1  POZIOMU
                SPNOWY:
0143                    END

LOAD B:C

FIRST ADDRESS 0100
LAST  ADDRESS 0120
BYTES READ    0021
RECORDS WRITTEN 01
```

7. POPRAWIANIE I URUCHAMIANIE PROGRAMÓW POD KONTROLĄ DDT

Program DDT.COM przeznaczony jest do wspomagania uruchamiania i/lub poprawiania programów asemblerowych typu .HEX lub postaci wykonywalnej typu .COM, poprzez kontrolowanie określonych sekwencji programu lub całego programu. Wywołanie programu DDT odbywa się następująco:

1/ A > DDT < cr >

2/ A > DDT < nazwa zbioru > . < typ > < cr >

gdzie typ = { HEX / COM }

Pierwszy sposób powoduje umieszczenie samego programu DDT w pamięci RAM. Drugi sposób umieszcze program DDT i zbiór /program/ użytkownika w pamięci RAM.

Działaniem programu sterują komendy /dyrektywy/, które podaje się, gdy program po załadowaniu do pamięci zgłosi się kreską, tzn. "-". W dalszej części przedstawione zostaną szczegółowo komendy programu DDT.

Komenda A /Assembly/

Umożliwia wprowadzanie bezpośrednio do pamięci rozkazów asemblerowych. Postać komendy jest następująca:

A s

gdzie:

s = adres heksadecymalny 4 znakowy /dwa bajty/

Program DDT przyjmuje podany adres, przenosząc sterowanie do trybu czytania rozkazów asemblerowych z operandami w postaci heksadecymalnej. Każdy następny wiersz jest wprowadzany przez czytaniem następnego rozkazu. Wprowadzenie pustej linii kończy tryb pracy komendy.

Komenda D /display/

Komenda umożliwia wyprowadzanie zawartości pamięci w postaci heksadecymalnej i znakowej /kod ASCII/.

Postać komendy jest następująca:

1/ D

2/ Ds

3/ Ds, f

Pierwsza postać komendy umożliwia wyprowadzenie zawartości pa-

mięci od adresu 100H, wyświetlając 16 linii. Każda wyświetlona linia ma postać:

aaaa bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb cccccccccc cccc

gdzie:

aaaa = adres w postaci heksadecymalnej

bb = zawartość programu w postaci hex

c = zawartość pamięci /programu/ w postaci znakowej /kod ASCII/ przy czym znaki nie mające odpowiedników graficznych przedstawione są jako kropka ".".

Komenda wyprowadza zarówno małe jak i duże litery. Każda linia przedstawia 16 bajtów zawartości pamięci.

Druga postać komendy umożliwia wyprowadzenie zawartości pamięci poczynając od adresu /hex/ podanego w komendzie jako operand. Trzecia postać komendy umożliwia wyprowadzenie zawartości pamięci poczynając od adresu podanego pierwszym operandem /s/ do adresu podanego drugim operandem /f/.

Komenda F /Fill/

Umożliwia wprowadzenie do pamięci od adresu określonego pierwszym operandem /s/ do adresu określonego drugim operandem /f/ wartości określonej trzecim operandem /c/. Postać komendy jest następująca:

F s, f, c

gdzie:

s = adres początkowy /hex/

f = adres końcowy /hex/

c = stała hex /bajt/

Komenda G /Go/

Inicjuje realizację programu, umożliwiając warunkowe określenie do dwóch punktów kontrolnych. Może wystąpić w jednej z następujących postaci:

1/ G

2/ Gs

3/ Gs, b

4/ Gs, b, c

5/ G, b

6/ G, b, c

Pierwsza postać startuje program, poczynając od aktualnej wartości licznika programowego i aktualnego stanu CPU. bez ustawionego punktu kontrolnego /powtórne przekazanie sterowania do programu DDT może być realizowane poprzez rozkaz RST 7/.
Druga postać startuje program od adresu określonego operandem /s/.
Trzecia postać pozwala określić adres początkowy realizacji programu oraz adres /b/ po osiągnięciu którego, realizacja programu jest zatrzymana, /adres b musi być w obszarze adresowym programu/.
Czwarta postać umożliwia określenie dwóch punktów kontrolnych, jeden o adresie b, drugi o adresie c. Osiągnięcie obu punktów kontrolnych kończy realizację testowanego programu a oba punkty kontrolne są likwidowane.
Postać piąta i szósta przyjmuje wartość licznika PC na podstawie wartości pochodzących z aktualnego stanu procesora, umożliwiając ustawienie jednego lub dwóch punktów kontrolnych. Realizacja testowanego programu przebiega od adresu początkowego do punktu kontrolnego. Oznacza to, że nie ma ingerencji programu DDT między adresem początkowym a adresem przerwania. W ten sposób, jeśli testowany program nie zawiera przerwań, sterowanie nie może być przekazane do programu DDT bez wykonania rozkazu RST 7. Gdy napotkany zostanie punkt kontrolny program DDT zatrzymuje swoją realizację programu, wyprowadzając

znak : wd

gdzie:
d = adres przerwania

Komenda I /Input/

Umożliwia wprowadzenie nazwy zbioru /programu/ do bloku strony zerowej o adresie 5CH. Postać komendy jest następująca:

I ⟨ nazwa zbioru ⟩ . ⟨ typ ⟩

$$\text{typ} = \left\{ \begin{matrix} \text{HEX} \\ \text{COM} \end{matrix} \right\}$$

lub

I ⟨ nazwa zbioru ⟩

Jeżeli chcemy wprowadzić program do pamięci, to musimy użyć jeszcze komendy R.

Komenda L /List/

Używana jest do wyprowadzania postaci asemblerowej wskazanego obszaru pamięci. Postać komendy jest następująca:

1/ L

2/ Ls

3/ Ls, f

Pierwsza postać komendy wyprowadza 12 linii desasemblerowego kodu zgodnie z aktualną listą adresową.
Druga postać umożliwia ustawienie adresu początkowego desasemblerowego kodu i również powoduje wyprowadzenie 12 linii.
Trzecia postać komendy umożliwia wyprowadzenie kodu od adresu początkowego /s/ do adresu końcowego /f/.

Komenda M /Move/

Umożliwia przesuwanie obszarów programu lub danych z jednego miejsca pamięci do innego obszaru. Postać komendy jest następująca:

Ms, f, d

gdzie:

s = adres początkowy obszaru przesuwanego

f = adres końcowy obszaru przesuwanego

d = adres początkowy, od którego rozpocznie się przesuwanie obszaru /s-f/

operacja jest wykonywana jeżeli s < f

Komenda R /Read/

Używane jest w połączeniu z komendą I do czytania zbioru /programu/ typu HEX lub COM z dysku do pamięci. Postać komendy jest następująca:

1/ R

2/ Rb

gdzie:

b = opcjonalny adres, który jest dodawana do adresu programu lub danych w czasie ładowania programu do pamięci

Jeśli b jest pominięte, to przyjmuje się domyślnie b = 0000H. Adres ładowania dla każdego rekordu jest pobierany z rekordów zbioru typu HEX, a dla zbioru typu COM przyjmowany jest jako wartość 100H. Ładowanie programu nie powinno niszczyć obszaru

strony zerowej tj. obszaru Ø-ØFFH. Po załadowaniu programu /zbioru/ program DDT wyprowadza komunikat:

NEXT PC
nnnn pppp

gdzie:
nnnn = następny adres po załadowanym programie
pppp = wartość licznika rozkazów /aktualna/ programu

Komenda S /Set/

Umożliwia badanie zawartości wskazanego bajtu pamięci i wprowadzenie ewentualnie nowej wartości pod wskazany adres. Postać komendy jest następująca:

Ss

gdzie:
s = heksadecymalny adres pamięci RAM
W odpowiedzi na tę komendę program DDT podaje adres pamięci i aktualną zawartość pod tym adresem. Jeśli wciśnięty zostanie klawisz RETURN, to zawartość ta nie ulegnie zmianie. Gdy napisana zostanie wartość, to zostanie ona zapamiętana pod wskazanym adresem. Możliwe jest kolejne sprawdzanie lub zmiana zawartości komórek pamięci do momentu, gdy wprowadzona zostanie błędna wartość lub znak kropki ".".

Komenda T /Trace/

Umożliwia selektywne śladowanie realizacji programu od 1 do 65535 kroków programowych. Postać komendy jest następująca:

Tn

gdzie:
n = ilość kroków programowych
Po wykonaniu zadanych ilości kroków, program zatrzymuje się pisząc na kontroli adres w postaci:

= hhhh

gdzie:
hhhh = następny adres programu do wykonania
Program realizowany z funkcja śledzenia wykonywany jest około 500 razy wolniej, ponieważ program DDT przejmuje sterowanie po wykonaniu każdego rozkazu programu użytkownika.

Komenda U /Untrace/

Jest podobna do komendy T za wyjątkiem tego, że nie są wyprowadzane bezpośrednio stany CPU po każdym kroku programowym. Tryb bez śladowania programu umożliwia badanie od 1 do 65535 kroków programowych i jest używany głównie do przejęcia sterowania, gdy osiągnięto punkty kontrolne. Wszystkie reguły obowiązujące dla komendy T odnoszą się również do U

Komenda X /Examine/

Umożliwia wyprowadzenie i zmianę stanu mikroprocesora /rejestry, SP, PC, PSW/. Postać komendy jest następująca:

1/ X

2/ Xr

gdzie:

r = rejestry mikroprocesora 8080

C = przeniesienie /carry/

Z = bit zera /Z/

M = bit znaku /minus/ /S/

E = bit parzystości /P/

I = przeniesienie połówkowe /Ax/

A = akumulator

B = para rejestrów B i C

D = para rejestrów D i E

H = para rejestrów H i L

S = licznik stosu /SP/

P = licznik programowy /PC/

W pierwszej postaci komendy, stan CPU wyprowadzany jest w następującej postaci:

CfZfMfEfIfA = bb B = dddd H = dddd S = dddd P = dddd instrukcja

gdzie:

f = wartość 1 lub Ø

bb = wartość bajtu /hex/

dddd = zawartość dwóch bajtów

instrukcja = postać instrukcji w programie

Druga postać komendy umożliwia wybór lub zmianę wartości odpowiednich stanów mikroprocesora.

Program DDT umożliwia wprowadzanie z konsoli np. nowej wartości rejestrów, SP, PSW itd.

Rejestry B, D, H są wprowadzane jako para rejestrów, bez względu na to, który rejestr zostanie wskazany do wprowadzania.
W załączeniu pokazano przykładową sesję testowania i uruchamiania programu w postaci zbioru typu HEX. Program znajduje się na dyskietce pod nazwą C.HEX. Opis programu przedstawiono w rozdziale 6.

```
DDT B:C.HEX
DDT VERS 1.0
NEXT PC
011A 0100
-L
  0100  LXI  SP,0200
  0103  MVI  E,0E
  0105  CALL 0110
  0108  MVI  E,0C
  010A  CALL 0110
  010D  CALL 0115
  0110  MVI  C,02
  0112  JMP  0005
  0115  MVI  C,00
  0117  JMP  0005
  011A  ??=  20
-G100,105
*0105
-X
COZOMOEOIO A=00 B=0000 D=000E H=0000 S=0200 P=0105 CALL 0110
-T
COZOMOEOIO A=00 B=0000 D=000E H=0000 S=0200 P=0105 CALL 0110*0110
-XP
P=0110 100
-X
COZOMOEOIO A=00 B=0000 D=000E H=0000 S=01FE P=0100 LXI  SP,0200
-A115
0115  JMP 0
0118
-L115,117
  0115  JMP  0000
  0118
-X
COZOMOEOIO A=00 B=0000 D=000E H=0000 S=01FE P=0100 LXI  SP,0200
-G,115

*0115
-G100,105
*0105
-X
COZIMOE1IO A=00 B=000C D=000E H=0000 S=0200 P=0105 CALL 0110
-T4
COZIMOE1IO A=00 B=000C D=000E H=0000 S=0200 P=0105 CALL 0110
COZIMOE1IO A=00 B=000C D=000E H=0000 S=01FE P=0110 MVI  C,02
COZIMOE1IO A=00 B=0002 D=000E H=0000 S=01FE P=0112 JMP  0005
COZIMOE1IO A=00 B=0002 D=000E H=0000 S=01FE P=0005 JMP  D400,D400
-X
COZIMOE1IO A=00 B=0002 D=000E H=0000 S=01FE P=D400 JMP  DAA2
-A117
0117  NOP
0118  L
-L118,L120
  0118  DCR  B
  0119  NOP
  011A  ??=  20
  011B  ??=  20
  011C  ??=  20
  011D  ??=  20
  011E  ??=  20
  011F  ??=  20
  0120  ??=  20
  0121
-L117,118
  0117  NOP
  0118  DCR  B

  0119
-X
COZIMOE1IO A=00 B=0002 D=000E H=0000 S=01FE P=D400 JMP  DAA2
```

8. PROGRAMOWANIE W JĘZYKU BASIC

8.1. Instrukcje języka BASIC

8.1.1. Zasady pisowni programów w BASIC-u

Język BASIC jest obecnie jednym z najpopularniejszych języków programowania wyższego rzędu dla mini i mikrokomputerów. Opracowano go w 1965 roku w Dartmouth College w USA. Jak sugeruje nazwa BASIC /Beginners All - purpose Symbolic Instruction Code/, jest to uniwersalny język programowania zaprojektowany z myślą o początkujących i nieprofesjonalnych użytkownikach komputerów. Reguły gramatyczne języka są maksymalnie proste, a programista otrzymuje szczegółową informację o rodzaju i lokalizacji występujących w programie błędów. BASIC mimo swej prostoty nie ustępuje uniwersalnością takim językom programowania jak FORTRAN czy ALGOL.

Popularność języka BASIC na mikrokomputerach wynika głównie z jego prostoty i łatwości implementacji na sprzęcie mikro. Kompilatory i interpretery języka są łatwe do napisania i co jest bardzo istotne, nie zajmują dużo pamięci. Stworzono wiele różnorodnych implementacji języka BASIC, w dalszej części skoncentrujemy się na wersji BASIC-80 Rev 5.21 CP/M Version Copyright 1977-1981 by Microsoft created: 28 Jul - 81 32824 Bytes free.

Formalne zasady pisowni programu w języku BASIC są bardzo proste. Program wprowadzany z klawiatury nie wymaga żadnych specjalnych pól zapisu. Pierwszą instrukcją programu jest pierwsza użytkowa instrukcja programu. Logiczny koniec programu jest wyznaczony przez wystąpienie instrukcji END, która nie musi być fizycznie ostatnią instrukcją programu.

Wiersz programu w języku BASIC ma następujący format:
n n n n n instrukcja przycisk CR
W jednym wierszu może być umieszczona więcej niż jedna instrukcja, lecz muszą być one rozdzielone znakiem dwukropka. Zalecane jest pisanie w jednym wierszu jednej instrukcji. Wiersz programu zaczynać się musi zawsze numerem linii, który musi mieścić się w zakresie 0 - 65529. Wskazane jest aby numeracja nie była ciągła, lecz kolejne wiersze numerowane z krokiem 10 lub większym. Numery linii wyznaczają kolejność wykonania instrukcji oraz sta-

nową etykiety instrukcji zapisanych w danej linii. Znakiem końca wiersza jest wciśnięcie klawisza CR. W jednym wierszu można maksymalnie umieścić 255 znaków.

Pisząc program mamy do dyspozycji pełny zbiór znaków dostępnych z klawiatury, chociaż niektóre znaki specjalne mają określone znaczenie.

8.1.2. Stałe w programie BASIC-u

W programach BASIC-u można pisać jawnie dwa rodzaje stałych:
- numeryczne,
- alfanumeryczne.

Stałe alfanumeryczne są ciągiem znaków ujętych w znaki ograniczające cudzysłowu. Maksymalna długość ciągu wynosi 255 znaków.

Przykłady stałych alfanumerycznych:

"PRZYKŁAD"

"1235 ZLOTYCH"

"WPROWADZENIE DO JEZYKA"

Stałe numeryczne są dodatnimi lub ujemnymi liczbami. Jest pięć typów stałych numerycznych:
- całkowite liczby z przedziału od - 32768 do + 32767, bez kropki dziesiętnej
- stałopozycyjne dodatnie lub ujemne liczby rzeczywiste, tzn. zawierające kropkę dziesiętną
- zmiennopozycyjne dodatnia lub ujemna liczba składająca się z mantysy, po której występuje litera E i wykładnik potęgi. Wykładnik musi być z zakresu od - 38 do + 38
- hexadecymalne są to liczby szesnastkowe poprzedzone znakiem & H
- aktualne są to liczby ósemkowe poprzedzone znakiem & O

8.1.3. Zmienne w programie BASIC-u

Zmienne wykorzystywane w programach BASIC-u identyfikowane są poprzez nazwę, której ostatni znak określa postać przechowywania zmiennej. Nazwa zmiennej może składać się z liter i cyfr, ale pierwszym znakiem musi być litera.

Możliwe są następujące typy zmiennych i odpowiadające im

znaki na ostatniej pozycji nazwy:

% zmienne całkowite /maksymalnie 4 cyfry/

! zmienne pojedynczej precyzji /maksymalnie 7 cyfr/

zmienne podwójnej precyzji /maksymalnie 16 cyfr/

$ zmienne znakowe /łańcuch znaków/ /maksymalnie 255 znaków/

Przykłady nazw:

K$ pozwala na przechowywanie ciągu znaków

PI% reprezentuje liczbę całkowitą

MAŁA! reprezentuje liczby pojedynczej precyzji

DUŻA# reprezentuje liczby podwójnej precyzji

Zmienne mogą być grupowane w tablice. Deklaracja tablic odbywa się przy pomocy instrukcji DIM, gdzie po nazwie tablicy określającej typ elementów podane są w nawiasie maksymalne wartości wskaźników. W przypadku tablic wielowymiarowych, maksymalne wartości wskaźników oddzielone są przecinkami. Dostęp do elementu tablicy następuje przez wypisanie nazwy tablicy z określonymi wartościami wskaźników w nawiasach.

Przykłady tablic

DIM K$ /15/	deklaruje tablicę 15-to elementową, której elementami są ciągi znaków
DIM TAB! /10, 5/	deklaruje tablicę dwuwymiarową, której elementami są liczby pojedynczej precyzji, pogrupowane w 10-ciu wierszach i 5-ciu kolumnach.

8.1.4. Wyrażenia i operatory

Zmienne i stałe można łączyć operatorami arytmetycznymi i logicznymi tworząc w ten sposób wyrażenia arytmetyczne i logiczne.

Jako operatory arytmetyczne wykorzystywane są następujące znaki umieszczone w kolejności malejących priorytetów:

^ dla operacji potęgowania

* dla operacji mnożenia

/ dla operacji dzielenia

+ dla operacji dodawania

- dla operacji odejmowania

Dla zapisu relacji mającej na celu porównanie dwóch wartości wykorzystywane są następujące znaki:

= relacja równości
<> relacja nierówności
< relacja mniejszości
> relacja większości
<= relacje mniejsze lub równe
>= relacje większe lub równe

Jako operatory logiczne wykorzystywane są zapisy NOT, AND, OR zgodnie z zasadami logiki matematycznej.

Wyrażenia arytmetyczne wykorzystywane są do wykonania obliczeń arytmetycznych. Połączenie nazw zmiennych i stałych programowych operatorami arytmetycznymi pozwala na wyliczanie pojedynczej wartości. Przy bardziej złożonych wyrażeniach celowe jest używanie nawiasów dla zwiększenia przejrzystości.

Przykłady wyrażeń arytmetycznych:

X + Y * 2 reprezentuje wartość wyrażenia x + 2y

X * Y/2 reprezentuje wartość wyrażenia $\frac{x \cdot y}{2}$

/ALA! + K%/ /KS! - 2 * /AL! - BL!/

Działania wykonywane są zgodnie z przyjętymi w matematyce priorytetami, stosowanie nawiasów powoduje w pierwszej kolejności wykonanie działań ujętych w nawiasy. W przypadku gdy w wyrażeniu występują dane różnych typów, wykonana jest odpowiednia konwersacja.

Operatory relacji i operatory logiczne wykorzystywane są głównie do porównania dwóch lub większej liczby wartości. Rezultatem takiego porównania jest wartość logiczna prawda /1/ lub fałsz /0/.

Wyrażenie

"ABC" < "ACD"

spowoduje badanie na relację mniejszości każdego znaku ciągu z lewej strony z każdym znakiem ciągu z prawej strony zgodnie z wartościami kodu ASCII, poczynając znak po znaku od lewej do prawej. Wynikiem tej relacji będzie prawda /1/.

Operator arytmetyczny + może być wykorzystany do wykonania operacji dodawanie dla danych numerycznych. Użycie znaku + dla danych alfanumerycznych powoduje konkatenację /połączenie/ ciągów znaków.

Przykład

Jeżeli pole TŻ ma wartość "TO JEST", a pole PRZŻ ma wartość "PRZYKŁAD", to operacja

T$ + PRZ$ da w wyniku ciąg "TO JEST PRZYKŁAD" natomiast operacja

T$ + "DOBRY" + PRZ$

da w wyniku ciąg znaków "TO JEST DOBRY PRZYKŁAD"

8.1.5. Instrukcja podstawiania

Jedną z najczęściej wykonywanych operacji w programach jest operacja podstawiania pod wartość określonej zmiennej wartość innej zmiennej lub wyrażenia. Operację taką piszemy w postaci

L E T zmienna wynikowa = wyrażenie lub zmienna

W szczególności wyrażenie może być stałą

Np.

```
L E T  A = B
L E T  A = ( B + C)/5
L E T  WYN$ = "WYNIK"
```

Słowo kluczowe L E T można pominąć pisząc instrukcję podstawiania zgodnie z zasadami zapisu matematycznej równości.

Wykonanie instrukcji podstawiania polega na podstawieniu w miejsce zmiennej występującej po lewej stronie znaku równości obliczonej wartości wyrażenia z prawej strony znaku równości lub wartości zmiennej z prawej strony znaku równości. Zarówno w trakcie obliczania wartości wyrażenia, jak i wykonywania operacji podstawiania wykonywane są niezbędne konwersje na postać pola wynikowego.

Przykładowo operacja

A% = 23.42

daje w wyniku wartość 23 w polu o nazwie A%, czyli wartość stałej występującej po prawej stronie znaku równości zamieniona została na postać zmiennej wynikowej, czyli na liczbę całkowitą. W trakcie konwersji wykonano jest zaokrąglenie. Dlatego też operacja

LET ZMIE% = 75.65

spowoduje, że w polu wynikowym ZMIE% otrzymamy wartość 76.

8.1.6. Instrukcja wejścia/wyjścia

Instrukcja wejścia/wyjścia można podzielić na dwie grupy:

- współpracy z klawiaturą i ekranem,

- współpracy ze zbiorami na dyskietce.

Dla wprowadzania danych z klawiatury służy instrukcja INPUT, która może przyjmować kilka postaci.

INPUT A

spowoduje wprowadzenie z klawiatury wartości i podstawienie jej pod zmienną o nazwie A. Przy czym wykonanie instrukcji INPUT realizowane jest w ten sposób, że realizacja programu jest zawieszana a na ekranie wypisywany jest znak ? i po nim należy wprowadzić określoną wielkość z klawiatury zakończoną znakiem nowej linii. Po wprowadzeniu wartości program jest dalej realizowany.

Jedną instrukcją INPUT można wprowadzać pojedyńczą zmienną lub ciąg zmiennych oddzielając je przecinkami

INPUT A, B, C

? 2, 3, 7

spowoduje wprowadzenie pod zmienną A wartości 2, i dalej odpowiednio

B = 3 a C = 7

Wprowadzane kolejno wartości należy oddzielać przecinkami.

Dla większej czytelności takiego sposobu wprowadzanie danych można przed wprowadzeniem zmiennej wydrukować tekst informujący o wprowadzeniu jakiej wartości chodzi w danej chwili.

Zapis instrukcji w postaci

INPUT "PODAJ DLUGOSC", N

spowoduje wydrukowanie tekstu zakończonego znakiem ? po którym wprowadzamy wartość dla zmiennej N.

Liczba wprowadzanych danych musi się dokładnie zgadzać z liczbą zmiennych występujących w operandzie instrukcji INPUT. Typy danych wprowadzanych muszą odpowiadać typom zmiennym z listy po instrukcji.

Dla wprowadzania informacji na monitor służy instrukcja PRINT,

która również może przyjmować kilka postaci.

Instrukcja PRINT bez operandów powoduje wydruk pustej linii.

Generalnie jedna instrukcja PRINT wyprowadza jedną linię wydruku, przy czym drukowane mogą być wartości zmiennych, teksty, jak i obliczone przed wydrukiem wartości wyrażeń.

Sekwencja instrukcji

10 A = 10

```
20 PRINT A + 5, A - 5, "WYNIK", 50/2
30 END
```

spowoduje wydruk w jednej linii

```
15  5  WYNIK  25
```

przy czym przy takiej postaci instrukcji wartości numeryczne są drukowane zawsze na 14-tu znakach. Używając jako separatora znaku średnika /;/ wydruk jest zagęszczany w ten sposób, że pola kolejne drukowane są bezpośrednio jedno po drugim.

Jeśli przecinek lub średnik wystąpi jako ostatni znak listy po instrukcji PRINT, to następna instrukcja PRINT spowoduje drukowanie w tej samej linii.

Program

```
 .
 .
 .
 50 PRINT "WPROWADZ WSPOLCZYNNIKI ROWNANIA KWADRATOWEGO"
 60 INPUT "WPROWADZ A, B, C"; A!; B!; C!
 70 PRINT "ROZWIAZANIE ROWNANIA";
 80 PRINT A!; "X2+"; B!; "x"; C!; "=0"
 .
 .   Instrukcje rozwiązujące równanie kwadratowe
 .
150 PRINT "WYNIKI:";
160 PRINT "X1 ="; X1; "X2 ="; X2!
```

Spowoduje następujący wydruk na monitorze

```
WPROWADZ WSPOŁCZYNNIKI ROWNANIA KWADRATOWEGO·
WPROWADZ A, B, C? 1, 1, - 2
ROZWIAZANIE ROWNANIA 1 X2 + 1 X - 2 = 0
WYNIKI: X1 = 1 X2 = - 2
```

A jaki wydruk spowoduje poniższy program?

```
10 PRINT "OBLICZANIE KWADRATU I SZESCIANU LICZBY"
20 INPUT "WPROWADZ LICZBE"; LICZ
30 PRINT "KWADRAT LICZBY; LICZ;
40 PRINT "WYNOSI -"; LICZ * LICZ
50 PRINT "SZESCIAN -"; LICZ ^ 3
```

Powyższy przykład pokazuje ponadto jak w instrukcji PRINT można przed wydrukiem ściągnąć wyliczenie wartości wyrażenia zapisanego w liście operandów.

Omówione postacie instrukcji PRINT powodują wyprowadzanie na monitor ekranowy. Wyprowadzenie danych na drukarkę dołączoną do

mikrokomputera można osiągnąć przy pomocy instrukcji LPRINT. Wszystkie zasady omówione przy instrukcji PRINT dotyczą również instrukcji LPRINT. Zakłada się, że długość wiersza drukarki wynosi 132 znaki.

Wymienione postacie instrukcji INPUT i PRINT służą do wprowadzania danych z klawiatury i wyprowadzania na ekran, wymienionych po instrukcji list zmiennych. Instrukcje te, wykorzystywane są również do wprowadzania i wyprowadzania listy zmiennych do i z zbioru na dyskietce. W takim przypadku po instrukcji, na początku ciągu zmiennych musi wystąpić nr zbioru poprzedzony znakiem #. Zbiór o określonym numerze musi być wcześniej otwarty przy pomocy instrukcji

OPEN

Instrukcja ta ma następującą postać

OPEN tryb, # nr zbioru, nazwa zbioru

gdzie:

- tryb może przyjmować wartości
. O dla zbioru sekwencyjnego wyjściowego
. I dla zbioru sekwencyjnego wejściowego
. R dla zbioru bezpośredniego wejściowo/wyjściowego
- nr zbioru jest liczbą z przedziału <1, 15> . liczba ta jest przypisana do zbioru tak długo, jak długo jest on otwarty i służy do identyfikacji zbioru w programie. Inne instrukcje WE/WY odnoszące się do tego zbioru poprzez tą liczbę identyfikują zbiór
- nazwa zbioru jest nazwą, pod którą zbiór jest identyfikowany przez system operacyjny, czyli jest to nazwa fizycznie zapisana na dyskietce.

Przykładowo

OPEN "I", # 2, "ZBIOR"

otwiera zbiór o nazwie ZBIOR z numerem 2 dla operacji wprowadzania.

Po skończonym przetwarzaniu zbioru dyskowego należy go zamknąć przy pomocy instrukcji CLOSE. W instrukcji zamknięcia podawany jest tylko numer zbioru, z którym zbiór był otwarty.

CLOSE # 2

powoduje zamknięcie zbioru otwartego instrukcją z poprzedniego przykładu.

Zamknięty zbiór może być ponownie otwarty w tym samym pro-

gramie z innym trybem otwarcia. Jeśli nie wystąpi instrukcja CLOSE do końca programu, to w momencie napotkania instrukcji końca END wszystkie niezamknięte zbiory z danego programu zostaną zamknięte.

Program współpracujący ze zbiorami na dyskietce poprzez instrukcje OPEN i CLOSE przekazują niezbędne informacje do systemu operacyjnego. Na podstawie informacji z tych instrukcji system operacyjny identyfikuje zbiór na dyskietce, wyszukuje go na nośniku sprawdzając poprawność zapisów, allokuje w wolnych obszarach dyskietki, itp.

Występujący w instrukcji OPEN tryb otwarcia zbioru decyduje o możliwych do wykonania na tym zbiorze operacji. Do zbiorów otwartych w trybie O możliwe są tylko operacje zapisywania informacji w zbiorze. Zbiór otwarty z tym trybem jest przez system zakładany na dyskietce o ile nie istnieje już zbiór o podanej nazwie. W przypadku gdy zbiór o podanej nazwie istnieje na dyskietce, dane są zapisywane w tym zbiorze niszcząc poprzednią jego zawartość.

Do zbiorów otwartych można odwoływać się przy pomocy poznanych już instrukcji INPUT i PRINT, w których przed listą zmiennych podany jest numer zbioru, którego dotyczy operacja

Instrukcje

```
70 INPUT # 1, A!, B$, RESZTA$
80 PRINT # 2, A!, B$, RESZTA$
```

spowodują przeczytanie trzech wymienionych zmiennych ze zbioru o numerze #1 i zapisanie ich do zbioru o numerze # 2.

Dane zapisywane w zbiorze na dyskietce przy pomocy instrukcji PRINT są zgodnie z zasadami omówionymi w odniesieniu do monitora. Instrukcja wprowadzania INPUT wymaga aby dane były oddzielone znakami przecinka lub CR, w związku z tym znana postać instrukcji PRINT nie może być w prosty sposób użyta jeśli chcemy te dane następnie czytać instrukcją INPUT.

Instrukcja PRINT powinna wstawiać wymagane separatory jako dane do zbioru. Wiersz 80 z poprzedniego przykładu powinien mieć postać

```
80 PRINT#2, A!; ","; B$; ","; RESZTA$
```

Taka postać instrukcji umieści w zbiorze na dyskietce trzy dane oddzielone znakami przecinka, co pozwoli na przeczytanie tych danych przy pomocy instrukcji INPUT.

Omówiony sposób zapisu danych jest kłopotliwy przy większej liczbie danych zapisywanych przy pomocy jednej instrukcji w takich sytuacjach należy korzystać z instrukcji

WRITE

która zapisuje dane do zbioru dyskowego oddzielając zapisywane dane przecinkami. Zasady zapisu pola operand w instrukcji WRITE są takie same jak w instrukcji PRINT.

Kontynuując omawiany przykład zapisu trzech danych, wiersz 80 przyjmuje postać

80 WRITE #2, A!, B$, RESZTA$

Instrukcja WRITE dotyczy tylko zbiorów dyskowych o dostępie sekwencyjnym.

Język programowania BASIC umożliwia również przetwarzanie zbiorów o dostępie bezpośrednim. Zbiory takie powinny być otwarte z trybem "R". Przetwarzanie tych zbiorów wspomagane jest instrukcjami

FIELD
LSET, RSET
GET
PUT

Instrukcja FIELD pozwala na określenie struktury rekordu zbioru o dostępie bezpośrednim. W polu operand tej instrukcji oprócz numeru zbioru podaje się rozmiar i nazwy pól w rekordzie pooddzielane przecinkami.

Opis jednego pola składa się z

n AS nazwa

gdzie

n - określa długość pola w znakach
AS - słowo kluczowe
nazwa - nazwa pola alfanumerycznego

W zbiorze o dostępie bezpośrednim przechowywane są w zasadzie tylko dane alfanumeryczne. Dane numeryczne mogą być pamiętane w zbiorze o dostępie bezpośrednim tylko i wyłącznie w postaci alfanumerycznej. Przy pomocy funkcji wbudowanych /patrz pkt. 9/ MKI$, MKS$, MKD$ można dokonać konwersji wartości numerycznych na postać znakową i odwrotnie przy pomocy funkcji CVI, CVS, CVD można dokonać konwersji ciągu znaków na postać numeryczną.

Instrukcja FIELD opisuje jedynie bufor zbioru o dostępie

bezpośrednim ale nie wypełnia go danymi

Zapis

FIELD #1, 10 AS POL$, 15 AS POL2$, 8 AS L$

opisuje rekord składający się z trzech pól, odpowiednio pola POL$ - 10 znaków, pola POL2$ - 15 znaków i pola L$ - 8 znaków.

Dla tego samego zbioru może być wykonana dowolna liczba instrukcji FIELD. Obowiązuje aktualnie aktywna instrukcja, czyli wykonanie następnej instrukcji FIELD anuluje poprzednią.

Należy pamiętać o ograniczeniach 128 znaków na długość rekordu, czyli suma znaków opisywanych pól w jednej instrukcji FIELD nie może tej granicy przekroczyć. Również nazwy zmiennych wymienionych w instrukcji FIELD nie mogą być używane w instrukcjach INPUT i LET.

Dla wypełnienia poszczególnych pól w buforze opisanym instrukcją FIELD, przed operacją wyprowadzania, służą instrukcje

LSET

RSET

Instrukcje te służą do umieszczenia danych z pamięci w buforze zbioru dyskowego o dostępie bezpośrednim. Instrukcja LSET rozpoczyna umieszczanie kolejnych znaków w polu bufora poczynając od lewej strony. Jeśli pole źródłowe jest krótsze od pola wynikowego następuje uzupełnienie spacjami z prawej strony. Gdy pole źródłowe jest dłuższe od pola wynikowego, gubione są znaki wykraczające poza rozmiar pola wynikowego z prawej strony.

Dla instrukcji RSET operacja dosuwania jest realizowana prawostronnie a ewentualne obcięcie realizowane jest z lewej strony pola za długiego.

.
.
.

```
50 FIELD #1, 10 AS P1$, 40 AS P2$, 50 AS P3$
60 INPUT "PODAJ ZAWOD",TEK$
70 LSET P1$=TEK$
80 INPUT "PODAJ IMIE I NAZWISKO",TEK$
90 LSET P2$=TEK$
100 INPUT "PODAJ ADRES",TEK$
110 LSET P3$=TEK$
```

Powyższy fragment programu wypełnia bufor zbioru o dostępie bezpośrednim otwartym z numerem 1. Kolejne instrukcje INPUT pozwalają na wprowadzenie danych a instrukcje LSET umieszczają

wprowadzone dane w odpowiednie pola bufora.

Zapis tak przygotowanego rekordu na dyskietkę realizowany jest przy pomocy instrukcji

PUT nr zbioru, numer rekordu

gdzie

nr zbioru - jest numerem zbioru nadanym mu w operacji otwarcia instrukcji OPEN

nr rekordu - jest numerem rekordu, pod którym rekord zostanie zapamiętany w zbiorze na dyskietce. Numer ten jest kluczem dostępu do rekordu w zbiorze. Maksymalna wartość numeru rekordu wynosi 32 767.

Instrukcja

```
120 PUT # 1, K%
```

spowoduje zapisanie rekordu utworzonego w przykładzie wyżej z numerem rekordu równym aktualnej wartości zmiennej K%.

Przetwarzając zbiory o dostępie bezpośrednim należy przestrzegać zasady, że każdy rekord posiada unikalny numer.

Czytanie rekordu ze zbioru o dostępie bezpośrednim z dyskietki do bufora opisanego przy pomocy instrukcji FIELD realizuje instrukcja

GET nr zbioru, nr rekordu

gdzie

nr zbioru - jest numerem zbioru nadanym mu w operacji otwarcia instrukcją OPEN

nr rekordu - jest numerem rekordu z jakim rekord został zapisany w zbiorze instrukcją PUT

Po wykonaniu się instrukcji GET bufor zbioru opisany przy pomocy instrukcji FIELD jest wypełniony danymi. W programie można używać nazw zmiennych z instrukcji FIELD.

Przeczytanie i wydrukowanie zawartości rekordu utworzonego w poprzednim przykładzie można zrealizować w następujący sposób:

```
 50 FIELD # 2, 10 AS ZAW$, 40 AS IM$, 50 AS AD$
 60 INPUT "PODAJ NUMER REKORDU", NUM%.
 70 GET # 2, NUM %
 80 PRINT "ZAWOD":; ZAW$
 90 PRINT "IMIE I NAZWISKO:"; IM$
100 PRINT "ADRES:"; AD$
```

Przetwarzanie zbiorów o dostępie bezpośrednim wymaga od użytkownika nieco doświadczeń w przetwarzaniu zbiorów dyskowych.

Zbiory te mogą być również przetwarzane sekwencyjnie i wówczas w instrukcjach GET i PUT nie podaje się numeru rekordu.

W programie napisanym w BASIC-u istnieje możliwość zdefiniowania ciągu wartości, który to ciąg może być czytany przy pomocy instrukcji READ. Dla zdefiniowania ciągu wartości służy instrukcja

DATA ciąg wartości

Wartości ciągu oddzielone są przecinkami o kolejne wystąpienie instrukcji DATA spowoduje dopisanie nowych wartości na koniec ciągu istniejącego.

Na przykład, wystąpienie dwóch instrukcji

```
10 DATA 1, 2, 3
20 DATA 4, 5
```

jest równoważne wystąpieniu jednej instrukcji

DATA 1, 2, 3, 4, 5

Pobieranie kolejnych wartości z ciągu realizowane jest przy pomocy instrukcji

READ nazwa zmiennej

Na początku wykonywania programu wartość wskaźnika ciągu ustawiana jest na 1. Pierwsze wystąpienie instrukcji READ spowoduje nadanie pierwszej wartości z ciągu dla pierwszej zmiennej występującej po READ oraz zwiększenie wskaźnika ciągu o 1.

Zapis

READ A, B

spowoduje dla wymienionych wcześniej deklaracji DATA nadanie wartości 1 dla A i 2 dla B. Następne wystąpienie

READ C

spowoduje nadanie wartości 3 dla C.

Próba czytania elementu gdy ciąg został wyczerpany spowoduje sygnalizację błędu.

Powrót ze wskaźnikiem ciągu na jego początek to znaczy do wartości 1, jest możliwy przez użycie instrukcji

RESTORE

Następna po RESTORE instrukcja READ spowoduje przeczytanie pierwszej wartości z pierwszej w programie deklaracji DATA.

Instrukcja DATA i READ mogą być wykorzystane w programie BASIC-u dla przechowania wektora, którego składowe są indeksowane od 1 do n, gdzie n jest ilością danych we wszystkich występujących w programie wierszy DATA. W wektorze takim można przecho-

wywać wielkości stałe, jak na przykład współczynniki wielomianu, którego wartość liczy program. Praktyczne wykorzystanie wektora stałych obrazuje poniższy przykład programu na obliczanie wartości wielomianu n-tego stopnia.

```
10 INPUT "PODAJ STOPIEN WIELOMIANU",N
20 INPUT "PODAJ WARTOSC X0",X0
30 RESTORE
40 LET W=0
50 LET I=0
60 READ A
70 W=W*X0+A
80 I=I+1
90 IF I<N+1 THEN GO TO 60
100 PRINT "WARTOSC DLA X0 = ";X0;" WYNOSI ";W
110 DATA 1,0,0,1,-2,1,5
120 END
```

W wierszu 10 podajemy stopień wielomianu, którego współczynniki są zapamiętane w wierszu 110. Zamieszczony przykład wiersza DATA reprezentuje równanie

$$x^6 + x^3 - 2x^2 + x + 5$$

Chcąc policzyć wartość innego wielomianu tym samym programem należy wprowadzić tylko nowe wartości w instrukcji DATA.

W wierszu 20 podajemy dla jakiej wartości x ma być obliczona wartość wielomianu.

Wiersze 40, 50, 70 i 80 występują operacje podstawienia. Przykład pokazuje wystąpienie słowa kluczowego LET lub pominięcie go.

Wiersz 90 zawiera instrukcje wyboru, która zostanie omówiona w dalszej części.

8.1.7. Instrukcja wyboru

Dla realizacji sytuacji decyzyjnych występujących w programie służy instrukcja

IF war THEN instrukcja - 1 ELSE instrukcja - 2

Przy czym opcja ELSE jest opcjonalna i jeśli ją pominiemy to realizowany jest tzw. wybór prosty czyli wykonywane są instrukcje występujące bezpośrednio po instrukcji IF. Zapis instrukcji IF musi zawrzeć się w jednym wierszu w całości. Jeśli występuje opcja ELSE to musi ona w całości być opisana w danym wierszu.

Instrukcję IF można używać w postaci wyboru pełnego gdy dla

prawdziwego warunku wykonuje się jedną lub kilka instrukcji i dla nieprawdziwego warunku wykonuje się również jedną lub kilka instrukcji.

Przykładowo instrukcja

IF A < B THEN PRINT "MNIEJSZY" ELSE PRINT "WIEKSZY"

jest typowym przykładem gdy w zależności od relacji A < B chcemy wydrukować tekst "MNIEJSZY" lub "WIEKSZY".

W szczególności instrukcję IF można zagnieżdżać w sobie co ilustruje przykład

IF A=B THEN B=C THEN PRINT "A-C" ELSE PRINT "A < > C"

Najczęściej jednak, z uwagi na ułomność instrukcji IF, występuje ona w połączeniu z instrukcją skoku bezwarunkowego. Taka postać instrukcji wykorzystywana jest wówczas gdy w zależności od wartości warunku chcemy wykonać nie jedną instrukcję ale dowolnie długi ich ciąg.

Najczęściej występująca struktura to np.

```
 50  IF  A = 5  THEN  GO  TO  100
 60  . . . .
       .                    operacje wykonywane przy niespełnionym
       .                    warunku tzn. A ≠ 5
       .
 90  GO  TO  150
100  . . .
       .                    operacje wykonywane przy spełnionym
       .                    warunku tzn. A = 5
       .
150  . . .
```

Oczywiście ciągi instrukcji realizowane przy spełnionym i niespełnionym warunku mogą być dowolnie długie, jak również warunek może stanowić dowolne złożenie różnych wyrażeń połączonych znakami relacji i operatorami logicznymi.

Szczególną uwagę należy zwracać na sytuację gdy zanurzamy instrukcje IF jedna w drugiej, aby organizowany przez nas punkt wyjścia z takiej konstrukcji był zgodny z logiką zapisanych warunków.

Przeanalizujemy wykorzystanie instrukcji wyboru w programie rozwiązującym równanie kwadratowe

```
10 INPUT "WPROWADZ WSPOLCZYNNIKI",A,B,C
20 IF A=0 THEN GO TO 110
30 DELTA=B*B-4*A*C
40 IF DELTA =0 THEN GO TO 70
50 PRINT "BRAK ROZWIAZAN RZECZYWISTYCH"
60 END
70 PDEL=SQR(DELTA)
80 PRINT "X1 = ";-B
89 PRINT "X1 = ";(-B-PDEL)/2*A
90 PRINT "X2 = ";(-B+PDEL)/2*A
100 END
110 PRINT "X = ";-C/B
120 END
```

W wierszach 20 i 40 występują instrukcje wyboru, które w zależności od wartości współczynnika A i wartości delty dokonują wyboru odpowiedniego algorytmu obliczeniowego. W wierszu 70 wykorzystano funkcje obliczającą wartość pierwiastka kwadratowego, szczegóły związane z wykorzystaniem funkcji zostaną omówione w dalszej części.

W przykładzie programu występują instrukcje końca programu na zakończenie każdego wybranego instrukcjami IF algorytmu obliczenia pierwiastków równania. Ponieważ dla konkretnych danych wybrany zostanie konkretny jeden algorytm obliczeniowy to również jedna instrukcja końca programu będzie wykonana.

8.1.8. Instrukcja powtarzania

Oprócz struktur sekwencji i wyboru, które zostały już omówione, dla konstrukcji programu niezbędna jest struktura umożliwiająca powtarzanie ciągu instrukcji. Powtarzanie może być realizowane dopóty, dopóki spełniony jest opisany warunek, lub też określamy liczbowo krotność powtórzeń ciągu instrukcji.

W Basicu można programować cykle programowe wykorzystując do tego celu instrukcje

FOR POCZ = A TO KON STEP B

gdzie

POCZ = A oznacza, że zmienna POCZ przyjmuje wartość początkową A i ciąg instrukcji występujący dalej realizowany jest aż do osiągnięcia wartości KON przy zmianie po każdym cyklu wartości POCZ o wartości B. Gdy pominiemy opcję STEP to krok wynosi 1.

Ostatnią instrukcją powtarzanego ciągu jest instrukcja

NEXT zmienna

Jest to typowa pętla sterowana liczbowo, która może być zanurzona jedna w drugiej. Zanurzając pętlę w pętli należy pamiętać, że nie jest dozwolone przecinanie się pętli tzn. pętla wewnętrzna musi się kończyć przed końcem pętli zewnętrznej.

Na przykład

```
30 FOR I = 1 TO N
40 FOR J = 1 TO M
50 READ TAB /I, J/
60 NEXT J
70 NEXT I
```

Spowoduje wypełnienie macierzy kolejnymi wartościami z ciągu DATA. Wypełnianie będzie realizowane kolumnami.

Jest oczywiste, że wartość pola KON powinna być większa od pola A, o ile wartość kroku B jest dodatnia. Napotkanie instrukcji NEXT ze zmienną sterującą taką samą, jak w instrukcji FOR powoduje jej zwiększenie o wartość kroku i porównanie z wartością końcową. Gdy wartość końcowa nie jest osiągnięta to wykonują się ponownie instrukcje poczynając od następnej instrukcji po FOR. W przeciwnym wypadku /gdy wartość końcowa jest osiągnięta lub przekroczona/ wykonują się dalsze instrukcje programu poczynając od instrukcji następnej po NEXT.

W szczególności krok może mieć wartość ujemną i wówczas relacje między wartością początkową i końcową muszą być odwrotne gdyż następuje zmniejszanie wartości początkowej o wartość kroku.

Wypełnienie macierzy tymi samymi danymi co w poprzednim przykładzie ale poczynając od elementu TAB nm do elementu TAB_{11}

```
30 FOR I = N TO 1 STEP -1
40 FOR J = M TO 1 STEP -1
50 READ TAB /I, J/
60 NEXT J
70 NEXT I
```

Oczywiście, w wyniku otrzymamy inne wartości macierzy gdyż pobieranie elementów z wektora DATA w obu przypadkach jest od początku do końca.

Sterowanie ilością powtórzeń, ciągu instrukcji może być realizowane przy pomocy odpowiednio zdefiniowanego warunku. Do tego typu pętli używa się instrukcji

```
WHILE  warunek
   .
   .
   .
zdania pętli
   .
   .
   .
WEND
```

Zdania pętli realizowane są tak długo jak długo warunek zdefiniowany po WHILE jest prawdziwy. Widać stąd wyraźnie, że zdania pętli winny wpływać na jego wartość o ile pętla ma być skończona.

```
K = 1
WHILE  K
   .
   .
   .
IF  A < B  THEN  K = 0
   .
   .
   .
WEND
```

Dla zobrazowania wykorzystania obu typów pętli oraz wzajemnego ich zanurzania przeanalizujemy następujący przykład.

```
10 DIM A(10)
20 FOR I=1 TO 10
30 INPUT A(I)
40 NEXT I
50 K=1
60 WHILE K
70 K=0
80 FOR I=1 TO 9
90 IF A(I)<=A(I+1) THEN GO TO 140
100 ROB=A(I)
110 A(I)=A(I+1)
120 A(I+1)=ROB
130 K=1
140 NEXT I
150 WEND
```

Ten fragment programu w wierszach 20 - 40 wprowadza do tablicy A kolejne wartości. Pozostałe wiersze mają za zadanie uporządkowanie wartości tablicy A według wzrastających wartości. Realizowane to jest metodą porównywania kolejnych elementów i ewentualnego przestawienia ich miejscami zgodnie z wymaganym po-

rządkiem. Operacje te powtarzane są tak długo dopóki nie nastąpi żadna operacja przestawiania elementów tablicy, czyli został osiągnięty wymagany porządek elementów tablicy.

8.1.9. Inne możliwości języka

Oprócz wymienionych podstawowych struktur programowania występują w BASIC-u również inne znane w językach wyższego rzędu możliwości i rozwiązania.

Można stosować funkcje standardowe wywołując je poprzez nazwę z odpowiednio zdefiniowanymi parametrami. W przykładach obrazujących funkcjonowanie innych instrukcji, używaliśmy niektóre z tych funkcji. Funkcje te wywołuje się poprzez nazwę, która może wystąpić samodzielnie lub jako element wyrażenia.
Np.

A = SIN (X)

Nazwa funkcji wraz z ujętymi w nawiasy wartościami wywołania oznacza wartość wywołanej funkcji w punkcie wywołania. W powyższym przykładzie A przyjmuje wartość sinusa w punkcie X.

Występujące w programie BASIC-u funkcje można podzielić na:

- arytmetyczne
- znakowe
- specjalne

Funkcje arytmetyczne to:

ABS - obliczenie wartości bezwzględnej wyrażenia
ATN - obliczenie wartości arcus tangens
SIN - obliczenie wartości sinusa wyrażone w radianach
COS - obliczenie wartości cosinusa
TAN - obliczenie wartości tangesa
LOG - obliczenie wartości logarytmu naturalnego
EXP - obliczenie wartości funkcji e do potęgi x
SQR - obliczenie wartości pierwiastka kwadratowego
INT - przyjmuje największą liczbę całkowitą < = X
FIX - oblicza część całkowitą
SGN - wartość 0 dla X = 0, 1 dla pozostałych X
CINT - dokonuje konwersji wyrażenia do postaci całkowitej
CSNG - dokonuje konwersji wyrażenia do pojedyńczej precyzji
CDBL - dokonuje konwersji wyrażenia do podwójnej precyzji

Funkcje znakowe to:

LEFT$ - pozwala na wycięcie z ciągu znaków dowolnego podciągu poczynając od lewej strony

RIGHT$ - analogicznie jak LEFT$ tylko od prawej strony

MID$ - pozwala na wycięcie z ciągu znaków dowolnego podciągu

CHR$ - podaje znak zgodnie z podanym kodem dziesiętnym ASCII

LEN - podaje długość ciągu w znakach

STRING$ - przyjmuje wartość ciągu znaków o zadanej długości, zgodnie z podanym kodem ASCII

Funkcje specjalne to:

TAB - przesuwa kursor do I - tej kolumny w wierszu

POS - podaje położenie kursora w wierszu

LPOS - analogicznie jak POS dla drukarki

SPC - powoduje wydrukowanie I spacji

SPACE$ - wartością funkcji jest ciąg spacji

EOF - przyjmuje wartość 1 gdy przy czytaniu zbioru dyskowego zostanie napotkany jego koniec

CVI - realizuje konwersję dwu-bajtowego ciągu znaków na liczbę typu całkowitego

CVS - realizuje konwersję cztero-bajtowego ciągu znaków na liczbę pojedyńczej precyzji

CVD - realizuje konwersję ośmio-bajtowego ciągu znaków na liczbę podwójnej precyzji

MKI$ - realizuje konwersję liczby całkowitej na postać 2-bajtowego ciągu znaków

MKS$ - realizuje konwersję liczby pojedyńczej precyzji na postać 4-bajtowego ciągu znaków

MKD$ - realizuje konwersję liczby podwójnej precyzji na postać 8-bajtowego ciągu znaków

Wymienione zostały tylko niektóre funkcje mające największe zastosowanie w praktyce programowania. Inne funkcje dostępne w BASIC-u mają zastosowanie przy programowaniu bardziej złożonych procedur.

Procedura wielokrotnie wykorzystywana w ramach danego programu jest najczęściej programowana w postaci podprogramu. Podprogram w języku BASIC nie ma specjalnej struktury, tzn. zaczyna się od dowolnej instrukcji oznaczonej numerem. Numer ten jest identyfikatorem, którego używamy w instrukcji wywołania podprogramu.

Dla wywołania podprogramu wykorzystujemy instrukcję

GOSUB nr wiersza

Sterowanie przekazywane jest do wiersza, którego numer wskazuje instrukcja GOSUB i podprogram wykonuje się do napotkania instrukcji

RETURN

która powoduje powrót do następnej po GOSUB instrukcji programu. Podprogram można wykorzystywać dowolną ilość razy w ramach programu. Podprogram może występować w dowolnym miejscu, tylko należy uważać aby sterowanie nie było mu przekazane pomyłkowo na przykład poprzez sekwencyjne przekazywanie sterowania. Aby zapobiec niezamierzonemu wejściu do podprogramu umieszcza się go po instrukcjach GO TO, STOP, END i innych zatrzymujących funkcjonowanie programu lub przekazujących sterowanie.

Poniższy przykład ilustruje sposób na wyprowadzania dowolnej liczby umieszczonej w zmiennej A poprzedzonej dwoma znakami gwiazdki i zakończonej dwoma znakami gwiazdki.

```
    .
    .
    .
 70 A = WAT
 80 GOSUB 500
    .
    .
    .
130 A = 15
140 GO SUB 500
    .
    .
    .
500 PRINT "* *"; A; "* *"
510 RETURN
```

Istnieje również możliwość wywoływania podprogramów zewnętrznych, ale jest to bardziej złożony zabieg wymagający dodatkowych deklaracji.

BASIC oprócz błędów formalnych sygnalizuje również błędy wykonania programu wadliwie funkcjonującego wypisując odpowiedni dla błędu komunikat i zawieszając wykonanie programu. Istnieje możliwość programowej obsługi błędu w przypadku gdy obsługa systemowa niezadowala użytkownika.

Instrukcja

ERR i ERL

pozwalają ponadto na otrzymanie informacji o kodzie błędu, który spowodował przerwanie prawidłowej sekwencji programu, oraz numerze linii, w której błąd został wykryty.

Dla zwiększenia przejrzystości kodu źródłowego programu można tekst programu uzupełniać wierszami komentarza. Wiersze komentarza piszemy w programie przy pomocy instrukcji

REM ciąg znaków

Ponadto BASIC dostarcza zbiór instrukcji sterujących, ułatwiających pisanie programu, jego edycję, uruchomienie i testowanie. Są to takie instrukcje jak:

NEW - instrukcja usuwa program z pamięci, czyli czyści pamięć przed wprowadzeniem nowego programu

RUN - umożliwia uruchomienie programu wcześniej przygotowanego. Uruchomienie może nastąpić od początku lub dowolnego numeru linii.

AUTO - służy do automatycznego generowania numerów wierszy programu po każdym wciśnięciu przycisku CR. Parametry pozwalają na podanie wartości początkowej oraz wartość przyrostu.

LIST - służy do wylistowania całego programu lub jego części określonej numerami linii

LLIST - wykonuje funkcje analogiczną do LIST na drukarkę

DELETE - pozwala na kasowanie jednej linii lub całego fragmentu programu

EDIT - instrukcja ta inicjuje pracę w trybie edytora, który posiada podobne funkcje jak program edytora opisany wcześniej.

RENUM - służy do zmiany numerów wierszy programu. W trakcie przenumerowywania zmieniane są również numery wierszy użyte jawnie jako operandy instrukcji programu.

TRON - instrukcje służące do aktywowania i dezaktywizowania funkcji śledzenia wykonywania poszczególnych instrukcji programu

CONT - instrukcja pozwalająca na wznowienie wykonania programu wcześniej zawieszonego.

8.2. Uruchamianie programów w BASIC-u

Uruchamianie programu w BASIC-u może być zrealizowane na dwa sposoby:

- z wykorzystaniem kompilatora języka BASIC
- z wykorzystaniem interpretera języka BASIC

Pierwszy sposób wykorzystywany jest wówczas gdy chcemy przygotować program w postaci wykonywalnej do eksploatacji użytkowej. Postać programu jest ostateczna i nie może być łatwo modyfikowana.

Program w pierwszym kroku tłumaczony jest na postać assemblerową i w postaci półskompilowanej zapisywany w zbiorze z rozszerzeniem .REL. Ponadto tworzony jest zbiór z rozszerzeniem .PRN, który zawiera wydruk z procesu translacji czyli program zapisany w kodzie assemblera. W drugim kroku należy użyć programu łączącego, który wykorzystując postać programu ze zbioru z rozszerzeniem .REL, wykona fazę łączenia i tworzy postać wykonywalną w zbiorze z rozszerzeniem .COM.

8.2.1. Interpreter języka BASIC

Interpreter języka programowania jest specjalnym sposobem wykonania programu napisanego w języku wyższego rzędu. Program napisany w języku wyższego rzędu przed wykonaniem, musi być zamieniony na postać języka maszynowego. Wykonują to specjalne programy zwane kompilatorami, które najczęściej operują na całych programach poddawanych procesowi kompilacji lub na nieco mniejszych częściach, w zależności od konkretnego języka programowania. W wyniku przebiegu kompilacji powstają inne, niż źródłowa, postacie programu, które zapamiętywane są w zbiorach na nośnikach magnetycznych. Taki sposób uruchamiania programów w BASIC-u omówimy w dalszej części.

Interpreter języka za jednostkę bierze każdą kolejną instrukcję języka źródłowego. Każda instrukcja jest tłumaczona na kod maszynowy i wykonywana. Interpreter nie tworzy żadnych postaci programu, które byłyby przechowywane i możliwe do wykorzystania w terminie późniejszym.

Interpreter języka BASIC wykorzystywany jest głównie na etapie testowania programu. Interpreter pobiera kolejne instrukcje

programu i każdą z nich interpretując, wykonuje. Chcąc program wykonać ponownie należy znów wywołać interpreter lub skorzystać z komendy RUN z adresem początkowym.

Istnieją dwa tryby pracy programu pod kontrolą interpretera. Jeden polega na tym, że piszemy cały program instrukcja po instrukcji z klawiatury. Po napisaniu całego programu możemy go wykonać używając komendy RUN. Program źródłowy znajduje się w pamięci mikrokomputera i użycie instrukcji NEW spowoduje jego nieodwracalne zniszczenie, podobnie jak każda operacja zapisu w pamięci operacyjnej powoduje jego zmianę ze zniszczeniem włącznie. Jeśli użytkownik chciałby zachować napisaną postać programu źródłowego, wówczas powinien użyć instrukcji

SAVE nazwa zbioru

Instrukcja ta służy do zapisania programu na dysku w zbiorze, którego nazwa umieszczona jest w operandzie instrukcji. Jeżeli zbiór o takiej nazwie już istnieje to wyprowadzany program zostanie wpisany w jego miejsce. W przeciwnym przypadku zbiór zostanie założony i domyślnie otrzyma rozszerzenie .BAS. /Rozszerzenie przyjmowane jest domyślnie również dla zbioru istniejącego/.

Interpreter języka BASIC w pierwszym trybie pracy wywołuje się przy pomocy wypisania na klawiaturze

MBASIC,

co spowoduje wywołanie interpretera z gotowością do pisania programu z klawiatury. Po skompletowaniu instrukcji całego programu, można go uruchomić przy pomocy komendy

RUN nr wiersza,

gdzie nr wiersza wskazuje numer dowolnej instrukcji do wykonania, która będzie pierwszą wykonywaną lub bez numeru co spowoduje wykonanie od pierwszej, zgodnie z numeracją, instrukcji programu. Drugi tryb pracy można uzyskać poprzez wypisanie na klawiaturze

MBASIC nazwa zbioru

co powoduje wyszukanie zbioru o podanej nazwie i przyjętym przez domniemanie rozszerzeniu .BAS i wykonanie jego zawartości jako kompletny program w BASIC-u.

Wyjście z interpretera osiąga się przez napisanie na klawiaturze słowa SYSTEM.

Pierwszy z załączonych przykładów obrazuje nam napisanie i

uruchomienie programu na obliczanie wartości wielomianu /por. pkt. 8.1.6/.

```
10 INPUT "PODAJ STOPIEN WIELOMIANU",N
20 INPUT "PODAJ WARTOSC X0",X0
30 RESTORE
40 LET W=0
50 LET I=0
60 READ A
70 W=W*X0+A
80 I=I+1
90 IF I<N+1 THEN GO TO 60
100 PRINT "WARTOSC DLA X0 = ";X0;" WYNOSI ";W
110 DATA 1,0,0,1,-2,1,5
120 END
Z
A)
A>RUN
A>PODAJ STOPIEN WIELOMIANU? 5

WARTOSC X0? 0

WARTOSC DLA X0 = 0 WYNOSI 5
```

Drugi przykład obrazuje wykonanie programu na wprowadzenie i uporządkowanie tablicy, w drugim z omawianych trybów współpracy z interpreterem. Program zamieszczony na końcu pkt. 8.1.8 został uzupełniony o instrukcje

```
142  PRINT
143  FOR  J = 1  TO  10
145  PRINT  A (I);
148  NEXT  J
```

pozwalające śledzić kolejne kroki porządkowania tablicy.

```
10 DIM A(10)
20 FOR I=1 TO 10
30 INPUT A(I)
40 NEXT I
50 K=1
60 WHILE K
70 K=0
80 FOR I=1 TO 9
90 IF A(I)=A(I+1) THE  GO TO 140
100 ROB=A(I)
110 A(I)=A(I+1)
120 A(I+1)=ROB
130 K=1
140 NEXT I
142 PRINT
143 FOR J=1 TO 10
145 PRINT A(J)
148 NEXT J
150 WEND
160 END
```

```
22,44,-78,1,0,67,98,99,11,100
22,-78,1,0,44,67,98,11,99,100
-78,1,0,22,44,67,11,98,99,100
- 8,0,1,22,44,11,67,98,99,100
-78,0,1,22,11,44,67,98,99,100
-78,0,1,11,22,44,67,98,99,100
-78,0,1,11,22,44,67,98,99,100
```

W operandzie instrukcji SAVE można jeszcze wybrać sposób zapamiętania programu na dyskietce. Program może być przechowywany w kodzie ASCII lub w postaci binarnej.

Dla załadowania, tak zapamiętanego programu ponownie do pamięci, służy instrukcja

LOAD nazwa zbioru

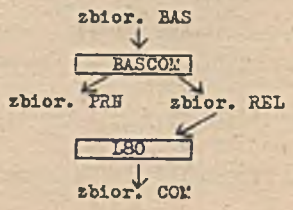
Instrukcja ta wprowadza do pamięci program ze zbioru dyskowego, w którym program był zapamiętany instrukcją SAVE. Podając po nazwie parametr R oddzielony przecinkiem, program po załadowaniu jest uruchamiany.

Drugi tryb pracy pod kontrolą interpretera, polega na wcześniejszym przygotowaniu programu i zapisaniu go w zbiorze na dysku z rozszerzeniem .BAS. Zbiór taki może być przygotowany przy pomocy programu edytora. Wywołanie interpretera z nazwą zbioru, w którym zapisany jest kompletny program powoduje umieszczenie go w pamięci i uruchomienie od pierwszej instrukcji.

Zalecany jest drugi tryb pracy, który pozwala na wykorzystanie wszystkich możliwości edytorskich programu edytor oraz, co jest bardzo istotne, chroni użytkownika przed nieumyślnym zniszczeniem tekstu programu źródłowego.

8.2.2. Kompilacja programu w BASIC-u

W odróżnieniu od pracy z wykorzystaniem interpretera, praca z kompilatorem koncentruje się głównie na tworzeniu zbiorów na dysku, które mogą być wykorzystane po przebiegu kompilacji. Przebieg procesu kompilacji obrazuje rysunek



Proces realizowany jest w dwóch fazach. Pierwsza, tłumaczy program źródłowy w języku BASIC na kod maszynowy, tworząc tak zwaną postać półskompilowaną, która jest wejściem do fazy drugiej, łączenia. Faza łączenia daje wynikową postać programu, która może być wykonywana pod kontrolą systemu operacyjnego CP/M poprzez wypisanie nazwy na klawiaturze, podobnie jak komendy systemu.

Wejściem do procesu kompilacji jest zapisany w zbiorze na dysku kompletny program w języku BASIC. Zbiór ten musi mieć rozszerzenie .BAS. Zbiór taki można utworzyć korzystając z programu edytora. W wyniku przebiegu powstają dwa zbiory. Jeden z rozszerzeniem .PRN zawiera wydruk przebiegu kompilacji, w szczególności postać programu w kodzie assemblera dla każdej instrukcji źródłowej BASIC-u /por. załącznik/. Drugi zbiór z rozszerzeniem .REL zawiera postać półskompilowaną, która jest zbiorem wejściowym do fazy łączenia, która daje w wyniku zbiór z rozszerzeniem .COM. Postać wynikowa, zawiera program ładowalny przy pomocy wypisania nazwy zbioru /bez rozszerzenia/ na klawiaturze, co powoduje ładowanie go do pamięci i wykonanie.

Nazwy zbiorów mogą być budowane zgodnie z ogólnie przyjętymi zasadami w systemie CP/M.

Ogólna postać komendy kompilacji jest następująca:

BASCOM nazwa programu - 1, nazwa programu - 2 = nazwa programu - 3

gdzie

nazwa programu - 1 - określa nazwę tworzonego zbioru zawierającego postać półskompilowaną. Zbiór ten otrzyma rozszerzenie .REL

nazwa programu - 2 - określa nazwę tworzonego zbioru zawierającego listing procesu kompilacji. Zbiór ten otrzyma rozszerzenie .PRN

nazwa programu - 3 - określa nazwę istniejącego zbioru zawierającego program źródłowy do kompilacji. Zbiór ten musi mieć rozszerzenie .BAS

Np.

BASCOM PROG, PROG = PROG

oznacza, że zbiór o pełnej nazwie PROG.BAS zostanie wyszukany i wzięty jako wejściowy do procesu kompilacji. W wyniku kompilacji zostaną utworzone dwa zbiory. Jeden o pełnej nazwie PROG.REL,

zawierający postać półskompilowaną a drugi o pełnej nazwie PROG.PRN zawierający listing kompilacji.

Przedstawiony w przykładzie sposób, nadawania tej samej nazwy dla wszystkich zbiorów biorących udział w procesie kompilacji jest bardzo praktyczny. Pozwala to na prosty sposób identyfikacji a rozszerzenie określa zarazem ich zawartość. Po poprawnym przebiegu zbiory z rozszerzeniami .REL i .PRN są najczęściej kasowane.

Następny krok, łączenie, wykonujemy przy pomocy programu L80 w sposób następujący:

L80 nazwa prog-1, nazwa prog-2/parametry

gdzie

nazwa prog-1 - określa nazwę istniejącego zbioru zawierającego postać półskompilowaną. Zbiór ten musi mieć rozszerzenie .REL

nazwa prog-2 - określa nazwę tworzonego zbioru zawierającego postać wykonywalną. Zbiór ten otrzyma rozszerzenie .COM

parametry - parametry przebiegu łączenia

Np.

L80 PROG, PROG/N/E

Spowoduje wykonanie fazy łączenia dla zbioru o pełnej nazwie PROG.REL i utworzenie nowego zbioru o pełnej nazwie PROG.COM z programem w postaci wykonywalnej.

Parametry fazy łączenia sterują przebiegiem łączenia. Szczegółowo nie będziemy ich omawiać z uwagi na to, że dotyczą najczęściej możliwości języka BASIC, o których nie mówiliśmy przy omawianiu instrukcji.

8.2.3. Diagnostyka procesu kompilacji

W trakcie przebiegu kompilacji na monitorze podawana jest informacja zbiorcza o ilości wykrytych błędów w tekście programu. Szczegółowa informacja o błędach znajduje się na wydruku, który powstaje w zbiorze z rozszerzeniem .PRN. Widać stąd, że w sytuacji gdy program zawiera błędy wymagana jest analiza zbioru z rozszerzeniem .PRN.

Sygnalizacja błędu polega na wypisaniu dwuznakowego kodu błędu oraz numeru linii, w której błąd został wykryty.

Błędy sygnalizowane mają dwa stopnie ciężaru gatunkowego. Błędy typu FATAL, które powodują, że program wymaga bezwzględnej poprawy, oraz ostrzeżenia /WARNING/, które nie muszą powodować błędnego wykonania.

Kody błędów typu FATAL

SN Błędy składni, sygnalizowany w przypadku:
niedozwolony argument nazwy
niedozwolony fragment stałej
niedozwolona składnia wyrażenia
niedozwolona lista argumentów funkcji
niedozwolona nazwa funkcji
niedozwolony formalny parametr funkcji
niedozwolony separator
niedozwolony format numeru linii
niedozwolona składnia podprogramu
błędny znak
pominięty nawias lewostronny
opuszczony znak minus
opuszczony operand w wyrażeniu
opuszczony nawias prawostronny
opuszczony przecinek
zbyt długa nazwa
powinno wystąpić GO TO lub GOSUB
niedozwolona składnia
błędny numer argumentu
parametr formalny powinien być unikalny
dopuszczalna tylko pojedyncza zmienna
opuszczone TO
niedozwolona zmienna indeksowa w FOR
opuszczone THEN
niedozwolona nazwa podprogramu

OM Przekroczenie pamięci, sygnalizowane w przypadku:
zbytu duża tablica
przekroczenie pamięci danych
zbyt duży numer linii
przekroczenie pamięci programu

SQ Błąd sekwencji:
podwójny numer linii
zdanie poza sekwencją

TM Błąd typu:
konflikt typu danych
zmienne muszą mieć zgodne typy
TC linia zbyt złożona:
wyrażenie zbyt złożone
zbyt dużo argumentów funkcji CALL
zbyt duży rozmiar
zbyt dużo zmiennych dla INPUT
BS Błędny zapis:
niedozwolony rozmiar wartości
zły numer zapisu
LL Linia zbyt długa
UC Nierozpoznana komenda:
zdanie nierozpoznane
komenda niedozwolona
OV Nadmiar
/O Dzielenie przez zero
DD Tablica już posiadająca rozmiar
FN Błędne FOR/NEXT
Zmienna indeksowa w FOR już w użyciu
FOR bez NEXT
NEXT bez FOR
FD Funkcja już zdefiniowana
UF Funkcja niezdefiniowana
WE Błędne WHILE/WEND
WHILE bez WEND
WEND bez WHILE

Kody błędów typu WARNING

ND Tablica bez określonego rozmiaru
SI Zdanie zignorowane

Błędy składni wykrywane są na etapie kompilacji programu źródłowego w BASIC-u. Ponadto sygnalizowane są błędy wykonania programu. Błędy te posiadają numery, które są wypisywane wraz z numerem linii, w której błąd został wykryty.

Błędy wykonania

2 Błąd składni
Linia jest nierozpoznawalna lub zawiera niedozwoloną sekwencję znaków w instrukcji DATA
3 RETURN bez GOSUB

Instrukcja RETURN jest nierozpoznawalna /nie wystąpiła instrukcja GOSUB/

4 Przekroczenie danych
Próba wykonania instrukcji READ podczas gdy ciąg danych instrukcji DATA został wyczerpany

5 Niedozwolone wykorzystanie funkcji
Parametr przekazany do funkcji powoduje jej błędne funkcjonowanie. Może to wystąpić w sytuacji
- niedobry lub bezsensownie długi zapis
- ujemny lub zerowy argument funkcji LOG
- ujemny argument funkcji SQR
- ujemna mantysa z niecałkowitym wykładnikiem
- konkatenowany ciąg znaków dłuższy od 255 itp.

6 Nadmiar stałoprzecinkowy lub nadmiar zmiennoprzecinkowy
Wynik obliczeń zbyt duży niż pozwalają na to formaty danych. Wynik przyjmuje wartość zero i obliczenia są kontynuowane

9 Zapis poza zakresem
Element tablicy dotyczy poza zakresem tablicy

11 Dzielenie przez zero

14 Przekroczenie ciągu znaków
Ciąg znaków przekracza przyjęty rozmiar ciągu

21 Nienadający się do wydrukowania błąd
Niezdefiniowany kod błędu nie pozwala na wydrukowanie komunikatu o błędzie

50 Nadmiar pola
Instrukcja FIELD specyfikuje więcej bajtów dla rekordu niż długość rekordu w zbiorze o dostępie bezpośrednim

52 Zły numer zbioru
Instrukcja odnosząca się do zbioru dyskowego wykorzystuje numer zbioru, który nie wystąpił w instrukcji OPEN

53 Zbiór nie znaleziony
Na dysku nie znaleziono zbioru, do którego następuje odwołanie w programie

54 Zły tryb pracy zbioru
Próba wykonania operacji na zbiorze niezgodnie z trybem jego otwarcia

55 Zbiór już otwarty
Próba ponownego otwarcia zbioru już otwartego

57 Błąd we/wy na zbiorze dyskowym

Nieusuwalny błąd we/wy na zbiorze dyskowym

61 Dysk zapełniony
Cały obszar na dysku wykorzystany

62 Próba wprowadzania po końcu zbioru
Instrukcja INPUT wystąpiła dla zbioru pustego lub po wykryciu znacznika końca zbioru

63 Zły numer rekordu
W instrukcji PUT lub GET, użyty został numer rekordu zero lub większy od maksymalnej wartości

64 Zła nazwa zbioru
Nazwa zbioru niedozwolona /np. zbyt dużo znaków w nazwie/

67 Zbyt dużo zbiorów
Próba utworzenia nowego zbioru na dysku, gdzie nie ma miejsca w katalogu /więcej niż 255 zbiorów/.

8.3. Przykłady programów w BASIC-u

8.3.1. Przykład 1

Program służy do zakładania i aktualizacji zbioru dyskowego o dostępie bezpośrednim. Kluczem dostępu do zbioru jest numer kolejny pracownika. Informacja o jednym pracowniku nie mieści się na 128 znakach, dlatego też informacja o jednym pracowniku umieszczona jest w dwóch kolejnych rekordach, których klucze wyliczane są z numeru pracownika wg algorytmu n * 10 i n * 10 + 1, gdzie n jest numerem pracownika.

Każda informacja obsługiwana jest przez wspólny podprogram. Ponadto po przetworzeniu informacji o imieniu i nazwisku istnieje możliwość porzucenia przetwarzania danego pracownika i przejścia do innego. Przetwarzanie realizowane jest aż do podania zera jako numer pracownika. Proszę zwrócić uwagę na to, że logiczny koniec programu jest w wierszu numer 40.

```
10 REM "ZAKLADANIE/AKTUALIZACJA ZBIORY PTI"
20 OPEN  R",#1,"PTI"
30 INPUT "NUMER REKORDU",NR%
40 IF NR% = 0 THEN END
50 FIELD #1, 10 AS IM$, 15 AS NA$, 15 AS TY$, 4 AS RO$, 15 AS UC$
55 N=NR%*10
60 GET #1,N
70 LET TEK$=" IMIE "
80 LET PW$=IM$
90 GOSUB 7000
100 IF PW$="*" THEN GO TO 30
110 LSET IM$ PW$
120 LET TEK$="NAZWISKO"
130 LET PW$=NA$
140 GOSUB 7000
150 IF PW$="*" THEN GO TO 30
160 LSET NA$=PW$
170 LET TEK$="TYTULY$"
180 LET PW$=TY$
190 GOSUB 7000
200 LSET TY$=PW$
210 LET TEK$="ROK UKONCZENIA STUDIOW"
220 LET PW$=RO$
230 GOSUB 700
240 LSET RO$=PW$
250 LET TEK$="UCZELNIA UKONCZONA"
260 LET PW$=UC$
270 GOSUB 7000
280 LSET UC$=PW$
450 N = NR%*10
460 PUT #1,N
470 FIELD #1, 25 AS MPR$, 40 AS ADR$, 10 AS TES$, 15 AS STA$
473 LET N=NR%*10+1
476 GET #1,N
480 LET TEK$="MIEJSCE PRACY - NAZWA INSTYTUCI"
490 LET PW$=MPR$
500 GOSUB 7000
510 LSET MPR$=PW$
520 LET TEK$="ADRES INSTYTUCI"
530 LET PW$=ADR$
540 GOSUB 7000
550 LSET ADR$=PW$
560 LET TEK$="TELEFON SLUZBOWY"
570 LET PW$=TES$
580 GOSUB 7000
590 LSET TES$=PW$
600 LET TEK$="STANOWISKO"
610 LET PW$=STA$
620 GOSUB 7000
630 LSET STA$=PW$
640 LET N=NR%*10+1
650 PUT #1,N
5000 GO TO 30
7000 LET PR$=" "
7010 PRINT TEK$
7020 PRINT "WARTOSCI W ZBIORZE : ",PW$
7023 INPUT "WYMAGANA MODYFIKACJA ? T/N ",TN$
7026 IF TN$ <> "T" THEN GO TO 7050
7030 INPUT "NOWA WARTOSC : ",PR$
7040 PW$=PR$
7050 PRINT
7060 RETURN
```

8.3.2. Przykład 2

Program służy do wydruku informacji ze zbioru z poprzedniego przykładu. W programie wykorzystano funkcje specjalne dla redakcji wydruku.

Ponadto zamieszczony jest wydruk procesu kompilacji wraz z wydrukiem powstałego w trakcie przebiegu zbioru zawierającego listing. Ponadto załączony jest przykładowy wydruk osiągnięty tym programem.

```
10 REM "WYDRUK PELNEJ ZAWARTOSCI REKORDU ZE ZBIORU PTI"
12 OPEN "R",#1,"PTI"
15 INPUT "NUMER REKORDU ? ",NRX
20 IF NRX=0 THEN NX=0 ELSE NX=NRX-1
30 LPRINT STRING$(5,10),STRING$(5,13)
35 LET NX=NX+1
40 LPRINT STRING$(46,42)
50 LPRINT "*";TAB(46);"*"
60 LPRINT "* INFORMACJA O CZLONKU KOLA PTI W SZCZECINIE *"
70 LPRINT "*";TAB(46);"*"
80 LPRINT STRING$(46,42)
90 LPRINT STRING$(3,10),STRING$(3,13)
100 LPRINT "NUMER W ZBIORZE :     ";NX
105 LPRINT "=================="
110 FIELD #1, 10 AS IM$, 15 AS NA$, 15 AS TY$, 4 AS RO$, 15 AS UC$
120 LET NRRX=NX*10
130 GET #1,NRRX
140 LPRINT STRING$(3,10),STRING$(3,13)
145 LPRINT "IMIE : ";IM$
147 LPRINT "*******"
148 LPRINT
150 LPRINT "NAZWISKO : ";NA$
155 LPRINT "**********"
160 LPRINT STRING$(2,10),STRING$(2,13)
165 LPRINT "TYTULY  : ";TY$
167 LPRINT
170 LPRINT "ROK UKONCZENIA UCZELNI : ";RO$
175 LPRINT
180 LPRINT "UKONCZONA UCZELNIA  :    ";UC$
220 FIELD #1, 25 AS MPR$, 40 AS ADR$, 10 AS TES$, 15 AS STA$
230 LET NRRX=NRRX+1
240 GET #1,NRRX
250 LPRINT "MIEJSCE PRACY"
255 LPRINT "-------------"
260 LPRINT "   INSTYTUCJA : ";MPR$
270 LPRINT "   ADRES      : ";ADR$
280 LPRINT "   TELEFON    : ";TES$
290 LPRINT "   STANOWISKO : ";STA$
1000 END
```

```
A>
A>TYPE PTIPR3.PRN
. . . . . .


BASCOM 5.30 - Copyright 1979,80,81 (C) by MICROSOFT - 25298 Bytes Free
 0014 0007        10 REM "WYDRUK PELNEJ ZAWARTOSCI REKORDU ZE ZBIORU PTI
 0014 0007        12 OPEN "R",#1,"PTI"
        ** 0014'I00000: CALL     $530
        ** 0017'L00010: L00012:
        ** 0017'        LXI      H,(const)
        ** 001A'        CALL     $DKN
        ** 001D'        LXI      B,0001
        ** 0020'        LXI      D,(const)
        ** 0023'        LXI      H,0000
        ** 0026'        CALL     $DKM
 0029 0007        15 INPUT "NUMER REKORDU ? ",NR%
        ** 0029'L00015: LXI      H,(const)
        ** 002C'        CALL     $INOA
        ** 002F'        DB       02
        ** 0030'        CALL     $IPUA
        ** 0033'        DB       01
        ** 0034'        DB       04
        ** 0035'        LXI      H,NR%
        ** 0038'        CALL     $IPUB
 003B 0009        20 IF NR%=0 THEN N%=0 ELSE N%=NR%-1
        ** 003B'L00020: LHLD     NR%
        ** 003E'        MOV      A,H
        ** 003F'        ORA      L
        ** 0040'        JNZ      I00001
        ** 0043'        LXI      H,0000
        ** 0046'        SHLD     N%
        ** 0049'        JMP      I00002
        ** 004C'I00001:
        ** 004C'        LHLD     NR%
        ** 004F'        DCX      H
        ** 0050'        SHLD     N%
        ** 0053'I00002:
 0053 000B        30 LPRINT STRING$(5,10),STRING$(5,13)
        ** 0053'L00030: CALL     $PROE
        ** 0056'        LXI      D,0005
        ** 0059'        LXI      H,000A
        ** 005C'        CALL     $ST$
        ** 005F'        CALL     $PV0D
        ** 0062'        LXI      H,000D
        ** 0065'        CALL     $ST$
        ** 0068'        CALL     $PV2D
 006B 000B        35 LET N%=N%+1
        ** 006B'L00035: LHLD     N%
        ** 006E'        INX      H
        ** 006F'        SHLD     N%
 0072 000B        40 LPRINT STRING$(46,42)
        ** 0072'L00040: CALL     $PROE
        ** 0075'        LXI      D,002E
        ** 0078'        LXI      H,002A
        ** 007B'        CALL     $ST$
        ** 007E'        CALL     $PV2D
 0081 000B        50 LPRINT "*";TAB(46);"*"
        ** 0081'L00050: CALL     $PROE
        ** 0084'        LXI      H,(const)
```

```
** 0087'          CALL    $PV1D
** 0088'          LXI     H,002E
** 008B'          CALL    $TABA
** 0090'          LXI     H,(const)
** 0093'          CALL    $PV1D
** 0096'          LXI     H,(const)
** 0099'          CALL    $PV2D
0000 000B      60 LPRINT "* INFORMACJA O CZŁONKU KOŁA PTI W SZCZECINIE *
** 009C'L00060: CALL    $PROE
** 009F'          LXI     H,(const)
** 00A2'          CALL    $PV2D
0000 000B      70 LPRINT "*";TAB(46);"*"
** 00A5'L00070: CALL    $PROE
** 00A8'          LXI     H,(const)
** 00AB'          CALL    $PV1D
** 00AE'          LXI     H,002E
** 00B1'          CALL    $TABA
** 00B4'          LXI     H,(const)
** 00B7'          CALL    $PV1D
** 00BA'          LXI     H,(const)
** 00BD'          CALL    $PV2D
0000 000B      80 LPRINT STRING$(46,42)
** 00C0'L00080: CALL    $PROE
** 00C3'          LXI     D,002E
** 00C6'          LXI     H,002A
** 00C9'          CALL    $ST$
** 00CC'          CALL    $PV2D
0000 000B      90 LPRINT STRING$(3,10),STRING$(3,13)
** 00CF'L00090: CALL    $PROE
** 00D2'          LXI     D,0003
** 00D5'          LXI     H,0004
** 00D8'          CALL    $ST$
** 00DB'          CALL    $PV0D
** 00DE'          LXI     H,000D
** 00E1'          CALL    $ST$
** 00E4'          CALL    $PV2D
0000 000B     100 LPRINT "NUMER W ZBIORZE ";
** 00E7'L00100: CALL    $PROE
** 00EA'          LXI     H,(const)
** 00ED'          CALL    $PV1D
** 00F0'          LHLD    HX
** 00F3'          CALL    $PV1X
0000 000B     105 LPRINT "*****************"
** 00F6'L00105: CALL    $PROE
** 00F9'          LXI     H,(const)
** 00FC'          CALL    $PV2D
0000 000B     110 FIELD #1: 10 AS IM$, 15 AS IN$, 15 AS TY$, 4 AS RO$, 1
** 00FF'L00110: LXI     H,0001
** 0102'          CALL    $FLD$
** 0105'          LXI     D,IM$
** 0108'          LXI     H,0000
** 010B'          CALL    $FLDD
** 010E'          LXI     D,IN$
** 0111'          LXI     H,000F
** 0114'          CALL    $FLDD
** 0117'          LXI     D,TY$
** 011A'          CALL    $FLDD
** 011D'          LXI     D,RO$
** 0120'          LXI     H,0004
** 0123'          CALL    $FLDD
** 0126'          LXI     D,UP$
** 0129'          LXI     H,0010
```

```
        ** 012C'            CALL     $FLDR
012F 001A        120 LET NRR%=N%*10
        ** 012F'L00120: LHLD     N%
        ** 0132'            CALL     $IMUG
        ** 0135'            DB       0A,00
        ** 0137'            SHLD     NRR%
013A 001C        130 GET $1,NRR%
        ** 013A'L00130: LXI      D,0001
        ** 013D'            LHLD     NRR%
        ** 0140'            CALL     $RGT
0143 001C        140 LPRINT STRING$(3,10)+STRING$(3,13)
        ** 0143'L00140: CALL     $PROE
        ** 0146'            LXI      D,0003
        ** 0149'            LXI      H,000A
        ** 014C'            CALL     $ST$
        ** 014F'            CALL     $PV0D
        ** 0152'            LXI      H,000D
        ** 0155'            CALL     $ST$
        ** 0158'            CALL     $PV2D
015B 001C        145 LPRINT "INIE : ";IN$
        ** 015B'L00145: CALL     $PROE
        ** 015E'            LXI      H,(const)
        ** 0161'            CALL     $PV1D
        ** 0164'            LXI      H,IM$
        ** 0167'            CALL     $PV2D
016A 001C        147 LPRINT "*******"
        ** 016A'L00147: CALL     $PROE
        ** 016D'            LXI      H,(const)
        ** 0170'            CALL     $PV2D
0173 001C        148 LPRINT
        ** 0173'L00148: CALL     $PROE
        ** 0176'            LXI      H,(const)
        ** 0179'            CALL     $PV2D
017C 001C        150 LPRINT "NAZUISKO : ";NA$
        ** 017C'L00150: CALL     $PROE
        ** 017F'            LXI      H,(const)
        ** 0182'            CALL     $PV1D
        ** 0185'            LXI      H,NA$
        ** 0188'            CALL     $PV2D
018B 001C        155 LPRINT "**********"
        ** 018B'L00155: CALL     $PROE
        ** 018E'            LXI      H,(const)
        ** 0191'            CALL     $PV2D
0194 001C        160 LPRINT STRING$(2,10)+STRING$(2,13)
        ** 0194'L00160: CALL     $PROE
        ** 0197'            LXI      D,0002
        ** 019A'            LXI      H,000A
        ** 019D'            CALL     $ST$
        ** 01A0'            CALL     $PV0D
        ** 01A3'            LXI      H,000D
        ** 01A6'            CALL     $ST$
        ** 01A9'            CALL     $PV2D
01AC 001C        165 LPRINT "TYTULY  : ";TY$
        ** 01AC'L00165: CALL     $PROE
        ** 01AF'            LXI      H,(const)
        ** 01B2'            CALL     $PV1D
        ** 01B5'            LXI      H,TY$
        ** 01B8'            CALL     $PV2D
01BB 001C        167 LPRINT
        ** 01BB'L00167: CALL     $PROE
        ** 01BE'            LXI      H,(const)
        ** 01C1'            CALL     $PV2D
```

```
01C4 001C        170 LPRINT "ROK UKONCZENIA UCZELNI : "$RO$
     ** 01C4'L00170: CALL     $PROE
     ** 01C7'         LXI      H,(const)
     ** 01CA'         CALL     $PV1D
     ** 01CD'         LXI      H,RO$
     ** 01D0'         CALL     $PV2D
01D3 001C        175 LPRINT
     ** 01D3'L00175: CALL     $PROE
     ** 01D6'         LXI      H,(const)
     ** 01D9'         CALL     $PV2D
01DC 001C        180 LPRINT "UKONCZONA UCZELNIA  :     "$UC$
     ** 01DC'L00180: CALL     $PROE
     ** 01DF'         LXI      H,(const)
     ** 01E2'         CALL     $PV1D
     ** 01E5'         LXI      H,UC$
     ** 01E8'         CALL     $PV2D
01EB 001C        220 FIELD $1, 25 AS MPR$, 40 AS ADR$, 10 AS TES$, 15 AS STA
     ** 01EB'L00220: LXI      H,0001
     ** 01EE'         CALL     $FLDA
     ** 01F1'         LXI      D,MPR$
     ** 01F4'         LXI      H,0019
     ** 01F7'         CALL     $FLDB
     ** 01FA'         LXI      D,ADR$
     ** 01FD'         LXI      H,0028
     ** 0200'         CALL     $FLDB
     ** 0203'         LXI      D,TES$
     ** 0206'         LXI      H,000A
     ** 0209'         CALL     $FLDB
     ** 020C'         LXI      D,STA$
     ** 020F'         LXI      H,000F
     ** 0212'         CALL     $FLDB
0215 0028        230 LET NRRZ=NRRZ+1
     ** 0215'L00230: LHLD     NRRZ
     ** 0218'         INX      H
     ** 0219'         SHLD     NRRZ
021C 0028        240 GET $1,NRRZ
     ** 021C'L00240: LXI      D,0001
     ** 021F'         LHLD     NRRZ
     ** 0222'         CALL     $RGT
0225 0028        250 LPRINT "MIEJSCE PRACY"
     ** 0225'L00250: CALL     $PROE
     ** 0228'         LXI      H,(const)
     ** 022B'         CALL     $PV2D
022E 0028        255 LPRINT "---------------"
     ** 022E'L00255: CALL     $PROE
     ** 0231'         LXI      H,(const)
     ** 0234'         CALL     $PV2D
0237 0028        260 LPRINT "  INSTYTUCJA : "$MPR$
     ** 0237'L00260: CALL     $PROE
     ** 023A'         LXI      H,(const)
     ** 023D'         CALL     $PV1D
     ** 0240'         LXI      H,MPR$
     ** 0243'         CALL     $PV2D
0246 0028        270 LPRINT "   ADRES       : "$ADR$
     ** 0246'L00270: CALL     $PROE
     ** 0249'         LXI      H,(const)
     ** 024C'         CALL     $PV1D
     ** 024F'         LXI      H,ADR$
     ** 0252'         CALL     $PV2D
0255 0028        280 LPRINT "   TELEFON     : "$TES$
     ** 0255'L00280: CALL     $PROE
     ** 0258'         LXI      H,(const)
```

```
        ** 025B'          CALL    $PV1D
        ** 025E'          LXI     H,TES$
        ** 0261'          CALL    $PV2D
 0264 0028       290 LPRINT "   STANOWISKO : ";STA$
        ** 0264'L00290: CALL     $PROE
        ** 0267'          LXI     H,(const)
        ** 026A'          CALL    $PV1D
        ** 026D'          LXI     H,STA$
        ** 0270'          CALL    $PV2D
 0273 0028       1000 END
        ** 0273'L01000: CALL     $END
 0276 0028
        ** 0276'          CALL    $END
 03DC 0032

00000 Fatal Error(s)
24079 Bytes Free

A>DIR
```

9. ZASTOSOWANIA MIKROKOMPUTERÓW

Zastosowania mikrokomputerów, dzięki ich właściwościom takim jak miniaturyzacja, mały pobór mocy, specjalizacja architektury i konfiguracji, niskie koszty zakupu i eksploatacji, obsługa i programowanie przez bezpośredniego użytkownika itp., obejmują rzeczywiście wszystkie dziedziny życia ludzkiego /zarówno związanego z pracą zawodową, jak i życiem domowym/.

Tradycyjny podział zastosowań informatyki /z punktu widzenia zaspokajanych potrzeb/ na:

- obliczenia naukowo-techniczne i zawodowe,
- przetwarzanie danych gospodarczych,
- sterowanie procesami technologicznymi,
- wyszukiwanie informacji,
- dydaktyka komputerowa,

ugruntował się, znacznie rozszerzył swoje zakresy w ramach wymienionych dziedzin, a jednocześnie dzięki właśnie zastosowaniu mikrokomputerów, wyodrębniły się przynajmniej dwie nowe dziedziny zastosowań, które nazwać można:

- informatyka domowa /obejmująca różnorodne czynności życia domowego związane zarówno z funkcjonowaniem gospodarstw domowych, jak również rozrywkę i wypoczynek/,
- obsługa informacyjna społeczeństwa /możliwa dzięki wyposażeniu gospodarstwa domowego w mikrokomputer jako terminala inteligentnego systemów informacyjnych/.

Jest to jak gdyby pionowy podział zastosowań. Należy jeszcze zwrócić uwagę na zjawisko poziomego ujęcia zastosowań. Pierwszy krąg zastosowań informatyki dotyczył przede wszystkim przedsiębiorstw produkcyjnych /wspomaganie i obsługa procesów wytwarzania/. Następnym kręgiem zastosowań było opanowanie przez informatykę sektora obsługi ludności i usług dla ludności a zatem administracji, handlu, banków, komunikacji itd. /co jest logicznym następstwem opanowania pierwszego kierunku zastosowań/. I najszerszy krąg, związany z mikrokomputerami, to zastosowania domowe.

Pomiędzy wymienionymi kręgami istnieją powiązania funkcjonalne i maszynowe, pozwalające na ich integrację. Informatyka domowa umożliwia połączenie wszystkich zastosowań.

Cechą charakterystyczną zastosowań mikrokomputerów jest za-

kres tych zastosowań ograniczony do pojedyńczego człowieka, stanowiska pracy. Cecha ta przejawia się we wszystkich wymienionych dziedzinach zastosowań.

O b l i c z e n i a n a u k o w o - t e c h n i c z n e, inżynierskie i innych zawodów obok tradycyjnych kierunków objęły szereg nowych zastosowań, wśród których najbardziej rozwinęły się:

1/ Projektowanie wspomagane

Udział kosztów projektowania, szczególnie w przedsięwzięciach technicznych stale rośnie. Najlepszym przykładem jest sama mikroelektronika, gdzie nakład pracy projektowej elementów VSLI sięga kilkudziesięciu osobolat, a sama złożoność projektu powoduje, że nie można go stworzyć metodami ręcznymi. Komputerowe wspomaganie projektowania nie ma w takich warunkach alternatywy i jest warunkiem rozwoju nauki i techniki w ogóle. Warunki umożliwiające komputerowe wspomaganie projektowania wymagają istnienia:

- języka specyfikacji zadania projektowego,
- systemu weryfikacji zadania,
- języka dialogu z komputerem,
- modelu zadania projektowego w komputerze.

Tradycyjne systemy CAD /Computer Aided Design/ zastępowane są dzięki mikrokomputerom systemami CIE /Computer Integrated Engineering/. Ten najwyższy stopień automatyzacji projektowania poprzedzają poziomy niższe tzn. wspomaganie projektowania oraz wykonywanie obliczeń inżynierskich dla potrzeb projektowania.

2/ Jako przykład gwałtownego rozwoju zastosowań zawodowych związanych z wykorzystaniem mikrokomputerów można wymienić medycynę. Zastosowania informatyki w medycynie mają długą tradycję, dopiero jednak zastosowanie mikrokomputerów spowodowało, że ich zakres obejmuje:

- wspomaganie medycznej działalności naukowej,
- wspomaganie praktyki medycznej,
- diagnostykę,
- nadzór nad ciężko chorymi,
- sterowanie urządzeniami analitycznymi,
- wykorzystanie w protezach i innych urządzeniach zastępujących funkcjonowanie określonych narządów.

Jest to zatem bardzo szeroki zakres od badań naukowych /statystyki medyczne ale również modele mięśni, komórek nerwowych, serca itp., modele symulacyjne ich funkcjonowania/ poprzez praktykę lekarską /diagnostyka, zbieranie i analiza informacji z badań analitycznych, zarządzanie szpitalem/ do bezpośredniego zastosowania w leczeniu /nadzorowanie chorych w intensywnym leczeniu czyli obserwowania stanu zdrowia pacjenta w sposób ciągły/, informowanie a także opisywanie przebiegu leczenia, sterowanie urządzeniami medycznymi, wspomaganie czynności wykrywanych przez trwale upośledzonych np. syntetyzatory mowy, czytniki pisma, maszyny do pisania dla niewidomych itp./. Medycyna jest przykładem wszechstronnego, zintegrowanego zastosowania mikrokomputerów w określonej dziedzinie, rozszerzającego w sposób nieporównywalny do działania tradycyjnego, jakość, skuteczność, niezawodność, efektywność, nowych, wspomaganych mikrokomputerem rozwiązań.

3/ Tradycyjne pakiety obliczeń inżynierskich, obejmujące zbiory powszechnie stosowanych algorytmów, doskonale spełniające swoje zadania w tradycyjnej informatyce /ich potrzeba i efektywność nigdy nie były kwestionowane/ znalazły się prawie w całości w oprogramowaniu użytkowym mikrokomputerów. Są to podstawowe algorytmy statystyki, ekonometrii, matematyki i z reguły obejmują:

- obliczenia statystyczne,
- działania na macierzach,
- całkowanie,
- różniczkowanie,
- działania na wielomianach,
- aproksymację i interpolację.

Możliwości zastosowania określonego algorytmu ograniczone są z jednej strony zasobami mikrokomputera, z drugiej strony wielkością problemu użytkownika. Użytkownik mikrokomputerów mając do dyspozycji języki wysokiego rzędu może zakres obliczeń dostosowywać i rozszerzać odpowiednio do swoich potrzeb.

P r z e t w a r z a n i e d a n y c h g o s p o d a r c z y c h na mikrokomputerach rozpatrywać można również w dwóch płaszczyznach:

- co jest nowego w stosunku do tradycyjnych zastosowań,

- co z zastosowań tradycyjnych przeniesiono na mikrokomputery.

1/ Niewątpliwie nowa jakość związana z mikrokomputerami mieści się w tzw. biurotyce /lub automatyzacji prac biurowych/. Jest to dziedzina, której tradycyjna informatyka nie była w stanie opanować w związku z trudnością doprowadzenia mocy obliczeniowych do stanowiska biurowego, różnorodnością czynności wykonywanych na takim stanowisku, a obejmującą obok obliczeń również problemy komunikacji, przesyłania informacji, przetwarzanie tekstów, pisanie na maszynie itp., i związaną z nią różnorodnością urządzeń technicznych wspierających te czynności. Wszystkie te różnorodne urządzenia powinny w formie zintegrowanej znaleźć się na jednym stanowisku pracy. Do pracowników biura - użytkowników systemu, zalicza się zarówno kierowników, jak i urzędników oraz pracowników sekretariatu. Czynności wykonywane w ramach automatyzacji biura przez poszczególne grupy pracowników prezentuje tablica 15. O skali problemu w USA świadczą następujące liczby: koszty pracy urzędników i sekretarek w USA stanowią 34 % ogólnych kosztów pracy /około 400 mld dolarów/ a liczba tej grupy pracowników wynosi około 15 mln. Jest to zatem obszar dużych potencjalnie efektów usprawnienia tych prac.

Oprogramowanie systemu automatyzacji biura wyróżnia następujące grupy programów:

- programy tworzenia dokumentów, umożliwiające bezpośrednie wprowadzenie dokumentu do kartoteki, edycję oraz korektę tekstu dokumentów,
- programy poczty elektronicznej, pozwalające na szybką, papierooszczędną wymianę informacji między pracownikami biura,
- programy organizacji i wyszukiwanie w kartotekach, ułatwiające obsługę kartotek, indeksowanie, wyszukiwanie z nich informacji,
- programy wyprowadzania informacji,
- programy obliczeniowe dotyczące np. rozliczeń zużycia materiałów, delegacji, obliczeń statystycznych i ekonomicznych,
- programy instruktarzowe, jak nauka obsługi systemu, maszynopisanie, oraz
- programy systemowe związane z utrzymaniem kartotek i opro-

Tablica 15

Struktura czasu pracy pracowników biura

Czynności	Przeciętny wskaźnik struktury czasu pracy w %			
	Dyrektorzy	Kierownicy	Urzędnicy	Sekretarki
Pisanie	9,8	17,2	17,8	5,5
Korespondencja	6,1	5,0	2,7	8,1
Zatwierdzanie	1,8	2,5	2,4	3,9
Wyszukiwanie inform.	3,0	6,4	6,4	-
Czytanie	8,7	7,4	6,3	1,7
Obsługa kartotek	1,1	2,0	2,5	4,6
Wyszukiwanie w kartotekach	1,8	3,7	4,3	2,8
Dyktowanie				
- sekretarce	4,9	1,7	0,4	-
- maszynie	1,0	0,9	0,0	-
Telefon	13,8	12,3	11,3	10,5
Obliczanie	2,3	5,8	9,6	-
Omawianie spraw z sekretarką	2,9	2,1	1,0	-
Planowane spotkania	13,1	6,7	3,8	-
Nieplanowane spotkania	8,5	5,7	3,4	-
Łączenie i sortowanie dokumentów	-	-	-	2,6
Pisanie na maszynie	-	-	-	37,0
Omawianie spraw z kierownikiem	-	-	-	4,3
Stenografowanie	-	-	-	5,5
Przygotowywanie poczty	-	-	-	1,4
Kalendarz spotkań	-	-	-	2,6
Odbiór poczty	-	-	-	2,2
Planowanie	4,7	5,5	2,9	-
Podróże	13,1	6,6	2,2	-
Kopiowanie	0,1	0,6	1,4	6,2
Korzystanie ze sprzętu	0,1	1,3	9,9	1,3
I n n e	3,1	6,6	11,7	1,8

gramowania automatycznego biura.

Zastosowanie mikrokomputerów w pracach biurowych stwarza realne podstawy zwiększenia wydajności pracy w tym zakresie.

2/ Z tradycyjnych zastosowań informatyki w przetwarzaniu danych gospodarczych przeniesiono na mikrokomputery wszystkie zastosowania /oczywiście z uwzględnieniem specyfiki tych zastosowań, tzn. małe, autonomiczne, stanowiskowe systemy/. Wymienić tutaj można systemy:
- gospodarki materiałowej,
- gospodarki finansowej,
- gospodarki kadrowo-płacowej,
- rozrachunków,
- fakturowania, itp.

System ewidencji finansowo-kosztowej opracowany na mikrokomputer PSPD-90 przez producenta MERA-KFAP w Krakowie posiada następujące możliwości.

System ewidencji finansowo-kosztowej zwany F-K, jest systemem uniwersalnym, może być eksploatowany w jednostkach gospodarczych różnego szczebla i branż, jak przedsiębiorstwa przemysłowe, budowlane, spółdzielnie typu produkcyjnego, usługowego itp.

Jest on zorientowany na mikrokomputer PSPD-90 wyposażony w pamięć wewnętrzną 8 K, drukarkę wierszową i monitor, oprogramowany jest przy wykorzystaniu systemu operacyjnego SOJK.

F-K bazuje na 9-cio znakowym symbolu konta analitycznego. Podstawowe zbiory systemu mogą obejmować:
- Kartoteka analityczna - do 5 000 pozycji /kont analitycznych/, mieści się na jednym dysku,
- Dekrety miesiąca - do 7 500 pozycji, co obejmuje jeden dysk,
- Zbiór rozrachunków - do 15 000 pozycji - maksimum dwie dyskietki.

Przetwarzanie w systemie F-K odbywa się okresowo, okresem jest miesiąc obliczeniowy, natomiast zbiór transakcyjny może być tworzony sukcesywnie, w miarę powstawania zaszłości gospodarczych.

System jest prosty w obsłudze, dokładny "przewodnik" po jego programach stanowi Instrukcja operatorska. Z tych względów może być obsługiwany przez pracownika działu księgowości.

Główne zadania systemu F-K sprowadzają się do:
- założenia bilansu otwarcia dla poszczególnych kont analitycznych na dzień 1.I. - dla wejścia systemu do eksploata-

cji; możliwe jest również wejście w trakcie roku, lecz wyłącznie bilansem otwarcia na dany miesiąc, nie obrotami narastającymi,

- założenia i modyfikacji zbioru nazw kont analitycznych; zbiór ten ma charakter opcyjny /oznacza to, że można go nie zakładać/,
- co miesięcznego tworzenia zbioru transakcji, zawierającego zaszłości miesiąca obliczeniowego /dekrety/.

W trakcie ich wprowadzania, przeprowadzana jest systemowa kontrola ich poprawności, zasadnicze znaczenie ma zwłaszcza kontrola na bilansowanie się zapisów w ramach każdego numeru dowodu w układzie:

  - suma Wn = suma Ma
  - suma Wn 400 - 469 = suma Ma 490 = suma Wn zespołu 5 i 6,

- automatycznego ustalania stanów i obrotów miesiąca każdego konta analitycznego,
- systemowego ustalania stanów i obrotów kont syntetycznych oraz na dowolną /podaną/ ilość znaków konta analitycznego; możliwa jest przy tym emisja informacji na wskazaną ilość:
  - tylko pierwszych znaków symbolu konta,
  - pierwszych i ostatnich znaków konta.

Salda we wszystkich zespołach kont ustalane są w ten sam sposób, przez porównanie sumy obrotów Wn i Ma i ustawienie salda po stronie większej w wysokości różnicy między tymi stronami.

- automatycznego tworzenia zbioru rozrachunków /wybieranie ze zbioru dekretów/, celem dołączenia ich do rozrachunków nie zrealizowanych do końca ubiegłego miesiąca, dla przeprowadzenia ich analizy w miesiącu obliczeniowym,
- systemowego prowadzenia zbioru wszystkich transakcji od początku roku dla wskazanych /wybranych/ kont,
- automatycznego zamknięcia kont na koniec roku oraz tworzenie bilansu otwarcia dla nowego roku,
- przeprowadzenie korekty do bilansu otwarcia, w wyniku badania bilansu przez biegłych.

Wymienione funkcje oraz funkcje pomocnicze /etykietowanie zbiorów, ich kopiowanie, ustawianie ilości wierszy na stronie tabulogramu/, realizują 43 programy, które są pogrupowane w 8

modułów. Nazwy i numery tych modułów tworzą makietę główną systemu, która ukazuje się na ekranie monitora po włączeniu stacji.

Uruchomienie programu polega na wybraniu z makiety głównej właściwego modułu /przez wprowadzenie jego numeru/, po czym na monitorze ukaże się z kolei jego makieta, zawierająca wykaz programów tego modułu. Teraz wystarczy podać z klawiatury numer interesującego programu i system przechodzi do wykonania jego procedur.

S t e r o w a n i e p r o c e s a m i t e c h n o l o g i c z n y m i /a do systemów tych zalicza się systemy pomiarowo-kontrolne i zastosowania w automatyce/ z wykorzystaniem mikrokomputerów stworzyło podobnie jak w innych zastosowaniach nowe możliwości. Problematykę tą, bardzo szeroką, można podzielić następująco:

- zastosowanie skupione /z pojedyńczym procesorem/,
- zastosowanie rozłożone przestrzennie,
- mikroprocesory w urządzeniach technicznych.

W zastosowaniach ze sterowaniem skupionym /centralnym/ procesor /program główny/ steruje pracami innych programów. Zastosowanie mikroprocesorów umożliwia:

- zmniejszenie i potanienie systemów sterowania, przez co stają się one opłacalne nawet w przypadku niewielkich procesów,
- zwiększenie niezawodności i elastyczności.

Te dwie przesłanki doprowadziły do zastosowań rozłożonych przestrzennie /lokalne sieci mikrokomputerów/. Przykładowy system sterowania rozproszonego TDC-2000 /Total Distributed Control firmy Honeywell/ zbudowany jest z kilku modułów o charakterze cyfrowym i analogowym. Podstawowym modułem jest regulator /mikroprocesor/ przyjmujący i wysyłający sygnały cyfrowe i analogowe. Ponadto system wyposażony jest w stanowisko operatora, który śledzi i kieruje przebiegiem procesu. Istnieją stanowiska przenośne umożliwiające lokalne ich włączenie do systemu w dowolnym miejscu. Kolejnymi modułami są drukarki i koncentratory danych. Koncentratory tworzą z wielu strumieni danych płynących z różnych przyrządów pomiarowych /przetworniki, czujniki, termometry/ jeden strumień. Różne moduły, których w systemie może być 63 łączy magistralę systemową. W innej sieci DANUBE można

dołączyć do 255 stacji, a szybkość transmisji dochodzi do 2 Mb/s.

Szczytowym osiągnięciem wykorzystania mikroprocesorów w urządzeniach technicznych są roboty przemysłowe. Robot sterowany mikrokomputerem składa się z części komputerowej, do pamięci której wprowadza się informacje o wykonywanych czynnościach. Mikrokomputer zgodnie z zapisanym programem za pośrednictwem szeregu elementów wykonawczych takich jak: silniki elektryczne, elektromagnesy itp., steruje ruchami części mechanicznej robota, która jest wyposażona w szereg uchwytów, podajników przemieszczających np. obrabiany element do narzędzia lub odwrotnie. Kolejnym etapem tego typu zastosowań są urządzenia kroczące.

S y s t e m y w y s z u k i w a n i a i n f o r m a c j i umożliwiają efektywne wykorzystanie w działalności ludzkiej zasobów informacyjnych określonych obiektów /przedsiębiorstw, instytucji, również komórek organizacyjnych/. Problem ten w tradycyjnych zastosowaniach, wymagających bardzo dużych zasobów komputera oraz skomplikowanych metod i algorytmów klasyfikacji, organizacji i wyszukiwania informacji w zastosowaniach mikrokomputerowych sprowadzony został przede wszystkim do systemów relacyjnych baz danych w bardzo prostych do eksploatacji wersjach. W oparciu o takie oprogramowanie możliwe jest tworzenie różnorodnych systemów użytkowych omówionych wcześniej /np. systemy przetwarzania danych/ na zasadzie wyszukiwania informacji. Opracowany w kraju przez Computer Studio Kajkowski Bank Danych - CSK umożliwia:

- zakładanie plików danych /w tym również kopiowanie i łączenie plików/,
- aktualizowanie plików danych /dopisywanie nowych rekordów, zmiany w rekordach i ich polach, usuwanie rekordów/,
- zastosowanie innych procedur /sortowania, indeksowania, scalania, wyszukiwania, drukowania, badania warunków, tworzenia dziennika dostępu do plików/,
- tworzenie systemów użytkowych.

Możliwa jest również współpraca systemu zarządzania bazą danych z innymi systemami tej firmy np. TEKS-CSK /system redagowania tekstów/. W sumie oprogramowanie to można bardzo efektywnie wykorzystać w odniesieniu do autonomicznych zbiorów związanych z konkretnym stanowiskiem pracy użytkownika. Przy minimalnych zasobach /48 kb pamięci operacyjnej/ umożliwia on operowanie

zbiorami obejmującymi 65 535 rekordów w pliku i 1000 znaków w rekordzie.

S y s t e m y d y d a k t y k i k o m p u t e r o w e j wspomagają procesy dydaktyczne. Komputer spełniać może w tych systemach wiele funkcji nauczania, jak nauczanie właściwe, testowanie postępów nauczania, dostarczanie uczącemu się informacji pomocniczych, rejestrowanie postępu itp.

Zaletą zastosowań mikrokomputerów w dydaktyce jest interakcyjny charakter komunikacji uczącego się a maszyną, wizualizacja podawanej przez mikrokomputer wiedzy na monitorze ekranowym, kontrola poprawności przyswajania wiedzy. Zastosowania mikrokomputerów w dydaktyce zaczynają się już od zapoznania się z samym mikrokomputerem, poprzez kontrolowane nauczanie języków programowania do dydaktyki nie związanej z informatyką np. nauka języków obcych, gry kierownicze, testy kontrolne itp. Mikrokomputery mogą stanowić terminale inteligentne dużych systemów dydaktyki komputerowej.

Mikroprocesory w z a s t o s o w a n i a c h d o m o w y c h znalazły szerokie zastosowanie zarówno w urządzeniach domowych jak również w "systemach" obsługujących gospodarstwo domowe. W pierwszej grupie zastosowań wymienić można mikroprocesory jako urządzenia sterujące działaniem pralki automatycznej, w automatycznych kuchenkach indukcyjnych, czy też samochodach. Dla przykładu mikroprocesory zastosowane w samochodach umożliwiają sterowanie pracą silnika, zapewniają zwiększenie bezpieczeństwa jazdy /kontrola zapięcia pasów, zamknięcia drzwi, programowanie hamowania itp./, a także informują kierowcę o wielu elementach jazdy /szybkość, przebyta odległość, czas wyjazdu itp./. W sumie optymalizują pracę samochodu /do 20 % zmniejszenia zużycia paliwa/, zapewniają większe bezpieczeństwo i komfort jazdy.

Do drugiej grupy zastosowań zaliczyć należy:

- planowanie budżetu domowego i kontrolę jego realizacji,
- terminarze,
- rozliczenia z zewnętrznymi kontrahentami /banki, handel, ubezpieczenia itp./,
- obliczenia zawodowe,
- podręczne bazy danych,
- również gry komputerowe itp.

Rozwój tego typu zastosowań mikrokomputerów domowych umożliwi prawdopodobnie w przyszłości przeniesienie prac zawodowych do domów.

Zastosowania mikrokomputerów jako terminali inteligentnych różnorodnych systemów stwarza możliwości szerokiej komunikacji pomiędzy pojedynczym użytkownikiem a skomputeryzowanym społeczeństwem. W postaci mikrokomputerów domowych znalazły swoje urzeczywistnienie tanie, dostępne dla wszystkich urządzenia techniczne umożliwiające automatyczną wymianę informacji ze społeczeństwem przy zachowaniu odrębności jednostki, bez naruszenia życia osobistego i praw ludzkich.

10. PRZEGLĄD KRAJOWEGO SPRZĘTU MIKROKOMPUTEROWEGO

10.1. AC 805

Mikrokomputer zbudowany na bazie mikroprocesora Z80 przeznaczony jest do zastosowań profesjonalnych jako pomoc w dydaktyce, czy też obliczeniach naukowo-technicznych. Minimalna konfiguracja AC 805 obejmuje:

- pamięć ROM - 12 kb,
- pamięć dynamiczna RAM - 16, 32 lub 48 kb,
- monitor ekranowy /24 linie po 32 znaki, 96 znaków ASCII, 32 znaki graficzne, 128 znaków programowalnych, możliwa pseudografika/,
- klawiatura typu QWERTY,
- układ współpracy z magnetofonem,
- sprzęgi: 2 porty szeregowe w standardzie RS 232C oraz port równoległy - 24 linie TTL /port równoległy umożliwia podłączenie do mikrokomputera nietypowych urządzeń peryferyjnych np. rejestratorów, sterowników, drukarek, ploterów/.

Wersja rozbudowana mikrokomputera może objąć:

- dyski elastyczne 5,25" /AC 815/,
- dyski twarde WINCHESTER /AC 825/.

Oprogramowanie mikrokomputera AC 805 obejmuje

- MONITOR o nazwie Mikro System,
- interpretor języka BASIC /zgodny ze standardem MICROSOFT/,
- edytor ekranowy,
- opcjonalnie system operacyjny kompatybilny z CP/M 2.2

Producentem mikrokomputerów rodziny AC 800 jest firma AMEPROD.

10.2. ComPAN-8

ComPAN-8 jest 8-bitowym mikrokomputerem profesjonalnym na bazie mikroprocesora 8080A. Podstawowe jego elementy to:

- procesor z kontrolerem przerwań i kontrolerem przesyłów. Możliwość adresowania do 2 Mb pamięci operacyjnej,
- moduł Video-RAM wyświetla na ekranie, niezależnie od procesora, treść pamięci obrazu w wybieranych programowo trybach: znakowym, semigraficznym lub graficznym. Zawartość

pamięci obrazów można kopiować na drukarce,
- pamięć RAM 128/512 kb,
- kontroler peryferii: dyski elastyczne maks. 4 mechanizmy z pojedynczą i podwójną gęstością zapisu /w standardzie dysk 5" z podwójną gęstością K 5600.10 prod. NRD/, drukarka graficzna DGM-82 lub DZM-180,
- sprzęgi: 2 szeregowe typu RS 232C oraz równoległy typu BSJ.

Mikrokomputer jest umieszczony w 8 pakietowej kasecie wbudowanej w monitorze CRT serii Mera 7900.

Oprogramowanie mikrokomputera ComPAN-8 obejmuje:
- system operacyjny zgodny z CP/M 2.2
- kompilatory i interpretery wielu języków /MAKROASEMBLER, BASIC, FORTH, FORTRAN, PASCAL/.
- oprogramowanie aplikacyjne ukierunkowane na implementację relacyjnych baz danych, grafiki komputerowej, terminali inteligentnych.

Opcjonalnie ComPAN może być wyposażony w system zgodny z ISIS-II z makroasemblerem oraz kompilatorami PL/M i FORTRAN.

ComPAN-8 jest szczególnie przydatny dla wspomagania projektowania inżynierskiego, wspomagania pracy operatorów procesów technologicznych, w diagnostyce medycznej, w automatyzacji badań naukowych /np. rejestracje przebiegu eksperymentów i przetwarzaniu ich wyników/ oraz jako terminal inteligentny.

Rozwój mikrokomputera ComPAN-8 zakłada wyposażenie w mikroprocesor 16-to bitowy i interfejsy ploterów.

Producentem mikrokomputera ComPAN-8 są Zakłady Urządzeń Komputerowych MERA-ELZAB.

10.3. ELWRO 523

ELWRO 523 jest mikrokomputerem zbudowanym na bazie mikroprocesora MCY 7880 /analog Intel 8080A/ o konstrukcji zblokowanej /tzn. wszystkie elementy mieszczą się w jednej obudowie/, przeznaczony przede wszystkim do zastosowań biurowych.

Podstawowa konfiguracja ELWRO 523 obejmuje:
- pamięć stała ROM 12 kb,
- pamięć operacyjną RAM 48 kb /z perspektywą rozszerzenia do 64 kb/,

- sprzęgi równoległe i 1 szeregowy,
- klawiatura,
- drukarka ROBOTRON 1152 z transporterem papieru,
- przystawka do kart kontowych ROBOTRON 1161,
- pamięć zewnętrzna na dyskach elastycznych PL x 45D.5 /jedna bądź dwie jednostki/,
- monitor ekranowy /16 x 64/.

Oprogramowanie obejmuje system operacyjny EWOS kompatybilny z CP/M 2.2 wraz z oprogramowaniem podstawowym oraz interpreterami języków ZIM i BASIC.

Prawdopodobnie mikrokomputer ELWRO 523 będzie jedynym, dostępnym powszechnie, mikrokomputerem biurowym dostępnym na rynku.

Producentem rodziny mikrokomputerów ELWRO 500 są Zakłady Elektroniczne ELWRO.

10.4. IMP-85

IMP-85 jest mikrokomputerem opartym na mikroprocesorze INTEL 8085. Pojemność pamięci operacyjnej RAM 16, 32, 48 lub 64 K bajtów. Pojemność pamięci stałej PROM wynosi 4 K bajtów. Pamięć zewnętrzna obejmuje 1 lub 2 mechanizmy dyskowe PL x 45D o pojemności odpowiednio 0,5 M bajta lub 1 M bajtów. Konsola operatorska - wejścia to: klawiatura alfanumeryczna i funkcyjna w układzie QWERTY, konsola operatorska - wyjścia to: monitor ekranowy alfanumeryczny 24 x 80. Dodatkowe interfejsy obejmują: równoległy /do drukarki DZM-180/ i szeregowy V 24 /do modemu transmisji danych/.

Oprogramowanie IMP-85 obejmuje system operacyjny IM PS zgodny z CP/M oraz BASIC i inne języki działające pod tym systemem, a także podstawowe oprogramowanie narzędziowe.

Producentem mikrokomputera IMP-85 jest przedsiębiorstwo polonijno-zagraniczne IMPOL II.

10.5. IMZ-80

Mikrokomputer zbudowany na bazie mikroprocesora Z80 przeznaczony jest do celów dydaktycznych, prac biurowych i obliczeń naukowo-technicznych. Podstawowe wyposażenie obejmuje:

- pamięć operacyjna RAM 64 K bajtów /w tym 4 Kb zajmuje pamięć obrazu i obszar roboczy klawiatury/,
- układ współpracy z magnetofonem,
- klawiatura typu QWERTY,
- układ sterujący monitorem ekranowym /25 wierszy po 40 znaków, zestaw znaków zawiera 96 znaków ASCII i 64 znaki semigraficzne/,
- zegar,
- generator sygnałów akustycznych.

Rozbudowa konfiguracji podstawowej może objąć:

- sprzęg dla drukarki DZM 180,
- sprzęg dla pamięci kasetowej PK3,
- szeregowy sprzęg komunikacyjny V 24,
- układ współpracy z napędami dysków elastycznych 5,25" i 8",
- moduł pamięci stałej EPROM - 16 Kb,
- programator pamięci EPROM.

Oprogramowanie wersji podstawowej stanowią LOADER umieszczony w pamięci stałej, MONITOR, BASIC, MACROASSEMBLER wczytywany z kasety magnetofonowej. Wyposażenie IMZ-80 w pamięć dyskową umożliwia wykorzystanie oprogramowania dodatkowego: systemu operacyjnego IMPS /odpowiednik CP/M/, interpretura języka BASIC /MBASIC/, kompilatorów języków BASIC 80, FORTRAN 80, PASCAL/M.

Producentem mikrokomputera IMZ-80 jest Przedsiębiorstwo Polonijno-Zagraniczne IMPOL II.

10.6. MERA-60 /SM-1633/

MERA-60 jest modularnym systemem mikrokomputerem przeznaczonym do sterowania procesami technologicznymi, obliczeń naukowych, inżynierskich i zawodowych. Podstawowymi elementami architektury systemu MERA-60 są:

- magistrala wewnątrzsystemowa służąca do wymiany informacji między pamięcią a procesorem lub urządzeniami wejścia-wyjścia obejmuje 16 linii danych i adresów oraz 23 linii sterujących,
- mikroprocesor serii K-590, /16 bitowy/,
- pamięć operacyjna - moduły pamięci dynamicznej RAM o pojemnościach 8, 16, 32, 64 kB,

- moduły interfejsowe - podłączenia konsoli operatora, urządzeń transmisji danych, 4 terminali, stacji taśmy papierowej, drukarki DZM-180, dysków elastycznych SP 60M, pamięci kasetowej PK-1, kasety CAMAC ze sterownikiem /nowości: pamięci dyskowe 5 MB SM 5400 /BRL/ SM 5401 /PRL/, pamięć taśmowa PT-305, pisak x-y/.

Konfiguracja bazowa systemu mikrokomputera Mera 60 /SM 1633/ zawiera:

- procesor M2 z pamięcią RAM 4 kB,
- pamięć operacyjna z dwóch modułów typu P2 8 kB lub P3 - 32 kB,
- monitor ekranowy Mera 7953,
- pamięć na dyskach elastycznych SP60M,
- stacja taśmy papierowej SPTP-3,
- drukarka DZM-180,
- kaseta typu CAMAC ze sterownikiem,
- moduł pamięci stałej MPR-60.

Mera-60 wyposażona jest w system operacyjny RT 60 V.04 oraz oprogramowanie transmisji zgodne z protokołem BSC. Może pracować jako terminal inteligentny w sieciach komputerowych z komputerem centralnym typu RIAD lub IBM 360/370. System operacyjny zapewnia korzystanie z następujących języków programowania: MACRO-ASSEMBLER, FORTRAN IV, APL, BASIC. Ponadto MERA-60 dysponuje oprogramowaniem umożliwiającym emulacje terminali IBM 3270 i 3780 wraz z pakietami programów specjalizowanych.

Producentem mikrokomputera jest Centrum Naukowo-Produkcyjne Systemów Sterowania MERA-STER.

10.7. MERITUM

MERITUM jest nieprofesjonalnym mikrokomputerem domowym zbudowanym na bazie mikroprocesora U 880 /analog Z80/. Jego pierwowzorem jest amerykański mikrokomputer TRS-80 Model II produkowany w latach 1978-1983 przez firmę Tandy Radio Shack. Podstawowa konfiguracja mikrokomputera MERITUM obejmuje:

- pamięć operacyjną typu EPROM - 14 kb,
  typu RAM - 16/17 kb,
  pamięć obrazu - 1 kb,
- układ współpracy z magnetofonem jako zewnętrzną pamięcią

masową,
- układ sterujący monitorem ekranowym /16 x 64 znaki lub 16 x 32 znaki wybierane z klawiatury/ semigrafika /o 48 liniach i 128 punktach w linii/
- klawiatura typu QWERTY,
- sprzęgi: szeregowy według standardu RS 232C oraz 3 równoległe we/wy.

Zamierzenia rozwojowe rodziny mikrokomputerów MERITUM idą w kierunku:
- rozszerzenia pamięci do 64 Kb,
- wyposażenia systemu w blok 2 dysków elastycznych.

Oprogramowanie podstawowe MERITUM stanowi 12 kb język BASIC-MERITUM uzupełniony o moduły zarządzające i obsługujące klawiaturę, porty we/wy, wyświetlenie oraz interfejsy zewnętrzne. Rozwój oprogramowania układa wyposażenie mikrokomputera w system operacyjny CP/M /MERITUM II/, kompilator języka PASCAL.

Producentem mikrokomputerów rodziny MERITUM są Zakłady Urządzeń Komputerowych MERA-ELZAB.

10.8. MK4501

Mikrokomputer MK 4501 zbudowany na bazie mikroprocesora Intel 8085A jest mikrokomputerem modułowym domowym przeznaczonym do prac biurowych. Podstawowa konfiguracja mikrokomputera obejmuje:
- pamięć operacyjna 16, 32, 48 lub 64 kb,
- układ sterujący monitorem ekranowym /24 lub 25 wierszy po 80 znaków/,
- klawiatura typu QWERTY,
- układy współpracy z klawiaturą, pamięcią na dyskach elastycznych, drukarka oraz sprzęg szeregowy V 24,
- pamięć zewnętrzna na dyskach elastycznych /jedna lub dwie jednostki PL x 45D/.

Podstawą oprogramowania MK 4501 jest system operacyjny kompatybilny z systemem CP/M wraz z kompilatorem języka FORTRAN-80, interpreterem BASIC 80 oraz kompilatorem BASIC 80.

Mikrokomputer MK 4501 można wykorzystać jako:
- inteligentny terminal w sieciach teleprzetwarzania,
- autonomiczny mikrokomputer do automatyzacji prac biuro-

wych.

Producentem mikrokomputera MK 4501 jest Krakowska Fabryka Aparatów Pomiarowych MERA-KFAP.

10.9. PSPD-90

Programowana Stacja Przetwarzania Danych, bo tak brzmi pełna nazwa mikrokomputera, jest mikrokomputerem zbudowanym na bazie mikroprocesora INTEL 8080A.

W skład podstawowej konfiguracji sprzętowej oferowanej przez producenta wchodzą:

- Pamięć ROM - 1 K bajtów,
- pamięć RAM - 11 K bajtów z czego 8 K bajtów jest dostępne dla programisty /od adresu 1000 do 2FFF szesnastkowo/, a w pozostałej części pamięci operacyjnej rezyduje system operacyjny /w tym 512 bajtów pamięci wizualizowanej/,
- Monitor ekrajowy, wyświetlający 512 znaków w postaci 16 linii po 32 znaki lub opcyjnie 16 linii x 64 znaki,
- Klawiatura alfanumeryczna typu QWERTY,
- Drukarka znakowo-mozaikowa DZM 180,
- Dwie jednostki dyskowe typu PL x 45 D, z których każda zawiera po dwie komory dla 8 calowych dysków elastycznych. Daje to w sumie około 1 M bajtów pamięci dyskowej dostępnej jednocześnie.

Najpopularniejsze zastosowanie mikrokomputera PSPD-go to:

- autonomiczne urządzenie do gromadzenia i przetwarzania danych
- urządzenie do przygotowania danych w ośrodku obliczeniowym /1 dyskietka = 2000 kart perforowanych/ oraz jako konwerter dysk - taśma 0,5 cala
- urządzenia do rejestracji i przetwarzania danych z procesorów technologicznych, tester pakietów elektronicznych
- terminal pracujący on line z systemem komputerowym /przez kanał czytnika - perforatora taśmy papierowej lub multiplekser/.

Oprogramowanie mikrokomputera PSPD-90 obejmuje:

- interpreter języka BASIC 8080,
- Assembler 8080,
- system operacyjny MACRODOS - 90,

- specjalistyczne pakiety programów do gromadzenia i przetwarzania danych /operacje zbierania danych, anulowania i modyfikacji formatów danych, operacje indeksowe szukania danych, operacje wejścia-wyjścia/.

Producentem mikrokomputera PSPD-90 jest Krakowska Fabryka Aparatów Pomiarowych MERA-KFAP.

10.10. ZX81

ZX 81 jest mikrokomputerem domowym zbudowanym na bazie mikroprocesora Z80A. Typowa konfiguracja ZX 81 obejmuje:

- pamięć stałą ROM - 8K
- pamięć RAM - 1K
- klawiatura membranowa 40-sto klawiszowa, zawierająca dla większości przycisków 6 znaczeń,
- czarno-biały odbiornik telewizyjny,
- magnetofon kasetowy jako pamięć masowa.

Możliwości mikrokomputera ZX 81 można zwiększyć poprzez

- dodatkową pamięć RAM 16 lub 64 k,
- drukarki ZX lub GP-50S.

Oprogramowanie obejmuej ASSEMBLER Z-80 i BASIC.

Producentem mikrokomputerów ZX 81 jest firma AMEPROD.

10.11. ZX SPECTRUM

Mikrokomputer domowy firmy SINCLAIR zbudowany na bazie mikroprocesora Z80A o rozbudowanych w stosunku dc ZX 81 możliwościach.

Konfiguracja podstawowa ZX SPECTRUM obejmuje:

- pamięć ROM z interpretorem BASIC /16 kb/,
- pamięć RAM /16 kb lub 48 kb/,
- ULA - tablica logiczna wejścia-wyjścia,
- klawiatura,
- monitor ekranowy,
- magnetofon kasetowy jako pamięć zewnętrzna.

ZX SPECTRUM ma możliwość nagrywania programów, bloków pamięci, bloków danych, obrazów z ekranu. Możliwości sprzętowe mogą być rozszerzone poprzez urządzenia dodatkowe /nie wszystkie dostępne w naszych warunkach/ a mianowicie:

- drukarka,
- interface 1, który łączy funkcje sprzęgu w standardzie RS232 i sprzęgu do lokalnej sieci /w sieci można łączyć do 64 mikrokomputerów SIECTRUM/,
- microdrive - szybka pamięć taśmowa o pojemności około 85 kb,
- interface 2 - przystawka umożliwiająca wykorzystanie dwóch manetek do sterowania,
- syntezatory mowy,
- układ rozpoznawania mowy,
- digitajzer,
- pióro świetlne.

Istnieje ponad 700 programów sprzedawanych dla ZX SPECTRUM /w tym około 80 % to gry/. Programy użytkowe obejmują ASSEMBLER, PASCAL, LISP, LOGO, PROLOG, FORTH.

11. SŁOWNIK TERMINÓW

ADRESOWANIE - określenie części pamięci komputerowej lub miejsca przeznaczenia danych lub programu za pomocą adresu

ALU /Arithmetic-Logic-Unit/ jednostka arytmetyczno-logiczna, blok mikroprocesora służący do wykonywania operacji arytmetycznych i logicznych

ASEMBLER - dokładniej, translator języka symbolicznego jest programem tłumaczącym, który umożliwia stosowanie instrukcji bezpośrednio odpowiadającym rozkazom wewnętrznym mikrokomputera

CAMAC - uniwersalny, modularny system elektroniczny, stosowany do automatyzacji zarówno procesów pomiarowych w eksperymentach naukowych, jak i sterowaniu procesami technologicznymi. System CAMAC zapewnia dwustronne połączenie i przekazywanie informacji pomiędzy układem pomiarowym a komputerem lub innymi urządzeniami sterującymi

CYKL MASZYNOWY - wydzielona część cyklu rozkazowego, w czasie której odbywa się adresowanie i przesyłanie danych lub właściwa realizacja rozkazu

DIGITAJZER - urządzenie umożliwiające wprowadzenie do mikrokomputera rysunki. Położenie specjalnie skonstruowanego wysiągnika przetwarzane jest na cyfrową informację o współrzędnych x, y

DMA /Direct Memory Acces/ - bezpośredni dostęp do pamięci, transmisja między pamięcią a urządzeniem zewnętrznym dokonuje się z pominięciem mikroprocesora

DOSTĘP BEZPOŚREDNI - sposób przetwarzania dyskowego zbioru danych polegający na przetwarzaniu tylko i wyłączenie jednego, wybranego rekordu ze zbioru. Wybór rekordu odbywa się przy pomocy tzw. klucza dostępu czyli pola, które w sposób jednoznaczny określa położenie rekordu w zbiorze

DOSTĘP SEKWENCYJNY - sposób przetwarzania zbioru danych polegający na przetwarzaniu rekordów zgodnie z ich kolejnością fizycznego występowania w zbiorze. Dostęp do n-tego rekordu wymaga przetworzenia m-1 rekordów go poprzedzających

DYSK ELASTYCZNY /dyskietka/ - elastyczny magnetyczny krążek zamknięty w ochronnej kopercie, średnica dysku zwykle wynosi 5 "lub 8" i pojemność odpowiednio 0,5 - 1 M bita lub 2-4 M bitów

DYSK typu WINCHESTER - pamięć dyskowa z zastosowaną nową techniką odczytu i zapisu informacji. Polega ona na odmiennym niż w dyskach tradycyjnych sposobem wprowadzania głowicy na żądaną wysokość lotu /głowice spoczywają na powierzchniach dysków, po uruchomieniu ślizgają się po powierzchniach by przy osiągnięciu 80 % normalnych obrotów unieść się nad powierzchnią/ oraz samą wysokością lotu /0,5 - 1 mikrometra, w tradycyjnych dyskach 2,5 mikrometra/. Daje to możliwość znacznego zagęszczenia zapisu. Cechy charakterystyczne pamięci dyskowych typu WINCHESTER: pojemność 30-300 i więcej Mb na wrzeciono, niewymienność pakietów, gęstość zapisu 2500-5000 bitów/cm, ilość ścieżek 200-400/cm

EPROM /Erasable PROM/,i EAROM /Electrically Alterable ROM/ - pamięć, którą użytkownik może wykorzystać wielokrotnie, wymazywanie informacji dokonuje się poprzez naświetlanie promieniami ultrafioletowymi lub metodami elektrycznymi

GRAFIKA KOMPUTEROWA - obejmuje metody i techniki tworzenia, przedstawiania, przechowywania i przekształcania rysunków za pomocą komputerów

INSTRUKCJA - zdanie języka rozpoznawalne przez asembler, zazwyczaj zbudowana z kodu operacji oraz operandu

INTERPRETER - specjalny program umożliwiający wykonanie programu napisanego w języku wyższego rzędu. Program ten pobiera kolejne instrukcje programu napisanego w języku wyższego rzędu, tłumaczy na kod maszynowy i wykonuje

JĘZYK WEWNĘTRZNY - postać binarna /uzyskana jako wynik pracy asemblera/ bezpośrednio interpretowana przez maszynę

KATALOG DYSKOWY - specjalny obszar na dysku, gdzie zapisane są zbiorcze informacje o zbiorach zapisanych na tym dysku oraz zajmowanych przez nie obszarach

KOMPILACJA - specjalny sposób przetwarzania programu źródłowego napisanego w języku wyższego rzędu mający na celu przygoto-

wanie go do wykonania na danym komputerze. Przetwarzanie to jest najczęściej wielofazowe i wykorzystuje zbiory na nośniku magnetycznym

KONKATENACJA - operacja wykonana na łańcuchach znaków mająca na celu sklejenie tych łańcuchów w jeden łańcuch wynikowy, gdzie następnym znakiem po ostatnim znaku łańcucha pierwszego jest pierwszy znak łańcucha drugiego itd. Konkatenowane łańcuchy nie ulegają zmianie

KONSOLA - urządzenie wejścia/wyjścia służące do komunikowania się z systemem. Najczęściej jest to klawiatura z ekranem lub urządzeniem piszącym

KOPROCESOR - typ mikroprocesora, który zawsze towarzyszy pewnemu mikroprocesorowi centralnemu. Jego zadaniem jest rozszerzenie listy rozkazów mikroprocesora centralnego i wykonywanie tych uzupełniających rozkazów równolegle z programem mikroprocesora centralnego

LICZNIK PROGRAMU /PC - Program Counter/ - 16-to bitowy rejestr zawierający adres wykonywanych rozkazów. Przy pobieraniu każdego bajtu z pamięci zewnętrznej i wchodzącego w skład rozkazu zawartość licznika jest automatycznie zwiększana o 1

MAGISTRALA - wielofunkcyjny zespół szyn

MIKROKOMPUTER - mikroprocesor wraz z pamięciami /RAM i ROM/ oraz układami wejściowo-wyjściowymi, a także oprogramowaniem podstawowym zawartym w ROM

MIKROPROCESOR - jednostka centralna mikrokomputera, wykonany w postaci jednego lub kilku układów scalonych

MOS - układ scalony zbudowany z tranzystorów unipolarnych /Metal-Oxide-Semiconductor czyli Metal-Tlenek-Półprzewodnik/. Pierwsze litery skrótów PMOS, NMOS i CMOS określają typ kanału w tranzystorze. Szybkość działania tranzystora n-MOS jest co najmniej dwukrotnie większe od p-MOS

MYSZKA - urządzenie za pomocą którego operator steruje bezpośrednio poruszaniem się kursora po ekranie. W połączeniu z odpowiednim oprogramowaniem "myszka" pozwala, przenosząc bezpośrednio ruchy operatora, wybierać odpowiednią funkcję, zaznaczać fragmenty danych do usunięcia itd.

OPERAND - część instrukcji programowej szczegółowo określająca wartości dla określonego kodu wykonywanej operacji

PAMIĘĆ DYSKIETKOWA - pamięć dyskowa wyposażona w dyski elastyczne, najczęściej 1-2 jednostki napędu oraz niezbędne układy elektroniczne

PORT - miejsce /droga/ przez które mikroprocesor komunikuje się z urządzeniem zewnętrznym w celu transmisji danych

PROGRAM ŁADUJĄCY - służy do ładowania programu /po kompilacji/ do pamięci operacyjnej. Z postaci absolutnej relokowalnej lub relokowalnej i konsolidującej do postaci absolutnej /adresacja w pamięci/

PROGRAMATOR - urządzenie wpisujące dane lub programy do pamięci

PROM /Programmable ROM/ - pamięć typu ROM, której zawartość jest wpisywana przez użytkownika, a nie przez producenta układów scalonych. Typowe pamięci PROM mają pojemność równą 8-16 K bitów w organizacji 1-2 K x 8 bitów. Informacje można zapisać tylko raz

PRZERWANIE - funkcje mikrokomputerów, spowodowane najczęściej pojawieniem się sygnału zewnętrznego na wyróżnionym wejściu /tzw. sygnał przerwań/ polegające na przejściu od aktualnie wykonywanego programu do programu związanego z obsługą zaistniałej sytuacji

PSEUDOINSTRUKCJA /dyrektywa/ - instrukcje, którym nie odpowiadają żadne rozkazy /maszynowe/, służące do sterowania przebiegiem tłumaczenia programu

RAM /Random Access Memory/ - pamięć umożliwiająca zarówno zapis, jak i odczyt danych

ROM /Read-Only-Memory/ - pamięć stała umożliwiająca tylko odczyt, której zawartość jest wpisywana w trakcie produkcji

ROZDZIELCZOŚĆ - cecha urządzenia graficznego wyrażająca zdolność obrazowania szczegółów rysunku

SYSTEM MIKROKOMPUTEROWY - sprzęt /mikrokomputer z urządzeniami zewnętrznymi/ oraz pełne oprogramowanie /podstawowe i użytkowe/

SYSTEM OPERACYJNY - zespół programów przeznaczonych do zarządza-

nia zasobami systemu, a mianowicie pamięcią, procesami, urządzeniami i programami, danymi

SYSTEM WIELOMIKROPROCESOROWY - system zbudowany z kilku mikroprocesorów na zasadzie specjalizacji mikroprocesorów, bądź też z kilku mikroprocesorów centralnych

STOS - miejsce w pamięci operacyjnej /sterowane licznikiem stosu/ pozwalające wpisywać dane i odczytywać je w odwrotnej kolejności

SZYNA - zespół linii danych lub linii adresowych lub linii kontrolno-sterujących, służących do przesyłania sygnałów między modułami systemu dołączonymi równolegle

UKŁAD PAMIĘCIOWY - układ scalony zawierający pamięć

ULA /Uncomitted Logic Array/ - tablica logiczna spełniająca funkcję układu wejścia-wyjścia w mikrokomputerach Sinclair'a

WOLUMIN - fizyczna jednostka nośnika magnetycznego przystosowana do przechowywania danych

WSKAŹNIK STOSU /SP-Stack Pointer/ - 16-to bitowy rejestr zawierający adres następnego, wolnego miejsca na stosie. Wskaźnik stosu zwiększa się o 1 przy wykonywaniu instrukcji powodujących zapisywanie informacji lub zmniejsza się o 1 przy odczytywaniu informacji ze stosu.

L I T E R A T U R A

1. Barron D.W., Asemblery i programy ładujące, PWN, Warszawa 1982
2. BASIC Compiler User's Manual, Microsoft 1979
3. BASIC-80 Reference Manual, Microsoft, 1979
4. Biuletyny Techniczno-Informacyjne MERA
5. Cellary W., Systemy wielomikroprocesorowe, w pracy Współczesne kierunki rozwoju informatyki, materiały Szkoły Jesiennej PTI, Rydzyna 1984
6. CP/M 2, Alteration Guide, Digital Research, California 1979
7. CP/M 2, Introduction, Digital Research, California 1979
8. Ed: Context Editor for the CP/M Disk System User's Manual, Digital Research, California 1979
9. Grabowski J., Koślarz S., Podstawy i praktyka programowania mikroprocesorów, WNT, Warszawa 1980
10. Informatyka - miesięcznik
11. Kobus R., Szyller J., Mikroprocesory, WP, Warszawa 1984
12. Korczak J., Automatyzacja prac biurowych, materiały konferencji INFOGRYF'83, TNOiK Szczecin, Politechnika Szczecińska, Szczecin 1983
13. Madnick S.E., Donovan J.J., Systemy operacyjne, PWN, Warszawa 1983
14. Marczyński R., Bąkowski P., Sochacki J., Mikroprocesory, WNT Warszawa 1979
15. Misiurewicz P., Układy mikroprocesorowe, WNT, Warszawa 1983
16. 8080/8085 Assembly Lauguage Programming Manual, Siemens nr 98 003 301 B
17. Pelka H., Od algebry połączeń do mikroprocesora, WKiŁ, Warszawa 1980
18. Podręcznik dla użytkownika. Asembler. Tom V, Impol-II
19. Prospekty firmowe