PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
„MERA-PIAP"
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

440

BE 10

Ośrodek Automatyki Elektrycznej

Pracownia Oprogramowania Wieloprocesorowych Systemów Automatyki i Robotów

POUFNE
z wyj. stron 1–8

Główny wykonawca mgr inż. Andrzej Aderek

Wykonawcy mgr inż. Aleksander Ustaszewski
mgr inż. Teresa Zagubień

Konsultant mgr inż. R. D. Tyrcha, mgr K. Najar

Nr zlecenia
9438 Etap 1.

Asembler skrośny dla mikroprocesora 8086 na minikomputer SM-4.
Etap 1. Uproszczona wersja jednomodułowa.

Zleceniodawca praca własna

Pracę rozpoczęto dnia październik 1983 **zakończono dnia** 17.02.1984.

Kierownik Pracowni
mgr inż. A. Aderek

p. o. Z-cy Dyrektora d/s Automatyki
dr inż. T. Gałązka

Kierownik Ośrodka
prof dr inż. T. Missala

Praca zawiera:

8	stron
1	rysunków
	fotografii
	tabel
	tablic
4	tabulogramy

Rozdzielnik:

Egz. 1	BOINTE
Egz. 2	OAE – 83
Egz. 3	
Egz. 4	
Egz. 5	
Egz. 6	

Nr. rejestr. 5197

Analiza deskryptorowa

Oprogramowanie + asembler skrośny + mikroprocesor

Analiza dokumentacyjna

Opracowanie zawiera krótki opis i instrukcję obsługi uproszczonej wersji asemblera skrośnego dla mikroprocesora 8086 na minikomputer SM - 4.

Tytuły poprzednich sprawozdań

681.32:621.372-181.48 Mikroprocesory

UKD

Druk „MERA-PIAP"/TW-62/5000 egz.

2.

SPIS TRESCI

1. WSTEP.

KROTKI OPIS PRODUKTU.

PROGRAM ASM86 JEST UPROSZCZONA WERSJA ASEMBLERA SKROSNEGO DLA MIKROPROCESORA 8086. PROGRAM DZIALA POD SYSTEMEM OPERACYJNYM RSX-11M NA MINIKOMPUTERACH SERII PDP-11 LUB ICH ODPOWIEDNIKACH (NP. SM4).

UZASADNIENIE OPRACOWANIA UPROSZCZONEJ WERSJI ASEMBLERA DLA MIKROPROCESORA 8086.

ZE WZGLEDU NA PILNA POTRZEBE DYSPONOWANIA SRODKIEM DO URUCHAMIANIA OPROGRAMOWANIA NA SPRZECIE MIKROPROCESOROWYM 16-BITOWYM OPARTYM O MIKROPROCESORY TYPU 8086 PODJETO DECYZJE OPRACOWANIA W PIAP ASEMBLERA SKROSNEGO PRACUJACEGO NA MINIKOMPUTERZE SM4, GDYZ MIMO USILNYCH PROB NIE UDALO SIE TAKIEGO PROGRAMU ZAKUPIC.

UPROSZCZONA WERSJA JEST PIERWSZYM ETAPEM TEJ PRACY I BEDZIE TYMCZASOWYM NARZEDZIEM PRZY OPRACOWYWANIU OPROGRAMOWANIA DLA ROBOTOW PRZEMYSLOWYCH ZLOZONYCH. W MIARE SUKCESYWNEGO OPRACOWYWANIA WERSJI DOCELOWEJ ASEMBLERA, OBECNA UPROSZCZONA WERSJA BEDZIE STOPNIOWO UDOSKONALANA. PRZEWIDUJE SIE ROWNIEZ MOZLIWOSC PEWNYCH ZMIAN DODATKOWYCH, JESLI OKAZE SIE, ZE JAKIES UPROSZCZENIA SA ZBYT UCIAZLIWE I ZMNIEJSZAJA EFEKTYWNOSC PRACY Z ASEMBLEREM.

WYMAGANA WIEDZA UZYTKOWNIKA

ZAKLADA SIE, ZE UZYTKOWNIK ZNA POZYCJE [1] I [2] LITERATURY ORAZ POSIADA ELEMENTARNA WIEDZE O WIELODOSTEPNYM SYSTEMIE OPERACYJNYM RSX-11M.

2. OPIS IMPLEMENTACJI.

2.1. WSTEP.

ASEMBLER OMAWIANY TUTAJ JEST UPROSZCZONA WERSJA ORYGINALNEGO MAKROASEMBLERA ROZPOWSZECHNIANEGO PRZEZ FIRME INTEL.

OPIS MAKROASEMBLERA INTELA MOZNA ZNALEZC W [1]. DALEJ ZAKLADAMY, ZE CZYTELNIK ZNA POWYZSZA POZYCJE.

2.2. NAZWY INSTRUKCJI.

MNEMONIKI INSTRUKCJI SA DOKLADNIE TAKIE SAME, JAK W [1].

2.3. IDENTYFIKATORY UZYTKOWNIKA.

POSTAC TAKA, JAK W [1], LECZ TYLKO PIERWSZE 6 ZNAKOW JEST ZNACZACE.

2.4. ARGUMENTY INSTRUKCJI.

2.4.1. NAZWY REJESTROW.

SA TAKIE, JAK W [1]. NAZW REJESTROW AX I AL NIE MOZNA POMIJAC.

2.4.2. ODWOLANIA DO PAMIECI.

POSTAC TAKA, JAK W [1], Z TYM ZE:
A) W WYRAZENIACH REJESTROWYCH CALE WYRAZENIE MUSI BYC ZAWARTE W JEDNEJ PARZE NAWIASOW KWADRATOWYCH ORAZ KOLEJNOSC SKLADNIKOW (JESLI WYSTEPUJA) MUSI BYC NASTEPUJACA:

```
REJESTR BAZOWY
REJESTR INDEKSOWY
STALA
```

NP.

```
[BP+SI+5], [SI-4]
```

B) ZAMIAST KONSTRUKCJI POSTACI "IDENT + N", GDZIE "IDENT" JEST IDENTYFIKATOREM UZYTKOWNIKA A "N" JEST STALA NUMERYCZNA, DOPUSZCZA SIE TYLKO KONSTRUKCJE POSTACI "IDENT [N]".
C) DOPUSZCZALNY JEST JEDYNIE OPERATOR PTR.

2.4.3. ETYKIETY.

DOPUSZCZALNE SA TYLKO TYPU NEAR (NIE MA DYREKTYWY LABEL), JEDNAKZE JEST MOZLIWE GENEROWANIE SKOKOW TYPU FAR POPRZEZ ZASTOSOWANIE OPERATORA PTR.

2.4.4. PROCEDURY.

SA TYLKO TYPU NEAR (ARGUMENT DYREKTYWY PROC JEST IGNOROWANY), A WIEC WSZYSTKIE INSTRUKCJE RET BEDA TYPU NEAR. CHCAC WYGENEROWAC RET TYPU FAR MOZNA UZYC DYREKTYWY DB LUB DW.
MOZLIWE JEST NATOMIAST GENEROWANIE WYWOLAN PROCEDUR TYPU FAR PRZEZ UZYCIE OPERATORA PTR.

2.4.5. OPERATORY.

ZREALIZOWANO OPERATORY SHORT I PTR. UZYCIE OPERATORA SHORT JEST OBOWIAZKOWE W PRZYPADKU SKOKOW WARUNKOWYCH W PRZOD.

2.4.6. STALE NUMERYCZNE.

STALE NUMERYCZNE MOGA BYC W POSTACI BINARNEJ, OKTALNEJ, DZIESIETNEJ I HEKSADECYMALNEJ.

2.5. DYREKTYWY.

ZREALIZOWANO NASTEPUJACE DYREKTYWY:

2.5.1. DYREKTYWA ORG.

JESLI NIE WYSTAPILA, TO ZA POCZATKOWA WARTOSC LICZNIKA MIEJSCA PRZYJMOWANE JEST ZERO.
DYREKTYWA ORG MOZE WYSTEPOWAC WIELOKROTNIE.

2.5.2. DYREKTYWY DB I DW.

DYREKTYWY TE DOPUSZCZAJA CO NAJWYZEJ JEDEN ARGUMENT (STALA NUMERYCZNA).

2.5.3. DYREKTYWY PROC I ENDP.

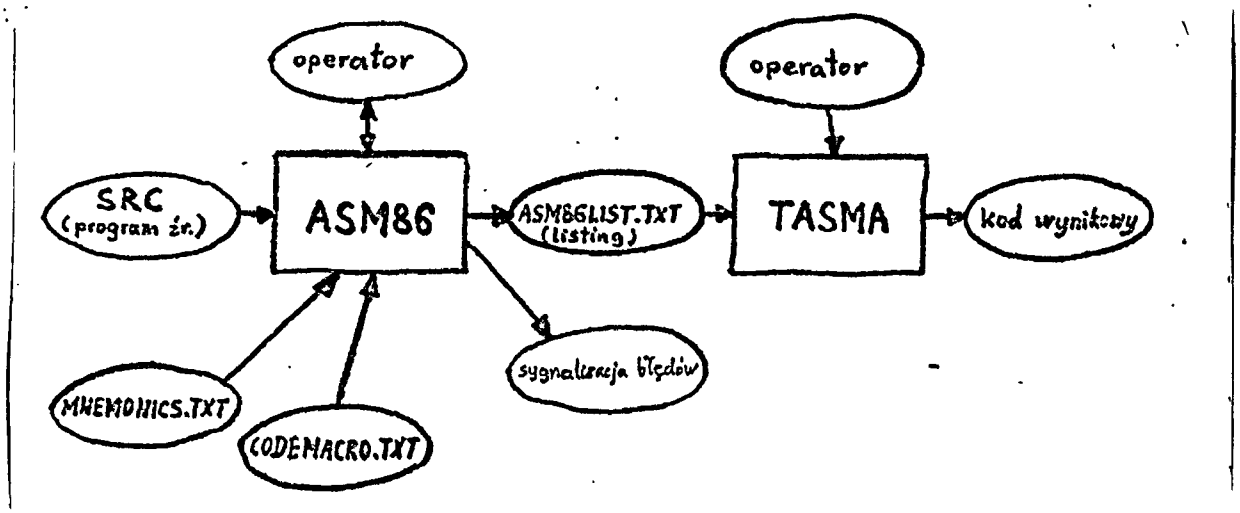
MOZNA DEFINIOWAC PROCEDURY TYPU NEAR. DYREKTYWA ENDP JEST IGNOROWANA.

2.5.4. DYREKTYWA END.

PROGRAM ZRODLOWY MUSI KONCZYC SIE DYREKTYWA END. OPCJONALNY ADRES STARTU JEST IGNOROWANY.

3. STRUKTURA OPROGRAMOWANIA REALIZUJACEGO ASEMBLER.

3.1. PRZEPLYW INFORMACJI.



RYS. 1. PRZEPLYW INFORMACJI W PROCESIE ASEMBLACJI.

OPROGRAMOWANIE REALIZUJACE ASEMBLER SKLADA SIE Z DWOCH PROGRAMOW: ASM86 I TASMA, ORAZ DWOCH PLIKOW POMOCNICZYCH O NAZWACH MNEMONICS.TXT I CODEMACRO.TXT.

PROGRAM ASM86 KODUJE SYMBOLICZNE PRZEDSTAWIENIA INSTRUKCJI MIKROPROCESORA 8086 DO POSTACI BINARNEJ I WLASCIWYM WYNIKIEM JEGO DZIALANIA JEST ZBIOR LISTINGOWY O NAZWIE ASM86LIST.TXT.

PROGRAM TASMA JEST PROGRAMEM POMOCNICZYM I SLUZY DO PRODUKOWANIA TASIEMKI PAPIEROWEJ ZAWIERAJACEJ KOD WYNIKOWY W POSTACI HEKSADECYMALNEJ.

3.2. OPIS PROGRAMU TASMA.

PROGRAM TASMA.PAS BUDUJE ZBIOR WYNIKOWY W FORMACIE HEXADECYMALNYM.

DANE DO PROGRAMU POBIERANE SA ZE ZBIORU ASM86LIST.TXT.JEST TO LISTING PROGRAMU ZRODLOWEGO,KTORY OTRZYMUJEMY W WYNIKU DZIALANIA PROGRAMU ASM86. PROGRAM TASMA.PAS WYPROWADZA NASTEPUJACE REKORDY:

START ADDRESS
EXTENDED ADDRESS
DATA
END OF FILE.

REKORD START ADDRESS JEST OPCJONALNY.WSZYSTKIE WYZEJ WYMIENIONE REKORDY ZAPISYWANE SA DO ZBIORU O NAZWIE PLIK. PO WYKONANIU PROGRAMU TASMA.PAS , ZBIOR PLIK SKLADOWANY JEST NA DYSKU.

4. INSTRUKCJA UZYTKOWANIA.

4.1. START PROGRAMU.

PROGRAM URUCHAMIANY JEST STANDARDOWA DYREKTYWA MCR'A W SPOSOB NASTEPUJACY:

RUN ASM86

PROGRAM ZGLASZA SIE NA TERMINALU TEKSTEM:

PIAP 8086 CROSSASSEMBLER

I NASTEPNIE WYPISUJE TEKST:

LISTING ?[1/0]

CHCAC OTRZYMAC LISTING NALEZY WPROWADZIC LICZBE 1, A JESLI NIE CHCEMY OTRZYMAC LISTINGU - LICZBE 0.

NASTEPNIE PROGRAM W PODOBNY SPOSOB ZAPYTUJE UZYTKOWNIKA, CZY MA BYC (DODATKOWO) WYPROWADZANY LISTING NA TERMINAL.

NASTEPNIE ZOSTAJE WYSWIETLONY NAPIS

OBJECT ?[1/0]

W ODPOWIEDZI NALEZY ZAWSZE WPROWADZIC LICZBE ZERO ! (W PRZYPADKU POMYLKI NALEZY PLIK OBJECT.DAT, KTORY ZOSTANIE UTWORZONY NA KONCIE UZYTKOWNIKA, USUNAC PROCESOREM SYSTEMOWYM PIP).

OD TEGO MONENTU PROGRAM ROZPOCZYNA PRACE.

4.2. PRZEBIEG ASEMBLACJI.

ASEMBLACJA PROGRAMU ZRODLOWEGO ODBYWA SIE W TRZECH PRZEBIEGACH. POCZATEK KAZDEGO PRZEBIEGU SYGNALIZOWANY JEST KOMUNIKATEM NA TERMINALU:

PASS N

GDZIE N - NUMER PRZEBIEGU.

ZAKONCZENIE PRACY PROGRAMU SYGNALIZOWANE JEST KOMUNIKATEM NA TERMINALU:

ASSEMBLY COMPLETE

4.3. DANE WEJSCIOWE.

DANE WEJSCIOWE (PROGRAM ZRODLOWY) POWINIEN ZNAJDOWAC SIE NA PLIKU TEKSTOWYM O NAZWIE SRC. PLIK TEN MOZNA PRZYGOTOWAC PROCESOREM SYSTEMOWYM EDI.

4.4. DANE WYJSCIOWE.

4.4.1. LISTING.

LISTING TWORZONY JEST PODCZAS TRZECIEGO PRZEBIEGU NA PLIKU O NAZWIE ASM86LIST.TXT.
PLIK TEN MOZNA POZNIEJ WYPROWADZIC NA PAPIER PRZY|POMOCY PROCESORA SYSTEMOWEGO PIP.

4.4.2. KOD WYNIKOWY.

KOD WYNIKOWY JEST UMIESZCZONY W ZBIORZE ASM86LIST.TXT. PODANY JEST NA LISTINGU.
CHCAC OTRZYMAC KOD WYNIKOWY NA TASMIE PAPIEROWEJ W FORMACIE HEKSADECYMALNYM (ZGODNYM Z OPISEM W [2]) NALEZY UZYC PROGRAMU TASMA, DOSTARCZANEGO RAZEM Z PROGRAMEM ASM86.

4.4.3. SYGNALIZACJA BLEDOW.

BLEDY SYGNALIZOWANE SA NA MONITORZE W SPOSOB NASTEPUJACY:

```
      "LINIA PROGRAMU ZRODLOWEGO"
      !   ERROR N   IN LINE M
```

GDZIE

"LINIA PROGRAMU ZRODLOWEGO" -- LINIA, W KTOREJ WYSTAPIL BLAD
N -- NUMER BLEDU
M -- NUMER LINII

LISTA BLEDOW ZNAJDUJE SIE W ROZDZIALE 5.

4.5. EFEKTY UBOCZNE.

PODCZAS PRACY PROGRAM TWORZY POMOCNICZY PLIK O NAZWIE RESERVE, KTORY MOZNA USUNAC PROCESOREM SYSTEMOWYM PIP.

4.6. INSTRUKCJA UZYTKOWANIA PROGRAMU TASMA.

PROGRAM TASMA URUCHAMIA SIE POD MCR PRZEZ ROZKAZ:

```
      MCR) RUN TASMA
```

PROGRAM ZGLASZA SIE PRZEZ WYDRUKOWANIE NA TERMINALU:

```
 ***    ASM86-CREATING OBJECT TAPE    ****
[
SEGMENT BASE ADDRESS(FOUR HEXADECIMAL DIGITS)=...
```

NASTEPNIE PROGRAM CZEKA NA WPROWADZENIE CZTERECH CYFR HEKSADECYMALNYCH, KTORE NIE MUSZA BYC ZAKONCZONE LITERA H. WPROWADZAJAC DANE OKRESLAMY BAZOWY ADRES SEGMENTU(USBA). JESLI WPROWADZONE ZNAKI NIE BEDA NALEZALY DO ZBIORU ZNAKOW KODU HEKSADECYMALNEGO, WOWCZAS WYDRUKOWANE ZOSTANIE:

```
[
SYNTAX ERROR
```

I POJAWI SIE PONOWNIE PYTANIE O WARTOSC USBA.
NASTEPNIE GDY WYDRUKOWANE ZOSTANIE:

```
[
WOULD YOU LIKE START ADDRESS RECORD? YES OR NO
```

W ZALEZNOSCI OD ODPOWIEDZI YES/NO BEDZIE LUB TEZ NIE TWORZONY START ADDRESS RECORD. JESLI ODPOWIEDZ BEDZIE YES, WOWCZAS WYDRUKOWANE

ZOSTANA DWA DODATKOWE TEKSTY, PO KTORYCH NALEZY (WEDLUG KOLEJNOSCI WYSTAPIENIA)WPROWADZIC ZAWARTOSC REJESTROW: CODE SEGMENT(CS) I INSTRUCTION POINTER (IP).ZASADY WPROWADZANIA SA TAKIE SAME JAK W PRZYPADKU WPROWADZENIA WARTOSCI USBA.
W TYM MOMENCIE PROGRAM MA JUZ WSZYSTKIE DANE POTRZEBNE DO WYKONANIA. PO ZAKONCZENIU WYKONYWANIA SIE PROGRAMU, UTWORZONY NA DYSKU ZBIOR O NAZWIE PLIK, MOZEMY WYPROWADZIC NA TASME PAPIEROWA PRZY POMOCY DYREKTYWY:
>FLX PP:=PLIK./RS

4.7. INSTALACJA PROGRAMU.

ABY ZAINSTALOWAC PROGRAM NALEZY W DIRECTORY, POD KTORYM BEDZIEMY PRACOWAC, UMIESCIC NASTEPUJACE ZBIORY:

ASM86.TSK
MNEMONICS.TXT
CODEMACRO.TXT
TASMA.TSK

5. LISTA BLEDOW.

1, 8 - POWTORNA DEFINICJA IDENTYFIKATORA
2, 9 - PRZEPELNIENIE SKOROWIDZA NAZW UZYTKOWNIKA
3 - BRAK NAZWY W POLU ETYKIETY DYREKTYWY PROC
4 - BRAK WYRAZENIA W POLU ARGUMENTOW DYREKTYWY ORG
5 - BLEDNY POCZATEK LINII
6 - BRAK MNEMONIKA W POLU MNEMONIKA
7 - DYREKTYWA NIE ZAIMPLEMENTOWANA W TEJ WERSJI ASEMBLERA
10 - BRAK WYRAZENIA W POLU ARGUMENTOW DYREKTYWY ORG
11 - BRAK ':' PO IDENTYFIKATORZE DEFINIUJACYM ETYKIETE
12 - ZA DUZO PREFIKSOW W JEDNEJ LINII
13 - BLEDNA POSTAC ARGUMENTOW
14 - NIE DOPASOWANO CODEMACRO
15 - PREFIKS NIE MOZE MIEC ARGUMENTOW
16 - TRZECI PRZEBIEG POTRZEBUJE WIECEJ BAJTOW DLA KODU INSTRUKCJI NIZ ZOSTALO ZAREZERWOWANE W DRUGIM PRZEBIEGU
17 - NIEZDEFINIOWANY IDENTYFIKATOR
18 - BLAD W OPERATORZE PTR
19 - BLEDNA POSTAC STALEJ NUMERYCZNEJ
20 - BRAK REJESTRU INDEKSOWEGO W WYRAZENIU REJESTROWYM
21 - NIEPRAWIDLOWA POSTAC WYRAZENIA REJESTROWEGO
22 - KONTYNUACJA ARGUMENTU ZA WYRAZENIEM REJESTROWYM
23 - BRAK ZNAKU ']'
24 - JESLI ARGUMENT JEST STALA TO MOZE ZA NIM WYSTAPIC TYLKO PRZECINEK, SREDNIK LUB KONIEC LINII
25 - PO WYRAZENIU REJESTROWYM POWINIEN WYSTAPIC PRZECINEK, SREDNIK LUB KONIEC LINII
26 - ZA DUZO PREFIKSOW LUB SAME PREFIKSY

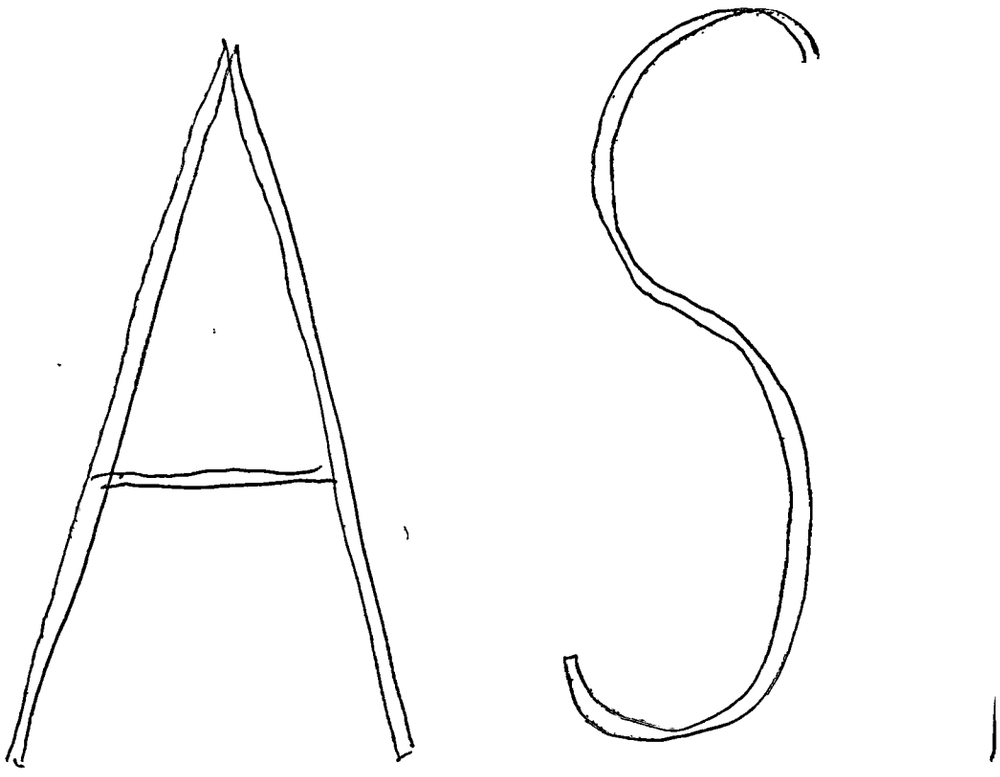
6. SPIS LITERATURY.

[1] "8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL FOR 8080/8085 - BASED DEVELOPMENT SYSTEMS",
MANUAL ORDER NUMBER: 121623-001 REV. A
COPYRIGHT 1980, INTEL CORPORATION, 3065 BOWERS AVENUE, SANTA CLARA, CA 95051

[2] "MCS-86 ABSOLUTE OBJECT FILE FORMATS", AN INTEL TECHNICAL SPECIFICATION, ORDER NUMBER: 9800821A

[3] WISNIEWSKA J., "WYBRANE INFORMACJE O SYSTEMIE RSX-11M", OSRODEK OBLICZENIOWY, INSTYTUT INFORMATYKI UW

```
IEF142I    STEP WAS EXECUTED - COND CODE 0000
IEF285I    SYS84031.T143144.RF000.KBROLA.LOADSET          DELETED
IEF285I    VOL SER NOS= PIAP01.
IEF285I    SYS84031.T143144.RF000.KBROLA.GOSET            PASSED
IEF285I    VOL SER NOS= PIAP01.
IEF285I    SYS84031.T143144.RF000.KBROLA.SYSUT1           DELETED
IEF285I    VOL SER NOS= SYSFF1.
IEF285I    SYS84031.T143144.SF000.KBROLA.R0000002         SYSOUT
IEF285I    VOL SER NOS= PIAP01.
IEF373I STEP /LKED     / START 84031.1434
IEF374I STEP /LKED     / STOP  84031.1435 CPU    0MIN 00.98SEC MAIN  90K LCS    0K
XXGO     EXEC  PGM=*.LKED.SYSLMOD,COND=((8,LT,ASM),(4,LT,LKED))          00000300
//GO.DDIN DD DSNAME=ALA,UNIT=2400,VOL=SER=SPEC00,DISP=OLD,               00002070
//    LABEL=(1,NL)                                                      00002080
//GO.DDOUT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)           00002090
//GO.SYSUDUMP DD SYSOUT=A                                              00002100
//                                                                     00002110
IEF236I ALLOC. FOR KBROLA   GO,      EX
IEF237I 132   ALLOCATED TO PGM=*.DD
IEF237I 201   ALLOCATED TO DDIN
IEF237I 132   ALLOCATED TO DDOUT
IEF237I 132   ALLOCATED TO SYSUDUMP
```



```
E          ;D     6
          (*                                                    *)
          (*                  A S M 8 6                         *)
          (*                                                    *)
          (******************************************************)

PROGRAM AS8086 (TTY);

(*-------------------------------------------------------------------------)
(       CROSSASSEMBLER DLA MIKROPROCESORA INTEL 8086
(  JEST TO UPROSZCZONA WERSJA ASEMBLERA DLA TEGO MIKROPROCESORA.
(-------------------------------------------------------------------------*)

(*********************************************************************)
(*                                                                   *)
(*               S T A L E     G L O B A L N E                     *)
(*                                                                   *)
(*********************************************************************)

CONST
  EXTLINEWIDTH = 73;  (* = LINEWIDTH + 1 *)
  FULL = 0;
  IDLENGTH = 6; (* W WERSJI DOCELOWEJ 31 *)
  LINEWIDTH = 72;  (* DLUGOSC LINII PROGRAMU ZRODLOWEGO *)
  MAXUSER = 128; (* ROZMIAR SLOWNIKA NAZW UZYTKOWNIKA *)
  MLEN = 6;  (* MAKSYMALNA ILOSC ZNAKOW JAKA MOZE ZAWIERAC MNEMONIK *)
        (*  UWAGA: WARTOSC MLEN JEST TYMCZASOWA. DOCELOWO BEDZIE MLEN *)
        (*           >= 7   *)
  MLENPACK = 2;  (* DLUGOSC TABLICY TYPU MNEM, W KTOREJ PRZECHOWYWANE *)
                 (* SA MNEMONIKI W POSTACI SPAKOWANEJ                 *)
        (*  UWAGA: DOCELOWO MLENPACK BEDZIE INNE  *)
  NOCM = 271;            (* LICZBA CODEMACRO'SOW *)
  NOMNEM = 145;                  (* LICZBA MNEMONIKOW INSTRUKCJI I DYREKTYW *)
  maxmn = 3; { maksymalna liczba mnemoników w jednej linii }
(*********************************************************************)
(*                                                                   *)
(*               T Y P Y     G L O B A L N E                       *)
(*                                                                   *)
(*********************************************************************)

TYPE
  ARGKIND = RECORD
              SPECIFIER: PACKED ARRAY [0..7] OF BOOLEAN;
              MODIFIER: PACKED ARRAY [0..3] OF BOOLEAN;
              RANGE: PACKED ARRAY [0..7] OF BOOLEAN;
              REG: 0..7;  (* OKRESLA POLE REG *)
              OFFSET: INTEGER;
              ANONYMOUS: BOOLEAN;
              BX, BP, SI, DI, CS, SS, DS, ES: BOOLEAN;
              SEGFIX: RECORD
                        YES: BOOLEAN;
                        WHAT: 0..3
                      END
            END;
  ARGMTS = RECORD
             NUMBER: 0..2;  (* LICZBA ARGUMENTOW *)
             ARG: ARRAY [1..2] OF ARGKIND
           END;
  ALFA = PACKED ARRAY [1..10] OF CHAR;
  BYTE86 = 0..255;
  CHARPOS = INTEGER;  (* WERSJA BEZ DYREKTYWY EQU *)
  DECFILE = RECORD
              DEV: ARRAY [1..4] OF CHAR;
```

```
            DIR: RECORD
                   GROUP: 0..377B;
                   MEMBER: 0..377B
                 END;
            FILENAME: ARRAY [1..9] OF CHAR;
            FILETYPE: ARRAY [1..3] OF CHAR;
            VERSION: 0..77777B
          END;
  IDENTIFIER = PACKED ARRAY [1..IDLENGTH] OF CHAR;
  IDLOCATION = INTEGER;  (* POLOZENIE NAZWY W SLOWNIKU UZYTKOWNIKA *)
  MNEM = ARRAY [1..MLENPACK] OF INTEGER;
        (*-------------------------------------------------------------*)
        (*      DO PRZECHOWYWANIA MNEMONIKA INSTRUKCJI LUB DYREKTYWY   *)
        (*  W POSTACI SPAKOWANEJ.                                      *)
        (*-------------------------------------------------------------*)
  REXVAL = RECORD
             BX, BP, SI, DI, CONS: BOOLEAN;
             VALUE: INTEGER
           END;   (* WYNIK ROZPOZNANIA REGISTER EXPRESSION *)
  TYPE86 = (IBYTE, WORD, DWORD, QUORD, TBYTE, NEAR, FAR );
  WORD86 = ARRAY [1..2] OF BYTE86;
  WORDCHAR = PACKED ARRAY[1..2] OF CHAR;

(*************************************************************)
(*                                                           *)
(*              Z M I E N N E   G L O B A L N E              *)
(*                                                           *)
(*************************************************************)

VAR
  ACTPARAMETERS: ARGMTS;
        (*-------------------------------------------------------------*)
        (* INFORMACJA O PARAMETRACH AKTUALNYCH INSTRUKCJI              *)
        (*-------------------------------------------------------------*)

  ARGS: ARRAY [1..NOCM] OF INTEGER;
        (*-------------------------------------------------------------*)
        (*  TABLICA ARGS ZAWIERA ZAKODOWANA W JEDNYM SLOWIE            *)
        (*  INFORMACJE O ARGUMENTACH FORMALNYCH POSZCZEGOLNYCH         *)
        (*  CODEMACRO. SPOSOB ZAKODOWANIA JEST NASTEPUJACY:            *)
        (*                                                             *)
        (*  BIT 15: OKRESLA KTORY ARGUMENT POSIADA RANGE               *)
        (*      0 - DRUGI ARGUMENT (SOURCE)                            *)
        (*      1 - PIERWSZY ARGUMENT (DESTINATION)                    *)
        (*   LUB JESLI LICZBA ARGUMENTOW WYNOSI ZERO TO BIT 15 = 1     *)
        (*  I BITY 1-0 ROWNE 0 OZNACZAJA, ZE DANE CODEMACRO OKRESLA    *)
        (*  PREFIKS ( NP. LOOP, REP )                                  *)
        (*                                                             *)
        (*      BITY 14-12: RANGE. RODZAJ ZAKRESU JEST ZAKODOWANY      *)
        (*  W TYM POLU JAKO LICZBA Z PRZEDZIALU [0, 7] W SPOSOB        *)
        (*  NASTEPUJACY:  (-128,127) - 0, (0, 63) - 1, (3) - 2,        *)
        (*  (1) - 3, (CL) - 4, (DX) - 5, (ES) - 6, (SS,DS) - 7         *)
        (*                                                             *)
        (*      BITY 11-10: MODIFIER DRUGIEGO ARGUMENTU, SPOSOB        *)
        (*  KODOWANIA:  B - 0, W - 1, D - 2, BRAK MODIFIER'A - 3       *)
        (*                                                             *)
        (*      BITY 9-8: MODIFIER PIERWSZEGO ARGUMENTU                *)
        (*                                                             *)
        (*      BITY 7-5: SPECIFIER DRUGIEGO ARGUMENTU, SPOSOB KODO-   *)
        (*  WANIA:  A - 0, C - 1, D - 2, E - 3, M - 4, R - 5, S - 6,   *)
        (*  X - 7.                                                     *)
        (*                                                             *)
        (*      BITY 4-2: SPECIFIER PIERWSZEGO ARGUMENTU               *)
```

```
        (*                                                              *)
        (*      BITY 1-0: LICZBA ARGUMENTOW, SPOSOB KODOWANIA:          *)
        (*  BRAK ARGUMENTOW - 0, JEDEN ARGUMENT - 1, DWA ARGUMENTY - 2, *)
        (*  DWA ARGUMENTY, Z KTORYCH JEDEN POSIADA RANGE - 3 (WTEDY     *)
        (*  BIT 15 OKRESLA, KTORY ARGUMENT POSIADA RANGE)               *)
        (*                                                              *)
        (*  UWAGA: DLA DWOCH INSTRUKCJI (AAD, AAM) ARGS [K] ZAWIERA     *)
        (*  NA BITACH 9-2 DRUGI BAJT KODU OPERACYJNEGO.                 *)
        (*--------------------------------------------------------------*)

CDMCR: TEXT;
        (*--------------------------------------------------------------*)
        (*  ZAWIERA OPIS CODEMACRO'SOW POTRZEBNY DO WYPELNIENIA         *)
        (*  TABLIC ARGS, CODING, FINISH                                 *)
        (*--------------------------------------------------------------*)

CM: INTEGER;
        (*--------------------------------------------------------------*)
        (* NUMER AKTUALNIE ROZPATRYWANEGO CODEMACRO                     *)
        (*--------------------------------------------------------------*)

                maxmn
CMS: ARRAY [1..2] OF INTEGER;
        (*--------------------------------------------------------------*)
        (*  DOPASOWANE NUMERY CODEMACRO'SOW (ZERO OZNACZA, ZE NIE       *)
        (*  DOPASOWANO)                                                 *)
        (*--------------------------------------------------------------*)

CODING: ARRAY [1..NOCM] OF INTEGER;
        (*--------------------------------------------------------------*)
        (*      TABLICA CODING OKRESLA PROCEDURE KODUJACA DANE          *)
        (*  CODEMACRO ORAZ CZESC LUB CALY KOD OPERACYJNY. INFORMACJA    *)
        (*  TA UMIESZCZONA JEST W ELEMENCIE TABLICY W SPOSOB NASTEPU-   *)
        (*  JACY:  BITY 15-11: NUMER PROCEDURY KODUJACEJ, BITY 10-8:    *)
        (*  DRUGA (TRZYBITOWA) CZESC KODU OPERACYJNEGO (UMIESZCZANEGO   *)
        (*  W BAJCIE MODRM NIEKTORYCH INSTRUKCJI), BITY 7-0: PIERWSZA   *)
        (*  CZESC KODU OPERACYJNEGO (UMIESZCZANA W PIERWSZYM BAJCIE     *)
        (*  KODU INSTRUKCJI).                                           *)
        (*--------------------------------------------------------------*)

COMMENT: BOOLEAN;
        (*--------------------------------------------------------------*)
        (*  JESLI COMMENT = TRUE TO LINIA JEST KOMENTARZEM              *)
        (*--------------------------------------------------------------*)

CURRCHAR: CHAR;
        (*--------------------------------------------------------------*)
        (*  BIEZACY ZNAK. NA OGOL CURRCHAR = LINE [CURRPOS]. JEDNAKZE   *)
        (*  JESLI JEST ZAIMPLEMENTOWANA DYREKTYWA EQU TO CURRCHAR       *)
        (*  MOZE BYC ZNAKIEM Z DEFINICJI CIAGU ZNAKOW.                  *)
        (*--------------------------------------------------------------*)

CURRDATE: ALFA;
        (*--------------------------------------------------------------*)
        (*  AKTUALNA DATA                                               *)
        (*--------------------------------------------------------------*)

CURRPOS: INTEGER;
        (*--------------------------------------------------------------*)
        (*  WSKAZNIK BIEZACEGO ZNAKU W WIERSZU (LINE [CURRPOS] JEST     *)
        (*  BIEZACYM ZNAKIEM                                            *)
        (*--------------------------------------------------------------*)

DAYTIME: ALFA;
```

```
    (*-------------------------------------------------------------------*)
    (*  AKTUALNY CZAS                                                    *)
    (*  (TYPE ALFA = PACKED ARRAY [1..10] OF CHAR)                       *)
    (*-------------------------------------------------------------------*)

  ENTRY: ARRAY [1..NOMNEM] OF INTEGER;
    (*-------------------------------------------------------------------*)
    (*        TABLICA ENTRY W ZALEZNOSCI OD TEGO CZY DANY                *)
    (*        MNEMONIK ODPOWIADA INSTRUKCJI CZY TEZ DYREKTYWIE           *)
    (*        OKRESLA POCZATEK LANCUCHA CODEMCROSOW DLA DANEJ            *)
    (*        INSTRUKCJI LUB TEZ PROCEDURE OBSLUGUJACA DANA              *)
    (*        DYREKTYWE.                                                 *)
    (*-------------------------------------------------------------------*)

  EOLINE: BOOLEAN;
    (*-------------------------------------------------------------------*)
    (*  OKRESLA CZY JEST KONIEC LINII (CURRPOS>LINELENGTH)               *)
    (*-------------------------------------------------------------------*)

  EOPROGRAM: BOOLEAN;
    (*-------------------------------------------------------------------*)
    (*  ZMIENNA LOGICZNA USTAWIANA DO WARTOSCI TRUE WIERSZEM Z           *)
    (*  DYREKTYWA "END"                                                  *)
    (*-------------------------------------------------------------------*)

  ERR: BOOLEAN;
    (*-------------------------------------------------------------------*)
    (*  ZMIENNA USTAWIANA PRZEZ PROCEDURE ERROR. INFORMUJE, ZE           *)
    (*  WYSTAPIL BLAD.                                                   *)
    (*  NA POCZATKU PRZETWARZANIA LINII USTAWIAMY ERR = FALSE            *)
    (*-------------------------------------------------------------------*)

  FATAL: BOOLEAN;
    (*-------------------------------------------------------------------*)
    (*  OZNACZA BLAD KONCZACY PRACE ASEMBLERA                            *)
    (*-------------------------------------------------------------------*)

  FILL: TEXT;
    (*-------------------------------------------------------------------*)
    (*  ZAWIERA NAZWY MNEMONIKOW I ZWIAZANE Z NIMI DANE                  *)
    (*-------------------------------------------------------------------*)

  FINISH: PACKED ARRAY [1..NOCM] OF BOOLEAN;
    (*-------------------------------------------------------------------*)
    (*        FINISH [K] OKRESLA CZY DANE CODEMACRO JEST KONCEM          *)
    (*        LANCUCHA CODEMACRO DLA DANEGO MNEMONIKA:                   *)
    (*          FINISH [K] = TRUE OZNACZA, ZE CODEMACRO O NUMERZE        *)
    (*                       K JEST KONCEM LANCUCHA                      *)
    (*        UWAGA: KIERUNEK DODATNI W LANCUCHU JEST OKRESLONY          *)
    (*               ODWROTNIE DO WZROSTU K.                             *)
    (*-------------------------------------------------------------------*)

  I: INTEGER;
    (*-------------------------------------------------------------------*)
    (*  DO OBSLUGI PETLI                                                 *)
    (*-------------------------------------------------------------------*)

  ID: IDENTIFIER;
    (*-------------------------------------------------------------------*)
    (*  AKTUALNIE ROZPATRYWANY IDENTYFIKATOR                             *)
    (*-------------------------------------------------------------------*)

  INS: PACKED ARRAY [1..NOMNEM] OF BOOLEAN;
```

```
        (*-------------------------------------------------------------*)
        (*      OKRESLA CZY DANY MNEMONIK JEST MNEMONIKIEM INSTRUKCJI  *)
        (*  (INS [K] = TRUE), CZY TEZ DYREKTYWY (INS [K] = FALSE)      *)
        (*-------------------------------------------------------------*)

J: INTEGER;
        (*-------------------------------------------------------------*)
        (*  DO OBSLUGI PETLI                                           *)
        (*-------------------------------------------------------------*)

LABELFIELD: RECORD
              LAB, NAME: BOOLEAN;
              CONT: IDENTIFIER
            END;
        (*-------------------------------------------------------------*)
        (*  OPISUJE POLE ETYKIETY. JESLI LABFIELD.LAB = TRUE TO JEST   *)
        (*  ETYKIETA W POLU ETYKIETY I LABELFIELD.CONT ZAWIERA TA      *)
        (*  ETYKIETE (BEZ ':'). JESLI LABELFIELD.NAME = TRUE TO JEST   *)
        (*  NAZWA W POLU ETYKIETY I LABELFIELS.CONT ZAWIERA TA NAZWE.  *)
        (*-------------------------------------------------------------*)

LC: INTEGER;
        (*-------------------------------------------------------------*)
        (*  LOCATION COUNTER - LICZNIK BIEZACEGO MIEJSCA               *)
        (*-------------------------------------------------------------*)
LC1: integer;    {wartość lc na początku linii (potrzebna dla celów procedury objout)}
LINE: ARRAY [1..EXTLINEWIDTH] OF CHAR;
        (*-------------------------------------------------------------*)
        (*  W LINE JEST PRZECHOWYWANY BIEZACY WIERSZ. DODANO DODATKO-  *)
        (*  MIEJSCE ZA KONCEM LINII (LINE [LINEWIDTH + 1]) ABY UPRO-   *)
        (*  SCIC W WIELU PRZYPADKACH SPRAWDZENIE WYSTAPIENIA WARUNKU   *)
        (*  KONCA LINII. PO WCZYTANIU BIEZACEGO WIERSZA, OBCIETEGO     *)
        (*  EWENTUALNIE DO WIELKOSCI LINEWIDTH, DODAJE SIE             *)
        (*  LINE [LINELENGTH + 1] := ' ', GDZIE LINELENGTH JEST        *)
        (*  DLUGOSCIA BIEZACEGO WIERSZA.                               *)
        (*-------------------------------------------------------------*)

LINENUMBER: INTEGER;
        (*-------------------------------------------------------------*)
        (*  NUMER KOLEJNY LINII PROGRAMU ZRODLOWEGO                    *)
        (*-------------------------------------------------------------*)

LINELENGTH: INTEGER;
        (*-------------------------------------------------------------*)
        (*  DLUGOSC BIEZACEGO WIERSZA                                  *)
        (*-------------------------------------------------------------*)

LISTFILE: TEXT;
        (*-------------------------------------------------------------*)
        (*  NA TEN ZBIOR WYPROWADZANY JEST WLASCIWY LISTING.           *)
        (*-------------------------------------------------------------*)

LISTING: BOOLEAN;
        (*-------------------------------------------------------------*)
        (*  OKRESLA CZY WYPROWADZAC LISTING (NA ZBIOR LISTFILE).       *)
        (*  LISTING = TRUE OZNACZA, ZE TAK, NATOMIAST                  *)
        (*  LISTING = FALSE OZNACZA, ZE WLASCIWY LISTING NIE BEDZIE    *)
        (*  PRODUKOWANY (JEDNAKZE NIE WYKLUCZA LISTINGU NA TERMINALU)  *)
        (*-------------------------------------------------------------*)

LNONPAGE: INTEGER;
        (*-------------------------------------------------------------*)
        (*  ZMIENNA LNONPAGE ZAWIERA AKTUALNA ILOSC WYDRUKOWANYCH      *)
```

```
        (*  WIERSZY NA AKTUALNEJ STRONIE. ZMIENNA TA JEST UAKTU-      *)
        (*  ALNIANA PRZEZ PROCEDURE LIST.                              *)
        (*-------------------------------------------------------------*)

LOC: WORD86;
        (*-------------------------------------------------------------*)
        (*  ADRES KODU WYNIKOWEGO                                      *)
        (*-------------------------------------------------------------*)

LSTFLSPEC: DECFILE;
        (*-------------------------------------------------------------*)
        (*  OKRESLA PARAMETRY ZBIORU LISTINGOWEGO W SYSTEMIE           *)
        (*  ZBIOROW FILES-11.                                          *)
        (*-------------------------------------------------------------*)

MN: ARRAY [1..NOMNEM] OF MNEM;
        (*-------------------------------------------------------------*)
        (*  W TEJ TABLICY PRZECHOWYWANE SA WSZYSTKIE MNEMONIKI         *)
        (*  INSTRUKCJI I DYREKTYW                                      *)
        (*-------------------------------------------------------------*)

MNEMADDR: ARRAY [1..maxmn] OF INTEGER;
        (*-------------------------------------------------------------*)
        (*  ADRESY MNEMONIKOW (W TABLICY MN), KTORE WYSTAPILY W LINII  *)
        (*-------------------------------------------------------------*)

MNEMNUMBER: 0..maxmn;
        (*-------------------------------------------------------------*)
        (*  LICZBA MNEMONIKOW W LINII INSTRUKCJI                       *)
        (*-------------------------------------------------------------*)

NOPRINTED: BOOLEAN;
        (*-------------------------------------------------------------*)
        (*  OKRESLA CZY BIEZACA LINIA BYLA DRUKOWANA CZY NIE           *)
        (*  (Z POWODU BLEDU)                                           *)
        (*-------------------------------------------------------------*)

OBJ: RECORD
       NOBYTES: 0..9;
       CONTENTS: ARRAY [1..9] OF BYTE86
     END;
        (*-------------------------------------------------------------*)
        (*  OBJ ZAWIERA BAJTY KODU WYNIKOWEGO. W JEDNEJ LINII          *)
        (*  MOZNA MAKSYMALNIE DRUKOWAC 9 BAJTOW KODU WYNIKOWEGO.       *)
        (*  WYDRUK JEST W POSTACI HEXADECYMALNEJ.                      *)
        (*-------------------------------------------------------------*)

OBJECT: FILE OF INTEGER;
        (*-------------------------------------------------------------*)
        (*  NA TEN ZBIOR JEST WYPROWADZANY W PIERWSZEJ KOLEJNOSCI KOD  *)
        (*  WYNIKOWY (W POSTACI HEXADECYMALNEJ). JEST TO ZBIOR DYSKO-  *)
        (*  WY. POZNIEJ MOZNA KOD WYNIKOWY WYPERFOROWAC NA TASMIE      *)
        (*  PAPIEROWEJ.                                                *)
        (*  DOCELOWO BEDZIE DO PLIK TYPU TEXT.                         *)
        (*-------------------------------------------------------------*)

OBJNO: BOOLEAN;
        (*-------------------------------------------------------------*)
        (*  ZMIENNA UZYWANA PRZEZ PROCEDURE LIST. OBJNO = TRUE OZNA-   *)
        (*  CZA, ZE WYPROWADZAC LISTING BEZ KODU WYNIKOWEGO (NP.       *)
        (*  W PRZYPADKU LINII KOMENTARZA LUB NIEKTORYCH DYREKTYW)      *)
        (*-------------------------------------------------------------*)
```

```
OUT: BOOLEAN;
      (*-------------------------------------------------------------*)
      (*  OKRESLA CZY PRODUKOWAC KOD WYNIKOWY                        *)
      (*-------------------------------------------------------------*)

PAGELENGTH: INTEGER;
      (*-------------------------------------------------------------*)
      (*  ILOSC WIERSZY NA JEDNEJ STRONIE (RAZEM Z NAGLOWKIEM)       *)
      (*-------------------------------------------------------------*)

PAGENUM: INTEGER;
      (*-------------------------------------------------------------*)
      (*  NUMER KOLEJNEJ STRONY                                      *)
      (*-------------------------------------------------------------*)

PAGEWIDTH: INTEGER;
      (*-------------------------------------------------------------*)
      (*  SZEROKOSC LISTINGU (W ZNAKACH)                             *)
      (*-------------------------------------------------------------*)

PASS: 1..4;
      (*-------------------------------------------------------------*)
      (*  OKRESLA KTORY PRZEBIEG PRACY ASEMBLERA JEST AKTUALNIE      *)
      (*  WYKONYWANY.                                                *)
      (*-------------------------------------------------------------*)

PROCLEN: INTEGER;
      (*-------------------------------------------------------------*)
      (*  ZMIENNA GLOBALNA OKRESLAJACA Z JAKA PROCEDURA MAMY W DA-   *)
      (*  NEJ CHWILI DO CZYNIENIA (NEAR LUB FAR),                    *)
      (*       PROCLEN = 0      GDY PROCEDURA TYPU NEAR              *)
      (*       PROCLEN = 255    GDY PROCEDURA TYPU FAR               *)
      (*-------------------------------------------------------------*)

RESERVE: FILE OF INTEGER;
      (*-------------------------------------------------------------*)
      (*       PRZEJSCIOWY PLIK, W KTORYM PAMIETANA JEST ILOSC BAJ-  *)
      (*  TOW ZAREZERWOWANA DLA KAZDEJ INSTRUKCJI (LINIA PROGRAMU    *)
      (*  ZRODLOWEGO) PODCZAS DRUGIEGO PRZEBIEGU.                    *)
      (*-------------------------------------------------------------*)

SOURCE: TEXT;
      (*-------------------------------------------------------------*)
      (*  PLIK ZAWIERAJACY PROGRAM ZRODLOWY                          *)
      (*-------------------------------------------------------------*)

SRCFLSPEC: DECFILE;
      (*-------------------------------------------------------------*)
      (*  SPECYFIKACJA PLIKU ZAWIERAJACEGO PROGRAM ZRODLOWY          *)
      (*-------------------------------------------------------------*)

TERMINAL: BOOLEAN;
      (*-------------------------------------------------------------*)
      (*  OKRESLA CZY WYSYLAC ROWNIEZ LISTING NA TERMINAL.           *)
      (*-------------------------------------------------------------*)

TRUNC: BOOLEAN;
      (*-------------------------------------------------------------*)
      (*  W PRZYPADKU WYSYLANIA LISTINGU NA TERMINAL TRUNC = TRUE    *)
      (*  POWODUJE OBCINANIE LINII DO DLUGOSCI 72 ZNAKOW, NATOMIAST  *)
      (*  TRUNC = FALSE POWODUJE WYDRUK DLUZSZYCH LINII W            *)
      (*  KOLEJNYCH WIERSZACH, Z TYM ZE WIERSZE KONTNUACJI ROZPO-    *)
      (*  CZYNAJA SIE OD KOLUMNY GDZIE ZACZYNA SIE LISTING.          *)
```

```
        (*--- ---------- ---------- ---------- ---------- ---------- ---*)

  USERIDENT: ARRAY [1..MAXUSER] OF RECORD
                                     NAME: ARRAY [1..2] OF INTEGER;
                                     OFFSET: INTEGER;
                                     ITYPE: TYPE86;
                                     DETOFFSET: BOOLEAN
                                   END;
        (*---------- ---------- ---------- ---------- ---------- ---------- ---*)
        (*  UPROSZCZONY SLOWNIK IDENTYFIKATOROW UZYTKOWNIKA.                    *)
        (*  OPIS POL:                                                           *)
        (*    NAME       - NAZWA UZYTKOWNIKA PRZECHOWYWANA W POSTACI            *)
        (*                 SPAKOWANEJ, SAME SPACJE OZNACZAJA, ZE               *)
        (*                 MIEJSCE JEST WOLNE.                                 *)
        (*    DETOFSET   - POKAZUJE CZY OFFSET ZOSTAL JUZ ZDEFINIOWANY         *)
        (*---------- ---------- ---------- ---------- ---------- ---------- ---*)

        (*****************************************)
        (*                                       *)
        (*              C A N 3                  *)
        (*                                       *)
        (*****************************************)

FUNCTION CAN3 (VAR X: ARRAY [INTEGER] OF CHAR;
               K: INTEGER): INTEGER;

(*****************************************************************)
(*                                                               *)
(*  FUNKCJA KODUJE TRZY ZNAKI X [K], X [K+1], X [K+2] W JEDNO SLOWO   *)
(*  MASZYNOWE PDP-11 (LICZBA INTEGER)                            *)
(*                                                               *)
(*****************************************************************)

VAR
  I, S: INTEGER;

        (*****************************************)
        (*                                       *)
        (*              C A N 1                  *)
        (*                                       *)
        (*****************************************)

FUNCTION CAN1 (X: CHAR): INTEGER;

(*****************************************************************)
(*                                                               *)
(*  FUNKCJA CAN1 PRZYPORZADKOWUJE KOLEJNYM ZNAKOM Z TABLICY (1)  *)
(*                                                               *)
(*               0123456789                                      *)
(*               9?@ABCDEFG              (1)                     *)
(*               HIJKLMNOPQ                                      *)
(*               RSTUVWXYZ                                       *)
(*                                                               *)
(*  KOLEJNE LICZBY CALKOWITE 0, 1, 2, ...                        *)
(*  NP. CAN (' ') = 0, CAN1 ('0') = 1, ITD.                      *)
(*  JESLI ZNAK NIE NALEZY DO ZBIORU ZNAKOW Z TABLICY (1) TO JEST *)
(*  ON KODOWANY JAKO SPACJA.                                     *)
(*                                                               *)
(*****************************************************************)

BEGIN  (* CAN1 *)
  IF (X >= '0') AND (X <= '9') THEN
    CAN1 := ORD (X) - ORD ('0') + 1
```

```
    ELSE IF (X >= '?') AND (X <= 'Z') THEN
      CAN1 := ORD (X) - ORD ('?') + 11
    ELSE IF X = ' ' (* ORD (X) = 137B *) THEN
      CAN1 := 39
    ELSE CAN1 := 0
  END  (* CAN1 *);

  BEGIN  (* CAN3 *)
    CAN3 := CAN1 (X [K]) + 40 * CAN1 (X [K+1]) + 1600 * CAN1 (X [K+2])
  END  (* CAN3 *);

  (*****************************************************************************)

          (*********************************************)
          (*                                           *)
          (*              M N E M F I L L              *)
          (*                                           *)
          (*********************************************)

  PROCEDURE MNEMFILL;

  (*****************************************************************************)
  (*                                                                           *)
  (*      PROCEDURA WYPELNIA TABLICE MN MNEMONIKAMI INSTRUKCJI I DYRE-         *)
  (*  KTYW (W PORZADKU ALFABETYCZNYM) ORAZ TABLICE INS, MOWIACA O TYM          *)
  (*  CZY MNEMONIK JEST INSTRUKCJA (INS [K] = TRUE GDY MN [K] JEST             *)
  (*  MNEMONIKIEM INSTRUKCJI), ORAZ TABLICE ENTRY ZAWIERAJACA ADRESY           *)
  (*  POCZATKOW LANCUCHOW CODEMACRO'SOW DLA POSZCZEGOLNYCH MNEMONIKOW          *)
  (*  INSTRUKCJI LUB NUMERY PROCEDUR OBSLUGUJACYCH DYREKTYWY DLA MNEMO-        *)
  (*  NIKOW BEDACYCH DYREKTYWAMI ASM86.                                        *)
  (*      POTRZEBNE DANE PROCEDURA CZERPIE Z PLIKU TEKSTOWEGO O NAZWIE         *)
  (*  ZEWNETRZNEJ MNEMONICS.TXT, A O NAZWIE WEWNETRZNEJ FILL: TEXT.           *)
  (*  JEDNA LINIA TEGO PLIKU ZAWIERA INFORMACJE O JEDNYM MNEMONIKU            *)
  (*  W POSTACI NASTEPUJACEJ:                                                  *)
  (*              MN      INS     ENTRY                                        *)
  (*  UWAGA: POLE MN (MNEMONIK) KONCZY SIE SPACJA, A ENTRY PRZEDSTAWIO-        *)
  (*  NE JEST W POSTACI 0/1.                                                   *)
  (*  NP.                                                                      *)
  (*      ADD     1    152                                                     *)
  (*      DB      0    3                                                       *)
  (*                                                                           *)
  (*  STALE GLOBALNE:                                                          *)
  (*      MLEN                                                                 *)
  (*      NOMNEM                                                               *)
  (*      MLENPACK                                                             *)
  (*                                                                           *)
  (*  ZMIENNE GLOBALNE:                                                        *)
  (*      FILL                                                                 *)
  (*                                                                           *)
  (*  PROCEDURY GLOBALNE:                                                      *)
  (*      CAN3                                                                 *)
  (*                                                                           *)
  (*****************************************************************************)

  VAR
    I, L, TMPINS: INTEGER;
    ID: IDENTIFIER;
    OK: BOOLEAN;

  PROCEDURE READID;

  VAR
    I: INTEGER;
```

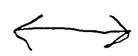
```
    EOMNEM: BOOLEAN;

  BEGIN  (* READID *)
    (* PRZESKOCZ POCZATKOWE SPACJE *)
    WHILE FILL  = ' ' DO GET (FILL);

    (* WCZYTAJ IDENTYFIKATOR MNEMONIKA *)
    EOMNEM := FALSE;
    FOR I := 1 TO MLEN DO
    IF (FILL  <> ' ') AND NOT EOMNEM THEN READ (FILL, ID [I])
    ELSE BEGIN EOMNEM := TRUE; ID [I] := ' ' END
  END  (* READID *);

  BEGIN  (* MNEMFILL *)
    OK := TRUE;
    FOR I := 1 TO NOMNEM DO
    BEGIN  (* FOR I *)
      IF EOF (FILL) THEN
      BEGIN
        WRITELN (TTY, ' EOF (MNEMONICS.TXT)');
        OK := FALSE
      END
      ELSE IF OK THEN
      BEGIN
        READID;
        READ (FILL, TMPINS, ENTRY [I]);
        IF TMPINS = 0 THEN INS [I] := FALSE
        ELSE INS [I] := TRUE;
        FOR L := 1 TO MLENPACK DO MN [I] [L] := CAN3 (ID, 3 * L - 2)
      END
    END  (* FOR I *)
  END  (* MNEMFILL *);

(*****************************************************************************)

          (*********************************************)
          (*                                           *)
          (*              F I L L C O D E N            *)
          (*                                           *)
          (*********************************************)

  PROCEDURE FILLCODEN;

  (*****************************************************************************)
  (*                                                                           *)
  (*       PROCEDURA WYPELNIA TABLICE ARGS, CODING, FINISH. DANE PROCE-        *)
  (*   DURA CZERPIE Z DYSKOWEGO PLIKU CODEMACRO.TXT (ZMIENNA CDMCR).           *)
  (*   PLIK TEN ZAWIERA NOCM WIERSZY. WIERSZ I-TY ZAWIERA INFORMACJE PO-       *)
  (*   TRZEBNA DO WYPELNIENIA ARGS [I], CODING [I], FINISH [I].                *)
  (*       FORMAT DANYCH:                                                      *)
  (*   I   ARGS [I]   CODING [I]   FINISH [I]                                  *)
  (*   Z TYM, ZE WARTOSC FINISH [I] JEST PRZEDSTAWIONA JAKO 0 I 1.
  (*   NP.
  (*   52 12773 11 0
  (*                                                                           *)
  (*****************************************************************************)

  VAR
    I, K, TMP: INTEGER;

  BEGIN  (* FILLCODEN *)
    FOR I := 1 TO NOCM DO
    BEGIN  (* FOR I *)
```

```
    READ (CODMCR, K, ARGS [I], CODING [I], TMP);
    IF TMP = 0 THEN FINISH [I] := FALSE
    ELSE FINISH [I] := TRUE
  END  (* FOR I *)
END  (* FILLCODEM *);

(*****************************************************************************)

        (*********************************************)
        (*                                           *)
        (*              R E A D L I N E              *)
        (*                                           *)
        (*********************************************)

PROCEDURE READLINE;

(*****************************************************************************)
(*                                                                           *)
(*      PROCEDURA WCZYTUJE LINIE PROGRAMU ZRODLOWEGO DO ZMIENNEJ             *)
(*  LINE. LINIA WCZYTYWANA JEST Z PLIKU O SPECYFIKACJI ZAWARTEJ W            *)
(*  ZMIENNEJ SRCFLSPEC: DECFILE, (PLIK JEST REPREZENTOWANY PRZEZ             *)
(*  ZMIENNA SOURCE: TEXT)                                                    *)
(*      W MOMENCIE WEJSCIA DO PROCEDURY READLINE PLIK SOURCE JEST            *)
(*  W TAKIM STANIE, ZE WSKAZNIK POZYCJI JEST NA POCZATKU LINII DO            *)
(*  PRZECZYTANIA.                                                            *)
(*      JESLI LINIA JEST DLUZSZA NIZ LINEWIDTH TO JEST OBCINANA,             *)
(*  TAK ZE TYLKO PIERWSZE LINEWIDTH ZNAKOW ZNAJDZIE SIE W LINE.              *)
(*      ZAWSZE LINE [LINEWIDTH + 1] = ' '.                                   *)
(*      JESLI LINIA ZRODLOWA JEST KROTSZA, TO KONCOWE MIEJSCA W LINE         *)
(*  WYPELNIANE SA SPACJAMI A ZMIENNA LINELENGTH PRZYJMUJE WARTOSC            *)
(*  ROWNA AKTUALNEJ DLUGOSCI WIERSZA PROGRAMU ZRODLOWEGO.                    *)
(*                                                                           *)
(*                                                                           *)
(*****************************************************************************)

VAR
  I: INTEGER;

BEGIN  (* READLINE *)
  I := 1;
  WHILE NOT (EOLN (SOURCE) OR (I > LINEWIDTH)) DO
  BEGIN  (* WHILE *)
    READ (SOURCE, LINE [I]);
    I := I + 1
  END  (* WHILE *);
  READLN (SOURCE);
  LINELENGTH := I - 1;
  IF LINELENGTH < LINEWIDTH THEN
  FOR I := LINELENGTH + 1 TO LINEWIDTH DO LINE [I] := ' ';
  IF LINELENGTH >= 1 THEN
  BEGIN
    CURRPOS := 0;
    EOLINE := FALSE
  END
  ELSE EOLINE := TRUE
END  (* READLINE *);

(*****************************************************************************)

        (*********************************************)
        (*                                           *)
        (*              D O L I N E                  *)
        (*                                           *)
```

```
(*************************************************)

PROCEDURE DOLINE;

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA DOLINE PRZETWARZA JEDNA LINIE PROGRAMU ZRODLOWEGO. *)
(*  JESLI LINIA BYLA LINIA KOMENTARZA TO USTAWIA COMMENT = TRUE, *)
(*  W PRZECIWNYM PRZYPADKU DOLINE WYKONUJE NASTEPUJACE CZYNNOSCI: *)
(*   1) JESLI WYSTAPILA ETYKIETA LUB NAZWA W POLU ETYKIETY - WCZYTUJE *)
(*      NAZWE LUB ETYKIETE DO ZMIENNEJ LABELFIELD. ZMIENNA       *)
(*      LABELFIELD OKRESLA TEZ CZY WYSTAPILA ETYKIETA CZY NAZWA. *)
(*   2) BADA POLE MNEMONIKA. JESLI JEST TAM IDENTYFIKATOR TO SPRAW- *)
(*      DZA CZY JEST TO MNEMONIK DYREKTYWY LUB INSTRUKCJI I W ZALEZ- *)
(*      NOSCI OD WYNIKU SPRAWDZENIA PRZEKAZUJE STEROWANIE DO PROCEDU- *)
(*      RY "INSTRUCTN" LUB "DIRECTIVE" LUB SYGNALIZUJE BLAD.    *)
(*                                                               *)
(*  ZMIENNE GLOBALNE:                                            *)
(*      LABELFIELD: RECORD                                       *)
(*                    LAB, NAME: BOOLEAN;                        *)
(*                    CONT: IDENTIFIER                           *)
(*                  END                                          *)
(*                                                               *)
(*      COMMENT: BOOLEAN                                         *)
(*                 -  OKRESLA CZY LINIA JEST LINIA KOMENTARZA    *)
(*                                                               *)
(*      CURRCHAR: CHAR  -  BIEZACY ZNAK                          *)
(*                                                               *)
(*      INS: PACKED ARRAY [1..NOMNEM] OF BOOLEAN                 *)
(*                                                               *)
(*  PROCEDURY GLOBALNE:                                          *)
(*      DIRECTIVE                                                *)
(*      ERROR                                                    *)
(*      IDENT                                                    *)
(*      INSTRUCTN                                                *)
(*      MNEMONIC                                                 *)
(*      NEXTITEM                                                 *)
(*                                                               *)
(*****************************************************************)

VAR
  ID: IDENTIFIER;
        (*-------------------------------------------------------*)
        (*  AKTUALNIE ROZPATRYWANY IDENTYFIKATOR                 *)
        (*-------------------------------------------------------*)
  ACTMNEM: INTEGER;
        (*-------------------------------------------------------*)
        (*  ADRES AKTUALNIE ROZPATRYWANEGO MNEMONIKA             *)
        (*-------------------------------------------------------*)

        (***********************************************)
        (*                                             *)
        (*              E R R O R                      *)
        (*                                             *)
        (***********************************************)

PROCEDURE ERROR (N: INTEGER);

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA DRUKUJE INFORMACJE O BLEDZIE NA TERMINALU W FORMACIE *)
(*  NASTEPUJACYM:                                                *)
(*
```

```
(*       "BIEZACA LINIA"                    (TYLKO PIERWSZE 72 ZNAKI)      *)
(*       !  ERROR N IN LINE M                                             *)
(*                                                                        *)
(*   GDZIE:                                                               *)
(*       N  -  NUMER BLEDU                                                *)
(*       M  -  NUMER BIEZACEJ LINII                                       *)
(*                                                                        *)
(*   ZMIENNE GLOBALNE:                                                    *)
(*                                                                        *)
(*       LINENUMBER: INTEGER                                              *)
(*       NOPRINTED: BOOLEAN                                               *)
(*       LINE: ARRAY [1..EXTLINEWIDTH] OF CHAR                            *)
(*       ERR: BOOLEAN                                                     *)
(*                                                                        *)
(**************************************************************************)

VAR
  I: INTEGER;

BEGIN  (* ERROR *)
  ERR := TRUE;
  IF NOPRINTED THEN
  BEGIN
    WRITELN (TTY, ' ');
    FOR I := 1 TO LINELENGTH DO WRITE (TTY, LINE [I]);
    WRITELN (TTY);
    NOPRINTED := FALSE
  END;
  WRITELN (TTY, '  !  ERROR ', N:4, '      IN LINE ', LINENUMBER:5)
END  (* ERROR *);

(**************************************************************************)
(*                                                                        *)
(*                    D E C A N 3                                         *)
(*                                                                        *)
(**************************************************************************)

PROCEDURE DECAN3 (X: INTEGER;
                  VAR Y: ARRAY [INTEGER] OF CHAR;
                  K: INTEGER);

(**************************************************************************)
(*                                                                        *)
(*  X    - ZAWIERA ZAKODOWANE TRZY ZNAKI                                  *)
(*  Y      DO TEJ TABLICY ZOSTANIE PRZEKAZANY WYNIK (DO Y.[K],            *)
(*         Y [K+1], Y [K+2])                                              *)
(*                                                                        *)
(**************************************************************************)

  VAR
    I: INTEGER;

(**************************************************************************)
(*                                                                        *)
(*                    D E C A N 1                                         *)
(*                                                                        *)
(**************************************************************************)

FUNCTION DECAN1 (X: INTEGER): CHAR;

(**************************************************************************)
```

```
(*                                                                  *)
(*  JEST FUNKCJA ODWROTNA DO CAN1. JESLI LICZBA X NIE PRZEDSTAWIA   *)
(*  DOPUSZCZALNEGO ZNAKU, TO WYNIKIEM FUNKCJI DECAN1 JEST SPACJA.   *)
(*                                                                  *)
(******************************************************************)

BEGIN  (* DECAN1 *)
  IF (X >= 1) AND (X <= 10) THEN  (* 0 - 9 *)
  DECAN1 := CHR (60B + X - 1)
  ELSE IF (11 <= X) AND (X <= 36) THEN
  DECAN1 := CHR (77B +X -11)
  ELSE IF X = 39 THEN
  DECAN1 := ' '
  ELSE DECAN1 := ' ';
END  (* DECAN1 *);

BEGIN  (* DECAN3 *)
  FOR I := 1 TO 3 DO
  BEGIN
    Y [K+I-1] := DECAN1 (X MOD 40);
    X := X DIV 40
  END
END  (* DECAN3 *);

(******************************************************************)

        (*****************************************)
        (*                                       *)
        (*              E Q U A L                *)
        (*                                       *)
        (*****************************************)

FUNCTION EQUAL (VAR ID: IDENTIFIER; STRING S): BOOLEAN;

(******************************************************************)
(*                                                                *)
(*  EQUAL = TRUE  IFF  ID = S                                     *)
(*                                                                *)
(*                                                                *)
(*  TYPY GLOBALNE:                                                *)
(*      IDENTIFIER                                                *)
(*                                                                *)
(*  PROCEDURY GLOBALNE:                                           *)
(*      DIFF                                                      *)
(*                                                                *)
(******************************************************************)

        (*****************************************)
        (*                                       *)
        (*              D I F F                  *)
        (*                                       *)
        (*****************************************)

FUNCTION DIFF (VAR ID: IDENTIFIER; STRING S): INTEGER;

(******************************************************************)
(*                                                                *)
(*  DIFF = -1    IF ID < S                                        *)
(*          0    IF ID = S                                        *)
(*         +1    IF ID > S                                        *)
(*                                                                *)
(*                                                                *)
(*  STALE GLOBALNE:                                               *)
```



```
function equal (var id: identifier; string s): boolean;

label
  1;

var max,
  i: integer;
  ok: boolean;

begin { equal }
  max := size(s);
  i := 1;
1: if id[i] = s[i-1] then
     if i = idlength then ok := true
     else if id[i+1] = '␣' then
       if i = max then ok := true
       else if s[i] = '␣' then ok := true
       else ok := false
     else if i = max then ok := false
     else if s[i] = '␣' then ok := false
     else
     begin
       i := i+1;
       goto 1
     end
   else ok := false;

  equal := ok
end {equal};
```

```
(*      IDLENGTH                                                        *)
(*                                                                      *)
(************************************************************************)

  VAR
    I, MAX, TDIFF: INTEGER;

  BEGIN  (* DIFF *)
    MAX := SIZE (S);
    IF MAX > IDLENGTH THEN MAX := IDLENGTH;
    TDIFF := 0;
    I := 1;
    REPEAT
      IF ID [I] < S [I - 1] THEN TDIFF := -1
      ELSE IF ID [I] > S [I-1] THEN TDIFF := 1;
      I := I + 1;
    UNTIL (TDIFF <> 0) OR (I > MAX);
    DIFF := TDIFF
  END  (* DIFF *);

  BEGIN  (* EQUAL *)
    EQUAL := (DIFF (ID, S) = 0)
  END  (* EQUAL *);

(************************************************************************)

        (*********************************************)
        (*                                           *)
        (*                F I N D                    *)
        (*                                           *)
        (*********************************************)

PROCEDURE FIND (USERNAME: IDENTIFIER;
                VAR LOC: IDLOCATION;
                VAR F: BOOLEAN);

(************************************************************************)
(*                                                                      *)
(*  PROCEDURA FIND SZUKA NAZWY "USERNAME" W SKOROWIDZU UZYTKONIKA.      *)
(*  JESLI NAZWA ZOSTANIE ZNALEZIONA TO ZMIENNA F PRZYJMIE WARTOSC       *)
(*  TRUE, A ZMIENNA LOC JEST ADRESEM TEJ NAZWY W SLOWNIKU NAZW UZYT-    *)
(*  KOWNIKA. JESLI NAZUA NIE ZOSTANIE ZNALEZIONA TO ZMIENNA F PRZYJ-    *)
(*  MIE WARTOSC FALSE A ZMIENNA LOC BEDZIE WSKAZYWAC WOLNE MIEJSCE,     *)
(*  NA KTORE MOZNA WPROWADZIC NAZWE WRAZ Z JEJ ATRYBUTAMI PROCEDURA     *)
(*  ENTER. JEZELI LOC=FULL (CONST FULL=0) TO NIE MA JUZ MIEJSCA         *)
(*  W SLOWNIKU I NAZWY NIE MOZNA WSTAWIC.                               *)
(*                                                                      *)
(*  PROCEDURY GLOBALNE:                                                 *)
(*       EQUAL                                                          *)
(*       DECAN3                                                         *)
(*                                                                      *)
(************************************************************************)

        (*  WERSJA UPROSZCZONA - PRZESZUKIWANIE LINIOWE *)

VAR
  Y: ARRAY [0..5] OF CHAR;
  FOUND, DIREND: BOOLEAN;

FUNCTION EMPTY: BOOLEAN;

VAR
  I: INTEGER;
```

```
  TEMP: BOOLEAN;

BEGIN  (* EMPTY *)
  TEMP := TRUE;
  FOR I := 0 TO 5 DO TEMP := TEMP AND (Y [I] = ' ');
  EMPTY := TEMP
END  (* EMPTY *);

BEGIN  (* FIND *)
  LOC := 1;
  FOUND := FALSE;
  DIREND := FALSE;
  WHILE (NOT FOUND) AND (LOC <= MAXUSER) AND (NOT DIREND) DO
  BEGIN  (* WHILE *)
    DECAN3 (USERIDENT [LOC].NAME [1], Y, 0);
    DECAN3 (USERIDENT [LOC].NAME [2], Y, 3);
    IF EMPTY THEN DIREND := TRUE
    ELSE IF EQUAL (USERNAME, Y) THEN FOUND := TRUE
    ELSE LOC := LOC + 1
  END  (* WHILE *);
  F := FOUND;
  IF LOC > MAXUSER THEN LOC := 0
END  (* FIND *);

(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*              E N T E R                        *)
        (*                                               *)
        (*************************************************)

PROCEDURE ENTER (ID: IDENTIFIER;
                 TP: TYPE86;
                 DETO: BOOLEAN;
                 O: INTEGER;
                 VAR RES: INTEGER);

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA ENTER WPROWADZA IDENTYFIKATOR UZYTKOWNIKA DO SLOWNIKA   *)
(*  IDENTYFIKATOROW UZYTKOWNIKA.                                 *)
(*  IDENTYFIKATOR UZYTKOWNIKA MOZE BYC:                          *)
(*      ETYKIETA  NEAR                                           *)
(*       --"--    FAR                                           *)
(*                                                               *)
(*      ZMIENNA BYTE, WORD, DWORD, QWORD, TBYTE                  *)
(*                                                               *)
(*      NAZWA PROCEDURY NEAR, FAR                                *)
(*                                                               *)
(*      (NAZWA SEGMENTU)                                         *)
(*                                                               *)
(*                                                               *)
(*  OPIS PARAMETROW:                                             *)
(*   ID                 -  IDENTYFIKATOR, KTORY BEDZIE WSTAWIANY *)
(*   TP                 -  TYP IDENTYFIKATORA ( IBYTE=1, WORD=2, DWORD=3,
(*                  QWORD=4, TBYTE=5, NEAR=6, FAR=7 )            *)
(*   DETO       -  OKRESLA CZY OFFSET PRZEDSTAWIA WAZNY OFFSET   *)
(*   O          -  OFFSET IDENTYFIKATORA                         *)
(*   RES        -  WYNIK WSTAWIANIA:                             *)
(*                 RES > 0   -   IDENTYFIKATOR ZOSTAL WSTAWIONY  *)
(*                               (W TYM PRZYPADKU JEST TO ADRES W *)
(*                               W TABLICY)                      *)
(*
```

```
(*                  -1          -   IDENTYFIKATOR JUZ SIE ZNAJDOWAL      *)
(*                                  POPRZEDNIO W TABLICY (KONFLIKT IDEN- *)
(*                                  TYFIKATOROW)                         *)
(*                  -2          -   PRZEPELNIENIE SLOWNIKA IDENTYFIKA- \ *)
(*                                  TOROW UZYTKOWNIKA                    *)
(*                                                                       *)
(*                                                                       *)
(*  PROCEDURY GLOBALNE:                                                  *)
(*       FIND                                                            *)
(*       CAN3                                                            *)
(*                                                                       *)
(*************************************************************************)

VAR
  LOC: IDLOCATION;
  FOUND: BOOLEAN;

BEGIN  (* ENTER *)
  FIND (ID, LOC, FOUND);
  IF FOUND THEN RES := -1
  ELSE
  BEGIN  (* NIE ODNALEZIONO *)
    IF LOC <> 0 THEN
    WITH USERIDENT [LOC] DO
    BEGIN  (* WITH USERIDENT [LOC] *)
      NAME [1] := CAN3 (ID, 1);
      NAME [2] := CAN3 (ID, 4);
      ITYPE := TP;
      DETOFFSET := DETO;
      IF NOT DETOFFSET THEN OFFSET := 0 (* else offset := 0 *)
    END  (* WITH USERIDENT [LOC] *)
    ELSE RES := -2
  END  (* NIE ODNALEZIONO *)
END  (* ENTER *);

(*************************************************************************)

        (*************************************************)
        (*                                               *)
        (*                G E T C H A R                  *)
        (*                                               *)
        (*************************************************)

PROCEDURE GETCHAR;

(*************************************************************************)
(*                                                                       *)
(*  PROCEDURA GETCHAR POBIERA DO ZMIENNEJ GLOBALNEJ CURRCHAR ZNAK        *)
(*  WSKAZYWANY PRZEZ CURRPOS (LINE [CURRPOS]) I PRZESUWA WSKAZNIK        *)
(*  POZYCJI O 1 DO PRZODU. JESLI WSKAZNIK POZYCJI PRZESUNIE SIE ZA       *)
(*  KONIEC LINII TO ZMIENNA EOLINE PRZYBIERZE WARTOSC TRUE, JESLI        *)
(*  PRZY WEJSCIU DO PROCEDURY GETCHAR JEST EOLINE = TRUE TO WARTOSC      *)
(*  CURRPOS NIE ZMIENIA SIE A DO CURRCHAR ZOSTANIE PODSTAWIONA SPACJA    *)
(*                                                                       *)
(*                                                                       *)
(*  STALE GLOBALNE:                                                     *)
(*       LINELENGTH                                                      *)
(*                                                                       *)
(*  ZMIENNE GLOBALNE:                                                   *)
(*       CURRPOS                                                         *)
(*       EOLINE                                                          *)
(*       LINE                                                            *)
(*                                                                       *)
(*************************************************************************)
```

```
BEGIN  (* GETCHAR *)
  IF EOLINE THEN CURRCHAR := ' '
  ELSE
  BEGIN
    CURRPOS := CURRPOS + 1;
    IF CURRPOS <= LINELENGTH THEN
      IF LINE [CURRPOS] = CHR (11B) THEN CURRCHAR := ' '
      ELSE CURRCHAR := LINE [CURRPOS]
    ELSE
    BEGIN
      CURRCHAR := ' ';
      EOLINE := TRUE
    END
  END
END  (* GETCHAR *);

(*****************************************************************)

        (*********************************************)
        (*                                           *)
        (*              N E X T I T E M              *)
        (*                                           *)
        (*********************************************)

PROCEDURE NEXTITEM;

(*****************************************************************)
(*                                                               *)
(*      JESLI BIEZACY ZNAK (CURRCHAR) JEST SPACJA LUB ZNAKIEM TABULA- *)
(*  CJI TO NEXTITEM PRZESUWA WSKAZNIK POZYCJI CURRPOS (UAKTUALNIAJAC  *)
(*  JEDNOCZESNIE CURRCHAR) TAK DLUGO AZ ZOSTANIE NAPOTKANY ZNAK ROZNY *)
(*  OD SPACJI LUB TABULACJI POZIOMEJ LUB WYSTAPI KONIEC LINII. JESLI  *)
(*  WYSTAPI KONIEC LINII TO NEXTITEM USTAWI EOLINE = TRUE ORAZ        *)
(*  CURRCHAR = ' '.                                                  *)
(*                                                                   *)
(*                                                                   *)
(*  ZMIENNE GLOBALNE:                                                *)
(*      EOLINE                                                       *)
(*      CURRCHAR                                                     *)
(*                                                                   *)
(*  PROCEDURY GLOBALNE:                                                 *)
(*      GETCHAR                                                      *)
(*                                                                   *)
(*****************************************************************)

BEGIN  (* NEXTITEM *)
  WHILE NOT EOLINE AND (CURRCHAR = ' ') DO GETCHAR
END  (* NEXTITEM *);

(*****************************************************************)

        (*********************************************)
        (*                                           *)
        (*              P O S                        *)
        (*                                           *)
        (*********************************************)

PROCEDURE POS (K: CHARPOS);

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA POS PRZESUWA WSKAZNIK POZYCJI CURRPOS NA POZYCJE K.    *)
```

```
(*  UAKTUALNIA ZMIENNA CURRCHAR (BIEZACY ZNAK). JESLI PRZY WEJSCIU     *)
(*  DO POS BYL KONIEC LINII (EOLINE = TRUE) A "K" JEST TAKIE, ZE       *)
(*  1 <= K <= LINELENGTH TO BEDZIE EOLINE = FALSE. JESLI PRZESUNIECIE  *)
(*  BEDZIE ZA KONIEC LINII (K > LINELENGTH) TO POD CURRCHAR ZOSTANIE   *)
(*  PODSTAWIONA SPACJA I ZOSTANIE USTAWIONE EOLINE = TRUE, A CURRPOS   *)
(*  = LINELENGTH + 1. JESLI K < 1 TO ZOSTANIE WYKONANE POS (1)         *)
(*                                                                     *)
(*                                                                     *)
(*  STALE GLOBALNE:                                                    *)
(*       LINELENGTH                                                    *)
(*                                                                     *)
(*  ZMIENNE GLOBALNE:                                                  *)
(*       CURRCHAR                                                      *)
(*       CURRPOS                                                       *)
(*       EOLINE                                                        *)
(*       LINE                                                          *)
(*                                                                     *)
(*********************************************************************)

BEGIN  (* POS *)
  IF (K >= 1) AND (K <= LINELENGTH) THEN
  BEGIN
    CURRPOS := K;
    EOLINE := FALSE;
    CURRCHAR := LINE [CURRPOS]
  END
  ELSE IF K > LINELENGTH THEN
  BEGIN
    CURRPOS := LINELENGTH + 1;
    CURRCHAR := ' ';
    EOLINE := TRUE
  END
  ELSE
  BEGIN
    CURRPOS := 1;
    CURRCHAR := LINE [CURRPOS];
    EOLINE := FALSE
  END
END  (* POS *);

(*********************************************************************)

        (*************************************************)
        (*                                               *)
        (*                  S A V E                      *)
        (*                                               *)
        (*************************************************)

PROCEDURE SAVE (VAR POSITION: CHARPOS);

(*********************************************************************)
(*                                                                   *)
(*  PROCEDURA ZAPAMIETUJE W ZMIENNEJ POSITION POZYCJE BIEZACEGO      *)
(*  ZNAKU (CURRPOS)                                                  *)
(*                                                                   *)
(*********************************************************************)

BEGIN  (* SAVE *)
  POSITION := CURRPOS
END  (* SAVE *);

(*********************************************************************)
```

```
        (*********************************************)
        (*                                           *)
        (*              B E T W E E N                *)
        (*                                           *)
        (*********************************************)

FUNCTION BETWEEN (A, X, B: CHAR): BOOLEAN;

(*****************************************************************)
(*                                                               *)
(*  BETWEEN PRZYJMUJE WARTOSC TRUE GDY A <= X <= B               *)
(*                                                               *)
(*****************************************************************)

  BEGIN  (* BETWEEN *)
    BETWEEN := ( X >= A ) AND ( X <= B )
  END  (* BETWEEN *);

(*****************************************************************)

        (*********************************************)
        (*                                           *)
        (*              C O N S T E X P R            *)
        (*                                           *)
        (*********************************************)

FUNCTION CONSTEXPR (VAR VALUE: INTEGER): BOOLEAN;

(*****************************************************************)
(*                                                               *)
(*  JESLI BIEZACY CIAG ZNAKOW JEST STALA TO CONSTEXPR = TRUE     *)
(*  I ZMIENNA VALUE JEST ROWNA WARTOSCI TEJ STALEJ, A BIEZACY ZNAK *)
(*  ZOSTAJE USTAWIONY NA PIERWSZY ZNAK NASTEPNEJ JEDNOSTKI LEKSYKAL- *)
(*  NEJ.                                                         *)
(*      STALA MOZE ROZPOCZYNAC SIE ZNAKIEM MINUS LUB PLUS.       *)
(*  JESLI BIEZACY CIAG ZNAKOW NIE JEST STALA TO CONSTEXPR = FALSE *)
(*  I POZA TYM WSZYSTKO POZOSTAJE BEZ ZMIAN.                     *)
(*                                                               *)
(*                                                               *)
(*  PROCEDURY GLOBALNE:                                          *)
(*      SAVE                                                     *)
(*      NEXTITEM                                                 *)
(*      BETWEEN                                                  *)
(*      NUMCONST                                                 *)
(*      POS                                                      *)
(*                                                               *)
(*  ZMIENNE GLOBALNE:                                            *)
(*      EOLINE                                                   *)
(*      CURRCHAR                                                 *)
(*                                                               *)
(*****************************************************************)

  VAR
    MINUS, OK: BOOLEAN;
    POSITION: CHARPOS;

        (*********************************************)
        (*                                           *)
        (*              N U M C O N S T              *)
        (*                                           *)
        (*********************************************)

PROCEDURE NUMCONST (VAR VAL: INTEGER);
```

```
(*************************************************************************)
(*                                                                       *)
(*  PROCEDURA OBLICZA  WARTOSC STALEJ ZNAKOWEJ I PODSTAWIA JA 'POD       *)
(*  ZMIENNA VAL. STALA ZNAKOWA ZNAJDUJE SIE W TABLICY LINE,              *)
(*  A PIERWSZY ZNAK STALEJ JEST W LINE[CURRPOS]. PROCEDURA  NIE          *)
(*  ZMIENIA TABLICY LINE. WSKAZNIK PO  WYJSCIU Z PROCEDURY               *)
(*  WSKAZUJE NA NASTEPNY ZNAK PO STALEJ.                                 *)
(*                                                                       *)
(*  ZMIENNE GLOBALNE:                                                    *)
(*                        LINE                                           *)
(*                        CURRPOS                                        *)
(*                                                                       *)
(*  PROCEDURY GLOBALNE:                                                  *)
(*                        ERROR                                          *)
(*                                                                       *)
(*                                                                       *)
(*************************************************************************)

TYPE Y=SET OF CHAR;
VAR
 NUMERIC:SET OF '0'..'9';
 ALFA:SET OF 'A'..'Z';
 QSET,BSET,HSET:Y;
 LP,LK:INTEGER;
 S:CHAR;

(*.....................................................................*)

 FUNCTION HEXBIN (X:CHAR):INTEGER;

(*                                                                       *)
(*  FUNKCJA  OBLICZA  WARTOSC  ZNAKU X                                   *)
(*                                                                       *)
BEGIN
 IF X IN NUMERIC THEN HEXBIN:=ORD(X)-48
                 ELSE HEXBIN:=ORD(X)-55
END (*  HEXBIN *);

(*.....................................................................*)

FUNCTION WYRAZ (X: Y; LKK: INTEGER): BOOLEAN;

(*                                                                       *)
(*  PROCEDURA SPRAWDZA CZY WYSTEPUJACE W STALEJ ZNAKOWEJ ZNAKI SA        *)
(*  DOPUSZCZALNE.JESLI WYKONANIE PROCEDURY ZAKONCZY SIE POWODZENIEM      *)
(*  TO FUNKCJA PRZYJMUJE WARTOSC  TRUE , W PRZECIWNYM RAZI FALSE.        *)
(*                                                                       *)
VAR
 I:INTEGER;
BEGIN
 WYRAZ:=TRUE;
 FOR I:=LP TO LKK DO
   IF NOT (LINE[I] IN X ) THEN WYRAZ:=FALSE
END(* WYRAZ *);

(*.....................................................................*)


FUNCTION POPRAWNE:BOOLEAN;

(*                                                                       *)
(*  FUNKCJA SPRAWDZA POPRAWNOSC STALYCH,ROZPOZNAJE TYP STALEJ            *)
```

```
(*  I WYWOLUE FUNKCJE WYRAZ Z DOPUSZCZALNYM ZBIOREM ZNAKOW KTORE       *)
(*  MOGA WYSTAPIC W STALEJ.                                            *)
(*
VAR
   Z:CHAR;
BEGIN
  Z:=LINE[LK];
  IF Z='B' THEN POPRAWNE:=WYRAZ(BSET,LK-1)
  ELSE
    IF Z='D' THEN POPRAWNE:=WYRAZ(NUMERIC,LK-1)
    ELSE
      IF (Z IN NUMERIC) THEN POPRAWNE:=WYRAZ(NUMERIC,LK)
      ELSE
        IF Z='H' THEN BEGIN
     IF LINE[LP] IN NUMERIC THEN POPRAWNE:=WYRAZ(HSET,LK-1)
                            ELSE POPRAWNE:=FALSE
                                 END
        ELSE
          IF Z='Q' THEN POPRAWNE:=WYRAZ(QSET,LK-1)
          ELSE
            POPRAWNE:=FALSE
END(* POPRAWNE *);

(*.....*)


PROCEDURE KONWERSJA(POTX:INTEGER);

                                                                       *)
(*                                                                     *)
(*  PROCEDURA DOKONUJE KONWERSJI STALEJ, ZAPISANEJ W KODZIE  O PODS-   *)
(*  STAWIE  POTX I PODSTAWIA JEJ WARTOSC POD ZMIENNA VAL.              *)
(*                                                                     *)
(*
VAR
 SUM,POT,PP:INTEGER;
BEGIN
 LK:=LK-1;SUM:=0;VAL:=0;
 REPEAT
SUM:=HEXBIN(LINE[LP]);
IF LP=LK THEN POTX:=1;
VAL:=(VAL+SUM)*POTX;
LP:=LP+1
UNTIL LP>LK
END (* PROC KONWERSJA *);

(*.....*)


BEGIN
  VAL := 0;
 NUMERIC:=['0','1','2','3','4','5','6','7','8','9'];
 ALFA:=['A','B','C','D','E','F','H','O','G','I','J','K',
       'L','M','N','O','P','R','S','T','U','X','V',
       'W','Y','Z'];
 BSET:=['0','1'];
 HSET:=NUMERIC OR['A','B','C','D','E','F'];
 QSET:=['0','1','2','3','4','5','6','7'];
 LP := CURRPOS;
 WHILE  (LINE[CURRPOS] IN ALFA)
        OR (LINE[CURRPOS] IN NUMERIC)
 DO CURRPOS:=CURRPOS+1 ;
 LK:=CURRPOS-1;
 IF POPRAWNE THEN
  BEGIN  (* POPRAWNE *)
```

```
      CASE LINE[LK] OF
      'B'                                :KONWERSJA(2);
      'Q'                                :KONWERSJA(8);
      '1','2','3','4','5','6','7',
      '8','9','0'                        :BEGIN
                                           LK:=LK+1;
                                           KONWERSJA(10)
                                         END;
      'D'                                :KONWERSJA(10);
      'H'                                :KONWERSJA(16)
      END ;
      IF CURRPOS > LINELENGTH THEN
      BEGIN
        CURRCHAR := ' ';
        EOLINE := TRUE
      END
      ELSE
      BEGIN
        CURRCHAR := LINE [CURRPOS];
        NEXTITEM
      END
    END  (* POPRAWNE *)
    ELSE  ERROR(19)
  END(* PROC NUMCONSTANT *);

(*****************************************************************)

  BEGIN  (* CONSTEXPR *)
    SAVE (POSITION);
    OK := FALSE;
    MINUS := FALSE;
    IF CURRCHAR = '-' THEN
    BEGIN
      MINUS := TRUE;
      GETCHAR;
      NEXTITEM
    END
    ELSE IF CURRCHAR = '+' THEN
    BEGIN
      GETCHAR;
      NEXTITEM
    END;
    IF BETWEEN ('0', CURRCHAR, '9') THEN
    BEGIN
      NUMCONST (VALUE); (* !!! SPRAWDZIC JAK SIE WYWOLUJE !!! *)
      OK := TRUE;
      IF MINUS THEN VALUE := - VALUE
    END;
    CONSTEXPR := OK;
    IF NOT OK THEN POS (POSITION)
  END  (* CONSTEXPR *);

(*****************************************************************)

          (*************************************************)
          (*                                               *)
          (*                 I D E N T                     *)
          (*                                               *)
          (*************************************************)

  FUNCTION IDENT (VAR ID: IDENTIFIER): BOOLEAN;

(*****************************************************************)
```

```
(*                                                                    *)
(*  1. JESLI JEDNOSTKA SYNTAKTYCZNA JEST IDENTYFIKATOREM TO:          *)
(*    A) IDENT = TRUE                                                 *)
(*    B) ID = DANY IDENTYFIKATOR                                      *)
(*    C) BIEZACY ZNAK JEST PIERWSZYM ZNAKIEM NASTEPNEJ                *)
(*       JEDNOSTKI SYNTAKTYCZNEJ                                      *)
(*  2. JESLI NIE MA IDENTYFIKATORA TO IDENT = FALSE                   *)
(*     I INNE ZMIENNE POZOSTAJA BEZ ZMIAN                             *)
(*  3. JESLI PRZY WEJSCIU BYLO EOLINE = TRUE TO IDENT = FALSE         *)
(*                                                                    *)
(**********************************************************************)

  VAR
    I, J: INTEGER;

  BEGIN
    IF BETWEEN ('?', CURRCHAR, '@')
       OR BETWEEN ('A', CURRCHAR, 'Z')
       OR (CURRCHAR = ' ') THEN  (* BEGIN OF IDENTIFIER *)
    BEGIN
      I := 0;
      IDENT := TRUE;
      REPEAT
        I := I + 1;
        IF I <= IDLENGTH THEN ID [I] := CURRCHAR;
        GETCHAR
      UNTIL NOT (BETWEEN ('?', CURRCHAR, 'Q')
            OR BETWEEN ('A', CURRCHAR, 'Z')
            OR (CURRCHAR = ' ') OR BETWEEN ('0', CURRCHAR, '9'));
      IF I < IDLENGTH THEN
      FOR J := I+1 TO IDLENGTH DO ID [J] := ' ';
      NEXTITEM
    END
    ELSE IDENT := FALSE
  END  (* IDENT *);

(**********************************************************************)

          (*********************************************)
          (*                                           *)
          (*              D I R E C T I V E            *)
          (*                                           *)
          (*********************************************)

PROCEDURE DIRECTIVE;

(**********************************************************************)
(*                                                                    *)
(*       PROCEDURA PRZETWARZA LINIE PROGRAMU ZRODLOWEGO BEDACA DYREK- *)
(*  TYWA.                                                             *)
(*       PRZY WEJSCIU DO PROCEDURY STAN PRZETWARZANIA LINII JEST      *)
(*  NASTEPUJACY:                                                      *)
(*    1) ZMIENNA LABELFIELD OKRESLA ZAWARTOSC POLA ETYKIETY           *)
(*    2) ZMIENNA ACTMNEM JEST ADRESEM PIERWSZEGO MNEMONIKA ZA POLEM   *)
(*       ETYKIETY I WIADOMO, ZE TEN MNEMONIK JEST MNEMONIKIEM         *)
(*       DYREKTYWY                                                    *)
(*    3) CURRCHAR ZAWIERA PIERWSZY ZNAK NASTEPNEJ JEDNOSTKI SYNTAKTY- *)
(*       CZNEJ (TZN. JEDNOSTKI ZA PIERWSZYM MNEMONIKIEM)              *)
(*                                                                    *)
(*       W SWOIM GLOWNYM CIELE PROCEDURA DOKONUJE JEDYNIE ROZPOZNANIA *)
(*  RODZAJU DYREKTYWY I ODPOWIEDNIO PRZEKAZUJE STEROWANIE DO PROCEDUR *)
(*  OBSLUGUJACYCH POSZCZEGOLNE DYREKTYWY. JESLI DANA DYREKTYWA NIE    *)
(*  JEST ZAIMPLEMENTOWANA (CHOCIAZ JEJ MNEMONIK ZNAJDUJE SIE W SKORO- *)
```

```
(*  WIDZU MNEMONIKOW), TO PROCEDURA PRZEKAZUJE TEN WYNIK W ZMIENNEJ    *)
(*  RES.                                                               *)
(*       ZNACZENIE RES:                                                *)
(*  RES = 0              BEZ BLEDU                                     *)
(*  RES = 1              DYREKTYWA NIE ZAIMPLEMENTOWANA W TEJ WERSJI   *)
(*                       ASEMBLERA                                     *)
(*                                                                     *)
(*       RODZAJE DYREKTYW:                                             *)
(*  ASSUME, DB, DD, DQ, DT, DW, END, ENDP, ENDS, EQU, EVEN,           *)
(*  EXTERN, GROUP, LABEL, NAME, ORG, PARA, PROC, PUBLIC, PURGE,       *)
(*  RECORD, SEGMENT, STRUC,                                           *)
(*                                                                     *)
(*       ZAIMPLEMENTOWANE DYREKTYWY:                                   *)
(*  DB, DW, END, ORG, PROC, ENDP                                      *)
(*                                                                     *)
(*********************************************************************)

VAR
  RES: INTEGER;

        (*****************************************)
        (*                                       *)
        (*              D D B                    *)
        (*                                       *)
        (*****************************************)

PROCEDURE DDB;

VAR
  RESULT, VALUE: INTEGER;
  LOC: IDLOCATION;
  FOUND: BOOLEAN;

BEGIN  (* DDB *)
  IF PASS = 1 THEN
  IF LABELFIELD.NAME THEN
  BEGIN  (* LABELFIELD.NAME *)
    ENTER (LABELFIELD.CONT, IBYTE, FALSE, 0, RESULT);
    IF RESULT = -1 THEN ERROR (8); (* POWTORNA DEFINICJA
                                      IDENTYFIKATORA *)
    IF RESULT = -2 THEN ERROR (9)    (* PRZEPELNIENIE SKOROWIDZA
                                        UZYTKOWNIKA *)
  END;  (* LABELFIELD.NAME *)
  IF PASS = 2 THEN
  BEGIN  (* PASS = 2 *)
    IF LABELFIELD.NAME THEN
    BEGIN  (* LABELFIELD.NAME *)
      FIND (LABELFIELD.CONT, LOC, FOUND);
      USERIDENT [LOC].OFFSET := LC;
      USERIDENT [LOC].DETOFFSET := TRUE
    END;  (* LABELFIELD.NAME *)
    LC := LC + 1
  END  (* PASS = 2 *);
  IF PASS = 3 THEN
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
    IF NOT CONSTEXPR (VALUE) THEN VALUE := 0;
    CONTENTS [NOBYTES + 1] := VALUE MOD 256;
    NOBYTES := NOBYTES + 1;
    LC := LC + 1
  END  (* WITH OBJ *)
END  (* DDB *);
```

```
(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*                  D D W                        *)
        (*                                               *)
        (*************************************************)

PROCEDURE DDW;

VAR
  RESULT, VALUE: INTEGER;
  LOC: IDLOCATION;
  FOUND: BOOLEAN;

BEGIN  (* DDW *)
  IF PASS = 1 THEN
  IF LABELFIELD.NAME THEN
  BEGIN  (* LABELFIELD.NAME *)
    ENTER (LABELFIELD.CONT, WORD, FALSE, 0, RESULT);
    IF RESULT = -1 THEN ERROR (8); (* POWTORNA DEFINICJA
                                      IDENTYFIKATORA *)
    IF RESULT = -2 THEN ERROR (9)    (* PRZEPELNIENIE SKOROWIDZA
                                        UZYTKOWNIKA *)

  END;  (* LABELFIELD.NAME *)
  IF PASS = 2 THEN
  BEGIN  (* PASS = 2 *)
    IF LABELFIELD.NAME THEN
    BEGIN  (* LABELFIELD.NAME *)
      FIND (LABELFIELD.CONT, LOC, FOUND);
      USERIDENT [LOC].OFFSET := LC;
      USERIDENT [LOC].DETOFFSET := TRUE
    END;  (* LABELFIELD.NAME *)
    LC := LC + 2
  END  (* PASS = 2 *);
  IF PASS = 3 THEN
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
    IF NOT CONSTEXPR (VALUE) THEN VALUE := 0;
    CONTENTS [NOBYTES + 1] := VALUE ~~DIV~~ 256;
    CONTENTS [NOBYTES + 2] := VALUE ~~MOD~~ 256;
                                    div
    NOBYTES := NOBYTES + 2;
    LC := LC + 2
  END  (* WITH OBJ *)
END  (* DDW *);

(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*                  D E N D                      *)
        (*                                               *)
        (*************************************************)

PROCEDURE DEND;

BEGIN  (* DEND *)
  EOPROGRAM := TRUE
END  (* DEND *);

(*****************************************************************)

        (*************************************************)
```

```
        (*                                                    *)
        (*                  D E N D P                         *)
        (*                                                    *)
        (******************************************************)

PROCEDURE DENDP;

(*-------- ---------- ---------- ---------- ---------- ---------- ----------
(       W TEJ WERSJI PROGRAMU NIC NIE ROBI. W NASTEPNYCH WERSJACH
(  BEDZIE ZMIENIAC PROCEDURE, W KTOREJ ZAKRESIE SIE ZNAJDUJEMY, A PRZEZ
(  TO MOZE SIE ROWNIEZ ZMIENIC PROCLEN (W ZALEZNOSCI OD TEGO CZY PRO-
(  CEDURA JEST TYPU NEAR LUB FAR).
(---------- ---------- ---------- ---------- ---------- ---------- ----------*)

BEGIN  (* DENDP *)
END  (* DENDP *);

(*****************************************************************************)

        (******************************************)
        (*                                        *)
        (*                D O R G                 *)
        (*                                        *)
        (******************************************)

PROCEDURE DORG;

VAR
  VALUE: INTEGER;

BEGIN  (* DORG *)
  IF CONSTEXPR (VALUE) THEN LC := VALUE
  ELSE ERROR (10)   (* BRAK WYRAZENIA W POLU ARGUMENTOW DYREKTYWY ORG *)
END  (* DORG *);

(*****************************************************************************)

        (******************************************)
        (*                                        *)
        (*                D P R O C               *)
        (*                                        *)
        (******************************************)

PROCEDURE DPROC;

VAR
  RESULT: INTEGER;
  LOC: IDLOCATION;
  FOUND: BOOLEAN;

BEGIN  (* DPROC *)
  IF PASS = 1 THEN
  IF LABELFIELD.NAME THEN
  BEGIN  (* LABELFIELD.NAME *)
    ENTER (LABELFIELD.CONT, NEAR, FALSE, 0, RESULT);
    IF RESULT = -1 THEN ERROR (1); (* POWTORNA DEFINICJA
                                      IDENTYFIKATORA *)
    IF RESULT = -2 THEN ERROR (2)   (* PRZEPELNIENIE SKOROWIDZA
                                       UZYTKOWNIKA *)
  END  (* LABELFIELD.NAME *)
  ELSE ERROR (3);  (* BRAK NAZWY W POLU ETYKIETY *)
  IF PASS = 2 THEN
  BEGIN  (* PASS = 2 *)
```

```
    FIND (LABELFIELD.CONT, LOC, FOUND);
    USERIDENT [LOC].OFFSET := LC;
    USERIDENT [LOC].DETOFFSET := TRUE
  END  (* PASS = 2 *);
  IF PASS = 3 THEN OBJNO := TRUE
END  (* DPROC *);

(*****************************************************************)

BEGIN  (* DIRECTIVE *)
  RES := 0;
  CASE ENTRY [ACTMNEM] OF
    1: DDB;
    2: DDW;
    3: DEND;
    4: DORG;
    5: DPROC;
    6: DENDP;
    OTHERS: RES := 1
  END  (* CASE *);
  IF RES = 1 THEN ERROR (7)
END  (* DIRECTIVE *);

(*****************************************************************)

        (*********************************************)
        (*                                           *)
        (*              M N E M O N I C              *)
        (*                                           *)
        (*********************************************)

FUNCTION MNEMONIC (VAR LOC: INTEGER): BOOLEAN;

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA BADA CZY JEDNOSTKA SYNTAKTYCZNA JEST MNEMONIKIEM DYREK- *)
(*  TYWY LUB INSTRUKCJI. JESLI TAK, TO MNEMONIC = TRUE I ZMIENNA LOC  *)
(*  JEST ADRESEM MNEMONIKA W TABLICY MN, A BIEZACY ZNAK (CURRCHAR)    *)
(*  JEST PIERWSZYM ZNAKIEM NASTEPNEJ JEDNOSTKI SYNTAKTYCZNEJ.         *)
(*  JESLI JEDNOSTKA SYNTAKTYCZNA NIE JEST MNEMONIKIEM TO MNEMONIC =   *)
(*  FALSE I CURRCHAR NIE ZMIENIA SIE, JESLI MNEMONIC = FALSE TO LOC   *)
(*  WSKAZUJE NA RODZAJ BLEDU:                                         *)
(*       LOC = 1       IDENTYFIKATOR, ALE NIE BEDACY MNEMONIKIEM     *)
(*       LOC = 2       NIE IDENTYFIKATOR                             *)
(*                                                                   *)
(*****************************************************************)

LABEL
  1;

CONST
  LESS = -1; EQUAL = 0; GREATER = 1;

VAR
  POSITION: CHARPOS;
  L, I, J, RELATION: INTEGER;
  OK: BOOLEAN;
  Y: ARRAY [1..MNLEN] OF CHAR;

BEGIN  (* MNEMONIC *)  SAVE (POSITION);
  IF IDENT (ID) THEN
  BEGIN (* PRZESZUKAJ TABLICE MNEMONIKOW *)
    I := 1; J := NOMNEM;
```

```
    OK := FALSE;
    REPEAT
      LOC := (I + J) DIV 2;
      FOR L := 1 TO MLENPACK DO DECAN3 (MN [LOC] [L], Y, 3 * (L - 1)+1);
      (*  PORÓWNANIE IDENTYFIKATORA Z MNEMONIKIEM Z TABLICY MNEMONIKÓW*)
      L := 1;
1:    IF ID [L] = Y [L] THEN
        IF (L = MLEN) THEN RELATION := EQUAL
        ELSE IF (ID [L+1] = ' ') AND ( Y [L+1] = ' ') THEN
                                              RELATION := EQUAL
        ELSE IF ID [L+1] = ' ' THEN RELATION := LESS
        ELSE IF Y [L+1] = ' ' THEN RELATION := GREATER
        ELSE
        BEGIN
          L := L + 1;
          GOTO 1
        END
      ELSE IF ID [L] < Y [L] THEN RELATION := LESS
      ELSE RELATION := GREATER;
      CASE RELATION OF
        LESS: J := LOC - 1;
        GREATER: I := LOC + 1;
        EQUAL: OK := TRUE
      END
    UNTIL OK OR (J < I);
    IF OK THEN MNEMONIC := TRUE
    ELSE
    BEGIN
      MNEMONIC := FALSE;
      ~~LOC := 1~~ POS (POSITION)
    END
  END  (* PRZESZUKAJ TABLICE MNEMONIKÓW *)
  ELSE
  BEGIN
    MNEMONIC := FALSE;
    ~~LOC := 2~~
  END
END  (* MNEMONIC *);

(*****************************************************************)

        (*****************************************)
        (*                                       *)
        (*            I N S T R U C T N          *)
        (*                                       *)
        (*****************************************)

PROCEDURE INSTRUCTN;

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA PRZETWARZA LINIE, KTÓRA JEST INSTRUKCJA.           *)
(*  STAN WEJŚCIOWY JEST NASTĘPUJĄCY:                             *)
(*   1) ZMIENNA LABELFIELD OKREŚLA ZAWARTOŚĆ POLA ETYKIETY       *)
(*   2) ZMIENNA ACTMNEM JEST ADRESEM PIERWSZEGO MNEMONIKA ZA POLEM *)
(*      ETYKIETY I WIADOMO, ŻE TEN MNEMONIK JEST MNEMONIKIEM     *)
(*      INSTRUKCJI                                               *)
(*   3) CURRCHAR ZAWIERA PIERWSZY ZNAK NASTĘPNEJ JEDNOSTKI SYNTAKTY- *)
(*      CZNEJ (TZN. JEDNOSTKI ZA PIERWSZYM MNEMONIKIEM)          *)
(*                                                               *)
(*  ZMIENNE GLOBALNE:                                            *)
(*      LABELFIELD: RECORD                                       *)
(*                    LAB, NAME: BOOLEAN;                        *)
```

```
(*                    CONT: IDENTIFIER                                  *)
(*                  END                                                 *)
(*       ACTMNEM: INTEGER                                               *)
(*       MNEMADDR: ARRAY [1..2] OF INTEGER                              *)
(*                    - ADRESY KOLEJNYCH MNEMONIKOW, KTORE WYSTAP       *)
(*                      PILY W LINII                                    *)
(*                                                                      *)
(*  PROCEDURY GLOBALNE:                                                 *)
(*       PREFX                                                          *)
(*       MNEMONIC                                                       *)
(*       ARGREAD                                                        *)
(*       CMMATCH                                                        *)
(*       CODE                                                           *)
(*                                                                      *)
(************************************************************************)

LABEL
  1;

VAR
  E, RES, ER: INTEGER;
  LOC: IDLOCATION;
  FOUND: BOOLEAN;
  PRE : boolean;
        (*********************************************)
        (*                                           *)
        (*              P O T E G A                  *)
        (*                                           *)
        (*********************************************)

FUNCTION POTEGA(K:INTEGER):INTEGER;

(*************************************************************************)
(*                                                                       *)
(*  PROCEDURA PODAJE WARTOSC K-TEJ POTEGI  2.                            *)
(*                                                                       *)
(*************************************************************************)
VAR
  WYNIK,I:INTEGER;
BEGIN
  WYNIK:=1;
  FOR I:=1 TO K DO WYNIK:=WYNIK+WYNIK;
  POTEGA:=WYNIK
END (* POTEGA *);

(************************************************************************)

        (*********************************************)
        (*                                           *)
        (*              F I E L D                    *)
        (*                                           *)
        (*********************************************)

FUNCTION FIELD (X, K, L: INTEGER): INTEGER;

(*************************************************************************)
(*                                                                       *)
(*  FUNKCJA FIELD WYDOBYWA LICZBE Z BITOW K, K-1, ..., L  LICZBY X       *)
(*  (16-BITOWEJ)                                                         *)
(*                                                                       *)
(*************************************************************************)

  VAR
```

```
    I: INTEGER;

        (*****************************************)
        (*                                       *)
        (*            D E C O D E                *)
        (*                                       *)
        (*****************************************)

PROCEDURE DECODE(VAR W,P:INTEGER;K,L:INTEGER);

(*************************************************************************)
(*                                                                       *)
(*  PROCEDURA WYCINA BITY K..L ( WLACZNIE ) ZE ZMIENNEJ W TYPU           *)
(*  INTEGER.ZMIENNE K I L SA TYPU INTEGER I NALEZA DO PRZEDZIALU         *)
(*  <0,15>, A PONADTO K>=L . WARTOSC WYCIETYCH BITOW PODSTAWIA           *)
(*  POD ZMIENNA P.                                                       *)
(*                                                                       *)
(*  PROCEDURY GLOBALNE                                                   *)
(*      POTEGA                                                           *)
(*                                                                       *)
(*************************************************************************)
VAR
   Q:INTEGER;
BEGIN
  P:=W;
  Q:=POTEGA(K+1);          if l=15 then
  P:=P MOD Q;               if w<0 then p:=1 else p:=0
  Q:=POTEGA(L);            else p:= (w div potega (l)) mod potega (k-l+1
  P:=P DIV Q
END(* DECODE *);

  BEGIN  (* FIELD *)
    DECODE (X,I,K,L);
    FIELD := I
  END  (* FIELD *);

(*************************************************************************)

        (*****************************************)
        (*                                       *)
        (*            P R E F X                  *)
        (*                                       *)
        (*****************************************)

FUNCTION PREFX (LOC: INTEGER): BOOLEAN;

(*************************************************************************)
(*                                                                       *)
(*  PROCEDURA SPRAWDZA CZY MNEMONIK Z POZYCJI LOC JEST PREFIXEM          *)
(*                                                                       *)
(*  ZMIENNE GLOBALNE:                                                    *)
(*      ARGS                                                             *)
(*      ENTRY                                                            *)
(*                                                                       *)
(*  PROCEDURY GLOBALNE:                                                  *)
(*      FIELD                                                            *)
(*                                                                       *)
(*************************************************************************)

BEGIN  (* PREFX *)
  PREFX := (FIELD (ARGS [ENTRY [LOC]], 15, 15) = 1)
           AND (FIELD (ARGS [ENTRY [LOC]], 1, 0) = 0 )
END  (* PREFX *);
```

```
(*****************************************************************)

        (*************************************)
        (*                                   *)
        (*          A R G R E A D            *)
        (*                                   *)
        (*************************************)

PROCEDURE ARGREAD (VAR RESULT: ARGNTS;
                   VAR ERRORV: INTEGER);

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA CZYTA ARGUMENTY INSTRUKCJI ASM86                   *)
(*                                                               *)
(*  PARAMETRY:                                                   *)
(*   RESULT     -  INFORMACJA O PRZECZYTANYCH PARAMETRACH        *)
(*   ERRORV     -  =0 GDY BEZ BLEDU                              *)
(*                                                               *)
(*****************************************************************)

VAR
  ARGCOUNTER: INTEGER;
  OK: BOOLEAN;

        (*************************************)
        (*                                   *)
        (*          A R G U M E N T          *)
        (*                                   *)
        (*************************************)

FUNCTION ARGUMENT (VAR RESULT: ARGKIND; VAR VERROR: INTEGER): BOOLEAN;

(*****************************************************************)
(*                                                               *)
(*  FUNKCJA ROZPOZNAJE CZY BIEZACA JEDNOSTKA LEKSYKALNA JEST     *)
(*  ARGUMENTEM INSTRUKCJI. JESLI TAK TO ARGUMENT = TRUE I W ZMIENNEJ *)
(*  RESULT PRZEKAZANY JEST WYNIK ROZPOZNANIA,                    *)
(*  W ZMIENNEJ VERROR PRZEKAZANY JEST NUMER EWENTUALNEGO BLEDU (JESLI *)
(*  VERROR = 0 TO NIE WYKRYTO BLEDU)                             *)
(*                                                               *)
(*                                                               *)
(*  STALE GLOBALNE:                                              *)
(*                                                               *)
(*  ZMIENNE GLOBALNE:                                            *)
(*                                                               *)
(*  PROCEDURY GLOBALNE:                                          *)
(*                                                               *)
(*****************************************************************)

  CONST
    A = 0;
    C = 1;
    D = 2;
    E = 3;
    M = 4;
    R = 5;
    S = 6;
    X = 7;
    B = 0;
    W = 1;
    R128127 = 0;
```

```
    R063 = 1;
    R3 = 2;
    R1 = 3;
    RCL = 4;
    RDX = 5;
    RES = 6;
    RSSDS = 7;

  VAR
    POSITION, POS1: CHARPOS;
    I, PTROVR: INTEGER;
    T: TYPE86;
        (*--- ---------- ---------- ---------- ---------- ---------- ---*)
        (*  PTROVR OKRESLA CZY WYSTAPIL OPERATOR PTR.                  *)
        (*  JESLI PTROVR = 0 TO OPERATOR PTR NIE WYSTAPIL, W PRZECIW-  *)
        (*  NYM RAZIE PTROVR WSKAZUJE JAKI TYP NARZUCIL OPERATOR PTR.  *)
        (*---- ---------- ---------- ---------- ---------- ---------- ---*)
    SHORTOPR, FOUND, ARGEND, OK: BOOLEAN;
        (*---------- ---------- ---------- ---------- ---------- ---------- ---*)
        (*  SHORTOPR OKRESLA CZY WYSTAPIL OPERATOR SHORT.              *)
        (*---------- ---------- ---------- ---------- ---------- ---------- ---*)
        (*  OK = TRUE JESLI ZOSTAL ROZPOZNANY ARGUMENT                 *)
        (*---------- ---------- ---------- ---------- ---------- ---------- ---*)
    V: PACKED ARRAY [0..7] OF BOOLEAN;
    LOC: IDLOCATION;
    EFFECT: REXVAL;

        (*********************************************)
        (*                                           *)
        (*              E O A R G                    *)
        (*                                           *)
        (*********************************************)

FUNCTION EOARG: BOOLEAN;

(*****************************************************************************)
(*                                                                           *)
(*  FUNKCJA EOARG SPRAWDZA CZY BIEZACA JEDNOSTKA SYNTAKTYCZNA JEST           *)
(*  KONCEM ARGUMENTU (KONCEM ARGUMENTU JEST PRZECINEK, SREDNIK, LUB          *)
(*  KONIEC LINII). JESLI TAK TO BIEZACY ZNAK (CURRCHAR) ZOSTANIE             *)
(*  USTAWIONY NA POCZATEK NASTEPNEJ JEDNOSTKI SYNTAKTYCZNEJ                  *)
(*  I EOARG PRZYBIERZE WARTOSC TRUE, JESLI NIE TO EOARG PRZYBIERA            *)
(*  WARTOSC FALSE I NIC SIE NIE ZMIENIA.                                     *)
(*                                                                           *)
(*                                                                           *)
(*  PROCEDURY GLOBALNE:                                                      *)
(*      NEXTITEM                                                             *)
(*                                                                           *)
(*  ZMIENNE GLOBALNE:                                                        *)
(*      CURRCHAR                                                             *)
(*                                                                           *)
(*****************************************************************************)

  BEGIN  (* EOARG *)
    IF EOLINE OR (CURRCHAR = ',') OR (CURRCHAR = ';') THEN
    BEGIN
      EOARG := TRUE;
      IF CURRCHAR = ';' THEN REPEAT GETCHAR UNTIL EOLINE
      ELSE
      BEGIN
        GETCHAR;
        NEXTITEM
      END
```

```
    END
    ELSE EOARG := FALSE
  END  (* EOARG *);

(*****************************************************************)

        (*********************************************)
        (*                                           *)
        (*              R E G E X P R                *)
        (*                                           *)
        (*********************************************)

FUNCTION REGEXPR (VAR RESULT: REXVAL): BOOLEAN;

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA BADA CZY JEDNOSTKA LEKSYKALNA JEST WYRAZENIEM REJESTRO- *)
(*  WYM.                                                         *)
(*  JESLI TAK TO REGEXPR = TRUE I BIEZACY ZNAK JEST PIERWSZYM ZNA-   *)
(*  KIEM NASTEPNEJ JEDNOSTKI LEKSYKANEJ.                         *)
(*  JESLI REGEXPR = TRUE TO PROCEDURA PODAJE CZY WYSTAPIL        *)
(*  BASE REGISTER, INDEX REGISTER, CONSTEXPR I PODAJE WARTOSC    *)
(*  CONSTEXPR.                                                   *)
(*  (TYPE REXVAL = RECORD                                        *)
(*                   BX, BP, SI, DI, CONS: BOOLEAN;              *)
(*                   VALUE: INTEGER                              *)
(*                 END                          )                *)
(*                                                               *)
(*  ZMIENNE GLOBALNE:                                            *)
(*      ERR             - ZMIENNA LOGICZNA USTAWIANA PRZEZ PROCEDURE  *)
(*                        ERROR                                  *)
(*                                                               *)
(*  PROCEDURY GLOBALNE:                                          *)
(*      SAVE                                                     *)
(*      POS                                                      *)
(*      CONSTEXPR                                                *)
(*      EQUAL                                                    *)
(*      NEXTITEM                                                 *)
(*                                                               *)
(*****************************************************************)

VAR
  POSITION, POS1: CHARPOS;
  OK: BOOLEAN;

BEGIN  (* REGEXPR *)
  ERR := FALSE;
  WITH RESULT DO
  BEGIN  (* WITH *)
    SAVE (POSITION);
    OK := FALSE;
    BX := FALSE;
    BP := FALSE;
    SI := FALSE;
    DI := FALSE;
    CONS := FALSE;
    IF CURRCHAR = '[' THEN
    BEGIN  (* [ *)
      GETCHAR;
      NEXTITEM;
      IF CONSTEXPR (VALUE) THEN CONS := TRUE
      ELSE IF IDENT (ID) THEN
      BEGIN
```

```
            BX := EQUAL (ID, 'BX');
            BP := EQUAL (ID, 'BP');
            SI := EQUAL (ID, 'SI');
            DI := EQUAL (ID, 'DI');
            IF BX OR BP THEN
            BEGIN  (* REJESTRY BAZOWE *)
              IF CONSTEXPR (VALUE) THEN CONS := TRUE
              ELSE IF CURRCHAR = '+' THEN
              BEGIN
                GETCHAR;
                NEXTITEM;
                IF IDENT (ID) THEN
                BEGIN
                  SI := EQUAL (ID, 'SI');
                  DI := EQUAL (ID, 'DI');
                  IF SI OR DI THEN
                  BEGIN
                    IF CONSTEXPR (VALUE) THEN CONS := TRUE
                  END
                  ELSE ERROR (20)
                END
                ELSE ERROR (20)
              END
            END  (* REJESTRY BAZOWE *)
            ELSE IF SI OR DI THEN
              IF CONSTEXPR (VALUE) THEN CONS := TRUE
          END
          ELSE ERROR (21);
          IF NOT ERR THEN
          BEGIN
            IF CURRCHAR = ']' THEN
            BEGIN
              OK := TRUE;
              GETCHAR;
              NEXTITEM;
            END
            ELSE ERROR (23)
          END;
          REGEXPR := OK;
          IF NOT OK THEN POS (POSITION)
        END  (* '[' *)
      END  (* WITH *)
    END  (* REGEXPR *);

(*****************************************************************)

          (***********************************************)
          (*                                             *)
          (*                 P T R                       *)
          (*                                             *)
          (***********************************************)

    PROCEDURE PTR (VAR K: INTEGER);

    (*****************************************************************)
    (*                                                               *)
    (*  PROCEDURA PTR BADA CZY WYSTAPIL OPERATOR PTR (TYPE OVERRIDING *)
    (*  OPERATOR). JESLI PTR WYSTAPIL TO CURRCHAR JEST PIERWSZYM ZNAKIEM *)
    (*  DALSZEJ CZESCI ARGUMENTU, A JESLI NIE WYSTAPIL TO CURRCHAR SIE *)
    (*  NIE ZMIENIA.                                                 *)
    (*                                                               *)
    (*  ZNACZENIE PARAMETROW:                                        *)
    (*  K = 1       -  PTR WYSTAPIL Z TYPEM BYTE                     *)
```

```
  (*   K = 2        -  PTR WYSTAPIL Z TYPEM WORD                          *)
  (*   K = 3        -  PTR WYSTAPIL Z TYPEM DWORD                         *)
  (*   K = 4        -  PTR WYSTAPIL Z TYPEM QWORD                         *)
  (*   K = 5        -  PTR WYSTAPIL Z TYPEM TBYTE                         *)
  (*   K = 6        -  PTR WYSTAPIL Z TYPEM NEAR                          *)
  (*   K = 7        -  PTR WYSTAPIL Z TYPEM FAR                           *)
  (*   K = 0        -  OPERATOR PTR NIE WYSTAPIL                          *)
  (*                                                                      *)
  (************************************************************************)

  VAR
    POSITION: CHARPOS;
    ID1, ID2: IDENTIFIER;

  (************************************************************************)
  PROCEDURE K0;

  BEGIN  (* K0 *)
    K := 0;
    POS (POSITION)
  END  (* K0*);
  (************************************************************************)

  BEGIN  (* PTR *)
    SAVE (POSITION);
    IF IDENT (ID1) THEN
    BEGIN
      IF IDENT (ID2) THEN
      BEGIN
        IF EQUAL (ID2, 'PTR') THEN
        BEGIN
          IF EQUAL (ID1, 'BYTE') THEN K := 1
          ELSE IF EQUAL (ID1, 'WORD') THEN K := 2
          ELSE IF EQUAL (ID1, 'DWORD') THEN K := 3
          ELSE IF EQUAL (ID1, 'QWORD') THEN K := 4
          ELSE IF EQUAL (ID1, 'TBYTE') THEN K := 5
          ELSE IF EQUAL (ID1, 'NEAR') THEN K := 6
          ELSE IF EQUAL (ID1, 'FAR') THEN K := 7
          ELSE K0
        END
        ELSE K0
      END
      ELSE K0
    END
    ELSE K0
  END  (* PTR *);

  (************************************************************************)

          (************************************************)
          (*                                              *)
          (*                S H O R T                     *)
          (*                                              *)
          (************************************************)

  FUNCTION SHORT: BOOLEAN;

  (************************************************************************)
  (*                                                                      *)
  (*  FUNKCJA LOGICZNA OKRESLAJACA CZY ETYKIECIE DAC ATRYBUT D LUB W      *)
  (*     MIANOWICIE:                                                      *)
  (*                                                                      *)
  (*                SHORT = TRUE    JESLI WYSTAPIL OPERATOR SHORT LUB     *)
```

```
(*                          ((OFFSET ETYKIETY) - (LOCATION COUNTER))      *)
(*                          NALEZY DO PRZEDZIALU [-126,129]               *)
(*                                                                        *)
(*       SHORT = FALSE    JESLI NIESPELNIONE WARUNKI DLA "TRUE"           *)
(*                                                                        *)
(**************************************************************************)

  BEGIN  (* SHORT *)
    IF SHORTOPR THEN SHORT := TRUE
    ELSE IF NOT USERIDENT [LOC].DETOFFSET THEN SHORT := FALSE
    ELSE IF ((USERIDENT [LOC].OFFSET - LC) >= -126)
            AND ((USERIDENT [LOC].OFFSET - LC) <= 129) THEN SHORT :=TRUE
    ELSE SHORT := FALSE
  END  (* SHORT *);

(**************************************************************************)

  BEGIN  (* ARGUMENT *)
    WITH RESULT DO
    BEGIN  (* WITH RESULT *)
      SAVE (POSITION);
      OK := FALSE;
      VERROR := 0;
      SHORTOPR := FALSE;
      ANONYMOUS := FALSE;
      SEGFIX.YES := FALSE;
      FOR I := 0 TO 7 DO
      BEGIN
        SPECIFIER [I] := FALSE;
        RANGE [I] := FALSE
      END;
      FOR I := 0 TO 3 DO MODIFIER [I] := FALSE;
      BX := FALSE;
      BP := FALSE;
      SI := FALSE;
      DI := FALSE;
      CS := FALSE;
      SS := FALSE;
      DS := FALSE;
      ES := FALSE;
      OFFSET := 0;
      PTR (PTROVR);
      SAVE (POS1);
      IF IDENT (ID) THEN
        IF EQUAL (ID, 'SHORT') THEN SHORTOPR := TRUE
        ELSE POS (POS1);
      IF IDENT (ID) THEN
      BEGIN  (* IDENTYFIER *)
        ARGEND := EOARG;
        IF EQUAL (ID, 'AL') AND ARGEND THEN
        BEGIN
          OK := TRUE;
          SPECIFIER [A] := TRUE;
          SPECIFIER [R] := TRUE;
          SPECIFIER [L] := TRUE;
          MODIFIER [B] := TRUE;
          REG := 0
        END
        ELSE IF EQUAL (ID, 'AX') AND ARGEND THEN
        BEGIN
          OK := TRUE;
          SPECIFIER [A] := TRUE;
          SPECIFIER [R] := TRUE;
```

```
    SPECIFIER [E] := TRUE;
    MODIFIER [W] := TRUE;
    REG := 0
  END
  ELSE
  BEGIN  (* REJESTR OGOLNY 16-BITOWY *)
    V [1] := EQUAL (ID, 'CX');
    V [2] := EQUAL (ID, 'DX');
    V [3] := EQUAL (ID, 'BX');
    V [4] := EQUAL (ID, 'SP');
    V [5] := EQUAL (ID, 'BP');
    V [6] := EQUAL (ID, 'SI');
    V [7] := EQUAL (ID, 'DI');
    IF (V [1] OR V [2] OR V [3] OR V [4] OR V [5]
        OR V [6] OR V [7]) AND ARGEND THEN
    BEGIN
      OK := TRUE;
      SPECIFIER [R] := TRUE;
      SPECIFIER [E] := TRUE;
      MODIFIER [W] := TRUE;
      RANGE [RDX] := V [2];
      FOR I := 1 TO 7 DO IF V [I] THEN REG := I;
    END
  END  (* REJESTR OGOLNY 16-BITOWY *);
  IF NOT OK THEN
  BEGIN  (* REJESTR OGOLNY 8-BITOWY *)
    V [1] := EQUAL (ID, 'CL');
    V [2] := EQUAL (ID, 'DL');
    V [3] := EQUAL (ID, 'BL');
    V [4] := EQUAL (ID, 'AH');
    V [5] := EQUAL (ID, 'CH');
    V [6] := EQUAL (ID, 'DH');
    V [7] := EQUAL (ID, 'BH');
    IF (V [1] OR V [2] OR V [3] OR V [4] OR V [5]
        OR V [6] OR V [7]) AND ARGEND THEN
    BEGIN
      OK := TRUE;
      SPECIFIER [R] := TRUE;
      SPECIFIER [E] := TRUE;
      MODIFIER [B] := TRUE;
      FOR I := 1 TO 7 DO IF V [I] THEN REG := I;
      RANGE [RCL] := V [1]
    END
  END  (* REJESTR OGOLNY 8-BITOWY *);
  IF NOT OK THEN
  (*  NIE AKUMULATOR I NIE REJESTR OGOLNY, A WIEC MOZE ODWOLANIE
  DO PAMIECI LUB ETYKIETA *)

  BEGIN  (* NIE A,R,S *)
  (*  NAJPIERW SPRAWDZIMY JAKIEGO TYPU JEST IDENTYFIKATOR.
  ODWOLANIA DO PAMIECI MOGA BYC TYPU B, W, D,  A DO KODU
  (ETYKIETY) MOGA BYC TYPU NEAR I FAR *)

    FIND (ID, LOC, FOUND)
    IF FOUND THEN
    BEGIN  (* IDENTYFIKATOR UZYTKOWNIKA *)
      T := USERIDENT [LOC].ITYPE;
          (*  POD T PODSTAWIANY TYP IDENTYFIKATORA UZYTKOWNIKA *)
      IF (T = IBYTE) OR (T = WORD) OR (T = DWORD) THEN
      BEGIN  (* MEMORY *)
        IF ARGEND THEN  (* DIRECT MEMORY REFERENCE *)
        BEGIN  (* DIRECT *)
          OK := TRUE;
```

(wspomniane rejestry segmentowe)

(patrz poprzednie listing)

```
            SPECIFIER [X] := TRUE;
            SPECIFIER [B] := TRUE;
            SPECIFIER [M] := TRUE;
            CASE T OF
              IBYTE: MODIFIER [0] := TRUE;
              WORD: MODIFIER [1] := TRUE;
              DWORD: MODIFIER [2] := TRUE
            END;
            OFFSET := USERIDENT [LOC].OFFSET
          END  (* DIRECT *)
        ELSE IF REGEXPR (EFFECT) THEN
        BEGIN  (* MEMORY REFERENCE *)
          IF EOARG THEN
          BEGIN
            OK := TRUE;
            SPECIFIER [B] := TRUE;
            SPECIFIER [M] := TRUE;
            CASE T OF
              IBYTE: MODIFIER [0] := TRUE;
              WORD: MODIFIER [1] := TRUE;
              DWORD: MODIFIER [2] := TRUE
            END;
            OFFSET := USERIDENT [LOC].OFFSET + EFFECT.VALUE;
            IF NOT (EFFECT.BX OR
                    EFFECT.BP OR
                    EFFECT.SI OR
                    EFFECT.DI) THEN
            (* DIRECT MEMORY REFERENCE *)
            SPECIFIER [X] := TRUE
            ELSE
            BEGIN
              BX := EFFECT.BX;
              BP := EFFECT.BP;
              SI := EFFECT.SI;
              DI := EFFECT.DI
            END
          END
          ELSE ERROR (25)
        END  (* MEMORY REFERENCE *)
      END  (* MEMORY *)
      ELSE IF ((T = NEAR) OR (T = FAR)) AND ARGEND THEN  (* CODE *)
      BEGIN  (* LABEL *)
        (* W TEJ WERSJI DOPUSZCZAMY JAKO LABEL-EXPRESSION
        JEDYNIE ETYKIETE *)

        OK := TRUE;
        SPECIFIER [C] := TRUE;
        OFFSET := USERIDENT [LOC].OFFSET;
        IF T = NEAR THEN
          IF SHORT THEN MODIFIER  [B] := TRUE
          ELSE MODIFIER [W] := TRUE
        ELSE MODIFIER [D] := TRUE
      END  (* LABEL *)
    END  (* IDENTYFIKATOR UZYTKOWNIKA *)
  END  (* NIE A,B,C *);
  IF NOT OK THEN
  (* IDENTYFIKATOR NIE JEST AKUMULATOREM, REJESTREM, ODWOLANIEM
  DO PAMIECI, ETYKIETA, MOZE WIEC JEST REJESTREM SEGMENTOWYM *)

  BEGIN  (* SEGMENT REGISTER *)
    CS := EQUAL (ID ,'CS');
    SS := EQUAL (ID ,'SS');
    DS := EQUAL (ID ,'DS');
```

```
    ES := EQUAL (ID ,'ES');
    IF (CS OR SS OR DS OR ES) AND ARGEND THEN
    BEGIN
      OK := TRUE;
      SPECIFIER [S] := TRUE;
      IF ES THEN RANGE [RES] := TRUE;
      IF SS OR DS THEN RANGE [RSSDS] := TRUE
    END
  END  (* SEGMENT REGISTER *);
  IF NOT OK THEN ERROR (17)  (* NIEROZPOZNANY IDENTYFIKATOR *)
END  (* IDENTYFIKATOR *)

ELSE IF CONSTEXPR (OFFSET) THEN
BEGIN  (* DATA *)
  IF EOARG THEN
  BEGIN
    OK := TRUE;
    SPECIFIER [D] := TRUE;
    IF (-256 <= OFFSET) AND (OFFSET <= 255)
      THEN MODIFIER [B] := TRUE
    ELSE MODIFIER [W] := TRUE;

    (* TERAZ ZBADAJ RANGE *)

    IF (-128 <= OFFSET) AND (OFFSET <= 127) THEN
    RANGE [R128127] := TRUE;
    IF (OFFSET >= 0) AND (OFFSET <= 63) THEN
    RANGE [R063] := TRUE;
    IF OFFSET = 3 THEN RANGE [R3] := TRUE;
    IF OFFSET = 1 THEN RANGE [R1] := TRUE
  END
  ELSE ERROR (24)
END  (* DATA *)
ELSE IF CURRCHAR = '[' THEN
BEGIN  (* REGISTER EXPRESION *)
  IF REGEXPR (EFFECT) THEN
  IF EOARG THEN
  BEGIN  (* EOARG *)
    OK := TRUE;
    SPECIFIER [E] := TRUE;
    SPECIFIER [M] := TRUE;
    IF PTROVR = 0 THEN
    BEGIN
      ANONYMOUS := TRUE;
      MODIFIER [B] := TRUE;
      MODIFIER [W] := TRUE;
      MODIFIER [D] := TRUE
    END;
    OFFSET := EFFECT.VALUE;
    IF NOT (EFFECT.BX OR EFFECT.BP OR EFFECT.SI OR EFFECT.DI) THEN
    SPECIFIER [X] := TRUE
    ELSE
    BEGIN
      BX := EFFECT.BX;
      BP := EFFECT.BP;
      SI := EFFECT.SI;
      DI := EFFECT.DI
    END
  END  (* EOARG *)
  ELSE ERROR (22)
END  (* REGISTER EXPRESSION *);
IF PTROVR <> 0 THEN
BEGIN
```

if ES then reg:=0 else if CS then reg:=1 else if SS then reg:=2 else reg:=3;

reg := offset mod 8 ; (potrzebne jest to dla proc 14 kodującej instrukcj esc)

```
        ANONYMOUS := FALSE;
        FOR I := 0 TO 3 DO MODIFIER [I] := FALSE;
        IF PTROVR = 1 THEN MODIFIER [B] := TRUE
        ELSE IF PTROVR IN [2, 6] THEN MODIFIER [W] := TRUE
        ELSE IF PTROVR IN [3, 7] THEN MODIFIER [D] := TRUE
        ELSE
        BEGIN
          ERROR (18);
          OK := FALSE
        END
      END;
      ARGUMENT := OK;
      IF NOT OK THEN
      BEGIN
        POS (POSITION);
        VERROR := 1
      END
    END  (* WITH RESULT *)
  END  (* ARGUMENT *);

  BEGIN  (* ARGREAD *)
    WITH RESULT DO
    BEGIN  (* WITH RESULT *)
      NUMBER := 0;
      ERRORV := 0;
      WHILE (NUMBER < 2) AND (ERRORV = 0) DO
      IF ARGUMENT (ARG [NUMBER + 1], ERRORV) THEN NUMBER := NUMBER + 1;
      IF NOT (EOLINE OR (CURRCHAR = ';')) THEN ERRORV := 100
      ELSE ERRORV := 0
    END  (* WITH RESULT *)
  END  (* ARGREAD *);

(*****************************************************************************)

          (*********************************************)
          (*                                           *)
          (*                C M M A T C H              *)
          (*                                           *)
          (*********************************************)

  FUNCTION CMMATCH: BOOLEAN;

(*****************************************************************************)
(*                                                                           *)
(*  PROCEDURA DOPASOWYWUJE CODEMACRO ODPOWIEDNIO DO MNEMONIKA I POS-         *)
(*  TACI ARGUMENTOW.                                                         *)
(*      DANE:                                                                *)
(*  MNEMNUMBER          - LICZBA MNEMONIKOW (JESLI SA DWA TO PIERWSZY        *)
(*                        JEST PREFIKSEM)                                    *)
(*  ACTPARAMETERS       - ZAWIERA INFORMACJE O AKTUALNYCH PARAMETRACH        *)
(*                        OSTATNIEGO MNEMONIKA W LINII                       *)
(*  MNEMADDR: ARRAY [1..2] OF INTEGER                                        *)
(*                      - ADRESY KOLEJNYCH MNEMONIKOW                        *)
(*                                                                           *)
(*      WYNIK:                                                               *)
(*  CMS: ARRAY [1..2] OF INTEGER                                             *)
(*                        - DOPASOWANE NUMERY CODEMACRO'SOW, JESLI           *)
(*                          CMMATCH = TRUE                                   *)
(*  CMMATCH = TRUE  OZNACZA,  ZE DOPASOWANO DO WSZYSTKICH MNEMONIKOW         *)
(*                  (WRAZ Z ARGUMENTAMI) ODPOWIEDNIE CODEMACRO'SY            *)
(*                  I WTEDY CMS ZAWIERA ADRESY TYCH CODEMACRO'SOW            *)
(*                                                                           *)
(*  PROCEDURY GLOBALNE:                                                      *)
                                                                          *)
```

```
(*       MATCH                                                          *)
(*                                                                       *)
(*************************************************************************)

VAR
  TMP, OK: BOOLEAN;
  I: INTEGER;

        (*****************************************)
        (*                                       *)
        (*             M A T C H                 *)
        (*                                       *)
        (*****************************************)

FUNCTION MATCH (VAR READARGS: ARGMTS; CODEM: INTEGER): BOOLEAN;

(*************************************************************************)
(*                                                                       *)
(*  FUNKCJA PORÓWNUJE ODCZYTANE PARAMETRY AKTUALNE (ARGUMENTY)           *)
(*  Z PARAMETRAMI W CODEMACRO.                                           *)
(*                                                                       *)
(*       OPIS PARAMETRÓW:                                                *)
(*  READARGS     -  ODCZYTANE ARGUMENTY                                  *)
(*  CODEM        -  NUMER CODEMACRO, Z KTÓRYM PORÓWNUJEMY                *)
(*                                                                       *)
(*       WYNIK:                                                          *)
(*  MATCH        =  TRUE  JEŚLI ZGODNOŚĆ CECH ODCZYTANYCH PARAMETRÓW     *)
(*                        Z WZORCEM W CODEMACRO                          *)
(*                =  FALSE  JEŚLI BRAK ZGODNOŚCI                         *)
(*                                                                       *)
(*  (TYPE ARGMTS = RECORD                                                *)
(*                   NUMBER: 0..2; (LICZBA ARGUMENTÓW)                   *)
(*                   ARG: ARRAY [1..2] OF ARGKIND                        *)
(*                 END                                 )                 *)
(*                                                                       *)
(*************************************************************************)

VAR
  I: INTEGER;
  OK: BOOLEAN;

        (*****************************************)
        (*                                       *)
        (*             P A R M E Q U             *)
        (*                                       *)
        (*****************************************)

FUNCTION PARMEQU (I: INTEGER): BOOLEAN;

(*************************************************************************)
(*                                                                       *)
(*  PRODEDURA PORÓWNUJE CZY ZACHODZI ZGODNOŚĆ MIĘDZY I-TYM PARAMETREM    *)
(*  FORMALNYM I AKTUALNYM. PARAMETR FORMALNY JEST OKREŚLONY PRZEZ        *)
(*  NUMER CODEMACRO (ZMIENNA GLOBALNA CODEM). PARAMETR AKTUALNY JEST     *)
(*  OPISANY W ZMIENNEJ GLOBALNEJ READARGS (PRZECZYTANE ARGUMENTY).       *)
(*                                                                       *)
(*************************************************************************)

VAR
  OK: BOOLEAN;

BEGIN  (* PARMEQU *)
  OK := FALSE;
```

```
    (* ZGODNOSC SPECYFIKATOROW *)

    WITH ACTPARAMETERS DO
    IF ARG [I].SPECIFIER [FIELD (ARGS [CODEM], 4+3*(I-1), 2+3*(I-1))]
    THEN  (* SPECYFIKATORY ZGODNE *)
    BEGIN  (* POROWNANIE MODYFIKATOROW I ZAKRESOW *)
      IF FIELD (ARGS [CODEM], 9+2*(I-1), 8+2*(I-1)) = 3 THEN OK := TRUE
      ELSE
      IF ARG [I].ANONYMOUS THEN
        IF NUMBER = 1 THEN OK := FALSE ELSE OK := TRUE
      ELSE OK := ARG [I].MODIFIER [FIELD (ARGS [CODEM],
                                          9+2*(I-1),8+2*(I-1))];
      IF OK THEN
      BEGIN  (* POROWNANIE ZAKRESOW *)
        (* NAJPIERW SPRAWDZAMY CZY WYSTEPUJE RANGE W DANYM CODEMACRO *)
        IF ((I=1) AND (FIELD (ARGS [CODEM], 15, 15) = 1)) OR
           ((I=2) AND ( FIELD (ARGS [CODEM], 1, 0) = 3) AND
            (FIELD (ARGS [CODEM], 15, 15) = 0)) THEN
          OK := ARG [I].RANGE [FIELD (ARGS [CODEM], 14, 12)]
      END  (* POROWNANIE ZAKRESOW *)
    END  (* POROWNANIE MODYFIKATOROW I ZAKRESOW *);
    PARMEQU := OK
  END  (* PARMEQU *);

(*****************************************************************************)

  BEGIN  (* MATCH *)
    WITH READARGS DO
    BEGIN  (* WITH *)

      (*  POROWNANIE LICZBY PARAMETROW FORMALNYCH I AKTUALNYCH *)

      I := FIELD (ARGS [CODEM], 1, 0);
      IF I = 3 THEN I := 2;
      IF I = NUMBER THEN  (* ZGODNOSC ILOSCI PARAMETROW AKTUALNYCH
                             I FORMALNYCH *)
                          (* NUMBER = LICZBA PARAMETROW AKTUALNYCH *)
      BEGIN  (* ZGODNOSC ILOSCI *)
        OK := TRUE;
        I := 0;
        WHILE I <> NUMBER DO
        BEGIN
          I := I + 1;
          OK := OK AND PARMEQU (I)
        END;
        IF OK THEN MATCH := TRUE ELSE MATCH := FALSE
      END  (* ZGODNOSC ILOSCI *)
      ELSE MATCH := FALSE
    END (* WITH *)
  END  (* MATCH *);

(*****************************************************************************)

  BEGIN  (* CMMATCH *)
    TMP := TRUE;
    FOR I := 1 TO MNEMNUMBER DO
    BEGIN  (* FOR I *)
      CMS [I] := ENTRY [ MNEMADDR [I]];
      (* ZACZYNAMY DOPASOWYWAC OD POCZATKU LANCUCHA CODEMACRO'SOW
         I JEDZIEMY DO KONCA LANCUCHA. POROWNANIE POLEGA NA POROWNANIU
         PARAMETROW AKTUALNYCH I FORMALNYCH *)
      LOOP
```

if prefx (mnemaddr [i]) then ok := true
else
begin {not prefix}

```
      OK := MATCH (ACTPARAMETERS, CMS [I]);
      EXIT IF OK OR FINISH [CMS [I]];
      CMS [I] := CMS [I] - 1
    END  (* LOOP *);
    TMP := TMP AND OK
  END  (* FOR I *);
  (* JESLI PO ZAKONCZENIU POWYZSZEJ PETLI TMP = TRUE, TO DLA
  WSZYSTKICH MNEMONIKOW ZOSTALY DOPASOWANE CODEMACRO'SY I ICH MNEMONIKI
  ZNAJDUJA SIE W CMS *)

  CMMATCH := TMP
END  (* CMMATCH *);

(*****************************************************************)

        (*********************************************)
        (*                                           *)
        (*              C O D E                      *)
        (*                                           *)
        (*********************************************)

PROCEDURE CODE;

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA KODUJE INSTRUKCJE WYKORZYSTUJAC DOPASOWANE CODEMACRO   *)
(*  (WZORZEC KODOWANIA). KOD WYNIKOWY JEST WSTAWIANY DO ZMIENNEJ OBJ. *)
(*      PROCEDURA CODE DZIALA PODCZAS DRUGIEGO I TRZECIEGO PRZEBIEGU *)
(*  ASEMBLERA.                                                   *)
(*      GLOWNYM ZADANIEM PROCEDURY CODE PODCZAS DRUGIEGO PRZEBIEGU   *)
(*  JEST JEDYNIE OKRESLENIE O ILE TRZEBA ZWIEKSZYC LC (W DRUGIM PRZE- *)
(*  BIEGU NIE SA JESZCZE ZNANE WSZYSTKIE OFFSETY, A WIEC NIE MOZNA   *)
(*  JESZCZE GENEROWAC PELNEGO KODU WYNIKOWEGO). TEN PRZYROST LC JEST *)
(*  ZAPAMIETANY I PODCZAS TRZECIEGO PRZEBIEGU BEDZIE TO ILOSC BAJTOW *)
(*  PRZEZNACZONA DLA INSTRUKCJI. INSTRUKCJA MOZE ZAJAC MNIEJ BAJTOW  *)
(*  I WTEDY WOLNE BAJTY ZOSTANA WYPELNIONE KODEM OPERACJI NOP.       *)
(*      PRZYROSTY LC SA ZAPAMIETYWANE W TYMCZASOWYM PLIKU            *)
(*  INCREMENTS: FILE OF INTEGER. TRZECI PRZEBIEG KORZYSTA Z TEGO     *)
(*  PLIKU.                                                       *)
(*      PROCEDURA KORZYSTA Z DANYCH PRZEKAZANYCH PRZEZ NASTEPUJACE   *)
(*  ZMIENNE GLOBALNE:                                            *)
(*  LC: INTEGER                  - LOCATION COUNTER                  *)
(*  PASS: 1..3             - NUMER PRZEBIEGU                     *)
(*  ACTPARAMETERS: ARGNTS                                        *)
(*                         - INFORMACJA O ARGUMENTACH            *)
(*  CMS: ARRAY [1..2] OF INTEGER                                 *)
(*                         - NUMERY DOPASOWANYCH CODEMACRO       *)
(*  MNEMNUMBER: 1..2       - LICZBA MNEMONIKOW                   *)
(*                                                               *)
(*      WYNIK DZIALANIA PROCEDURA PRZEKAZUJE W ZMIENNYCH:        *)
(*  OBJ                          - OBJ.NOBYTES = LICZBA BAJTOW ZAREZERWOWANYCH *)
(*                           DLA INSTRUKCJI (PRZEBIEG 2) LUB ZAJETYCH *)
(*                           (PRZEBIEG 3) PRZEZ INSTRUKCJE       *)
(*                           OBJ.CONTENTS = BAJTY KODU WYNIKOWEGO *)
(*                                                               *)
(*****************************************************************)

CONST
  A = 0; C = 1; D = 2; E = 3; M = 4; R = 5; S = 6; X = 7; B = 0; W = 1;

VAR
  I, TMP, CM, OLDNOBYTES: INTEGER;
        (*********************************************************)
```

```
        (*  CM          - NUMER AKTULANEGO CODEMACRO (UZYWANY PRZEZ   *)
        (*                PROCEDURY KODUJACE)                         *)
        (*****************************************************)

        (*************************************************)
        (*                                               *)
        (*              W C O D E                        *)
        (*                                               *)
        (*************************************************)

PROCEDURE WCODE (VAR W: INTEGER; X, K, L: INTEGER);

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA WSTAWIA DO ZMIENNEJ W TYPU INTEGER LICZBE X , LICZBA  X *)
(*  WSTAWIANA JEST NA BITY K..L (WLACZNIE).NUMERACJA BITOW W SLOWIE   *)
(*  JEST NASTEPUJACA: NAJSTARSZY BIT MA NUMER 15, NAJMLODSZY NUMER 0. *)
(*                                                               *)
(*                                                               *)
(*  PROCEDURY GLOBALNE:                                          *)
(*      POTEGA                                                   *)
(*                                                               *)
(*                                                               *)
(*****************************************************************)
VAR
  WP,WK,XP:INTEGER;
BEGIN
  WP:=POTEGA(L);
  WP:=W MOD WP;
  WK:=POTEGA(K+1);
  WK:=W DIV WK;
  WK:=WK*POTEGA(K+1);
  W:=WK+WP;
  XP:=POTEGA(K-L+1);
  X:=X MOD XP;
  X:=X*POTEGA(L);
  W:=W+X;
END (* CODE *);

        (*************************************************)
        (*                                               *)
        (*              M O D R M                        *)
        (*                                               *)
        (*************************************************)

PROCEDURE MODRM (P, Q: INTEGER);

(*****************************************************************)
(*                                                               *)
(*  PROCEDURA KODUJE DRUGI BAJT INSTRUKCJI (BAJT MODRM).         *)
(*                                                               *)
(*  ZNACZENIE PARAMETROW:                                        *)
(*                                                               *)
(*  P           - OKRESLA SPOSOB KODOWANIA POLA REG              *)
(*                P = 0  NA POLU REG UMIESZCZONE ZOSTAJE 3-BITOWE  *)
(*                       ROZSZERZENIE KODU OPERACJI (TO ROZSZERZENIE *)
(*                       ZNAJDUJE SIE NA BITACH 10-8 W CODING [CM]) *)
(*                P = 1  NA POLU REG UMIESZCZONY ZOSTAJE NUMER REJE- *)
(*                       STRU OKRESLONY PRZEZ POSTAC PIERWSZEGO    *)
(*                       ARGUMENTU (DESTINATION)                   *)
(*                P = 2  NA POLU REG UMIESZCZONY ZOSTAJE NUMER REJE- *)
(*                       STRU OKRESLONY PRZEZ POSTAC DRUGIEGO ARGU- *)
(*                       MENTU (SOURCE)                            *)
```

```
(*                                                                    *)
(*  Q           - OKRESLA SPOSOB KODOWANIA POL MOD I R/M. SPOSOB TEN  *)
(*                (TRYB ADRESOWANIA) ZALEZY Z KOLEI OD POSTACI ARGU-  *)
(*                MENTU. I TAK DLA                                    *)
(*                Q = 1  BRANY POD UWAGE JEST PIERWSZY ARGUMENT       *)
(*                       (DESTINATION)                                *)
(*                Q = 2  BRANY POD UWAGE JEST DRUGI ARGUMENT (SOURCE) *)
(*                                                                    *)
(*  ZMIENNE GLOBALNE:                                                 *)
(*      ACTPARAMETERS: ARRAY [1..2] OF ARGKIND                        *)
(*                (AKTUALNE PARAMETRY)                                *)
(*      CM: INTEGER   (NUMER AKTUALNIE ROZPATRYWANEGO CODEMACRO)      *)
(*      CODING: ARRAY [1..NOCM]   (TABLICA ZAWIERAJACA KODY OPERACYJ- *)
(*                        NE INSTRUKCJI)                              *)
(*      OBJ: RECORD                                                   *)
(*             NOBYTES: 0..9;                                         *)
(*             CONTENTS: ARRAY [1..9] OF BYTE86                       *)
(*           END         (TU LADOWANY JEST KOD WYNIKOWY)              *)
(*                                                                    *)
(**********************************************************************)

VAR
  MODFIELD, REGFIELD, RMFIELD: INTEGER;

        (*********************************************)
        (*                                           *)
        (*              M O D R M B Y T E            *)
        (*                                           *)
        (*********************************************)

FUNCTION MODRMBYTE: INTEGER;

(**********************************************************************)

BEGIN  (* MODRMBYTE *)
  MODRMBYTE := 64 * MODFIELD + 8 * REGFIELD + RMFIELD
END  (* MODRMBYTE *);

(**********************************************************************)

BEGIN  (* MODRM *)
  WITH ACTPARAMETERS DO
  BEGIN  (* WITH ACTPARAMETERS *)
    IF P = 0 THEN REGFIELD := FIELD (CODING [CM], 10, 8)
    ELSE REGFIELD := ARG [P].REG;
    WITH ARG [Q] DO
    BEGIN  (* WITH ARG [Q] *)
      IF SPECIFIER [R] THEN  (* DRUGI PARAMETR JEST REJESTREM *)
      BEGIN  (* REGISTER *)
        MODFIELD := 3;
        RMFIELD := REG;
        OBJ.CONTENTS [OBJ.NOBYTES + 1] := MODRMBYTE;
        OBJ.NOBYTES := OBJ.NOBYTES + 1;
      END  (* REGISTER *)
      ELSE IF SPECIFIER [X] THEN  (* DIRECT MEMORY REFERENCE *)
      BEGIN  (* X *)
        MODFIELD := 0;
        RMFIELD := 6;
        WITH OBJ DO
        BEGIN  (* OBJ *)
          CONTENTS [NOBYTES + 1] := MODRMBYTE;
          CONTENTS [NOBYTES + 2] := OFFSET MOD 256;
          CONTENTS [NOBYTES + 3] := OFFSET DIV 256;
```

```
          NOBYTES := NOBYTES + 3
        END  (* OBJ *)
      END  (* X *)
      ELSE  (* MEMORY *)
      BEGIN  (* M *)

        (* USTAWIENIE POLA "MOD" *)

        IF OFFSET = 0 THEN
          IF NOT (BP AND NOT SI AND NOT DI) THEN MODFIELD := 0
          ELSE MODFIELD := 1
        ELSE IF (OFFSET >= -128) AND (OFFSET <= 127) THEN MODFIELD := 1
        ELSE MODFIELD := 2;

        (* USTAWIENIE POLA R/M *)

        IF BX AND SI THEN RMFIELD := 0
        ELSE IF BX AND DI THEN RMFIELD := 1
        ELSE IF BP AND SI THEN RMFIELD := 2
        ELSE IF BP AND DI THEN RMFIELD := 3
        ELSE IF SI THEN RMFIELD := 4
        ELSE IF DI THEN RMFIELD := 5
        ELSE IF BP THEN RMFIELD := 6
        ELSE IF BX THEN RMFIELD := 7;

        WITH OBJ DO
        BEGIN  (* WITH OBJ *)
          CONTENTS [NOBYTES + 1] := MODRMBYTE;
          NOBYTES := NOBYTES + 1;
          IF MODFIELD = 1 THEN
          BEGIN
            CONTENTS [NOBYTES + 1] := OFFSET;
            NOBYTES := NOBYTES + 1
          END
          ELSE IF MODFIELD = 2 THEN
          BEGIN
            CONTENTS [NOBYTES + 1] := OFFSET MOD 256;
            CONTENTS [NOBYTES + 2] := OFFSET DIV 256;
            NOBYTES := NOBYTES + 2
          END
        END  (* WITH OBJ *)
      END  (* M *)
    END  (* WITH ARG [Q] *)
  END  (* WITH ACTPARAMETERS *)
    END  (* MODRM *);

(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*              D B C O D E                      *)
        (*                                               *)
        (*************************************************)

PROCEDURE DBCODE;

BEGIN  (* DBCODE *)
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
    CONTENTS [NOBYTES + 1] := FIELD (CODING [CM], 7, 0);
    NOBYTES := NOBYTES + 1
  END  (* WITH OBJ *)
END  (* DBCODE *);
```

```
(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*                D W C O D E                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE DWCODE;

BEGIN  (* DWCODE *)
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
    CONTENTS [NOBYTES + 1] := FIELD (CODING [CM], 7, 0);
    CONTENTS [NOBYTES + 2] := FIELD (ARGS [CM], 9, 2);
    NOBYTES := NOBYTES + 2
  END  (* WITH OBJ *)
END  (* DWCODE *);

(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*                S E G F I X                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE SEGFIX (K: INTEGER);

BEGIN  (* SEGFIX *)
END  (* SEGFIX *);

(*****************************************************************)

        (*************************************************)
        (*                                               *)
        (*                D B                            *)
        (*                                               *)
        (*************************************************)

PROCEDURE DB (K: INTEGER);

BEGIN  (* DB *)
  WITH ACTPARAMETERS DO
  WITH OBJ DO
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    CONTENTS [NOBYTES + 1] := ARG [K].OFFSET;
    NOBYTES := NOBYTES + 1
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* DB *);

        (*************************************************)
        (*                                               *)
        (*                D W                            *)
        (*                                               *)
        (*************************************************)

PROCEDURE DW (K: INTEGER);

BEGIN  (* DW *)
  WITH ACTPARAMETERS DO
  WITH OBJ DO
```

```
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    CONTENTS [NOBYTES + 1] := FIELD (ARG [K].OFFSET, 7, 0);
    CONTENTS [NOBYTES + 2] := FIELD (ARG [K].OFFSET, 15, 8);
    NOBYTES := NOBYTES + 2
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* DW *);

        (*********************************************)
        (*                                           *)
        (*                D D                        *)
        (*                                           *)
        (*********************************************)

PROCEDURE DD;

VAR
  I: INTEGER;

BEGIN  (* DD *)
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
    FOR I := 1 TO 4 DO CONTENTS [NOBYTES + I] := 0;
    NOBYTES := NOBYTES + 4
  END  (* WITH OBJ *)
END  (* DD *);

        (*********************************************)
        (*                                           *)
        (*                R E L W                    *)
        (*                                           *)
        (*********************************************)

PROCEDURE RELW;

VAR
  TEMP: INTEGER;

BEGIN  (* RELW *)
  WITH ACTPARAMETERS DO
  WITH OBJ DO
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    TEMP := ARG [1].OFFSET - (LC + 3);
    CONTENTS [NOBYTES + 1] := FIELD (TEMP, 7, 0);
    CONTENTS [NOBYTES + 2] := FIELD (TEMP, 15, 8);
    NOBYTES := NOBYTES + 2
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* RELW *);

        (*********************************************)
        (*                                           *)
        (*                N O S E G F I X            *)
        (*                                           *)
        (*********************************************)

PROCEDURE NOSEGFIX (SEGREG, K: INTEGER);

BEGIN  (* NOSEGFIX *)
END  (* NOSEGFIX *);

        (*********************************************)
        (*                                           *)
        (*                R 5 3 C O D E              *)
        (*                                           *)
```

```
        (*************************************************)

PROCEDURE R53CODE (K: INTEGER);

BEGIN  (* R53CODE *)
  WITH ACTPARAMETERS DO
  WITH OBJ DO
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    CONTENTS [NOBYTES + 1] := 8 * FIELD (CODING [CM], 7, 3)
                              + ARG [K],REG;
    NOBYTES := NOBYTES + 1
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* R53CODE *);

        (*************************************************)
        (*                                               *)
        (*              R 5 3 C O P 3                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE R53COP3;

BEGIN  (* R53COP3 *)
  WITH ACTPARAMETERS DO
  WITH OBJ DO
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    CONTENTS [NOBYTES + 1] := 8 * FIELD (CODING [CM], 7, 3)
                              + ARG [1],OFFSET DIV 8;
    NOBYTES := NOBYTES + 1
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* R53COP3 *);

        (*************************************************)
        (*                                               *)
        (*              R E L B                          *)
        (*                                               *)
        (*************************************************)

PROCEDURE RELB;

VAR
  TEMP: INTEGER;

BEGIN  (* RELB *)
  WITH ACTPARAMETERS DO
  WITH OBJ DO
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    TEMP := ARG [1].OFFSET - (LC + 2);
    IF TEMP < 0 THEN TEMP := TEMP + 256;
    CONTENTS [NOBYTES + 1] := TEMP;
    NOBYTES := NOBYTES + 1
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* RELB *);

        (*************************************************)
        (*                                               *)
        (*              R 3 2 3 C D C                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE R323CDC;

BEGIN  (* R323CDC *)
```

```
  WITH ACTPARAMETERS DO
  WITH OBJ DO
  BEGIN  (* WITH ACTPARAMETERS AND OBJ *)
    CONTENTS [NOBYTES + 1] := FIELD (CODING [CM], 7, 0);
    WCODE (CONTENTS [NOBYTES + 1], ARG [1].REG, 4, 3);
    NOBYTES := NOBYTES + 1
  END  (* WITH ACTPARAMETERS AND OBJ *)
END  (* R323CDC *);

        (*****************************************)
        (*                                       *)
        (*              R 4 1 3 C P R C          *)
        (*                                       *)
        (*****************************************)

PROCEDURE R413CPRC;

BEGIN  (* R413CPRC *)
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
    CONTENTS [NOBYTES + 1] := FIELD (CODING [CM], 7, 0);
    WCODE (CONTENTS [NOBYTES + 1], PROCLEN MOD 1, 3, 3);
    NOBYTES := NOBYTES + 1
  END  (* WITH OBJ *)
END  (* R413CPRC *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 1                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC1;

BEGIN  (* PROC1 *).
  DBCODE
END  (* PROC1 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 2                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC2;

BEGIN  (* PROC2 *)
  DBCODE
END  (* PROC2 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 3                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC3;

BEGIN  (* PROC3 *)
  SEGFIX (1);
  DBCODE;
  MODRM (0, 1);
  DB (2)
```

```
END  (* PROC3 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 4                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC4;

BEGIN  (* PROC4 *)
  SEGFIX (1);
  DDCODE;
  MODRM (0, 1);
  DW (2)
END  (* PROC4 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 5                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC5;

BEGIN  (* PROC5 *)
  DDCODE;
  DB (2)
END  (* PROC5 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 6                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC6;

BEGIN  (* PROC6 *)
  DDCODE;
  DW (2)
END  (* PROC6 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 7                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC7;

BEGIN  (* PROC7 *)
  SEGFIX (1);
  DDCODE;
  MODRM (2, 1)
END  (* PROC7 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 8                *)
        (*                                       *)
        (*****************************************)
```

```
PROCEDURE PROC8;

BEGIN  (* PROC8 *)
  SEGFIX (2);
  DBCODE;
  MODRM (1, 2)
END  (* PROC8 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 9                *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC9;

BEGIN  (* PROC9 *)
  SEGFIX (1);
  DBCODE;
  MODRM (0, 1)
END  (* PROC9 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 10               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC10;

BEGIN  (* PROC10 *)
  DBCODE;
  DD
END  (* PROC10 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 11               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC11;

BEGIN  (* PROC11 *)
  DBCODE;
  RELW
END  (* PROC11 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 12               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC12;

BEGIN  (* PROC12 *)
  NOSEGFIX (0, 2);
  SEGFIX (1);
  DBCODE
END  (* PROC12 *);

        (*****************************************)
```

```
        (*                                                    *)
        (*                  P R O C 13                        *)
        (*                                                    *)
        (******************************************************)

PROCEDURE PROC13;

BEGIN  (* PROC13 *)
  R53CODE (1)
END  (* PROC13 *);

        (******************************************************)
        (*                                                    *)
        (*                  P R O C 14                        *)
        (*                                                    *)
        (******************************************************)

PROCEDURE PROC14;

BEGIN  (* PROC14 *)
  SEGFIX (2);
  R53COP3;
  MODRM (1, 2)
END  (* PROC14 *);

        (******************************************************)
        (*                                                    *)
        (*                  P R O C 15                        *)
        (*                                                    *)
        (******************************************************)

PROCEDURE PROC15;

BEGIN  (* PROC15 *)
  DBCODE;
  DB (1)
END  (* PROC15 *);

        (******************************************************)
        (*                                                    *)
        (*                  P R O C 16                        *)
        (*                                                    *)
        (******************************************************)

PROCEDURE PROC16;

BEGIN  (* PROC16 *)
  DBCODE;
  RELB
END  (* PROC16 *);

        (******************************************************)
        (*                                                    *)
        (*                  P R O C 17                        *)
        (*                                                    *)
        (******************************************************)

PROCEDURE PROC17;

BEGIN  (* PROC17 *)
  DBCODE;
  MODRM (1, 2)
END  (* PROC17 *);
```

```
        (*************************************************)
        (*                                               *)
        (*              P R O C 18                       *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC18;

BEGIN  (* PROC18 *)
  SEGFIX (1);
  DBCODE
END  (* PROC18 *);

        (*************************************************)
        (*                                               *)
        (*              P R O C 19                       *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC19;

BEGIN  (* PROC19 *)
  R53CODE (1);
  DB (2)
END  (* PROC19 *);

        (*************************************************)
        (*                                               *)
        (*              P R O C 20                       *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC20;

BEGIN  (* PROC20 *)
  R53CODE (1);
  DU (2)
END  (* PROC20 *);

        (*************************************************)
        (*                                               *)
        (*              P R O C 21                       *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC21;

BEGIN  (* PROC21 *)
  SEGFIX (2);
  DBCODE;
  DU (2)
END  (* PROC21 *);

        (*************************************************)
        (*                                               *)
        (*              P R O C 22                       *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC22;

BEGIN  (* PROC22 *)
```

```
  SEGFIX (1);
  DBCODE;
  DV (1)
END  (* PROC22 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 23               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC23;

BEGIN  (* PROC23 *)
  NOSEGFIX (0, 1);
  SEGFIX (2);
  DBCODE
END  (* PROC23 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 24               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC24;

BEGIN  (* PROC24 *)
END  (* PROC24 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 25               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC25;

BEGIN  (* PROC25 *)
  DBCODE;
  DB (1)
END  (* PROC25 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 26               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC26;

BEGIN  (* PROC26 *)
  R323CDC
END  (* PROC26 *);

        (*****************************************)
        (*                                       *)
        (*              P R O C 27               *)
        (*                                       *)
        (*****************************************)

PROCEDURE PROC27;
```

```
BEGIN  (* PROC27 *)
  SEGFIX (1);
  DBCODE;
  MCDRM (0, 1)
END  (* PROC27 *);

        (*************************************************)
        (*                                               *)
        (*                 P R O C 28                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC28;

BEGIN  (* PROC28 *)
  R413CPRC;
  DW (* 1)
END  (* PROC28 *);

        (*************************************************)
        (*                                               *)
        (*                 P R O C 29                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC29;

BEGIN  (* PROC29 *)
  R413CPRC
END  (* PROC29 *);

        (*************************************************)
        (*                                               *)
        (*                 P R O C 30                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC30;

BEGIN  (* PROC30 *)
  NOSEGFIX (0, 1);
  DBCODE
END  (* PROC30 *);

        (*************************************************)
        (*                                               *)
        (*                 P R O C 31                    *)
        (*                                               *)
        (*************************************************)

PROCEDURE PROC31;

BEGIN  (* PROC31 *)
  R53CODE (2)
END  (* PROC31 *);

(*****************************************************************

BEGIN  (* CODE *)
  WITH OBJ DO
  BEGIN  (* WITH OBJ *)
  FOR I := 1 TO MNEMNUMBER DO
  BEGIN  (* FOR I *)
```

```
      CM := CMS [I];
      TMP := OBJ.NOBYTES;
      CASE FIELD (CODING [CM], 15, 11) OF
        1: PROC1;
        2: PROC2;
        3: PROC3;
        4: PROC4;
        5: PROC5;
        6: PROC6;
        7: PROC7;
        8: PROC8;
        9: PROC9;
        10: PROC10;
        11: PROC11;
        12: PROC12;
        13: PROC13;
        14: PROC14;
        15: PROC15;
        16: PROC16;
        17: PROC17;
        18: PROC18;
        19: PROC19;
        20: PROC20;
        21: PROC21;
        22: PROC22;
        23: PROC23;
        24: PROC24;
        25: PROC25;
        26: PROC26;
        27: PROC27;
        28: PROC28;
        29: PROC29;
        30: PROC30;
        31: PROC31;
        OTHERS:
      END  (* CASE *);
      LC := LC + (OBJ.NOBYTES - TMP)
    END  (* FOR I *)
    IF PASS = 2 THEN WRITE (RESERVE, OBJ.NOBYTES);
    IF PASS = 3 THEN
    BEGIN  (* PASS = 3 *)
      READ (RESERVE, OLDNOBYTES);
      IF OBJ.NOBYTES > OLDNOBYTES THEN
      BEGIN
        ERROR (16);
        OBJ.NOBYTES := OLDNOBYTES
      END
      ELSE IF OBJ.NOBYTES < OLDNOBYTES THEN
      FOR I := OBJ.NOBYTES+1 TO OLDNOBYTES DO
      BEGIN
        OBJ.CONTENTS [I] := 144;     (* =90H - KOD OPERACJI NOP *)
        OBJ.NOBYTES := OBJ.NOBYTES + 1;
        LC := LC + 1
      END
    END  (* PASS = 3 *)
    END  (* WITH OBJ *)
  END  (* CODE *);

  (*************************************************************

  BEGIN  (* INSTRUCTN *)

    (* SPRAWDZENIE POLA ETYKIETY *)
```

```
  IF LABELFIELD.NAME THEN
  BEGIN
    ERROR (11); (* BRAK ':' PO DEFINICJI ETYKIETY *)
    (* PRZYJMUJEMY, ZE PROGRAMISTA ZAPOMNIAL NAPISAC ':' *)
    (*  KOREKCJA BLEDU PROGRAMISTY : *)
    LABELFIELD.LAB := TRUE;
    LABELFIELD.NAME := FALSE
  END;

  (* ROZPOZNANIE DALSZEJ CZESCI LINII *)

  MNEMNUMBER := 1;
  MNEMADDR [1] := ACTMNEM;
  IF PREFX (ACTMNEM) THEN
  BEGIN  (* PREFIX *)
    IF MNEMONIC (ACTMNEM) THEN
    BEGIN
      IF NOT PREFX (ACTMNEM) THEN
      BEGIN
        MNEMNUMBER := MNEMNUMBER + 1;
        MNEMADDR [MNEMNUMBER] := ACTMNEM
      END
      ELSE ERROR (12)  (* NIEDOPUSZCZAMY DWOCH PREFIKSOW W JEDNYM
                          WIERSZU *)
    END
    ELSE IF EOLINE OR (CURRCHAR = ';') THEN GOTO 1
    ELSE ERROR (15);
    IF NOT PREFX (MNEMNUMBER) AND (PASS <> 1) THEN
    BEGIN  (* CZYTAJ ARGUMENTY *)
      ARGREAD (ACTPARAMETERS, E);
      IF E <> 0 THEN ERROR (1)  (* BLEDNA POSTAC ARGUMENTU *)
    END
  END  (* PREFX *);

    (* KONIEC ROZPOZNANIA LINII *)

    (*************************************************************)
    (* W TYM MOMENCIE STAN JEST NASTEPUJACY:                     *)
    (*   MNEMNUMBER      -  ZAWIERA LICZBE MNEMONIKOW W LINII (MOGA *)
    (*                      BYC MAKSYMALNIE 2)                     *)
    (*   LABELFIELD      -  ZAWIERA INFORMACJE O ZAWARTOSCI POLA    *)
    (*                      ETYKIETY                               *)
    (*   ACTPARAMETERS -  JESLI JEST PRZEBIEG 2 LUB 3 TO ZAWIERA   *)
    (*                    INFORMACJE O ARGUMENTACH. W PRZYPADKU    *)
    (*                    PRZEBIEGU 2 JEST TO INFORMACJA NIEPELNA. *)
    (*************************************************************)

  IF PASS = 1 THEN
  BEGIN  (* PASS = 1 *)
    IF LABELFIELD.LAB THEN  (* JEST ETYKIETA - WPROWADZ JA DO
                               SKOROWIDZA *)
    BEGIN
      ENTER (LABELFIELD.CONT, NEAR, FALSE, 0, RES);
      IF RES = -1 THEN ERROR (1);  (* POWTORNA DEFINICJA IDENT. *)
      IF RES = -2 THEN ERROR (2)   (* PRZEPELNIENIE SKOROWIDZA *)
    END
  END  (* PASS = 1 *)
  ELSE
  BEGIN  (* PASS = 2 OR PASS = 3 *)
    (*  JESLI PRZEBIEG 2 I ETKIETA TO WPROWADZ JEJ OFFSET DO SKOROWIDZA
        UZYTKOWNIKA *)
    IF (PASS = 2) AND LABELFIELD.LAB THEN
```

```
    BEGIN
      FIND (LABELFIELD.CONT, LOC, FOUND);
      USERIDENT [LOC].OFFSET := LC;
      USERIDENT [LOC].DETOFFSET := TRUE
    END;
    ARGREAD (ACTPARAMETERS, ER);
    IF ER = 0 THEN
    BEGIN  (* ER = 0 *)
1:    IF CHMATCH THEN  (* DOPASOWANO CODEMACRO *)
      CODE
      ELSE ERROR (14)  (* NIE DOPASOWANO CODEMACRO *)
    END  (* ER = 0 *)
    ELSE ERROR (13)  (* ZLA POSTAC ARGUMENTOW *)
  END  (* PASS = 2 OR PASS = 3 *)
END  (* INSTRUCTN *);

(*****************************************************************)

BEGIN  (* DOLINE *)
  IF NOT EOLINE THEN
  BEGIN  (* NOT EOLINE *)
    NOPRINTED := TRUE;

    (* PRZETWORZENIE POLA ETYKIETY *)

    LABELFIELD.LAB := FALSE;
    LABELFIELD.NAME := FALSE;
    COMMENT := FALSE;
    GETCHAR;
    IF CURRCHAR = ' ' THEN  (* NIC NIE MA W POLU ETYKIETY *)
    NEXTITEM
    ELSE IF IDENT (ID) THEN
    BEGIN  (* IDENTYFIKATOR NA POCZATKU LINII *)
      WITH LABELFIELD DO
      BEGIN  (* WITH LABELFIELD *)
        IF CURRCHAR = ':' THEN
        BEGIN
          GETCHAR;
          NEXTITEM;
          LAB := TRUE
        END
        ELSE NAME := TRUE;
        CONT := ID
      END  (* WITH LABELFIELD *)
    END  (* IDENTYFIKATOR NA POCZATKU LINII *)
    ELSE IF CURRCHAR = ';' THEN COMMENT := TRUE  (* LINIA KOMENTARZA *)
    ELSE ERROR (5);  (* BLEDNY POCZATEK LINII *)

    (* KONIEC PRZETWARZANIA POLA ETYKIETY *)

    (* PRZETWARZANIE POLA MNEMONIKA *)

    IF NOT COMMENT THEN
      IF MNEMONIC (ACTMNEM) THEN
        IF INS [ACTMNEM] THEN INSTRUCTN
        ELSE DIRECTIVE
      ELSE ERROR (6)  (* BRAK MNEMONIKA *)

    (* KONIEC PRZETWARZANIA POLA MNEMONIKA *)

  END  (* NOT EOLINE *)
END  (* DOLINE *);
```

patrz pomocniczy listing.

end

```
mnemnumber := 0;
repeat
  mnemnumber := mnemnumber + 1;
  mnemaddr [mnemnumber] := actmnem;
  pre := prefx (actmnem)
until (not pre) or (mnemnumber = maxmn) or
                   (not mnemonic(actmnem));
if pre then error (26)
else
begin
```

```
(*************************************************************************)

        (*****************************************)
        (*                                       *)
        (*            C H A N G E                *)
        (*                                       *)
        (*****************************************)

PROCEDURE CHANGE(X:BYTE86;VAR TAB:WORDCHAR);


(*************************************************************************)
(*                                                                       *)
(*  PROCEDURA DOKONUJE KONWERSJI BAJTU (ZMIENNA X)  NA KOD               *)
(*  HEXADECYMALNY. PO WYKONANIU PROCEDURY WYNIK UMIEJSZCZONY JEST        *)
(*  W ZMIENNEJ TAB KTORA JEST TYPU WORDCHAR.                             *)
(*                                                                       *)
(*************************************************************************)
VAR
  WART,DZIELNA,K:BYTE86;
BEGIN
 WART:=8;DZIELNA:=128;K:=0;
 WHILE DZIELNA<>8 DO
 BEGIN
  IF(X DIV DZIELNA)<>0 THEN
  BEGIN
   K:=WART+K;
   X:=X-DZIELNA
  END;
  DZIELNA:=DZIELNA DIV 2;
  WART:=WART DIV 2
END(* LICZBA SZESNASTKOWA W POSTACI KX *);
 IF K>9 THEN TAB[2]:= CHR(K+55)
        ELSE TAB[2]:= CHR(K+48);
 IF X>9 THEN TAB[1]:= CHR(X+55)
        ELSE TAB[1]:= CHR(X+48)
END(* CHANGE *);
(*                                                                       *)

(*************************************************************************)

        (*****************************************)
        (*                                       *)
        (*            O B J O U T                *)
        (*                                       *)
        (*****************************************)

PROCEDURE OBJOUT;

(*-----------------------------------------------------------------------*)
(       PROCEDURA WYPROWADZA KOD WYNIKOWY NA DYSK                        )
(-----------------------------------------------------------------------*)

        (* WERSJA PROBNA - NIC NIE ROBI ! *)

BEGIN   (* OBJOUT *)
END   (* OBJOUT *);

(*************************************************************************)

        (*****************************************)
        (*                                       *)
```

patrz poprzedni listing

25 Jan 84

OBJOUT

```
procedure objout;

var i: integer;

begin { objout }
  if out then
  with obj do
  for i:= 1 to nobytes do
  begin { for i }
    write ( object , lc1 + i - 1);
    write ( object , contents [i])
  end { for i }

end { objout } ;
```

```
(*                    L I S T                                          *)
(*                                                                     *)
(*******************************************************************)

PROCEDURE LIST;

(*******************************************************************)
(*                                                                     *)
(*  PROCEDURA TWORZY LISTING NA ZBIORZE LISTFILE I/LUB TTY.            *)
(*                                                                     *)
(*  PROCEDURY GLOBALNE:                                                *)
(*                        CHANGE                                       *)
(*                                                                     *)
(*******************************************************************)
VAR
J, ILOSC,SZEROKOSC,LICZBA,P,K,RESZTA:INTEGER;
(*.....................................................................*)

FUNCTION MINIMUM(X,Y:INTEGER):INTEGER;

(*                                                                     *)
(*  PROCEDURA PODAJE MNIEJSZA WARTOSC Z LICZB X I Y.                   *)
(*                                                                     *)
BEGIN                                                                 *)
  IF X<Y THEN MINIMUM:=X
         ELSE MINIMUM:=Y
END (* MINIMUM *);
(*.....................................................................*)

PROCEDURE SPACJA(L:INTEGER;VAR PLIK:TEXT);

(*                                                                     *)
(*  PROCEDURA WPROWADZA DO PLIKU FILE L SPACJI.                        *)
(*                                                                     *)
VAR                                                                   *)
  I:INTEGER;
BEGIN
  FOR I:=1 TO L DO WRITE(PLIK,' ')
END (* SPACJA *);
(*.....................................................................*)

PROCEDURE POM(VAR PLIK:TEXT;POCZ,KON:INTEGER;VAR LUB:INTEGER);

(*.....................................................................*)

(*                                                                     *)
(*  PROCEDURA WPROWADZA TEST ZRODLOWY DO PLIKU  "PLIK".                *)
(*  TEST ZRODLOWY ZNAJDUJE SIE W TABLICY  " LINE ".JESLI ZNAK          *)
(*  W TABLICY LINE JEST "HT" WOWCZAS W ZALEZNOSCI OD NUMERU            *)
(*  KOLUMNY W KTOREJ TEN ZNAK WYSTAPIL WPROWADZANA JEST ODPOWIEDNIA    *)
(*  LICZBA SPACJI .                                                    *)
(*                                                                     *)
(*.....................................................................*)

VAR
  V,LICZX,Z:INTEGER;
BEGIN
 LICZX:=KON-POCZ+1;
 LUB:=0;
 Z:=POCZ;
 REPEAT
  IF ORD(LINE[ Z ]) = 11B THEN
```

```
    BEGIN
      V:=Z MOD 8;
      IF V=0 THEN V:=1
             ELSE V:=9-V;
      SPACJA(V,PLIK);
      LUB:=LUB+V;
      LICZX:=LICZX-V
    END
  ELSE
  BEGIN
    WRITE(PLIK,LINE[Z]);
    LICZX:=LICZX-1
  END;
  Z:=Z+1
UNTIL LICZX <=0
END (* POM *);

(*.....*)

PROCEDURE TYTUL(VAR PLIK:TEXT;Z:CHAR);

(*                                                                    *)
(*  PROCEDURA WYPROWADZA PIERWSZE LINIE LISTINGU BEDACE NAGLOWKIEM    *)
(*  JAK ROWNIEZ POWTARZA JE ZA KAZDYM RAZEM GDY PRZECHODZINY          *)
(*  NA NASTEPNA STRONE.                                               *)
(*                                                                    *)
VAR
  I:INTEGER;
BEGIN
WRITE(PLIK,'   MCS-86 MACRO  ASSEMBLER          ');
FOR I:=1 TO 10 DO WRITE(PLIK,CURRDATE[I]);
SPACJA(6,PLIK);
FOR I:=1 TO 10 DO WRITE(PLIK,DAYTIME[I]);
SPACJA(6,PLIK);
IF Z='L' THEN WRITELN(PLIK,'PAGE  ',PAGENUM);
WRITELN(PLIK);WRITELN(PLIK);
IF Z='T' THEN
BEGIN
  WRITELN(TTY,' ');
  WRITELN(TTY,' ')
END;
WRITELN(PLIK,'LOC  OBJ                    LINE     SOURCE');
WRITELN(PLIK);
IF Z='T' THEN WRITELN(TTY,' ')
END (* TYTUL *);
(*.....*)

PROCEDURE NAGLOWEK(VAR PLIK:TEXT);

(*                                                                    *)
(*  PROCEDURA WYPROWADZA  :  LOC , OBJ , LINENUMBER                   *)
(*                                                                    *)
VAR
  L,I,K:BYTE86;
  TAB:PACKED ARRAY[1..2] OF CHAR;
BEGIN
  L:=OBJ.NOBYTES;
  IF L <> 0 THEN
  BEGIN
  CHANGE(LOC[2],TAB);
  WRITE(PLIK,TAB[2],TAB[1]);
  CHANGE(LOC[1],TAB);
  WRITE(PLIK,TAB[2],TAB[1]);
```

```
      END
      ELSE SPACJA(4,PLIK);
      WRITE(PLIK,' ');
      FOR I:=1 TO L DO
      BEGIN
        K:=OBJ.CONTENTS[I];
        CHANGE(K,TAB);
        WRITE(PLIK,TAB[2],TAB[1])
      END;
      SPACJA(17-2*L,PLIK);
      WRITE(PLIK,LINENUMBER);
      SPACJA(5,PLIK)
    END (* NAGLOWEK *);
    (*.....*)
    BEGIN (* LIST *)
      IF LISTING THEN
      BEGIN
        IF LNONPAGE=0 THEN
        BEGIN
          TYTUL(LISTFILE,'L');
          LNONPAGE:=5;
        END
      ELSE IF LNONPAGE=PAGELENGTH THEN
           BEGIN
             PAGE(LISTFILE);
             PAGENUM:=PAGENUM+1;
             TYTUL(LISTFILE,'L');
             LNONPAGE:=5
           END;
      NAGLOWEK(LISTFILE);SZEROKOSC:=35;
      ILOSC:=MINIMUM(LINEWIDTH,PAGEWIDTH-SZEROKOSC);
      POM(LISTFILE,1,ILOSC,I);
      WRITELN(LISTFILE);LNONPAGE:=LNONPAGE+1
      END(* LISTING *);
      IF TERMINAL THEN
      BEGIN
        IF LINENUMBER=1 THEN TYTUL(TTY,'T');
        SZEROKOSC:=35;
        ILOSC:=MINIMUM(LINEWIDTH,37);
        NAGLOWEK(TTY);
        POM(TTY,1,ILOSC,I);
        WRITELN(TTY);
        IF ((NOT TRUNC) AND (ILOSC=37)) THEN
        BEGIN
          ILOSC:=LINEWIDTH-37+I;
          P:=39-I;
          REPEAT
            J:=MINIMUM(ILOSC,37);
            K:=P+J-1;
            WRITELN(TTY);
            SPACJA(35,TTY);
            POM(TTY,P,K,I);
            ILOSC:=ILOSC-J+I;
            P:=K-I+1
          UNTIL ILOSC=0;
          WRITELN(TTY)
        END (* TRUNC *)
      END(* TERMINAL *)
    END(* TRANSFER *);
    (*****************************************************************)
    (*****************************************************************)

    BEGIN  (* ASM86 *)
```

```
WRITELN (TTY, 'PIAP 8086 CROSSASSEMBLER '); WRITELN (TTY, ' ');

  (* USTAWIENIE WARUNKOW POCZATKOWYCH *)

  (*  USTAWIENIE SKOROWIDZA NAZW UZYTKOWNIKA *)
  FOR I := 1 TO MAXUSER DO
  FOR J := 1 TO 2 DO
  BEGIN
    USERIDENT [I].NAME [J] := 0;
    USERIDENT [I].OFFSET := 0
  END;

  (* USTAWIENIE DANYCH DLA LISTINGU *)
  LISTING := TRUE;<-> { write (tty,'listing ? [1/0] '); break; read(tty,i);
  IF LISTING THEN       if i=0 then listing := false else listing := true;
  REWRITE (LISTFILE, 'ASM86LIST.TXT');
  PAGEWIDTH := 132;
  LNONPAGE := 0;
  PAGELENGTH := 60;
  TRUNC := TRUE;
  TERMINAL := TRUE;-> { write (tty, 'terminal ? [1/0] '); break; read (tty,i);
  DATE (CURRDATE);      if i=0 then terminal := false else terminal := true;
  TIME (DAYTIME);
  DAYTIME [9] := ' ';
  DAYTIME [10] := ' ';
  PAGENUM := 1;

  RESET (FILL, 'MNEMONICS.TXT');
  RESET (COMCR, 'CODEMACRO.TXT');  write (tty, 'object ? [1/0] '); break; read (tty,i);
  REWRITE (OBJECT, 'OBJECT.DAT');  if i=0 then out := false else out := true;
  OUT := TRUE;                     if out then
  MNEMFILL;
        (* USTAWIENIE TABLIC OPISUJACYCH MNEMONIKI (MN, INS, ENTRY) *)
  FILLCODEN;
        (* USTAWIENIE TABLIC OPISUJACYCH CODEMACROSY *)
  LINE [LINEWIDTH + 1] := ' ';
  PROCLEN := 0;
  PASS := 1;
  REWRITE (RESERVE,'RESERVE.');
  FATAL := FALSE;
  WHILE (PASS <= 3) AND (NOT FATAL) DO
  BEGIN  (* WHILE PASS <= 3 *)
    WRITELN (TTY, 'PASS ', PASS:1);
    LC := 0;
    RESET (SOURCE, 'SRC');
    EOPROGRAM := FALSE;
    LINENUMBER := 1;
    IF PASS = 3 THEN RESET (RESERVE);
    WHILE NOT EOPROGRAM DO
    BEGIN  (* WHILE NOT EOPROGRAM *)
      OBJ.NOBYTES := 0;  LC1 := LC;
      LOC [1] := LC MOD 256;
      LOC [2] := LC DIV 256;
      READLINE;
        (* WCZYTANIE LINII PROGRAMU ZRODLOWEGO DO ZMIENNEJ LINE *)
      DOLINE;
        (* PRZETWORZENIE LINII PROGRAMU ZRODLOWEGO *)
      IF PASS = 3 THEN
      BEGIN  (* PASS = 3 *)
        OBJOUT;
                (* WYPROWADZENIE KOLEJNEJ CZESCI KODU WYNIKOWEGO
                   NA DYSK (ZMIENNA OBJECT: FILE OF INTEGER) *)
        LIST
```

```
        END;  (* PASS = 3 *)
        LINENUMBER := LINENUMBER + 1
      END  (* WHILE NOT EOPROGRAM *);
      PASS := PASS + 1
    END  (* WHILE PASS <= 3 *);
END  (* ASM86 *).
```

writeln (tty, 'ASSEMBLY COMPLETE')