Mikrokomputer K-202

Dokumentacja archiwalna polskiego minikomputera K-202

OPIS PODSTAWOWEGO JĘZYKA SYMBOLICZNEGO M.C. K-202

ASSK

Skanowanie i opracowanie:
Ryszard Zenker
www.zenker.com.pl/k-202
e-mail: k-202@tlen.pl
Puszczykowo - 2008

OŚRODEK DOSKONALENIA KADR TECHNICZNYCH OW/NOT
i
ZAKŁAD DOŚWIADCZALNY MINIKOMPUTERÓW
przy Instytucie Maszyn Matematycznych

(Opis wyłącznie do celów szkoleniowych)

OPIS PODSTAWOWEGO JĘZYKA SYMBOLICZNEGO M.C. K-202 ASSK

Warszawa, wrzesień 1972

S P I S T R E Ś C I

-SIMP-ZODOK- z. 507/72 n. 250+20 egz.

1. WSTĘP

Język programowania ASSK jest podstawowym językiem symbolicznym maszyny cyfrowej K-2Ø2. Elementami programu źródłowego w ASSK-u są:

- rozkazy z listy rzkazów K-2Ø2 z argumentami zapisanymi jawnie i symbolicznie,
- liczby i wyrażenia arytmetyczne,
- teksty,
- komentarze,
- dyrektywy.

Translator ASSK-u tłumaczy program źródłowy na język wewnętrzny maszyny, przy czym do pamięci operacyjnej /PAO/ K-2Ø2 wprowadzane są jedynie odpowiedniki trzech pierwszych /na powyższej liście/ elementów programu źródłowego, zwane w dalszym ciągu tego opisu elementami programu wynikowego. Wykrycie przez translator braku formalnej poprawności programu jest sygnalizowane przez podanie specyfikacji błędu na odpowiednim urządzeniu zewnętrznym. Pełna lista błędów zamieszczona jest w dodatku D.

2. DEFINICJE PODSTAWOWYCH SYMBOLI

Formalny opis składni języka źródłowego jest przedstawiony przy użyciu symboliki Backusa. Alfabetem języka jest podzbiór kodu ISO-7 /dodatek A/

<symbol podstawowy> ::= <litera> | <cyfra> |
<znaki specjalne> | <znaki sterujące> ;

<litera> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|
X|Y|Z;

<cyfra> ::= Ø|1|2|3|4|5|6|7|8|9;

<znaki specjalne> ::= !|"|'|(|)|[|]|<|>|*|+|-|,|
&|/|:|.|?;

<znaki sterujące> ::= SP | LF | CR;

gdzie:

SP - spacja
LF - nowa linia
CR - karetka

3. LICZBY

<liczba> ::= <liczba krótka> | <liczba zmiennoprzecinkowa> |
<parametr alfanumeryczny>;

<liczba krótka> ::= <liczba oktalna krótka> |
<liczba dziesiętna krótka>;

<liczba oktalna krótka> ::= Ø | Ø <cyfra oktalna> |
<liczba oktalna krótka> <cyfra oktalna>;

<cyfra oktalna> ::= Ø|1|2|3|4|5|6|7;

Translator wprowadza liczby z zakresu: .Ø ÷ Ø 177 777.

<liczba dziesiętna krótka> ::= <pierwsza cyfra dziesiętna> |
<liczba dziesiętna krótka> <cyfra> ;

<pierwsza cyfra dziesiętna> ::= 1|2|3|4|5|6|7|8|9;

Translator wprowadza liczby z zakresu: 1 ÷ 32 767

Przykłady: 13 1ØØ 29998 Ø Ø 123456 Ø 77

Uwaga: Liczba zmiennoprzecinkowa może być wprowadzona przy pomocy dyrektywy f ≡ lub odpowiedniego ekstrakodu. Operacje na liczbach zmiennoprzecinkowej i zmiennoprzecinkowej podwójnej precyzji wykonuje się przy pomocy ekstrakodów.

⟨parametr alfanumeryczny⟩ ::=
! ⟨znak w kodzie ISO-7⟩ ⟨znak w kodzie ISO-7⟩;

Parametr alfanumeryczny jest traktowany jak liczba i kompletowany w słowie w ten sposób, że znaki w kodzie ISO-7 są wprowadzane odpowiednio na starszy i młodszy byte, przy czym znak "!" jest wyróżniony i zamieniony na znak o kodzie Ø /puste/.

Przykłady: ! a l !! a ! Ø !

4. NAZWA

⟨nazwa⟩ ::= ⟨litera⟩ | # ⟨litera⟩ | % ⟨litera⟩ | @ ⟨litera⟩ |
⟨cyfra⟩ | % ⟨cyfra⟩ | @ ⟨cyfra⟩ |
⟨nazwa⟩ ⟨litera⟩ | ⟨nazwa⟩ ⟨cyfra⟩;

Translator identyfikuje jedynie sześć pierwszych znaków /nie licząc #, %, @ /.

Przykłady: a % la aleksandra adjt macro

5. WYRAŻENIA ARYTMETYCZNE

W translatorze ASSK dopuszcza się jedynie wyrażenia arytmetyczne typu dodawania.

⟨składnik⟩ ::= ⟨liczba krótka⟩ | ⟨nazwa⟩ |
⟨parametr alfanumeryczny⟩ | ⟨liczba krótka skalowana⟩ |
⟨nazwa skalowana⟩ | ⟨parametr alfanumeryczny skalowany⟩ ;

<liczba krótka skalowana> ::= <liczba krótka>/<numer skali>;
<nazwa skalowana> ::= <nazwa>/<numer skali> ;
<parametr alfanumeryczny skalowany> ::=
<parametr alfanumeryczny> / <numer skali>;
<numer skali> ::= Ø|1|2|3|4|5|6|7|8|9|1Ø|11|12|13|14|15|
<nazwa> ;

Uwaga: Nazwa jako numer skali musi być etykietą lub zmienną translacji zdefiniowaną o wartości z zakresu Ø ÷ 15. W przeciwnym wypadku potraktowane to będzie jako błąd.

Skalowanie polega na tym, że wartość składnika /traktowana jako informacja logiczna/ przesunięta jest w lewo tak, aby najmłodszy bit znalazł się na wskazanej pozycji. Warto zauważyć, że numer skali pokrywa się z oznaczeniem bitu w słowie maszyny.

Przykłady: 1Ø/5 k/ą
ala/2 ą/k
- 1/Ø 1/Ø

<operator typu dodawania> ::= + | - ;
<wyrażenie arytmetyczne> ::= <składnik>|
<operator typu dodawania> <składnik>|
<wyrażenie arytmetyczne><operator typu dodawania><składnik>;

Przykłady: +1Ø ala ala+2Ø rob 1-2Ø + al
1Ø/2 ala/14 a/13+2 rob/4 - ala/2
1ØØ + a/2 + rob - h - Ø13/7

Wyrażenie arytmetyczne, gdy nie zawiera nazw niezdefinio - wanych, czyli gdy znana jest jego wartość w chwili translacji, nazywane będzie dalej wyrażeniem określonym.

6. ETYKIETY

<etykieta> ::= <nazwa>;

<deklaracja etykiety> ::= <etykieta>:

Deklaracja etykiety polega na przypisaniu danej nazwie wartości równej bieżącemu adresowi miejsca pamięci operacyjnej, do którego zostanie wprowadzony przez translator kolejny element programu wynikowego. To samo miejsce pamięci może być oznaczone wieloma etykietami.

Etykieta "S" jest zastrzeżona, jej wartość jest zawsze zdefiniowana i równa bieżącemu adresowi pamięci. Wartość tej etykiety jest zwiększana po translacji kolejnego elementu programu o jego długość.

7. ZMIENNA TRANSLACJI

<zmienna translacji> ::= <nazwa>;

<deklaracja zmiennej translacji> ::=
<zmienna translacji> = <wyrażenie określone>;

Deklaracja zmiennej translacji polega na przypisaniu danej nazwie wartości równej wyrażeniu określonemu.

Przykłady:
a = 10.
b = a + 15 + 11 b.
c : d = s + a.

8. ROZKAZY

Rozkaz składa się z kodu operacji i części opisujących argumenty operacji. Rozkaz może posiadać dwa lub tylko jeden argument operacji.

<numer rejestru> ::= 1|2|3|4|5|6|7 <etykieta>|
<zmienna translacji>;

Uwaga: Etykieta i zmienna translacji muszą mieć wartość z przedziału 1 ÷ 7. W przeciwnym przypadku zostanie to wykryte przez translator jako błąd i zasygnalizowane.

<argument 1> ::= |Ø|<numer rejestru> ;
<argument 2> ::= <wyrażenie arytmetyczne> ;
<parametr krótki> ::= <wyrażenie arytmetyczne> ;

Uwaga: Wartość wyrażenia arytmetycznego musi być z przedziału - 63, + 63. Składniki wyrażenia arytmetycznego nie mogą być skalowane. Niespełnienie tych warunków jest sygnalizowane jako błąd.

<kod operacji> ::= <nazwa> ;

Lista kodów operacji rozkazów m.c. K-2Ø2 z podziałem funkcjonalnym przedstawiona jest w dodatku B.

W zależności od swojej funkcji i sposobu opisu argumentów rozkaz zajmuje od jednej do czterech komórek pamięci.

<B-modyfikacja > ::= & <numer rejestru> ;
<D-modyfikacja> ::= ' ;

Uwaga: Możliwy jest trzeci rodzaj modyfikacji argumentu 2 zwany pre-modyfikacją. Modyfikacji tego rodzaju podlega każdy rozkaz posiadający argument 2 i parametr krótki poprzedzony rozkazem o kodzie operacji "mod". Jest to jedyna możliwość modyfikacji rozkazów z krótkim parametrem.

<warunek> ::= Q|L|E|G|V|C|Y|I|X|1|2|3|4|5|6|7|

Q <warunek> | L warunek | E warunek |
G <warunek> | V warunek | C warunek |
Y <warunek> | I warunek | X warunek |
1 <warunek> | 2 warunek | 3 warunek |
4 <warunek> | 5 warunek | 6 warunek |
7 <warunek> ;

Uwaga: Każdy rozkaz może być warunkowy tzn. wykonany, gdy aktualny stan warunków w rejestrze RØ jest zgodny z maską warunków podaną w rozkazie i jest pomijany w przeciwnym przypadku.

8.1. Kodowanie rozkazów

Ze względu na różnorodną budowę, rozkazy można podzielić na grupy:

```
<rozkaz> ::= <rozkaz dwuargumentowy>|
             <rozkaz tylko z argumentem 1>|
             <rozkaz tylko z argumentem 2>|
             <rozkaz z parametrem krótkim>|
             <rozkaz przesyłania grupowego> ;
```

8.1.1. Rozkazy dwuargumentowe

```
<rozkaz dwuargumentowy> ::= <rozkaz dwuargumentowy krótki>.
|<rozkaz dwuargumentowy krótki z warunkiem>.
|<rozkaz dwuargumentowy długi>)
|<rozkaz dwuargumentowy długi z warunkiem>);
<rozkaz dwuargumentowy krótki> ::=<kod operacji>,<argument 1>,
<numer rejestru>|

<kod operacji> <argument 1>,<numer rejestru> <B-modyfikacja>|
<kod operacji>,<argument 1>,<numer rejestru> <D-modyfikacja>|

<kod operacji>,<argument 1>,<numer rejestru> <B-modyfikacja>
<D-modyfikacja>;

<rozkaz dwuargumentowy krótki z warunkiem> ::=
  <rozkaz dwuargumentowy krótki> ? <warunek>
<rozkaz dwuargumentowy długi>::=
  <kod operacji> ,(<argument 1>(<argument 2>|
  <kod operacji><argument 1>(<argument 2>    <B-modyfikacja>|
  <kod operacji><argument 1>(<argument 2>    <D-modyfikacja>|
  <D-modyfikacja> ;

<rozkaz dwuargumentowy długi z warunkiem> ::=
  <rozkaz dwuargumentowy długi > ? <warunek> ;
```

Do grupy tych rozkazów należą rozkazy o kodach operacji: ad, adc, su, co, and, or, orn, clmo, clbo, lo, lom, los, lob, st, stb, par.

```
Przykłady:  ad, 1, 2.          ad, 1, 2 § ala' .
            su, a, 3.          st, ala (a + b & c ' ? x 12)
            co, alfa, b.
            or, 6 (Ø1Ø/12 + zeta ')
```

8.1.2. Rozkazy posiadające tylko argument 1

<rozkaz tylko z argumentem 1>::= <kod operacji><argument 1>|.
<kod operacji>,<argument 1> ? <warunek>.;

Rozkazy tej grupy mają kody operacji: stop, neg, neo, nega, chan, lobi, vkey, shl, shv, shly, shvy, shlx, shvx, shr, shry, shrx, stxa, stxs.

Przykłady: nega, 3.
shv, vej ? X 267.

8.1.3. Rozkazy bez argumentu 1

<rozkaz bez argumentu 1> ::=<rozkaz bez argumentu 1 krótki>.|
<rozkaz bez argumentu 1 krótki z warunkiem> .|
<rozkaz bez argumentu 1 długi>)|
<rozkaz bez argumentu 1 długi z warunkiem >);

<rozkaz bez argumentu 1 krótki>::=
<kod operacji> , <numer rejestru>|
<kod operacji>, <numer rejestru> <B-modyfikacja>|
<kod operacji>, <numer rejestru> <D-modyfikacja>|
<kod operacji>, <numer rejestru> <B-modyfikacja>
<D-modyfikacja>;|

<rozkaz bez argumentu 1 krótki z warunkiem> ::=
<rozkaz bez argumentu 1 krótki> ? <warunek> ;

<rozkaz bez argumentu 1 długi> ::=<kod operacji>(<argument 2>|
<kod operacji>(<argument 2> <B-modyfikacja>|
<kod operacji>(<argument 2> <D-modyfikacja>|
<kod operacji>(<argument 2> <B-modyfikacja><D-modyfikacja>

<rozkaz bez argumentu 1 długi z warunkiem> ::=
<rozkaz bez argumentu 1 długi > ? <warunek>

Rozkazy tej grupy mają kody operacji: jp, jpl, jpe, jpg, jpr, jprl, jpre, jprg, lbar, sbar, mod, ex, veex, ados, zs.

Przykłady: jp, a jpg, a & b'.
jp, a ? el. jp, ala'.
zs (ala + eua / 2 & G') mod (Ø2Ø ? e)
ados (Ø 1ØØ + X')

8.1.4. Rozkazy z parametrem krótkim

<rozkaz z parametrem krótkim> ::=
<rozkaz z parametrem krótkim z argumentem 1>|
<rozkaz z parametrem krótkim bez argumentu 1> ;

<rozkaz z parametrem krótkim z argumentem 1> ::=
<kod operacji>, < argument 1>, <parametr krótki> .|
<kod operacji>,<argument 1>,<parametr krótki> ? <warunek>;

<rozkaz z parametrem krótkim bez argumentu 1> ::=
<kod operacji> , <parametr krótki>,|
<kod operacji> , <parametr krótki> ? <warunek> .;

Do grupy tych rozkazów należą rozkazy o kodach operacji:
adt, adot, cot, lot, adjt, lts, sts, oraz
jpt, jptl, jpte, jptg, jptv, jptx, jpty, jpti.

Uwaga: Jeżeli parametr krótki zawiera jedną lub więcej etykiet niezdefiniowanych to suma składników znanych w czasie translacji musi być z przedziału -63, +63. W przeciwnym przypadku zostanie zasygnalizowany błąd, nawet jeżeli ogólna wartość wyrażenia mieści się w tym przedziale.
Parametr krótki w rozkazach tej grupy /oprócz pierwszych czterech rozkazów/ jest względnym argumentem, względem bieżącego wskaźnika pamięci operacyjnej. Powstaje on na etapie translacji w wyniku odjęcia od wartości etykiety lub zmiennej translacji w parametrze krótkim /o ile ona tam występuje/ wskaźnika bieżącego adresu pamięci /wartości etykiety S/.

Przykłady:

```
jpt, w + 3.
jpte, - 2Ø.
sts, 1, 1 + rob - 1Ø.
```

8.1.5. Rozkazy przesyłania grupowego

<rozkaz przesyłania grupowego>::=
<rozkaz przesyłania grupowego krótki>.|
<rozkaz przesyłania grupowego krótki z warunkiem>.|
<rozkaz przesyłania grupowego długi>)|
<rozkaz przesyłania grupowego długi z warunkiem>);

<rozkaz przesyłania grupowego krótki>::=
<kod operacji> , <przedłużenie kodu operacji> , <numer rejestru>|
<kod operacji> , <przedłużenie kodu operacji> , <numer rejestru> <B-modyfikacja> ;

<rozkaz przesyłania grupowego krótki z warunkiem> ::=
<rozkaz przesyłania grupowego krótki> ? <warunek>;

<rozkaz przesyłania grupowego długi> ::=
<kod operacji> , <przedłużenie kodu operacji>(<argument 2>|
<kod operacji> , <przedłużenie kodu operacji>(<argument 2>
<B-modyfikacja>;

<rozkaz przesyłania grupowego długi z warunkiem> ::=
<rozkaz przesyłania grupowego długi> ? <warunek>;

gdzie:<przedłużenie kodu operacji> ::= Ø|1|2|3|4|5|6|7;

Do grupy tej należą rozkazy o kodach operacji: log, stg.

Przykłady:

```
log, 4, 3
log, 5  (beta +3 & a)
stg, 7  (x ? 127)
```

9. DYREKTYWY

<dyrektywa> ::= <kod dyrektywy> *
|<dyrektywa> <wyrażenie określone> .;

<kod dyrektywy> ::= prog|finprog|seg|finseg|macro|finmacro|
os|all|res|s|f|lab|nlab|int|out|e|t|trac|
ntrac;

Uwaga: Jeżeli kod dyrektywy zawiera więcej niż 6 liter, podawanie liter siódmej i następnych jest zbędne.W stosunku do kodów dyrektyw przyjęta jest bowiem zasada identyfikacji na podstawie pierwszych sześciu znaków.

Użycie dyrektywy w programie źródłowym powoduje wywołanie określonej procedury translatora. W poszczególnych wersjach translatora niektóre kody dyrektyw nie są rozpoznawane i jest sygnalizowany odpowiedni błąd. Zastosowanie i znaczenie poszczególnych dyrektyw omówione zostało w dodatku C.

10. KOMENTARZE I TEKSTY

<komentarz> ::= <komentarz zwykły>|<komentarz dynamiczny>;
<komentarz zwykły>::=
[<dowolny ciąg znaków kodu ISO-7 oprócz "]">];

Cały tekst zamknięty w nawiasach [ ]jest przez translator ignorowany, stanowi jedynie pomocniczy środek dla programisty.

<komentarz dynamiczny> ::=<<dowolny ciąg znaków kodu ISO-7 oprócz znaku ostrego nawiasu zamykającego>>

Tekst zamknięty nawiasami< >jest wyprowadzony przez urządzenie wyjściowe. Komentarz dynamiczny może być użyty między innymi do informowania operatora poprzez odpowiednie wypisy tekstów o etapie translacji programu.

<Tekst alfanumeryczny> ::= " <dowolny ciąg znaków kodu ISO-7 oprócz oudzysłowu ">

Translator umożliwia wprowadzenie do pamięci dowolnego tekstu alfanumerycznego. Tekst umieszczony jest po dwa znaki w słowie w kolejnych miejscach pamięci. Zakończony jest jednym lub dwoma znakami o kodzie 128 w zależności od parzystości liczby znaków w tekście.

Przykłady:

```
[Pętla - licznik pętli]
<Translacja drugiej części programu>
"Tekst alfanumeryczny"
```

11. STRUKTURA PROGRAMU

Program źródłowy ma strukturę blokową. Blokiem obejmującym całość jest blok ograniczony dyrektywami prog * i finprog *. Blok ten może zawierać bloki typu segment i macro /ograniczone odpowiednio dyrektywami seg * finseg * i macro * finmacro */. Blok typu segment może zawierać również bloki typu macro. Z chwilą wykrycia dyrektywy prog * translator powołuje dynamiczny słownik etykiet i zmiennych translacji - słownik translacji /ST/. Słownik ST jest sukcesywnie zwiększany w miarę pojawiania się w programie kolejnych deklaracji etykiet i zmiennych translacji. Poprzedzenie tych deklaracji dyrektywą all * powoduje, że odpowiednia etykieta lub zmienna translacji traktowana jest jako globalna. Odwołania w programie są natychmiast realizowane w chwili wykrycia przez translator o ile w słowniku ST występuje potrzebna do realizacji nazwa. Po wystąpieniu w programie dyrektywy finmacro * lub finseg * realizowane są na podstawie całej aktualnej zawartości słownika odwołania występujące w zamykanym bloku a dotychczas nie zrealizowane. Po zwinięciu danego bloku ze słownika ST zostają usunięte oprócz globalnych:

- etykiety deklarowane w tym bloku,
- zmienne translacji, które były deklarowane po raz pierwszy w zamykanym bloku /tzn. nie istniały w słowniku w momencie powoływania tego bloku dyrektywą macro * lub seg */.

Usuwane ze słownika ST etykiety i zmienne translacji nazywamy lokalnymi w danym bloku.

Wykrycie w programie deklaracji etykiety powoduje umieszczenie etykiety i jej wartości w słowniku ST, ale tylko wtedy gdy nazwa etykiety nie występuje w słowniku. W przeciwnym wypadku sygnalizowany jest błąd.

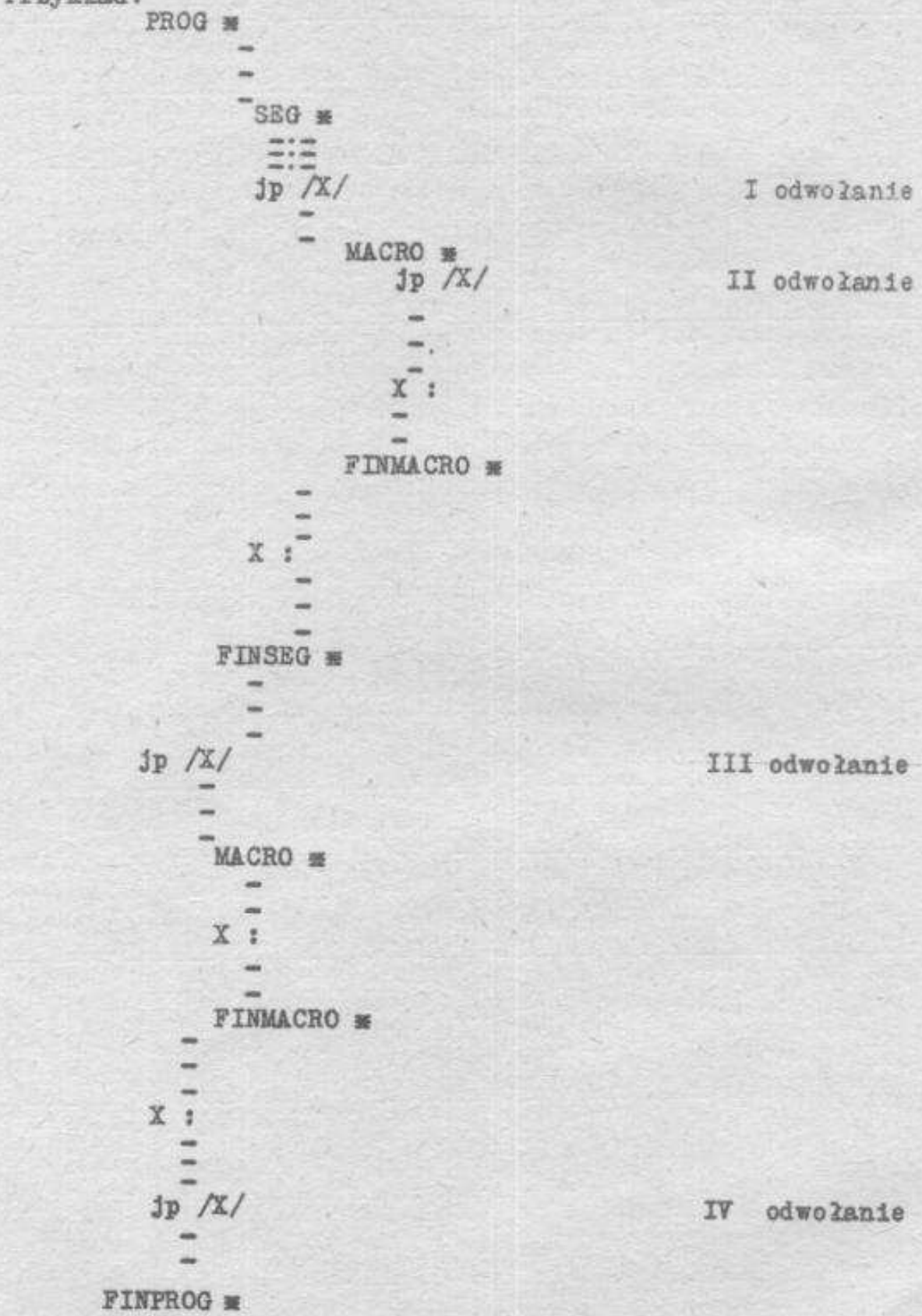
Wykrycie w programie deklaracji zmiennej translacji powoduje umieszczenie zmiennej translacji i jej wartości w słowniku. Jeżeli nazwa zmiennej istnieje w słowniku to:

1/ gdy jest to nazwa globalnej zmiennej translacji lub etykieta - zostaje zasygnalizowany błąd,

2/ gdy jest to nazwa zmiennej translacji, która była deklarowana po raz pierwszy w tym samym lub zewnętrznym bloku /tzn. pochodzi z tego samego lub wyższego poziomu/ - zmiennej translacji zostanie nadana nowa wartość w słowniku etykiet.

Dyrektywa finprog ж, informująca translator o końcu programu, powoduje jedynie realizację odwołań i wypis odwołań niezrealizowanych na urządzenie zewnętrzne, natomiast stan słownika ST nie ulega żadnym zmianom. Można wobec tego zadeklarować nazwy, do których są niezrealizowane odwołania i ponownie użyć dyrektywy finprog ж.

Przykład:


PROG *
SEG *
jp /X/
I odwołanie
MACRO *
jp /X/
II odwołanie
X :
FINMACRO *
X :
FINSEG *
jp /X/
III odwołanie
MACRO *
X :
FINMACRO *
X :
jp /X/
IV odwołanie
FINPROG *

W powyższym przykładzie etykiety X są lokalne i dlatego nie zachodzi kolizja. Gdyby jednak etykieta z poziomu prog była zdefiniowana na początku /przed którymś z bloków/ to byłaby ona w słowniku ST i kolidowała z nazwą z tego bloku -byłby zasygnalizowany błąd. Podobnie uglobalnienie etykiety X w jednym z bloków powodowałoby zlikwidowanie lokalności. Odwołanie I zrealizowane zostanie na poziomie seg. Odwołanie II na poziomie macro. Odwołanie III i IV na poziomie prog.

Przy translacji każdego programu /ujętego dyrektywami prog # i finprog #/ należy wstępnie ustalić szereg warunków:

1/ ustalić adres początkowy wprowadzenia programu do pamięci operacyjnej dyrektywą S #,

2/ zadeklarować urządzenie zewnętrzne - out #,

3/ określić sposób translacji programu:

bez wydruku listy etykiet i zm.translacji - nlab #

z listą etykiet - lab #

4/ zadeklarować urządzenie wejściowej - int #, z tą chwilą wystąpi start do translacji.

Standardowo ustalone są 2 i 3 warunek:

wyjściowe urządzenie - teletype

translacja bez wydruku listy etykiet i zmiennych translacji.

Zmiana ich wymaga użycia odpowiedniej dyrektywy.

Przykładowy program zliczania liczby jedynek w rejestrach R1, R2, R3. Wynik w R5.

```
 S = Ø 1ØØØ.
OUT = 1.
NLAB =
 INT = 1.
PROG =
```

< PROGRAM ZLICZANIA JEDYNEK >

```
A = 1.
B = 2.
C = 3.
D = 5.
E = 6.
M = 7.

OBLICZANIE: LOT, E, Ø.   [ZEROWANIE MODYFIKATORA]
            LO, M (Ø377) [UMIESZCZENIE MASKI]
            LO, 4   (TABLICA)
            LOM, E, A.
            LO, D, 4 & E'. [POBRANIE LICZBY JEDYNEK]
            CHAN, 1.
            LOM, E,A.
            AD, D, 4 & E'.
            LOM, E, B.
            AD, D, 4 & E'.
            CHAN, 2.
            LOM, E,A.
            AD, D, 4 & 6'.
            LOM, E, C.
            AD, D, 4 & E'.
            CHAN, 3.
            LOM, E, C.
            AD, D, 4 & E'.
```

```
EX, 4. ESYS.          [POWRÓT DO SYSTEMU]

TABLICA: Ø.           [ LICZBA JEDYNEK]
         1.           [1]
         1.           [2]
         2.           [3]
         1.           [4]
         2.           [5]

         ---------------------

         7.           [ 254 ]
         8.           [ 255 ]

FINPROG =

INT = 3.
```

Uwaga: Ze względu na brak miejsca nie wypisano w miejscu za - znaczonym kropkami wszystkich parametrów.

DODATEK A

KOD ISO - 7

HT	0001001	9
LF	0001010	10
FF	0001100	12
CR	.0001101	13
SP	.0100000	32
!	0100001	33
"	0100010	34
#	.0100011	35
$	0100100	36
%	.0100101	37
&	.0100110	38
'	0100111	39
(	0101000	40
)	.0101001	41
*	.0101010	42
+	0101011	43
,	.0101100	44
-	0101101	45
.	0101110	46
/	.0101111	47
0	0110000	48
1	.0110001	49
2	.0110010	50
3	0110011	51
4	.0110100	52
5	0110101	53
6	0110110	54
7	.0110111	55
8	.0111000	56
9	0111001	57
:	0111010	58
;	.0111011	59
<	0111100	60
=	.0111101	61
>	.0111110	62
?	0111111	63
@	.1000000	64
A	1000001	65
B	1000010	66
C	.1000011	67
D	1000100	68
E	.1000101	69
F	.1000110	70
G	1000111	71
H	1001000	72
I	.1001001	73
J	.1001010	74
K	1001011	75
L	.1001100	76
M	1001101	77
N	1001110	78
O	.1001111	79
P	1010000	80
Q	.1010001	81
R	.1010010	82
S	1010011	83
T	.1010100	84
U	1010101	85
V	1010110	86
W	.1010111	87
X	.1011000	88
Y	1011001	89
Z	1011010	90
[	.1011011	91
\	1011100	92
]	.1011101	93
↑	.1011110	94
←	1011111	95

DODATEK B

Mnemotechniczne kody rozkazów

Grupy:	8.1.1.	8.1.2.	8.1.3.	8.1.4	8.1.5
	Rozkazy umieszczania i pamiętania				
	LO	LOBI	LBAR	LOT	LOG
	LOM	RIC	SBAR	LTS	STG
	LOS			STS	
	LOB				
	ST				
	STB				
	Rozkazy arytmetyczne				
	AD	NEGA	ADOS		ADT
	ADC		ZS		ADOT
	SU				ADJT
	CO				COT
	Rozkazy logiczne				
	AND		NEG		
	OR		NEC		
	ORN				
	CLMO				
	CLBO				

Rozkazy przesuwania

SHL
SHV
SHLY
SHVY
SHLX
SHVX
SHR
SHRY
SHRX

Rozkazy skoków

JPAR

JP	JPT
JPL	JPTL
JPE	JPTE
JPG	JPTG
JPR	JPTV
JPRL	JPTX
JPRE	JPTY
JPRG	JPTI

Rozkazy operacji na bicie

STXA
STXZ

Rozkazy operacji na byte

CHAN

Rozkazy wejścia/wyjścia

PERO RKEY MESS
PERI
MERO
MERI

Rozkazy specjalne

STOP MOD
EX
REEX
PUF

DODATEK C

SPIS DYREKTYW

1. Dyrektywy ustalające strukturę blokową programu źródłowego

Zapis:

prog ※	finprog ※
seg ※	finseg ※
macro ※	finmacro ※

Znaczenie: Dyrektywy powyższe zawierają informację sterującą dla translatora o rozpoczęciu /prog ※, seg ※, macro ※/ lub zakończeniu odpowiedniego bloku. Dyrektywy kończące /finprog ※, finseg ※, finmacro ※/ powodują zwinięcie bloku, tzn. realizację odwołań i wymazanie ze słownika etykiet lokalnych w zamykanym bloku. Należy przy tym pamiętać, że etykiety i zmienne translacji z poziomu prog nie są wymazywane ze słownika ST i pozostają dostępne w translatorze podobnie jak globalne etykiety i zmienne translacji. Dopiero dyrektywa prog ※ powoduje, że w słowniku ST pozostają jedynie nazwy zastrzeżone /tj. nazwy ekstrakodów i etykieta S/.

2. Wróć do Systemu Operacyjnego

Zapis: OS ※

Znaczenie: Dyrektywa powoduje przejście z translatora ASSK-u do Systemu Operacyjnego, co jest sygnalizowane odpowiednim tekstem na monitorze.

3. Uglobalnij

Zapis: all ※

Znaczenie: Najbliższa występująca po dyrektywie all ※ etykieta bądź zmienna translacji będzie traktowana jako globalna /a więc dostępna z każdego poziomu/.

4. Rezerwuj

Zapis: res ※ <wyrażenie określone> .

Znaczenie: Wskaźnikowi adresu bieżącego PAO zostaje nadana nowa wartość, przy czym S = S + wyrażenie określone, podane w dyrektywie. Zarezerwowany w ten sposób obszar PAO zostaje wyzerowany.

Przykład: res = 1Ø.

5. Ustaw wskaźnik S

Zapis: S = <wyrażenie określone> .

Znaczenie: Dyrektywa nadaje wskaźnikowi adresu bieżącego PAO wartość równą wartości wyrażenia określonego.

6. Drukuj etykiety i zmienne translacji

Zapis: lab = <wyrażenie określone> .

Znaczenie: Poczynając od momentu pojawienia się tej dyrektywy na urządzeniu wyjściowym określonym dyrektywą out = będzie wyprowadzana lista etykiet i zmiennych translacji w n kolumnach, gdzie n ma wartość równą wyrażeniu określonemu /dozwolony zakres $1 \leq n \leq 5$/.

7. Anuluj druk etykiet i zmiennych translacji

Zapis: nlab =

Znaczenie: Dyrektywa anuluje użytą uprzednio dyrektywę lab =; lista etykiet i zmiennych translacji nie będzie wyprowadzana od momentu wystąpienia nlab =.

8. Określ urządzenie wejściowe translatora

Zapis: int =<wyrażenie określone> .

Znaczenie: Dyrektywa określa numer urządzenia wejściowego, z którego będą wprowadzane informacje do translatora.

9. Określ urządzenia wyjściowe translatora

Zapis: out = <wyrażenie określone> .

Znaczenie: Dyrektywa określa numer symbolicznego urządzenia wyjściowego, na które translator wyprowadza informacje o przebiegu translacji.

10. Rozszerz słownik translacji

Zapis: e ⁎ <wyrażenie określone> .

Znaczenie: Dyrektywa powoduje rozszerzenie słownika ST, które pozwala na zapisanie dodatkowo w słowniku ST takiej ilości etykiet bądź zmiennych translacji, jaka podana jest w wyrażeniu określonym. Na urządzeniu wyjściowym translatora wypisany zostaje nowy adres początkowy ciała translatora i sygnalizowany jest powrót do ASSK-u.

11. Czytaj liczby zmiennoprzecinkowe

Zapis: f ⁎ <wyrażenie określone>

Znaczenie: Dyrektywa powoduje wczytanie takiej ilości liczb zmiennoprzecinkowych, jaka podana jest w wyrażeniu określonym. Liczby powinny następować bezpośrednio po powyższej dyrektywie. Są one wczytywane przez ekstrakod "CZYTAJ LICZBĘ ZMIENNOPRZECINKOWĄ". Zapis liczby zmiennoprzecinkowej podany jest w opisie ekstrakodu.

12. Ustaw ekstrakod śledzenia

Zapis: t ⁎

Znaczenie: Dyrektywa ustala punkt kontrolny w programie; gdy translacja programu odbywa się po użyciu dyrektywy trac ⁎ w miejsce dyrektywy t ⁎ jest każdorazowo zapisywany do PAO ekstrakod, który w trakcie wykonywania programu powoduje wydruk bieżących zawartości rejestrów. Należy zauważyć, że dyrektywa t ⁎ zmienia długość programu.

13. Ignoruj zlecenie tracingu

Zapis: ntrac ⁎

Znaczenie: Występujące po tej dyrektywie dyrektywy t ⁎ są traktowane przez translator jak komentarze,

czyli ignorowane. Użycie t * nie pociąga zatem za sobą żadnych następstw.

14. Akceptuj zlecenie tracingu

Zapis: trac *

Znaczenie: Podane przy opisie dyrektywy t *.

Uwaga: Każdorazowo po wywołaniu ASSK-u /co jest sygnalizowane odpowiednim wydrukiem/ translator działa tak, jakby użytkownik użył już standardowego zestawu dyrektyw: int * 3, out * 3,nlab *, ntrac *.

DODATEK D

LISTA BŁĘDÓW ASSK-3

01 - Znak niedozwolony
02 - Znak specjalny w niedozwolonym miejscu
03 - Znak , (; ' /bez nazwy
04 - Błędny znak w liczbie
05 - Błędny znak w warunku
06 - Znak @ lub # w złym miejscu
07 -
010 - Złe INT OUT
011 - Etykieta niezdefiniowana w wyrazie określonym
012 - Zła składnia w adresie
013 - Zła część operacyjna
014 - Zła składnia rozkazu
015 - Zły parametr na początku
016 - Źle zak. dyrektywa
017 - Zła definicja =
020 - Powtórzona definicja
021 - Nieznana dyrektywa
022 - Niedozwolona dyrektywa
023 - Res 0 lub ujemne
024 - Błąd w ilości 1 ¤ lub f ¤
025 -
026 -
027 - Błąd w INT OUT

030 - Nadmiar liczby ,
031 - Zły numer rejestru
032 - T > 63
033 - Nadmiar adresu
034 -
035 -
036 -
037 -
040 - Niezdefiniowana skala
041 - Zły numer skali