1. Wstęp.

Język ASSK3 jest podstawowym językiem symbolicznym dla maszyny cyfrowej K-202.

Program wprowadzany przez translator ASSK3 może zawierać:

- rozkazy z listy rozkazów K-202 z argumentami zapisanymi symbolicznie
- stałe
- teksty
- komentarze
- dyrektywy

Translator sprawdza poprawność formalną wprowadzanego programu oraz dopuszczalny zakres stałych liczbowych. W przypadku wykrycia błędu zostaje na monitorze wypisana odpowiednia informacja.

Alfabetem translatora jest podzbiór kodu ISO-7 zawierający:

- cyfry 0 - 9
- litery A - Z
- znaki specjalne ! " ` ( ) [ ] * + , - . / : ; > = < ?
  oraz znaki sterujące spacja i nowa linia.

2. DEFINICJE

2.1. Liczby.

Translator ASSK-3 pozwala na wprowadzanie następujących rodzajów liczb:

1/ dziesiętna krótka,

2/ oktalna krótka,

3/ dziesiętna długa,

4/ dziesiętna zmiennoprzecinkowa,

5/ dziesiętna zp. podwójnej-precyzji

2.1.1. Liczba dziesiętna krótka jest ciągiem cyfr 0-9 nie rozpoczynającym się cyfrą 0. Zakres liczb (1-32767).

Przykłady: 13 108 29908.

2.1.2. Liczba oktalna krótka jest ciągiem cyfr 0-7 rozpoczynającym się cyfrą 0.

Zakres liczb (0-0177777).

Przykłady: 0 0100000 0525.

2.1.3. Wprowadzenie pozostałych typów liczb odbywa się za pomocą ekstrakodów wywoływanych przez translator i omówione będzie w dalszej, części opracowania /pkt.2.10.4/

2.2. Parametry alfanumeryczne.

Parametrem jest ciąg dwóch znaków wejściowych poprzedzony znakiem "!"

Znaki te wprowadzane są w kodzie ISO-7 odpowiednio na pozycje 0-7 i 8-15.

W ciągu tym wyjątkowo jest traktowany znak "!" który zamieniany jest na znak o wartości 0 /blank/.
Parametr alfanumeryczny jest przez translator traktowany jak liczba.
Przykłady: !a1 !!a
!x! !z0
W dalszej części opracowania pod pojęciem liczba rozumieć będziemy liczbę dziesiętną krótką, oktalną krótką lub parametr alfanumeryczny.

2.3. Nazwy.

Nazwą jest każdy ciąg liter i cyfr rozpoczynający się literą. Translator identyfikuje jedynie sześć pierwszych znaków /na symulatorze - cztery/.
Przykłady: a, z006, x1b3, macro.

2.4. Dyrektywy.

Dyrektywą jest "nazwa" zakończona znakiem "*".
W języku ASSK-3 zdefiniowane są następujące dyrektywy:

1/ prog*
2/ finprog*
3/ seg*
4/ finseg*
5/ macro*
6/ finmacro*
7/ all*
8/ res*
9/ s*
10/ g*

11/ l *

12/ f *

13/ d *

14/ int *

15/ out *

16/ o *

Zastosowanie i znaczenie dyrektyw , omówione będzie w dalszej części.

2.5. Etykiety.

Każda nazwa może być etykietą o ile w programie zostanie jej nadana wartość. Etykietę będziemy nazywali niezdefiniowaną do momentu pierwszego nadania jej wartości. Zdefiniowanie wartości etykiety nastąpić może dwoma sposobami:

1/ przez deklarację,

2/ przez oznaczenie miejsca pamięci.

2.5.1. Deklaracja ma postać:

etykieta = wyrażenie definiujące.

gdzie wyrażenie definiujące jest dowolną sumą algebraiczną liczb oraz etykiet zdefiniowanych. Deklaracja wykonywana moze być wiele razy.

Przykłady: a12=3. q=a12+6. b=!!b. q=q+6.

2.5.2. Miejsce pamięci oznacza się przez postawienie po etykiecie znaku ":" .

Etykiecie tej zostanie nadana wartość równa adresowi miejsca pamięci operacyjnej do którego wprowadzona będzie przez translator najbliższa informacja. Jedno miejsce pamięci oznaczone może być

wieloma etykietami. Etykieta oznaczająca miejsce pamięci nie może być definiowana w innym miejscu /ani przez ":" ani przez "="/

2.6. Wskaźnik adresu bieżącego.

Etykieta "S" jest zastrzeżona, jej wartość jest zawsze zdefiniowana i równa jest adresowi miejsca pamięci, w którym zostanie umieszczony przez translator najbliższy element tłumaczonego programu. Wartość tej etykiety zwiększa się więc po translacji każdego rozkazu o jego długość.

2.7. Skala.

Dowolny składnik wyrażenia arytmetycznego może być skalowany tzn. jego wartość /z uwzględnieniem znaku/ przesunięta w lewo tak, aby najmłodszy bit znalazł się na wskazanej pozycji. Jest to równoważne pomnożeniu tego składnika przez potęgę liczby 2, której wykładnik jest liczony wg. wzoru:

$$w=15-k.$$

gdzie k- nr skali.

Numer skali jest liczbą lub etykietą zdefiniowaną bez znaku. Czynnik skalowany zaznaczamy pisząc za nim znak "/" a następnie nr. skali.

Uwaga: skalowane mogą być również etykiety niezdefiniowane.

Pjzykłady : -1/0 a16/10 !!m/12.

2.8. Struktura programu.

Program ma strukturę blokową . Blokiem obejmującym całość jest blok ograniczony dyrektywami "prog*" i

"finprog✱". Blok ten może zawierać bloki typu segment i macro /ograniczone odpowiednio nawiasami "seg✱" , "finseg✱" i "macro✱", "finmacro✱"/. Blok typu segment zawierać może również bloki typu macro.

2.8.1. Etykiety lokalne i globalne.

Wszystkie etykiety zdefiniowane w dowolnym bloku są w tym bloku lokalne, tzn. zostają wymazane ze słownika etykiet w momencie zamykania bloku. Wyjątkiem są etykiety globalne oraz etykiety na poziomie "program", które nie są likwidowane. Bezpośrednio przed definicją etykiety globalnej musi stać dyrektywa "all✱". Etykieta globalna może być definiowana jednokrotnie /nawet poprzez "="/. W czasie translacji dostępne są wszystkie etykiety znajdujące się w słowniku, niezależne od poziomu zdefiniowania.

Przykłady:

```
prog✱
a:----
  ----
macro✱
d:----
  jp a
  jp d
  jp b
  ----
  finmacro✱
  ----
```

```
    b:jp(c)
      ----
      seg *
      ----
  all*c:----
      ----
    d:----
      ----
      jp(d)
      ----
      finseg *
      ----
      ----
    finprog *
```

2.9. Wyrażenia arytmetyczne .

Wyrażeniem arytmetycznym nazywamy dowolną sumę algebraiczną liczb i etykiet. Każdy ze składników wyrażenia może być ponadto skalowany. W przypadku gdy wyrażenie nie zawiera etykiet niezdefiniowanych nazywamy je "określonym".

2.10. Dyrektywy o postaci

nazwa * wyrażenie określone.

2.10.1. Zmiana adresu s*.

Dyrektywa ta powoduje ustawienie wskaźnika adresu bieżącego pamięci operacyjnej na zadaną wartość.

2.10.2.Rezerwacja res*.

Dyrektywa ta służy do rezerwacji obszaru pamięci operacyjnej, którego długość wskazana jest przez wyrażenie określone. Obszar ten ulega również wyzerowaniu.

2.10.3.Startowanie -g*.

Wyrażenie określone w tej dyrektywie wskazuje miejsce pamięci od którego należy rozpocząć wykonywanie programu. Zawartość rejestrów jest przypadkowa.

2.10.4.Wprowadzanie liczb wielosłowowych- l*,f*,d*.

Deklaracje te służą do wprowadzania liczb długich, zmiennoprzecinkowych i z.p. podwojnej precyzji. Wyrażenie określone wskazuje ilość wczytywanych liczb , które winny następować bezpośrednio za deklaracją.

2.10.5.Deklaracja urządzeń we-wy int*, out*.

Wyrażenie określone w tych deklaracjach wskazuje urządzenie wejściowe lub wyjściowe translatora na symulatorze wg. klucza:

1-czytnik taśmy papierowej

2-perforator taśmy papierowej

3-monitor

2.11.Kontrola zawartości miejsc pamięci

Dyrektywa o postaci:

o* wyrażenie określone . wyrażenie określone.

pozwala na wyprowadzenie oktalne adresów i zawartości obszaru pamięci operacyjnej ograniczonego wartościami podanych wyrażeń.

3.Komentarze i teksty.

3.1. Komentarz zwykły.

Cały tekst zamknięty w nawiasach [ ] i zawierający dowolne znaki kodu ISO-7 jest przez translator ignorowany.

3.2. Komentarz dynamiczny.

Tekst zamknięty w nawiasach < > jest wprowadzany na urządzenie wyjściowe translatora a przez sam translator jest pomijany.

3.3. Tekst.

Translator ASSK umożliwia wprowadzenie do pamięci dowolnego tekstu alfanumerycznego.Tekst ten winien być ograniczony parą znaków " / cudzysłów / i zawierać może pozostałe znaki kodu ISO-7.

Znaki tekstu umieszczane są po dwa w słowie i lokowane w kolejnych miejscach pamięci oraz uzupełniane jednym lub dwoma znakami "blank" do parzystej ilości.

4. ROZKAZY.

Rozkaz składa się z kodu operacji i części opisujących argumenty operacji.

Kod operacji jest nazwą z listy rozkazów m.c. K-202.

Rozkaz może posiadać dwa argumenty operacji /arg I i arg II/, lub jeden argument /arg I lub arg II/.

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

4.1. Argument I.

Argumentem I jest jeden z rejestrów podany w postaci liczby bez znaku z przedziału 0-7, lub etykiety o wartości z tego przedziału.

4.2. Argument II.

Argumentem II może być:

- jeden z rejestrów podany w postaci liczby lub etykiety z przedziału 1-7;
- komórka pamięci podana jako wyrażenie arytmetyczne o wartości 0-65536;
- parametr krótki podany w postaci liczby z przedziału /-63,63/ lub sumy algebraicznej liczb i etykiet o wartości z tego przedziału.

  Składniki sumy w ostatnim przypadku nie mogą być skalowane.

4.3. Kodowanie rozkazów.

Pola rozkazu oznaczamy:

A - I argument

C - II argument

T - II argument krótki

Ze względu na różnorodną budowę rozkazy można podzielić na grupy:

- rozkazy dwuargumentowe

Do tej grupy należą rozkazy o częściach operacyjnych:
ad, adc, su, co, and, or, orn, clmo, clbo, lo, lom,
los, lob, st, stb, jpar.
Kod rozkazów z tej grupy ma postać:

op,A,C.

dla rozkazów o arg II umieszczonym w rejestrze;

op,A(C)

dla rozkazów o argII umieszczonym w pamięci.

Przykłady:

ad,1,2.

su,a,3.

co,alfa,b.

or,6(010/14)

st,7(32000-abc)

- rozkazy posiadające tylko arg II

Do tej grupy należą rozkazy o częściach operacyjnych:
jp, jpl, jpe, jpg, jpr, jprl, jpre, jprg, lbar, sbar, mod,
ex, reex, ados, zs.
Kod rozkazów z tej grupy ma postć:

op,C.

dla rozkazów o arg II umieszczonym w rejestrze;

op(C)

dla rozkazów o arg II umieszczonym w pamięci.

Przykłady:

jp,7.

ex(3)

mod(a-b+c/011)

ados(rob-3)

- rozkazy posiadające tylko arg I

Do tej grupy należą rozkazy o częściach operacyjnych:
stop, neg, nec, nega, chan, lobi, rkey, shl, shv,
shly, shvy, shlx, shvx, shr, shry, shrx, stxa, stxz.
Kod rozkazów z tej grupy ma postać:

op,A.

Przykłady:

nega,3.

shv,rej.

- rozkazy z krótkim arg II /oznaczone literą T/

Rozkazy z tej grupy mogą posiadać albo arg I i arg II,
albo tylko arg II.
Argument II może być liczbą z przedziału /-63,63/.
Do tej grupy należą rozkazy o częściach operacyjnych:
adt, adot, cot, lot,
adjt, lts, sts, jpt, jptl, jpte, jptg, jptv, jptx,
jpty, jpti.
Kod rozkazów z tej grupy ma postać:

op,A,T. lub op,T.

Cztery pierwsze rozkazy posiadają arg II jako rzeczywisty efektywny argument operacji.
Przykłady:

adt,1,015.

lot,zeta,a-5.

cot,r,50.

Uwaga:Jeżeli krótki parametr jest podany jako wyrażenie arytmetyczne i zawiera jedną lub więcej etykiet niezdefiniowanych pozostałe składniki muszą być z przedziału /-63,63/.

W przeciwnym przypadku zostanie zasygnalizowany błąd, nawet jeżeli ogólna wartość wyrażenia mieści się w przyjętym przedziale.

W pozostałych rozkazach z tej grupy arg II jest efektywnym argumentem względnym, względem bieżącego wskaźnika pamięci operacyjnej.

Parametr T może być podany albo jako liczba albo wyrażenie arytmetyczne zawierające etykiety.

Wystąpienie pierwszej etykiety w drugiej grupie rozkazów krótkich, powoduje odjęcie od niej wskaźnika bieżącego adresu pamięci, dlatego pierwsza etykieta powinna być etykietą określającą miejsce w programie.

Do tej grupy odnosi się również uwaga zamieszczona wyżej.

Przykłady:

```
jpt,a+3.
lts,1,-20.
sts,par,1+rob-3.
```

- rozkazy przesyłania grupowego.

Do tej grupy należą rozkazy o częściach operacyjnych: log, stg.

Kod rozkazów z tej z grupy ma postać

op,A,C.

dla rozkazów o arg II umieszczonym w rejestrze

op,A(C)

dla rozkazów o arg II umieszczonym w komórce pamięci.

Część rozkazu oznaczona literą A nie oznacza tu numeru rejestru, lecz jest wartością liczbową określającą ilość pobieranych lub zapamiętanych rejestrów w grupie i nie może być podana symbolicznie.

Przykłady:

```
log,4,3.          - umieść w R1, R2
log,5(beta+3)     - umieść R1, R2, R3
log,5(beta-010)   - umieść R1, R2, R3
stg,6(x)          - pamiętaj R1 - R7
stg,7,r1.         - pamiętaj R5, R6, R7
```

Modyfikacja argumentu II.

Możliwe są trzy rodzaje modyfikacji argumentu II wykonywane w następującej kolejności:

- pre - modyfikacja

Modyfikacji tego rodzaju podlega każdy rozkaz posiadający arg II poprzedzony rozkazem o części operacyjnej "mod" /modyfikuj/

Argument efektywny rozkazu "mod" zostaje dodany do argumentu II rozkazu modyfikowanego. Jeden rozkaz może być poprzedzony co najwyżej trzema rozkazami "mod".

Przykłady:

```
mod,6. lo,1(r3)
          umieść w rejestrze R1
mod(0130) mod,1. ad,3(p12)
```

B - modyfikacja

Modyfikacja ta może być stosowana w rozkazach w których argII nie jest krótkim parametrem.

Polega ona na dodaniu do arg II rozkazu zawartości wybranego rejestru /R1-R7/

Rozkaz zawierający argument z B modyfikacją koduje się przez umieszczenie za opisem arguII znaku "(" lub ";".

Kod rozkazu ma postać:

op,A,C;B. lub

op,A(C;B) lub

op,C,B. lub

op(C;B)

Przykłady:

lo,3(06000;5)

ad,rej,rej2;rej3.

- D - modyfikacja

Modyfikacja ta może być stosowana w rozkazach w których arg II nie jest krótkim parametrem.

Nie może być stosowana w rozkazach o częściach operacyjnych "log" i "stg".

Jeżeli w rozkazie jest D=1 arg II jest traktowany jako adres efektywnego argumentu operacji. Rozkaz taki koduje się umieszczając za opisem arg II znak " ` ".

Kod rozkazu ma postać:

op,A,C`.

op,A(C`)

op,C`.

op(C`)

Przykłady:

lo,alfa,6`.

su,7(0100+rob`)

jp ,xyz`.

ex(we-5`)

Znak "`" umiaszczony na końcu arg II zapisanego jako wyrażenie arytmetyczne odnosi się do wyrtości liczbowej całego wyrażenia.

Rozkazy mogą posiadać 3 rodzaje modyfikacji:

Przykłady:

mod(x`) jp(tab1;7`)

lo,alfa(beta+1;c`)

4.4. Rozkazy warunkowe.

Każdy rozkaz może być warunkowy tzn. wykonywany gdy aktualny stan warunków w rejestrze RO jest zgodny z maską warunków podaną w rozkazie i jest pomijany w przeciwnym przypadku.

Rejestr RO posiada 16 pozycji o następujących nazwach i znaczeniach.

Q - wskaźnik systemu

L - " ustawiany w wyniku porównania

E - " " " "

G - " " " "

V - wskaźnik nadmiaru

C - " przeniesienia

Y - " przechowujący bit wychodzący przez rejestr w rozkazych przesuwania

I - wskaźnik systemu

X - wskaźnik ustawiany programowo do dowolnego wykorzystania przez programistę.

Rozkaz warunkowy koduje się przez umieszczenie za opisem argumentu II /dla rozkazów nie posiadających arg II za arg I / znaku "?". Za tym znakiem należy podać zestaw warunków.

Warunkami mogą być litery wymienione wyżej jako nazwy odpowiednich pozycji rejestru RO lub cyfry z przedziału 1 - 7.

Warunki nie mogą być zapisane symbolicznie.

Kod rozkazu warunkowego ma postać:

op,A?W.

op,A,C?W.

op,A(C?W).

op,C?W.

op(C?W)

op,A,T?W.

op,T?W.

Przykłady:

shl,3?1.

lot,1,x+3?117.

su,re(z-2;5`?g62)