POLITECHNIKA POZNAŃSKA

INSTYTUT AUTOMATYKI

60-965 Poznań, pl. M. Skłodowskiej-Curie 9

tel. 320-21 w. 321, 325

Zleceniodawca Zarząd Dróg i Mostów Poznań, ul.Wilczak 16	Nr umowy IA-U/11/74/75 NB
Nazwa opracowania Język symboliczny AWIK	Rodzaj opracowania Instrukcja programisty
Urządzenie	Egz. 4
	Nr archiwalny 76006

ZGPP A/741/72-1000

Politechnika Poznańska

Instytut Automatyki

J ę z y k s y m b o l i c z n y A W I K

- Instrukcja programisty -

Zleceniodawca: Zarząd Dróg i Mostów
Poznań, ul. Wilczak 16

Znak zlecenia: TRE/7132/75

Nasz znak: IA-U/11/74/75 B

Dyrektor Instytutu

Prof.dr hab.inż.Antoni Woźniak

Poznań, 1976

Autorzy:

dr inż.Jacek Martinek

mgr inż.Jan Nadolski

mgr Jerzy Bartoszek

1. Wstęp

AWIK jest językiem adresów symbolicznych maszyny WIK. WIK jest maszyną wirtualną czyli taką, która nie jest zrealizowana sprzętowo, lecz jest utworzona przez zespół programów maszyny cyfrowej Momik 8b-100. Organizację WIK-a opisano szczegółowo w [1].

AWIK jest językiem programowania niskiego poziomu, w którym adresy i nazwy danych można wyrażać przy pomocy nazw złożonych ze znaków alfanumerycznych i dobieranych przez programistę. Program napisany w języku AWIK (program źródłowy) jest ciągiem tzw. jednostek programu. Jednostką może być

- rozkaz
- dyrektywa
- definicja nazwy
- parametr
- tekst
- komentarz.

Podczas tłumaczenia programu źródłowego komenterze są ignorowane, dyrektywy służą do sterowania tłumaczeniem, zaś rozkazy, parametry i teksty stają się (po przetłumaczeniu) elementami programu wynikowego.

Program wynikowy jest tekstem uzyskiwanym po dwóch przebiegach translatora. Tekst ten składa się ze znaków alfanumerycznych i stanowi zapis stanu pamięci w kodzie szesnastkowym. Tak więc jedno słowo 8-bitowe pamięci (1 bajt) jest w tym zapisie reprezentowane przez dwa znaki szesnastkowe. */

---

*/ W zapisie szesnastkowym stosuje się następujące znaki: 0,1,2,...,9,A,B,C,D,E,F.

Program wynikowy zapisuje się do pamięci Momika przy pomocy specjalnego programu ładującego.

Translator języka AWIK zaprogramowano w LISP-ie. W ten sposób powstał tzw. translator skośny, dokonujący tłumaczenia programów dla maszyny Momik na innej maszynie, wyposażonej w system LISP-u. Przy takim rozwiązaniu tekst programu zapisanego w języku AWIK powinien stanowić dane możliwe do zaakceptowania przez system LISP-u. Wynika z tego, że słowa języka AWIK muszą być jednocześnie słowami w sensie języka LISP. */

2. Słowa języka AWIK

W języku AWIK istnieją trzy rodzaje słów:

- nazwy
- liczby całkowite
- ograniczniki

Nazwa jest to sznur złożony z liter, cyfr, znaków „+" i „-", rozpoczynający się literą. Nazwa nie może zawierać więcej niż 64 znaki.

Przykłady:

BUFOR

ALA

PR1-PR2

KV1+2

C12

VMAX-VMIN

---

*/ Realizacja dla e.m.c. K202

Liczby całkowite dzielą się na ósemkowe i dziesiętne. Liczba ósemkowa rozpoczyna się cyfrą 0, po której może następować do 6 pozycji znaczących. Zapisuje się na nich cyfry ósemkowe 0, 1, 2,...,7 , z tym, że największą akceptowaną liczbą ósemkową jest 0177777_8, bowiem zapis dwójkowy liczby ósemkowej nie może przekraczać 16 bitów.

Przykłady:

0, 01 , 013 , 011

Liczba dziesiętna składa się z cyfr dziesiętnych 0, 1, 2,....,9 poprzedzonych znakiem + lub -. Znak + można opóścić jeśli najstarsza cyfra liczby jest różna od 0. Liczba dziesiętna może składać się co najwyżej z 5 cyfr, a jej wartość powinna być zawarta w przedziale od -32768 do +32767, bowiem zapis dwójkowy liczby dziesiętnej także nie może przekraczać 16 bitów.

Przykłady:

+0, +012 , -016 , +12 , 12 , -12

Liczbami dziesiętnymi nie są:

0 , 33700, 7.2 , 5E2

Ogranicznikami są następujące znaki alfanumeryczne:

( ) [ ] ; , * "odstęp" "powrót karetki"

"zmiana wiersza" oraz para znaków :=

Zbiór słów można zdefiniować krótko w sposób następujący:

$$\langle \text{słowo} \rangle ::= \langle \text{nazwa} \rangle \mid \langle \text{liczba całkowita} \rangle \mid \langle \text{ogranicznik} \rangle$$

$$\langle \text{nazwa} \rangle ::= \langle \text{litera} \rangle \left\{ \langle \text{litera} \rangle \mid \langle \text{cyfra} \rangle \mid \langle \text{znak} \rangle \right\}_{0}^{63}$$

$$\langle \text{liczba całkowita} \rangle ::= \langle \text{liczba ósemkowa} \rangle \mid \langle \text{liczba dziesiętna} \rangle$$

$$\langle \text{liczba ósemkowa} \rangle ::= 0 \left\{ 0 \mid 1 \right\}_{0}^{1} \left\{ \langle \text{cyfra ósemkowa} \rangle \right\}_{0}^{5}$$

<liczba dziesiętna> ::= <znak> $\{$<cyfra>$\}_{1}^{5}$ | <cyfra1> $\{$<cyfra>$\}_{0}^{4}$

<ogranicznik> ::= (|) | [|] | ; | * | := | , | sp | cr | lf

<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> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<cyfra1> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<cyfra ósemkowa> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

<znak> ::= + | -

W tym zapisie $\{\ldots\}_{m}^{n}$ jest skrótem wskazującym, że treść ujęta w nawiasy może wystąpić od m do n razy.

3. Adresowanie

Pamięć maszyny wirtualnej WIK składa się z segmentów, z których każdy ma długość 256 słów 8-bitowych.

Adresowanie pamięci wirtualnej odbywa się przez cztery rejestry bazowe (zwane S-rejestrami) posiadające numery 0, 1, 2, 3. W S-rejestrach umieszcza się identyfikatory segmentów. Identyfikator segmentu jest liczbą 5-bitową (przy wykorzystaniu tylko pamięci operacyjnej) lub 16-bitową (przy współpracy z pamięcią dyskową).

Dwa bity określające numer S-rejestru i 8-bitów numeru słowa w segmencie składają się na 10-bitowy adres względny.

Programista pisząc program w języku AWIK posługuje się zatem 10-bitowymi adresami względnymi i adresuje pamięć względną

o maksymalnej pojemności 1024 słowa. Umiejscowienie pamięci względnej w pamięci wirtualnej zależy od stanu S-rejestrów w momencie wykonywania programu. W ogólnym przypadku segmenty kolejne w sensie pamięci względnej mogą nie być kolejnymi segmentami w pamięci wirtualnej. Identyfikatory segmentów zapisuje się do S-rejestrów przy pomocy specjalnych rozkazów opisanych dalej.

4. Definiowanie nazw

Nazwie nadaje się wartość w definicji nazwy lub w definicji etykiety.

Definicja nazwy posiada składnię

<definicja nazwy> ::= <nazwa> := <wyrażenie>

W momencie napotkania przez translator definicji nazwy, wartość wyrażenia występującego w tej definicji powinna być możliwa do obliczenia. Wynika stąd, że wyrażenie nie może zawierać nazw których jeszcze nie zdefiniowano.

Etykieta jest to nazwa (adres symboliczny) miejsca w programie. Wartością etykiety powinna być liczba całkowita z przedziału $[0 + 1023]$. Definicja etykiety posiada składnię:

<definicja etykiety> ::= (<nazwa>)

Nazwie występującej w definicji etykiety nadaje się wartość równą bieżącej wartości licznika adresów translatora.

5. Wyrażenia

Wyrażeniem w języku AWIK jest

- nazwa
- liczba całkowita
- napis zbudowany z wyrażeń przy pomocy jednego z operatorów PLUS, MINUS, TIMES, DIVBY, S i ewentualnie nawiasów (i).

Operatory PLUS, MINUS, TIMES, DIVBY i S są operatorami dwuargumentowymi. Operator PLUS oznacza dodawanie, MINUS - odejmowanie, TIMES - mnożenie, DIVBY - dzielenie. Operator S powoduje pomnożenie pierwszego operandu przez 256 i dodanie do tak otrzymanej liczby wartości drugiego operandu. Dzięki temu operatorowi możemy zapisywać adresy względne jako wyrażenia postaci

⟨nr S-rejestru⟩ S ⟨adres w ramach segmentu⟩

W sposób ścisły wyrażenia można opisać następująco:

⟨wyrażenie⟩ ::= ⟨term⟩ | ⟨term⟩⟨operator1⟩ ⟨wyrażenie⟩

⟨term⟩ ::= ⟨czynnik⟩ | ⟨czynnik⟩⟨operator2⟩⟨term⟩

⟨czynnik⟩ ::= ⟨nazwa⟩ | ⟨liczba całkowita⟩ | (⟨wyrażenie⟩)

⟨operator1⟩ ::= PLUS | MINUS

⟨operator2⟩ ::= TIMES | DIVBY | S

Poprawnymi wyrażeniami są np.:

+127

KUBA

VSR TIMES (3 PLUS 2)

2 PLUS VSR TIMES 3

—3 MINUS 7 PLUS 2

3 MINUS (7 PLUS -2)

(3 PLUS 7) TIMES SUMSR

A TIMES B DIVBY ODCHYLENIE

C S (40 TIMES 2 DIVBY X)

Błędne są następujące napisy:

3 + 2

- A TIMES B

2PLUS3

2S43

Nazwa, której nadano wartość nazywa się nazwą określoną.

Wyrażeniem określonym jest

- liczba
- nazwa określona
- wyrażenie zbudowane przy pomocy operatora z dwóch wyrażeń określonych.

Dla każdego wyrażenia określonego można w oczywisty sposób określić wartość.

6. Rozkazy arytmetyczno-logiczne

Rozkazy arytmetyczno-logiczne komputera WIK dzieli się na pięć grup:

a) rozkazy adresowe,

b) rozkazy z operandem bezpośrednim,

c) rozkazy z krótkim argumentem,

d) rozkazy bezargumentowe krótkie,

e) rozkazy bezargumentowe długie.

a) Rozkazy adresowe

Rozkaz adresowy posiada składnię:

⟨rozkaz adresowy⟩::=

⟨nazwa rozkazu adresowego⟩ ⟨ogr⟩ [I ⟨ogr⟩][X ⟨ogr⟩]⟨adres względny⟩

⟨nazwa rozkazu adresowego⟩ ::=

LA | LX | SA | SX | AD | SU | JP | JS | CO

⟨ogr⟩ ::= , | sp

⟨adres względny⟩ ::= ⟨wyrażenie⟩

Skrót {...} oznacza, że umieszczenie w rozkazie treści ujętej w nawiasy nie jest konieczne.

Wartość adresu względnego powinna być zawarta w przedziale od 0 do 1023.

Litera I oznacza adresację pośrednią, a litera X oznacza X-modyfikację.

Rozkazy adresowe posiadają następującą semantykę (patrz też [1])

Nazwa rozkazu adresowego	Treść	Opis operacji
LA	Ładuj A	/V/ → A
LX	Ładuj X	/V/ → X
SA	Pamiętaj A	/A/ → V
SX	Pamiętaj X	/X/ → V
AD	Dodaj do A	/A/ + /V/ → A
SU	Odejmij od A	/A/ - /V/ → A
ML	Mnóż logicznie przez A	/A/ ∧ /V/ → A
JP	Skocz	V → LR
JS	Skocz ze śladem	$/LR_{0-1}/ \rightarrow V$; $/LR_{2-9}/ \rightarrow V+1$ V + 2 → LR
CO	Porównaj arytmetycznie	Jeżeli /A/ < /V/ to 1→L, 0→E, 0→G
		Jeżeli /A/ = /V/ to 0→L, 1→E, 0→G
		Jeżeli /A/ > /V/ to 0→L, 0→E, 1→G

Przykłady

LA I X 24;

LA,I,X,24;

LX X ALA TIMES (21 DIVBY 7 MINUS 1);

ML 0377;

CO,+014;

Błędne są następujące napisy:

SU X PLUS 2;

JP I X S 3;

JS 32

b) Rozkazy z operandem bezpośrednim

Rozkaz z operandem bezpośrednim posiada składnię:

⟨rozkaz z operandem bezpośrednim⟩::=

⟨nazwa rozkazu z operandem bezpośrednim⟩⟨ogr⟩⟨operand⟩;

⟨nazwa rozkazu z operandem bezpośrednim⟩::=

LAS|LXS|ADS|XDS|SUS|XSS|JPS

⟨ogr⟩::= , |sp

⟨operand⟩::= ⟨wyrażenie⟩

Wartość operandu powinna być zawarta w przedziale od 0 do 255.

Rozkazy tej drupy posiadają następującą semantykę

Nazwa rozkazu	Treść	Opis operacji
LAS	Ładuj krótko A	OPERAND → A_{8-15}
LXS	Ładuj krótko X	OPERAND → X_{8-15}
ADS	Dodaj krótko do A	/A/ + OPERAND → A
XDS	Dodaj krótko do X	/X/ + OPERAND → X
SUS	Odejmij krótko od A	/A/ - OPERAND → A
XSS	Odejmij krótko od X	/X/ - OPERAND → X
JPS	Skocz krótko	OPERAND → LR_{2-9}
		LR_{0-1} bez zmian

Przykłady:

```
LAS 210;
LXS 030;
ADS, SUMA MINUS2;
XDS 2 TIMES VMAX;
```

napisy błędne:

```
SUS I 20;
XSS 430;
JPS - 2;
```

c) Rozkazy z krótkim argumentem

Rozkaz tej grupy posiada składnię:

⟨rozkaz z krótkim arg.⟩ ::=

⟨nazwa rozkazu z krótkim arg.⟩⟨ogr⟩⟨krótki argument⟩;

⟨nazwa rozkazu z krótkim arg.⟩ ::= LL|LR|CL|CR|AL|AR|K1|KØ

⟨ogr⟩::= , |sp

⟨krótki argument⟩::= ⟨wyrażenie⟩

Wartość krótkiego argumentu powinna być zawarta w przedziale od 0 do 15.

Rozkazy z krótkim argumentem posiadają następującą semantykę:

Nazwa rozkazu	Treść	Opis operacji
LL	Przesuń A logicznie w lewo	Zawartość A zostaje przesunięta w lewo- ilość pozycji jest określona w krótkim arg.
LR	Przesuń A logicznie w prawo	Zawartość A zostaje przesunięta w prawo - ilość pozycji jest określona krótkim arg.
CL	Przesuń A cyklicznie w lewo	Zawartość A zostaje przesunięta cyklicznie w lewo - ilość pozycji jest określona krótkim arg.
CR	Przesuń A cyklicznie w prawo	Zawartość A zostaje przesunięta cyklicznie w prawo - ilość pozycji jest określona krótkim arg.

AL	Przesuń A arytmetycznie w lewo	$A_{\emptyset}$ - bez zmian; A_{1-15} zostaje przesunięta w lewo o ilość pozycji określoną krótkim argumentem
AR	Przesuń A arytmetycznie w prawo	$A_{\emptyset}$ - bez zmian; A_{1-15} zostaje przesunięta w prawo o ilość pozycji określoną krótkim argumentem
K1	Przeskocz,gdy przycisk klawiatury wciśnięty	Gdy przycisk określony krótkim argumentem jest wciśnięty to 1 → P
KØ	Przeskocz gdy przycisk klawiatury wciśnięty	Gdy przycisk określony krótkim argumentem jest wciśnięty to 1 → P.

Przykłady:

LL ∅;

LR SMIN;

CL VMAX DIVBY SUMA;

napisy błędne:

AL - 1;

AR 16;

K1 2

d) Rozkazy bezargumentowe krótkie

Rozkaz bezargumentowy krótki posiada składnię:

⟨rozkaz bezargumentowy krótki⟩ ::= ⟨nazwa rozkazu bezargumentowego krótkiego⟩ ;

⟨nazwa rozkazu bezargumentowego krótkiego⟩ ::= JL|JE|JG|JV|JI|ZA|NA

Rozkazy bezargumentowe krótkie posiadają następującą semantykę:

Nazwa rozkazu	Treść	Opis operacji
JL	Przeskocz gdy L=1	Jeżeli L=1 to 1 → P
JE	Przeskocz gdy E=1	Jeżeli E=1 to 1 → P
JG	Przeskocz gdy G=1	Jeżeli G=1 to 1 → P
JV	Przeskocz gdy V=1	Jeżeli V=1 to 1 → P
JI	Przeskocz bezwarunkowo	1 → P
ZA	Zeruj A	$\emptyset \rightarrow A$
NA	Neguj logicznie A	$/\overline{A}/ \rightarrow A$

Przykłady:

JL;

JG;

e) Rozkazy bezargumentowe długie

Rozkaz tej grupy posiada składnię

⟨rozkaz bezargumentowy długi⟩ ::= OPT ⟨ogr⟩⟨nazwa rozkazu bezargumentowego długiego⟩;

⟨ogr⟩ ::= , |sp

nazwa rozkazu bezargumentowego długiego ::= AX|ALR|CHA|CHX|XA|LRA

Rozkazy bezargumentowe długie posiadają następującą semantykę:

Nazwa rozkazu	Treść	Opis operacji
AX	Prześlij A do X	/A/ → X
ALR	Prześlij A do LR	/A/ → LR
CHA	Zamień słowa w A	$/A_{0-7}/ \rightarrow A_{8-15}$; $/A_{8-15}/ \rightarrow A_{0-7}$
CHX	Zamień słowa w X	$/X_{0-7}/ \rightarrow X_{8-15}$; $/X_{8-15}/ \rightarrow X_{0-7}$
XA	Prześlij X do A	/X/ → A
LRA	Prześlij LR do A	/LR/ → A

Przykłady:

OPT AX;

OPT CHA;

7. Rozkazy synchronizacyjne

Rozkazy synchronizacyjne dzielą się na trzy grupy:

- rozkazy przesyłania informacji
- rozkazy tworzenia procesorów
- rozkazy sterowania pracą procesorów.

a) Rozkazy przesyłania informacji

Rozkaz tej grupy posiada składnię

⟨rozkaz przesyłania informacji⟩ ::=
⟨nazwa ro-prze-inf⟩⟨ogr⟩⟨nazwa procesora⟩⟨ogr⟩⟨adres⟩ ;

⟨nazwa ro-prze-inf⟩ ::= SME|WME|SAN|WAN

⟨ogr⟩::= , sp

⟨nazwa procesora⟩ ::= ⟨wyrażenie⟩

⟨adres⟩ ::= ⟨wyrażenie⟩

Wartość wyrażenia będącego nazwą procesora musi być zawarta w przedziale od 1 do 63 a w przypadku rozkazów WME i SAN może być zerem, natomiast wartość adresu musi być zawarta w przedziale od 0 do 1023.

Semantykę rozkazów tej grupy omówiono szczegółowo w 1 . Tutaj przytoczymy tylko krótką charakterystykę każdego rozkazu.

SME - wyślij komunikat - procesor-nadawca (wykonujący rozkaz SME) wysyła komunikat do procesora-odbiorcy, którego nazwa jest podana w rozkazie. Adres określa położenie treści komunikatu w pamięci względnej.

WME - czekaj na komunikat - sprawdzenie czy w kolejce komunikatów do procesora wykonującego rozkaz WME znajduje się komunikat od procesora o nazwie podanej w rozkazie. Jeżeli takiego komunikatu nie ma, to procesor zostaje zablokowany. W przeciwnym razie treść komunikatu znajdującego się w kolejce wpisuje się do pamięci od adresu wskazanego w rozkazie.

SAN - wyślij odpowiedź - wysłanie odpowiedzi do procesora o nazwie podanej w rozkazie. Adres określa położenie treści odpowiedzi w pamięci.

WAN - czekaj na odpowiedź - sprawdzenie czy w kolejce komunikatów do procesora wykonującego rozkaz WAN znajduje się odpowiedź od procesora o nazwie podanej w rozkazie. Jeżeli takiej odpowiedzi nie ma, to procesor zostaje zablokowany. W przeciwnym razie treść odpowiedzi zostaje przepisana do pamięci od miejsca wskazanego przez adres występujący w rozkazie.

Przykłady:

```
SME 1 632;
SME, RESZTA PLUS SUMA, BUFOR PLUS 72;
WME 0 10;
SAN , 0 , STOS1 TIMES 2;
WAN , 1 , 2 S 130;
WAN WX 3 S 12;
```

b) Rozkazy tworzenia procesorów

Rozkaz tej grupy posiada składnię:

⟨rozkaz tworzenia procesora⟩ ::=
⟨rozkaz tworzenia procesora ar-log⟩ |
⟨rozkaz tworzenia procesora we-wy-zew⟩

⟨rozkaz tworzenia procesora ar-log⟩ ::=
CAP⟨ogr⟩⟨nazwa procesora⟩⟨ogr⟩⟨adres⟩⟨ogr⟩⟨nazwa segmentu⟩
⟨ogr⟩⟨priorytet⟩ ;

⟨rozkaz tworzenia procesora we-wy-zew⟩ ::=
⟨nazwa rozkazu tworzenia pr-we-wy-zew⟩ ⟨ogr⟩⟨nazwa procesora⟩
⟨ogr⟩⟨nr urządzenia⟩ ;

⟨nazwa rozkazu tworzenia pr-we-wy-zew⟩ ::= CIP|COP|CEP

⟨ogr⟩::= , | sp

⟨nazwa procesora⟩ ::= ⟨wyrażenie⟩

⟨nazwa segmentu⟩ ::= ⟨wyrażenie⟩

⟨adres⟩ ::= ⟨wyrażenie⟩

⟨nr urządzenia⟩::= ⟨wyrażenie⟩

⟨priorytet⟩::= ⟨wyrażenie⟩

Ograniczenia wartości wyrażeń są następujące:

nazwa procesora	od 1 do 63
nazwa segmentu	od 0 do 31
adres	od 0 do 1023
nr urządzenia	od 0 do 31
priorytet	od 0 do 7

Krótka charakterystyka rozkazów tworzenia procesorów.

CAP - utwórz procesor arytmetyczno-logiczny. W rozkazie podaje się nazwę tworzonego procesora, adres - określający stan początkowy rejestru LR, nazwę segmentu - która zostanie umieszczona w S-rejestrze wskazanym przez dwa najstarsze bity "adresu" oraz priorytet procesora.

CIP - utwórz procesor wejścia. Procesor zostaje związany z urządzeniem wejściowym o numerze podanym w rozkazie.

COP - utwórz procesor wyjścia. Znaczenie jest podobne jak CIP.

CEP - utwórz procesor zewnętrzny. Znaczenie jest podobne jak CIP

Przykłady:

```
CAP, 60 , 2 S 30 , 30 , 3;
CAP , POPR PLUS 1 , 01023, 2 , 1;
CIP , OST PLUS 3 , 17;
```

c) Rozkazy sterowania pracą procesorów

Rozkaz należący do tej grupy musi mieć następującą składnię

⟨rozkaz sterowania pracą procesora⟩ ::=
⟨rozkaz sterowania bezadresowy⟩|⟨rozkaz sterowania z adresem⟩
⟨rozkaz kodowania S-rejestru⟩

⟨rozkaz sterowania bezadresowy⟩ ::=
⟨nazwa rozkazu sterowania bezadresowego⟩⟨ogr⟩⟨nazwa procesora⟩

⟨nazwa rozkazu sterowania bezadresowego⟩ ::= DEP|ST1|STP

⟨rozkaz sterowania z adresem⟩ ::= ST2 ⟨ogr⟩⟨nazwa procesora⟩⟨ogr⟩
⟨adres⟩ ;

⟨rozkaz ładowania S-rejestru⟩ ::=
⟨nazwa rozkazu ładowania S-rejestru⟩⟨ogr⟩⟨nazwa segmentu⟩ ;

⟨nazwa rozkazu ładowania S-rejestru⟩ ::= LSØ|LS1|LS2|LS3

⟨nazwa procesora⟩ ::= ⟨wyrażenie⟩

⟨adres⟩ ::= ⟨wyrażenie⟩

⟨nazwa segmentu⟩ ::= ⟨wyrażenie⟩

⟨ogr⟩ ::= , | sp

Ograniczenia wartości wyrażeń są następujące:

nazwa procesora	od 1 do 63
adres	od 0 do 1023
nazwa segmentu	od 0 do 31

Charakterystyka rozkazów sterowania pracą procesorów.

DEP - usuń procesor - usunięcie procesora o nazwie podanej w rozkazie znajdującego się w stanie zatrzymania lub zablokowania i zatrzymania.

ST1 - startuj procesor - wprowadzenie procesora o nazwie podanej w rozkazie w stan przetwarzania (gdy procesor określony przez "nazwę" był w stanie przetwarzania lub zatrzymania) lub zablokowania (gdy rozpatrywany procesor był w stanie zablokowania lub zablokowania i zatrzymania)

STP - zatrzymaj procesor - zatrzymanie procesora o nazwie podanej w rozkazie.

ST2 - startuj procesor od podanego adresu - startowanie procesora arytmetyczno-logicznego o określonej nazwie od adresu podanego w rozkazie.

LSØ - ładuj S-rejestr o numerze Ø - zapisanie do rejestru SØ procesora wykonującego rozkaz nazwy segmentu podanej w rozkazie.

LS1, LS2, LS3 - ładuj S-rejestr o numerze 1,2,3 - działanie analogiczne jak rozkazu LSØ.

Przykłady

```
DEP, 1;
ST1, MONITOR;
STP 24;
ST2 23  2 S 10;
LS0 31;
LS1 LASTN PLUS 1;
```

8. Dyrektywy

Dyrektywy są jednostkami programu źródłowego, które sterują działaniem translatora języka AWIK.

Składnia dyrektywy

⟨dyrektywa⟩ ::= ⟨dyrektywa bez argumentu⟩|⟨dyrektywa z argumentem

⟨dyrektywa bez argumentu⟩ ::=[⟨nazwa dyrektywy bez argumentu⟩]

⟨dyrektywa z argumentem⟩ ::=[⟨nazwa dyrektywy z argumentem⟩ ⟨wyrażenie⟩]

⟨nazwa dyrektywy bez argumentu⟩ ::= PROGEND|MODBEG|MODEND|GLOB|LAB|NLAB|STATE|TST|NTST|

nazwa dyrektywy z argumentem ::= SEG|ADR|RES|INT|INTO|OUT|OUT

Wartość wyrażenia będącego argumentem dyrektywy ADR musi być zawarta w przedziale od 0 do 1023.

Semantyka dyrektyw

PROGEND - koniec programu - powoduje zakończenie translacji.

MODBEG - początek modułu programu - wszystkie etykiety i nazwy (zmienne translacji) definiowane wewnątrz modułu, które nie są poprzedzone dyrektywą GLOB zostają zdefiniowane lokalnie, to znaczy definicja obowiązuje tylko wewnątrz modułu.

MODEND - koniec modułu programu

GLOB - dyrektywa pozwalająca zdefiniować globalnie ~~etykietę~~ nazwę wewnątrz modułu. Najbliższa występująca po dyrektywie GLOB definicja etykiety lub zmiennej translacji będzie definicją globalną.

LAB - polecenie wyprowadzania definicji etykiet i zmiennych translacji na wyjście operatorskie.

NLAB - unieważnienie polecenia wydanego dyrektywą LAB.

STATE - dyrektywa powodująca wpisanie do programu wynikowego rozkazu STA raportującego stan procesora. Wpisanie rozkazu STA ma miejsce jeśli dyrektywa STATE jest poprzedzona dyrektywą TST

TST - dyrektywy STATE poprzedzone dyrektywą TST powodują działania opisane wyżej

NTST - anuluje efekt wykonanej poprzednio dyrektywy TST.

SEG - ustalenie segmentu od którego umieszczany będzie przekład programu. W pierwszym przebiegu translacji dyrektywa SEG powoduje wyzerowanie zmiennej ADR translatora, natomiast w drugim przebiegu oprócz wyzerowania zmiennej ADR wyprowadza się na aktualne wyjście identyfikator segmentu służący do sterowania programem ładującym. Wartością argumentu dyrektywy powinna być liczba będąca nazwą (identyfikatorem) segmentu.

ADR - ustalenie wartości zmiennej ADR translatora czyli ustalenie adresu względnego. Argumentem dyrektywy powinno być wyrażenie określone o wartości z zakresu od 0 do 1023.

RES - zarezerwowanie komórek w ilości określonej wartością argumentu. Komórki (8-bitowe) zostają wyzerowane.

INT - ustalenie wejścia z którym współpracuje program translatora. Wartością argumentu powinien być numer urządzenia wejściowego.

INTO - ustalenie wejścia operatorskiego - wartością argumentu powinien być numer urządzenia wejściowego.

OUT - ustalenie wyjścia z którym współpracuje program translatora. Wartością argumentu powinien być numer urządzenia wyjściowego.

OUTO - ustalenie wyjścia operatorskiego - wartością argumentu powinien być numer urządzenia wyjściowego.

Urządzenia posiadają następujące numery:

Czytnik taśmy papierowej	1
Czytnik taśmy papierowej współpracujący z buforem początkowo wyzerowanym	10
Czytnik taśmy papierowej współpracujący z buforem - bez wyzerowania bufora	11
Perforator taśmy	1
Drukarka mozaikowa	2
Dalekopis operatora	3

Przykłady

```
[MODBEG]
[RES ILOSC PLUS X]
[SEG 27]
[ADR 2 S 15]
[INT 3]
[LAB]
```

9. Parametry i teksty

Jednostka programu zwana parametrem posiada następującą składnię

⟨parametr⟩ ::=⟨parametr 1-bajtowy⟩|⟨parametr 2-bajtowy⟩

⟨parametr 1-bajtowy⟩ ::= PAR ⟨ogr⟩⟨wyrażenie⟩⟨ogr⟩B;

⟨parametr 2-bajtowy⟩ ::= PAR ⟨ogr⟩⟨wyrażenie⟩⟨ogr⟩B ⟨ogr⟩B;

⟨ogr⟩ ::= , | sp

Umieszczenie parametru w programie źródłowym powoduje wpisanie wartości wyrażenia występującego w definicji do jednej lub dwóch kolejnych komórek pamięci. Jeżeli przy definiowaniu parametru 1-bajtowego wartość występującego w niej wyrażenia przekracza 255, to do pamięci wpisze się resztę z dzielenia wartości wyrażenia przez 256.

Składnia tekstu jest następująca

⟨tekst⟩::= TXT ⟨ogr⟩⟨tekst bez ;⟩ ;

⟨tekst bez ;⟩ ::={⟨znak1⟩ ⟨ogr⟩ }

⟨znak1⟩::=⟨litera⟩|⟨cyfra⟩|!| %|&|<|:| "|#|.|'|←|/|(|)|[|]

⟨ogr⟩::= ,|sp

Każdy znak tekstu zostaje umieszczony w jednym słowie pamięci.

10. Komentarze

Komentarz jest jednostką programu ignorowaną przez translator. Posiada składnię

⟨komentarz⟩ ::= [COM ⟨ogr⟩⟨ciąg słów⟩]

⟨ciąg słów⟩ ::={⟨słowo z wyj.]⟩⟨ogr⟩ }

⟨słowo z wyj.]⟩- dowolne słowo nie zawierające znaku]

⟨ogr⟩::= ,|sp

Przykład

[COM POCZATEK PETLI GŁOWNEJ]

11. Sygnalizacja i poprawa błędów

Translator języka AWIK został zrealizowany jako translator skrośny z wykorzystaniem systemu programowania LISP 1.5 [2] oraz translatora uniwersalnego [3], który jest programem napisanym w LISP-ie. Stąd mechanizm sygnalizacji i poprawy błędów jest dwustopniowy. Błędy leksykalne, czyli błędy w słowach języka AWIK podlegają regułom poprawy obowiązującym dla tej realizacji LISP-u, którą się wykorzystuje. Natomiast błędy strukturalne poprawia się według zasad wynikających ze sposobu działania translatora uniwersalnego.

Błędy leksykalne

Omówimy obecnie mechanizm sygnalizacji i poprawy błędów leksykalnych, który istnieje w systemie LISP-u dla K202 [2].

Wystąpienie w słowie błędnego znaku lub znaku $ (na niektórych klawiaturach zastępuje go znak £) powoduje wykrycie błędu, przerwanie wprowadzania znaków z wejścia i przejście do stanu "oczekiwanie na poprawę błędu". Na urządzenie operatorskie wyprowadzany jest napis

CORRECT:

po którym zostaje wypisane błędne słowo, np.

CORRECT : AR$

i funkcja RATOM LISP-u pozostaje w stanie "oczekiwanie na poprawę błędu" aż do momentu napisania na urządzeniu operatorskim znaku : lub * .

Podczas poprawy błędów można odczytywać dalsze znaki z wejścia aktualnego tak, aby nie były one poddawane analizie. Napisanie jednego ze znaków cr, lf, sp powoduje odczytanie z wejścia aktualnego jednego znaku i wydrukowanie go na urządzeniu operatorskim. Znak ten nie jest analizowany. Czynność może być powtarzana wielokrotnie.

W stanie "oczekiwanie na poprawę błędu" wszystkie znaki poza cr,lf,sp,:,* są ignorowane. Napisanie na urządzeniu operatorskim znaku ignorowanego powoduje wydruk po nim znaku ? i powtórne przejście funkcji RATOM LISP-u ze stanu "oczekiwanie na poprawę błędu" do stanu "poprawa błędu". Natomiast napisanie znaku * przeprowadza ze stanu "oczekiwanie na poprawę błędu" do stanu "błąd".

W stanie "poprawa błędu" można poprawić błędne słowo przez napisanie na urządzeniu operatorskim słowa poprawnego, zakończonego znakiem cr,lf,sp. Można także kończyć to słowo innym ogranicznikiem, jednak należy pamiętać, że zostaje on zapisany w buforze, co może być przyczyną błędnego działania funkcji wejścia wykonanej później.

Jeżeli w trakcie poprawy błędu zostanie znów wykryty błąd, to ponownie przechodzi się do stanu "oczekiwanie na poprawę błędu" i zostaje wydrukowany napis CORRECT : <błędne słowo >

Przejście do stanu "błąd",czyli napisanie na urządzeniu operatorskim znaku *, oznacza rezygnację z dalszej poprawy błędów. Na dalekopisie pojawia się napis

ERROR 2

i przerywa się wykonanie funkcji RATOM LISP-u, a tym samym działanie translatora języka AWIK.

Przykłady

1. Załóżmy, że z wejścia aktualnego przeczytano sznur

   DAA$

   wtedy na urządzeniu operatorskim pojawi się wydruk

   CORRECT : DAA$

   Chcąc poprawić słowo, operator pisze dalej na urządzeniu operatorskim

   : DATA

   i funkcja RATOM wprowadza słowo DATA.

2. Jeżeli, w sytuacji omówionej w poprzednim punkcie, operator chce opuścić błędne słowo, to pisze na urządzeniu operatorskim

: ␣

3. Jeżeli w tej samej sytuacji operator chce odczytać kolejny znak z wejścia aktualnego, to pisze

␣

i kolejny znak zostaje wczytany i wydrukowany na urządzeniu operatorskim. Jeżeli operator chce go wprowadzić, to pisze : i powtarza znak odczytany.

Uwaga

Zbiór słów w systemie LISP-u dla K202 jest bogatszy od zbioru słów języka AWIK. Stąd niektóre błędne słowa w sensie języka AWIK mogą zostać zaakceptowane, jeśli są równocześnie poprawnymi słowami LISP-u. Dotyczy to przede wszystkim kropki jako ogranicznika, oraz liczb zmiennoprzecinkowych, które nie występują w AWIK-u.

Znak * jako znacznik końca taśmy

Jeśli ogranicznik przeczytany przez funkcję RATOM LISP-u jest znakiem * , to następuje przejście w stan "zawieszenie". Na urządzeniu operatorskim drukowany jest napis

TAPE

i oczekuje się, aby operator napisał jeden znak na tym urządzeniu. Jeśli tym znakiem będzie * , to nastąpi przerwanie wykonywania funkcji RATOM, wydruk napisu

ERROR 1

i przerwanie działania translatora języka AWIK. Napisanie dowolnego, innego znaku powoduje powtórne wywołanie funkcji RATOM.

Dzięki temu można wykorzystać znak * jako znacznik końca taśmy przy pisaniu informacji na taśmie papierowej. Znak * chroni przed wypadnięciem taśmy z czytnika w przypadku, gdy na taśmie umieszczono błędnie zbyt małą liczbę jednostek leksykalnych, lub gdy informacje przygotowano na kilku taśmach i każda z nich kończy się znakiem * . Przeczytanie tego znaku zawiesza wykonanie funkcji RATOM. Można wymienić taśmę w czytniku i napisać na urządzeniu operatorskim dowolny znak (poza znakiem *), co powoduje powtórne wywołanie funkcji RATOM.

Błędy strukturalne

Translator języka AWIK wykrywa błędy strukturalne posługując się mechanizmem opisanym szczegółowo w [3]. Najpierw na urządzenie operatorskie zostaje wyprowadzony napis informujący o charakterze błędu, a następnie pojawia się sekwencja słów uznana za błędną. Wszystkie słowa wypisane w tej sekwencji zostają równocześnie usunięte z tekstu tłumaczonego. Następnie na urządzenie operatorskie wyprowadza się napis CORRECT i oczekuje się na wprowadzenie słowa sterującego przez operatora. Operator ma do wyboru kilka możliwości:

RTP - po napisaniu RTP następuje wprowadzenie jednego słowa z wejścia poprzednio aktywnego i wyprowadzenie tego słowa na urządzenie operatorskie. Następuje powrót do stanu poprawy błędów, jaki istniał po wyprowadzeniu słowa CORRECT.

COR - przechodzi się do stanu poprawy błędów, w którym operator może wpisać sekwencję słów w tekst tłumaczony. Sekwencję tę należy zakończyć słowem FIN.

FIN - oznacza koniec sekwencji słów, które mają być wpisane w tekst tłumaczony.

STOP - przerwanie poprawy błędów i przerwanie działania translatora.

Inne słowa powodują powtórny wydruk napisu CORRECT i przejście do stanu, w którym oczekuje się na wprowadzenie słowa sterującego przez operatora.

W trakcie wprowadzania sekwencji słów w tekst tłumaczony słowa sterujące RTP, FIN i STOP powodują taki sam efekt, jak w stanie po wydruku napisu CORRECT.

Przykład

Jeśli w tekście źródłowym wystąpi błędna jednostka

(BEGIN] zamiast (BEGIN)

wtedy na urządzeniu operatorskim pojawi się komunikat

(WRONG END OF LABEL)

BEGIN]

CORRECT

Wypisanie słów BEGIN oraz] oznacza, że zostały one usunięte z tekstu źródłowego. Operator powinien zatem napisać

COR BEGIN) FIN

Translator zdefiniuje etykietę BEGIN i będzie kontynuował translację tekstu źródłowego.

12. Uruchomienie translatora języka AWIK

Należy założyć do czytnika taśmę binarną systemu LISP i napisać na dalekopisie zlecenie czytania taśmy binarnej:

RB*

a następnie zlecenie uruchomienia ostatnio wczytanego programu:

A *

Na dalekopisie pojawi się napis

LISP IS READY

Należy wcisnąć klawisz O na pulpisie operatorskim jednostki centralnej. Wtedy pojawi się napis

LISP1:

Ustala się wejście pisząc

INPUT(10)

System odpowie:

+10

Należy wówczas założyć do czytnika taśmę translatora języka AWIK i napisać

START ( )

Nastąpi czytanie taśmy, które zakończy się wydrukiem zapisu

SCANTAPE

LISP1:

Należy wówczas napisać ponownie

START ( )

i czekać, aż pojawi się wydruk

STOP

LISP1:

Wówczas uruchamia się program pomocniczy

UNITRANS ( )

Którego zakończenie sygnalizowane jest napisem

NIL

LISP1:

Należy wtedy ustalić wejście, z którego będzie wprowadzany program zapisany w języku AWIK oraz wyjście, na które będzie się wyprowadzać tekst będący rezultatem translacji. Jeśli są to urządzenia o numerach 1 i 2, wtedy pisze się

CSET (NOFINP 1)

CSET (NOFOUT 2)

a następnie zakłada się do urządzenia nr 1 taśmę z programem tłumaczonym i pisze się

TRANS1 ( )

Po zakończeniu pierwszego przebiegu translacji zostaje wyprowadzony napis

(SECOND PASS)

Translator czeka wtedy na ponowne podłożenie taśmy od początku. Drugi przebieg rozpocznie się po napisaniu na dalekopisie znaku cr lub lf.

Poprawne zakończenie translacji sygnalizowane jest napisem

(END OF TRANSLATION)

LISP1:

Bibliografia

[1] W.Wojciechowski, J.Bartoszek, J.Martinek, Organizacja logiczna komputera wirtualnego WIK, Instytut Automatyki P.P., Poznań, listopad 1975.

[2] Praca zbiorowa. System programowania minikomputera K202 w języku LISP 1.5, Instrukcja programisty, Instytut Automatyki P.P., Poznań, luty 1974.

[3] J.Martinek, Translator uniwersalny sterowany składnią, Instytut Automatyki P.P., Poznań, listopad 1975.

D o d a t e k A

Ze względu na to, że pewne znaki które mogą być drukowane przez urządzenia wyjścia istniejące w systemie MERA 300 nie występują w kodzie ISO, zestawiono je poniżej w tabeli ukazującej odpowiedniość znaków kodu ISO i znaków drukowanych przez maszynę FACIT oraz drukarkę DZM 180.

ISO	FACIT	DZM 180
!	§	!
' (apostrof)	m^2	' (apostrof)
&	◇	&
<	?	<
←	ć	–
(	ś	(
)	m^3	)
[	Ł	Ź
]	ę	Ń

D o d a t e k B

Dodatek ten zawiera program przykładowy, który zamienia liczbę binarną mieszczącą się w dwóch bajtach na dodatnią liczbę całkowitą i wyprowadza ją na maszynę do pisania.
Program jest tak napisany, że może być traktowany jako podprogram uaktywniony przez rozkaz "skocz ze śladem".
Liczba binarna która ma być zamieniona na dziesiętną powinna być umieszczona w miejscu o nazwie KROB1.
Kolejne cyfry liczby dziesiętnej umieszczone są w pamięci począwszy od miejsca o nazwie BUF i wyprowadzane na zewnątrz przez procesor zewnętrzny o nazwie 2.
Zera nieznaczące są również wyprowadzane.

```
[SEG 027]
(PZL)
 [RES 2]
 [COM USTAWIANIE WSKAZNIKA PETLI I ZAWARTOSCI MIEJSCA MOD ]
 ZA;
 SA WSKP;
 LA WSKAZ;
 SA MOD;
(ETC1)
 LA WSKP;
 ADS 1;
 SA WSKP;
 [COM SPRAWDZENIE CZY ZAKONCZONO PRZETWARZANIE LICZBY ]
 LAS 6;
 CO WSKP;
 JG;
 JPS ETC4;
 [COM ZEROWANIE MIEJSCA  O  NAZWIE WARTOSC ]
 ZA;
 SA WARTOSC;
 [COM OBLICZANIE KOLEJNYCH CYFR LICZBY DZIESIETNEJ ]
 LA I MOD;
(ETC2)
 CO KROB1;
 JG;
 JI;
 JPS ETC3;
 LX WARTOSC;
 XDS 1;
 SX WARTOSC;
 AD I MOD;
(ETC3)
 SU I MOD;
 NA;
 ADS 1;
 AD KROB1;
 SA KROB1;
 [COM UMIESZCZENIE W BUFORZE BUF KODU ISO CYFRY
      KTORA MA BYC WYDRUKOWANA ]
 LAS 48;
 AD WARTOSC;
```

```
 OPT CHA;
 LX WSKP;
 SA X KOM MINUS 1;
 [COM ZMIANA ZAWARTOSCI MIEJSCA MOD ]
 ZA;
 LAS 2;
 AD MOD;
 SA MOD;
 JPS ETC1;
(ETC4)
 LA CRLF;
 SA KOM PLUS 5;
 SME 2 KOM;
 WAN 2 KOM;
 JP I PZL;
(BUF)     [RES 10]
(WSKP)    [RES 2]
(MOD)     [RES 2]
(WARTOSC)[RES 2]
(KROB1)   [RES 2]
(CRLF)    PAR 06412 B B;
(WSKAZ)   PAR L10000 B B;
(L10000)  PAR 10000 B B;
(L1000)   PAR 1000 B B;
(L100)    PAR 100 B B;
(L10)     PAR 10 B B;
(L1)      PAR 1 B B;
[PROGEND]
```