BIURO PROJEKTÓW PRZEMYSŁU SYNTEZY CHEMICZNEJ
PRACOWNIA TECHNIKI OBLICZENIOWEJ
Gliwice

SYSTEM AUTOMATYCZNEGO KODOWANIA SAKO

GLIWICE Listopad 1972r.

SYSTEM AUTOMATYCZNEGO KODOWANIA SAKO

„Prosynchem" R/624-72-120/R-23

Autorzy opracowania: Barbara Jermakowicz
Aleksandra Kasprowicz
Franciszek Kern
Stefan Sawicki
Jerzy Swianiewicz
Tadeusz Szponarski

Redakcja: Krzysztof Nowak

BIBLIOTEKA NARODOWA BN
III 473.879

197300 5889/2
S

P R Z E D M O W A

Język SAKO został zaprojektowany w latach 1959-1960, a następnie wykonano jego translatory dla maszyn: XYZ,ZAM-2 i ZAM-21.

W ciągu minionych dziesięciu lat wykształcono liczną rzeszę ludzi posługujących się tym językiem, a wiele ośrodków posiada pokaźne zbiory programów napisanych w SAKO.

Aby udostępnić kolejną maszynę wyżej wymienionej grupie programistów oraz dla zachowania dorobku programowego, postanowiliśmy wykonać translator języka SAKO dla maszyny ZAM-41.

Ponadto przypuszczamy, że liczne zalety języka i translatora prostota języka, szybkość translacji, optymalność programów wynikowych pozwolą na pozyskanie nowych użytkowników.

W stosunku do pierwotnej wersji języka, zostały wprowadzone pewne zmiany. Konieczność wprowadzenia zmian wynikła z tego, że niektóre elementy języka były związane z własnościami maszyn, dla których wykonywano translatory.

Różnice między językiem opisanym w podręczniku L.Łukaszewicza i A.Mazurkiewicza "System Automatycznego KOdowania SAKO", a językiem SAKO dla maszyny ZAM-41 podane są w dodatku C.

Niniejszy podręcznik składa się z trzech części:

I Opis języka
II Sposób korzystania w systemie SO-141
III Dodatki.

W części pierwszej opisany jest język SAKO. Paragrafy dotyczące operacji na wielkościach oktalnych oraz paragrafy opisujące zdania związane z wielkościami oktalnymi są oznaczone gwiazdkami. Przy rozwiązywaniu zagadnień czysto numerycznych wiedza w nich zawarta jest zbędna.

Część druga opisuje aparat programowy dostarczany przez translator języka SAKO i System Operacyjny SO-141 maszyny

ZAM-41, który to aparat może być wykorzystany przy uruchamianiu, magazynowaniu i wykorzystywaniu programów napisanych w języku SAKO.

Materiał zawarty w dodatkach można podzielić na trzy kategorie:

1. Tablice stanowiące wyciągi z opisów poszczególnych elementów oprogramowania.
2. Rozszerzenie wiadomości zawartych w częściach I i II.
3. Wyciągi z informacji zawartych w częściach I i II.

Niniejszy podręcznik jest adresowany do czytelników posiadających podstawowe wiadomości o programowaniu maszyn cyfrowych.

Translator języka został wykonany przez zespół pracowników Instytutu Maszyn Matematycznych w składzie:

Danuta Kosecka
Maria Łącka
Ewa Pawlikowska-Kozłowska
Iwona Trzpil
Zdzisława Wrotek
Wiktor Kozłowski
Adam Rykowski
Stefan Sawicki
Jerzy Swianiewicz
Jan Walasek

i inni.

Spis treści

str.

Część I
OPIS JEZYKA 1

WPROWADZENIE 3

STRUKTURA PROGRAMU 5

Klasyfikacja znaków 5
Wiersz 5
Zdanie 5
Rozdział 7

ELEMENTY ZDAN 8

Zwroty 9
Liczby całkowite 10
Liczby ułamkowe 12
Liczby oktalne 14
Liczby oktalne krótkie 14
Liczby oktalne długie 16

Nazwa 20
Zmienna 20
Deklaracja typu 21
Blok 22
Deklaracja typu 23
Deklaracja wymiaru 23
Przydział miejsc pamięci 24
Indeksowanie 26
Tablica 27
Funkcja definiowana 27
Funkcja języka 28
Podprogram 29

Etykieta 30

Wyrażenia arytmetyczne 31
Pierwotne wyrażenie arytmetyczne 32

Działania arytmetyczne 35
Potęga 36

str.

Iloczyn 39
Iloraz 41
Różnica 43
Suma 44

ZDANIA JĘZYKA SAKO 48

Zdanie arytmetyczne przypisujące 49
Deklaracja CAŁKOWITE 52
Deklaracja BLOK 54
Rezerwacja pamięci maszyny dla bloków 56

TABLICA dziesiętna 60
Postać danych tablicy 61
Różnice między zdaniami TABLICA i BLOK 61

TABLICA OKTALNA 64
Postać danych zdania TABLICA OKTALNA 64

STRUKTURA 67
Zakres ważności zdania STRUKTURA 70

SKOCZ DO 70

GDY A B 73

GDY A = B 75

SKOCZ WEDLUG 77

POWTORZ 78
Wskaźnik zasięgu powtarzania 78
Całkowite POWTORZ 81
Ułamkowe POWTORZ 83

Zdanie przypisujące wielokrotnie 87

GDY BYŁ NADMIAR 91

GDY KLUCZ 93

STOP 95

Zdania wejścia-wyjścia 97
DRUKUJ 98
Drukuj całkowite 99
Drukuj ułamkowe stałoprzecinkowe 101
Drukuj zmiennoprzecinkowe 103

str.

SPACJA 106
LINIA 108
STRONA 110
TEKST 111
TEKST WIERSZY 113
DRUKUJ WIERSZ 115
CZYTAJ 117
Przygotowanie danych 118
CZYTAJ WIERSZ 122
CZYTAJ OKTALNIE 124
DRUKUJ OKTALNIE 126
Zdanie przypisujące logiczne 128
Wyrażenie logiczne 128
Pierwotne wyrażenie logiczne 129
Działania logiczne 130
Przesunięcie 130
Iloczyn logiczny 133
Negacja 135
Suma logiczna 136
Przypisywanie wartości w zdaniu przypisującym logicznym 138
PISZ NA BEBEN 140
Pamięć bębnowa 140
Rozmieszczenie danych na bębnie 141
CZYTAJ Z BEBNA 143
WSPOŁPRACA Z TASMAMI MAGNETYCZNYMI 145
OTWORZ 148
ZAPISZ 151
ZAMKNIJ 154
ZNAJDZ 155
POSUN 157
POBIERZ 158
ZMIEN 160
BIBLIOTEKA 162
PRZESTAW 163

Komentarze ..... str.
JEZYK SAS 165
Formuła operacyjna 166
Podstawianie argumentów 168
Podstawianie wyników 169
Liczenie wartości funkcji 171
PODSTAW 175
WROC 177
PODPROGRAM 179
Współpraca podprogramów i programu głównego ... 180
Przykłady podstawianie i przekazywania wyników. 183
Przypisanie dziesiętnej wartości dla funkcji 188
Przypisanie oktalnej wartości dla funkcji 195
Programy wielorozdziałowe 198
ROZDZIAL 199
IDZ DO ROZDZIALU 200
KONIEC 203
Zdania warunkowej translacji 207
Część II
KORZYSTANIE Z TRANSLATORA JEZYKA SAKO 209
UWAGI OGOLNE 211
CZOŁOWKA SYSTEMU 213
Zdanie TYTUL 216
Zdanie PROBLEM 217
Definicja wejść-wyjść 218
Nazwy urządzeń..... 218
Nazwy podprogramów 220
Nazwy dekoderów 220
Przykłady czołówek systemu 221
CZOŁOWKA TRANSLATORA 221
TRANSLACJA 224
CHOWAJ SLOWNIKI 226
POST MORTEM 227
..... 228

str.

URUCHAMIANIE 229

Zasady rozmieszczenia programu na bębnie 230

BEBEN 232
PAO 233
NAZWA 236
OPISZ PROGRAM 237
IGNORUJ BLEDY 238
KLUCZE 239
LIKWIDUJ STOP 241
NIWELUJ NADMIARY 243
WYMAGANY BEBEN 245
MINIMALIZUJ 246
BIBLIOTEKA 247
ZAPISZ 248
POBIERZ 251
KONIEC 255

Zasady monitowania 257

Komponowanie programów SAKO z wybranych rozdziałów taśmy bibliotecznej z programami wynikowymi 259

TASMA MAGNETYCZNA BIBLIOTECZNA 261
ZAWARTOSC 262
TRANSLACJA (w formie rozszerzonej) 264
ZAPISZ (w rozszerzonej postaci) 266
Uogólnienie zdania IDZ DO ROZDZIALU 268

SYSTEM MAGAZYNOWANIA I AKTUALIZACJI DOKUMENTOW 269
Zakładanie biblioteki 270
Przetwarzanie tekstów 273

POST MORTEM objaśnienie tekstów wypisywanych przez ZAKO 276
Przyczyny zakończenia 276
Post mortem rozdziału 279

Część III
DODATKI 283

DODATEK A
Tablica zależności między kodem KW6 i kodami zewnętrznymi 285

str.

Opis tablicy 288

DODATEK B

Opis funkcji języka 290

Sinus 290

Cosinus 293

Tangens 290

Arcus sinus 291

Arcus cosinus 291

Arcus tangens 292

Pierwiastek kwadratowy 292

Pierwiastek sześcienny 292

Logarytm naturalny 293

Exponent 293

Arcus 293

Maximum 294

Minimum 294

Modulo 295

Dzielenie całkowite 295

Signum 296

Wartość bezwzględna 296

Część całkowita 296

Suma 297

Iloczyn 298

Scharakteryzuj wykonanie zdania dotyczącego taśm magnetycznych 299

Podaj numer bloku taśmy magnetycznej 299

Adres 299

Ilość elementów bloku 301

Dołącz urządzenie symboliczne 301

Wybierz urządzenie symboliczne 303

Odłącz urządzenie symboliczne 304

Czytaj całkowite 305

Działania dwuprecyzyjne 306

Czytanie liczb dwuprecyzyjnych 307

Drukowanie liczb dwuprecyzyjnych 308

Dodawanie liczb dwuprecyzyjnych 312

str.

Odejmowanie liczb dwuprecyzyjnych 312
Mnożenie liczb dwuprecyzyjnych 313
Dzielenie liczb dwuprecyzyjnych 313
Zamiana liczby ułamkowej jednoprecyzyjnej na dwuprecyzyjną 313
Zamiana liczby dwuprecyzyjnej na liczbę ułamkową zwykłą 314
Spis funkcji języka 315

DODATEK C

Porównanie języków SAKO dla maszyny ZAM-2 i ZAM-41 317

DODATEK D

Spis zdań języka SAKO 320
Zdania przypisujące 320
Zdania sterujące 321
Deklaracje 322
Zdania współpracy z pamięciami pomocniczymi ... 322
Tablice i teksty 323
Struktura 324
Zdania opisujące podprogramy 324
Zdania opisujące rozdziały 325
Zdania opisujące wstawki 325
Komentarze 325
Zdania wejścia-wyjścia 325

DODATEK E

Spis nazw rozkazów dopuszczalnych we wstawkach języka SAS 327

DODATEK F

Wstawki w języku SAS .....
Uwagi wstępne 330
Ogólna budowa wstawek 331
Komentarze 331
Liczba dziesiętna 332
Liczba oktalna 333
Stała tekstowa 334
Dyrektywa REZE 336

BIBLIOTEKA NARODOWA BN

str.

Adres 337
Adres stały 338
Adres zmienny 340
Rozkaz 343
Makrorozkaz 345

Budowa podprogramów i sposób podstawiania argumentów 346

Biblioteka programów i podprogramów 349

DODATEK G

Błędy etapu translacji 369
Spis błędów 370

DODATEK H

Błędy etapu wykonania 378
Spis błędów 378

DODATEK I

Spis zdań czołówki translatora SAKO 383

DODATEK J

Spis numerów błędów sygnalizowanych przez translator języka JOM 384

DODATEK K

Spis numerów błędów sygnalizowanych przez translator POPR 386

DODATEK L

Spis numerów przyczyn zakończenia programu 388

Literatura 391

CZĘŚĆ I

O P I S J Ę Z Y K A

W P R O W A D Z E N I E

Język SAKO (System Automatycznego KOdowania) został przewidziany przede wszystkim do rozwiązywania zagadnień numerycznych.

Zdanie dla maszyny opisane w tym języku będziemy nazywać programem źródłowym. Program źródłowy składa się z elementów opisanych w niniejszym podręczniku, zwanych zdaniami języka SAKO(krócej zdaniami).

Program źródłowy jest tłumaczony na język maszyny przy pomocy programu zwanego translatorem. Proces tłumaczenia będziemy nazywać translacją. Wyniki translacji są następujące:

- program w języku maszyny, zwany programem wynikowym
- opis programu dostarczany systemowi operacyjnemu
- informacje dostarczane programiście, takie jak opis programu i sygnalizacje błędów formalnych.

Translator jest podporządkowany Systemowi Operacyjnemu (SO-141) maszyny ZAM-41 i tworzy programy wynikowe, które mogą współpracować tylko z tym systemem. Urządzenie, z którego translator pobiera program źródłowy i kod w jakim jest zapisany program, zależą od wysterowania systemu operacyjnego.

Informacje dostarczane programiście wyprowadzane są również na urządzenia i w kodach określonych w Systemie Operacyjnym.

Urządzenia wejściowe i wyjściowe oraz kody danych i wyników dla programów wynikowych powinny być też określone przez System Operacyjny.

Translator przystosowany jest do pobierania programu źródłowego w kodzie wewnętrznym sześciobitowym(KW6) i wyprowadza wyniki też w tym kodzie. Również standardowe funkcje obsługujące wejście i wyjście programów wynikowych zdolne są przetwarzać teksty zapisane w KW6. Zadaniem SO-141 jest przetransponowanie innych kodów na KW6 na wejściu i z KW6 na określone kody na wyjściu.

W niniejszym opisie mówiąc o kodzie lub o wartościach kodu (bez dodatkowych objaśnień), będziemy zawsze rozumieli kod KW6 opisany w DODATKU A.

Aby skrócić opis, zostały wprowadzone następujące umowne oznaczenia:

1. Ciąg znaków zawartych w nawiasach kwadratowych oznacza, że wystąpienie tego ciągu w danym miejscu nie jest obowiązkowe. Np.

   LINIA [:] 5

   oznacza, że powyższy napis może przyjąć jedną z dwóch postaci:

   LINIA 5

   lub

   LINIA : 5

2. Kolumna napisów zawarta w nawiasach sześciennych oznacza jeden z wierszy. Np.

   { LINIA 5
   { LINIA : 5
   { LINII 5

   oznacza

   LINIA 5

   albo

   LINIA : 5

   albo LINII 5

3. Przy opisie wyrażeń arytmetycznych i logicznych oraz przy określeniu ogólnej postaci zdań używane będą ciągi znaków zaczynających się od małej litery (czasami podkreślone). Tego typu identyfikatory nie są konkretnymi wielkościami występującymi w zdaniu lecz stanowią ogólny symbol, którego znaczenie jest objaśnione w tekście. Np.

   LINIA : <u>a</u>

S T R U K T U R A P R O G R A M U

Klasyfikacja znaków

Każdy znak, wchodzący w skład tekstu stanowiącego program źródłowy, jest rozpatrywany przez translator na jeden z trzech sposobów:

1. Znak istotny - podlega analizie
2. Znak nieistotny - jest pomijany
3. Znak niedozwolony - powoduje sygnalizację błędu

Mówiąc o znakach(bez dodatkowych objaśnień) będziemy mieli na myśli zawsze tylko znaki istotne. Powyższym regułom nie podlegają ciągi znaków będące parametrami zdań typu TEKST i TABLICA. Do tej sprawy powrócimy przy omawianiu powyższych zdań. Przynależność znaków do odpowiednich grup podana jest w DODATKU A. Np. spacja (SP) jest znakiem nieistotnym tj. teksty: LINIA i L I N I A z punktu widzenia translatora są równoważne.

Wiersz

Wierszem będziemy nazywać taki ciąg znaków, który wewnątrz nie zawiera znaku nowej linii (NL) i ostatnim jego znakiem jest znak nowej linii (NL) . W szczególnym przypadku pojedynczy znak (NL) stanowi wiersz. Wiersz programu nie może zawierać więcej niż 120 znaków istotnych(ilość znaków nieistotnych jest dowolna).

Wierszem pustym będziemy nazywać taki wiersz, który zawiera same znaki nieistotne z wyjątkiem znaku kończącego wiersz (NL); pozostałe wiersze będziemy nazywać wierszami niepustymi. Wiersze puste są przez translator pomijane i w dalszych rozważaniach mówiąc o wierszach, będziemy mieli na myśli tylko wiersze niepuste.

Zdanie

Ze względu na składnię, program składa się z ciągu zdań. Ogólnie rzecz biorąc zdania mogą opisywać pewien proces obliczeniowy, charakter i rozmieszczenie danych lub też strukturę i organizację programu.

Ze względu na ich przeznaczenie, zdania można podzielić na następujące klasy:

1. Deklaracje
   Te zdania pozwalają na rezerwację pamięci i/lub określenie typu identyfikatorów.
2. Przypisujące
   Przy pomocy tych zdań programista opisuje jakie i na jakich wielkościach mają być wykonane działania oraz wskazuje miejsca, gdzie mają być przesłane wyniki obliczeń.
3. Sterujące
   Zbiór tych zdań pozwala programiście sterować procesem obliczeniowym w zależności od wyników otrzymanych w tym procesie.
4. Współpracy z pamięciami pomocniczymi
   Zdania z tej grupy pozwalają na przesyłanie bloków danych między pamięcią operacyjną maszyny a bębnową oraz między pamięciami taśmowymi i pamięcią operacyjną.
5. Tablice i teksty
   Zdania te pozwalają na wprowadzenie do programu wynikowego bloków stałych liczbowych lub tekstów.
6. Struktura
   Omawiane zdanie pozwala programiście zmienić dynamicznie strukturę zmiennych.
7. Zdania opisujące podprogramy
   Ten zbiór zdań pozwala na definiowanie podprogramów napisanych w języku SAKO.
8. Zdania opisujące rozdziały
   Pozwalają one na dzielenie programu na części zwane rozdziałami oraz na przekazywanie sterowania z jednego rozdziału do innego.
9. Zdania opisujące wstawki
   Informują translator, że fragmenty programu są napisane w języku SAS lub też, że część programu znajduje się na wyspecyfikowanej taśmie magnetycznej.
10. Komentarze
    Pomijane są przez translator, mogą objaśniać pewne kwestie związane z programem lub też służą jako informacje dla innych programów opracowujących tekst źródłowy.

Każde zdanie języka SAKO musi zajmować całkowitą ilość wierszy, a ze względu na ilość zajmowanych wierszy można podzielić je na trzy grupy:

a/ zajmujące jeden wiersz,

b/ takie, których listy parametrów mogą być przenoszone do następnych wierszy,

c/ zajmujące z założenia więcej niż jeden wiersz.

Przy opisie każdego zdania z grupy b będzie powiedziane, że listę można przenosić. Translator traktuje następny wiersz programu jako dalszy ciąg zdania poprzedniego, jeżeli trzy znaki: ,- i nowa linia (NL) kończą poprzedni wiersz. Brak w/w ciągu oznacza, że przeniesienia nie będzie. Do grupy c należą zdania klasy 5 tj. tablice i teksty.

Zdania języka SAKO można też sklasyfikować następująco:

1. Zdania wykonujące się
   Z każdego z tych zdań translator tworzy część programu wynikowego.
2. Zdania opisujące
   Zdania te dostarczają informacji translatorowi i nie są przetwarzane na części programu wynikowego.
3. Zdania wykonujące się i opisujące
   Posiadają one własności obu powyższych grup.

Rozdział

Program może być podzielony na niezależne części przy pomocy zdań ROZDZIAL. Każda z tych części tłumaczona jest przez translator na niezależny segment programu wynikowego zwany rozdziałem.

W czasie wykonywania programu wynikowego w pamięci maszyny znajduje się tylko jeden rozdział. Każdy z rozdziałów składa się z programu głównego oraz z ewentualnych podprogramów(w szczególnym przypadku program nie podzielony na rozdziały można uważać, że składa się z jednego rozdziału).

Podprogramy (jeżeli są) muszą występować w programie głównym; każdy z rozdziałów posiada własne podprogramy.

Ostatnim zdaniem programu musi być zdanie KONIEC.

E L E M E N T Y Z D A Ń

Pewne wielkości lub pojęcia mogą występować w różnych zdaniach i mają te same znaczenia.

Niezależne opisanie tych wielkości i pojęć zwanych elementami zdań, pozwoli przy opisie poszczególnych zdań na skoncentrowanie się na istotnych własnościach bez zbędnych dygresji.

Elementami zdań są: zwroty, stałe, identyfikatory, etykiety, wyrażenia arytmetyczne i wyrażenia logiczne.

Z W R O T Y

Definicja
Zwrot - jest to ściśle określony ciąg liter, który określa typ zdania i pozwala je wyróżnić spośród innych.

Zwrot, jeżeli zdanie go posiada, występuje zawsze na początku.

Dla skrócenia opisu, gdy wypisywanie pozostałych części zdania jest zbędne, zamiast całego zdania będziemy pisać tylko jego zwrot.

Przykłady zwrotów:

CZYTAJ
DKOCZ DO
SKOCZDO (równoważny powyższemu gdyż spacja jest znakiem nieistotnym)
PODPROGRAM
GDY

L I C Z B Y C A Ł K O W I T E

Definicja

Liczba całkowita - jest to ciąg cyfr dziesiętnych. Ciąg powinien zawierać co najmniej jedną cyfrę.

Zakres liczby: od 0 do 8 388 607 włącznie tj od 0 do $2^{23}-1$.

Liczba całkowita może być bez znaku lub też może być poprzedzona znakiem plus albo minus.

Liczbę całkowitą poprzedzoną znakiem minus (-) będziemy nazywać liczbą całkowitą ujemną, zaś liczbę całkowitą nie poprzedzoną żadnym znakiem lub poprzedzoną znakiem plus (+) będziemy nazywać liczbą dodatnią.

Liczba całkowita ujemna lub dodatnia umieszczona w pamięci maszyny, zajmuje jedno 24-bitowe słowo (słowo krótkie) .

Sposób przedstawienia liczby całkowitej w słowie maszyny będzie podany przy opisie stałych oktalnych krótkich.

Przykłady:

Poprawne liczby całkowite dodatnie:

0 (zero dodatnie)

91

8 388 607 (spacje nie są w zdaniach istotne, jest to maksymalna liczba całkowita dodatnia)

+ 0

+ 025 (jest to liczba równa 25)

00000000025 (jak wyżej)

Poprawne liczby całkowite ujemne:

-0 (zero ujemne)
-91
-8 388 607 (minimalna liczba całkowita)
-0015 (równoważna - 15)

Błędne liczby całkowite

25. (zawiera kropkę)
.123 (jak wyżej)
2,5 (zawiera przecinek)
8 388 608 (przekracza zakres)
-8 388 608 (również przekracza zakres)

L I C Z B Y U Ł A M K O W E

Definicja

Liczba ułamkowa - jest to stała, która może być zapisana w jednej z trzech niżej wymienionych postaci:
- jako liczba ułamkowa zwykła,
- jako liczba ułamkowa zwykła z dopisanym po niej wykładnikiem dziesiętnym,
- jako liczba całkowita z dopisanym po niej wykładnikiem dziesiętnym.

Liczba ułamkowa zwykła jest ciągiem cyfr dziesiętnych zawierającym przynajmniej jedną cyfrę oraz dokładnie jedną kropkę.
Wykładnik dziesiętny zaczyna się od litery E, po której musi występować dodatnia lub ujemna liczba całkowita.

Zakres liczby ułamkowej

Liczba ułamkowa może przyjmować wartości z zakresu od 2^{-256} do $(1-2^{-38}) \times 2^{255}$ włącznie lub wartość 0 ułamkowe.
W przybliżeniu jest to zakres od 0.8636 E-77 do 0.5789 E 77

Liczba ułamkowa może być poprzedzona znakiem plus (+) albo znakiem minus (-) lub może wystąpić bez znaku. Liczbę ułamkową poprzedzoną znakiem minus będziemy nazywać liczbą ułamkową ujemną, a liczbę ułamkową poprzedzoną znakiem plus lub występującą bez znaku będziemy nazywać liczbą ułamkową dodatnią.

Wykładnik dziesiętny określa potęgę dziesięciu, przez którą należy pomnożyć poprzedzającą go liczbę całkowitą lub też zwykłą liczbę ułamkową.
Liczba ułamkowa dodatnia albo ujemna umieszczona w pamięci maszyny zajmuje dwa kolejne słowa (stanowi ciąg 48 bitów). Kolejne dwa słowa maszyny będziemy czasami nazywać słowem długim.
Sposób przedstawienia liczby ułamkowej w długim słowie maszyny będzie podany przy opisie liczby oktalnej długiej.

Przykłady

Poprawne liczby ułamkowe

+0.	(zero ułamkowe dodatnie)
-0.	(zero ułamkowe ujemne)
19876.01	(liczba ułamkowa dodatnia)
.1234	
-.123 E-2	(jest równoważne: -0.123×10^{-2} tj. -0.00123)
7.0E+0	(tj. $7.0 \times 10^{0} = 7.0$)
+7.0E3 7.0E+3 7.0E+003 7E+3	tj. $7.0 \times 10^{3} = 7000.$

Błędne liczby ułamkowe:

1	(brak kropki to jest liczba całkowita)
3,14	(zawiera przecinek zamiast kropki)
1.2 E 125	(przekracza zakres)
.1 E-100	(przekracza zakres, nie jest równa zero)
.E2	(brak cyfr przed E)
+1.E	(brak liczby całkowitej po E)

*)L I C Z B Y O K T A L N E

Liczby oktalne służą do definiowania ciągów zero-jedynkowych 24 lub 48 bitowych.
Liczby oktalne definiujące ciągi 24 bitowe będziemy nazywać liczbami oktalnymi krótkimi.
Liczby oktalne definiujące ciągi 48 bitowe, będziemy nazywać liczbami oktalnymi długimi.
Ciąg 24 bitów będziemy nazywać krótkim słowem maszyny albo po prostu słowem maszyny zaś ciąg 48 bitów będziemy nazywać długim słowem maszyny.

Liczby oktalne krótkie

Definicja
Liczba oktalna krótka - jest to ciąg od 1 do 8 cyfr oktalnych, które to cyfry mogą być oddzielone od siebie kropkami. Do zbioru cyfr oktalnych należą następujące znaki: 0,1,2,...,7.

Kropki w ciągu cyfr służą tylko do oddzielania grup cyfr i są pomijane przez translator. Kropki nie mogą występować przed pierwszą cyfrą ciągu i po ostatniej.
Translator przetwarza liczbę oktalną krótką na słowo maszyny w sposób następujący:

1. Jeżeli liczba zawiera mniej niż 8 cyfr oktalnych, to uzupełniana jest z przodu (do ośmiu) cyframi zero.

2. Każda cyfra oktalna zastępowana jest ciągiem trzech bitów według niżej podanego przyporządkowania:

0 -	000	4 -	100
1 -	001	5 -	101
2 -	010	6 -	110
3 -	011	7 -	111

W powyższy sposób otrzymanemu ciągowi bitów, poczynając od lewej strony, przyporządkujemy kolejne numery poczynając od zera (tj. pierwszej cyfrze oktalnej będą odpowiadały bity o numerach 0,1 i 2, drugiej bity o numerach 3,4 i 5, ósmej zaś bity o numerach 21,22,23 (patrz rys.1)). Liczba całkowita dziesiętna zamieniana jest na słowo maszyny w sposób następujący:

Jeżeli liczba jest ujemna to bit zerowy słowa przyjmuje wartość 1. Jeżeli liczba jest dodatnia to bit zerowy słowa przyjmuje wartość 0. Wartość bezwzględna liczby przeliczana jest na postać binarną i umieszczona tak, aby pozycja jedności przypadła na 23 pozycję słowa (patrz rys.2).

Przykłady

<u>Poprawne</u> liczby oktalne krótkie:

37.1.2 (definiuje następujące słowo: 000 000 000 000 011 111 001 010)

12.34.56.70

4.000 000 0 (definiuje ciąg: 100 000 000 000 000 000 000 000)

<u>Błędne</u> liczby oktalne krótkie:

37.12. (zawiera kropkę na końcu ciągu)

123456701 (składa się z 9 cyfr)

1234581 (zawiera cyfrę 8)

W poniższych przykładach liczby wypisane w jednym wierszu (pierwsza liczba jest oktalna, a druga dziesiętna) tworzą identyczne słowa maszyny:

0.7	+7
0	+0
400.000.00	-0

Liczby oktalne długie

Definicja
Liczba oktalna długa - jest to ciąg od 9 do 16 cyfr oktalnych. Między cyframi mogą występować kropki.

Kropki w liczbie oktalnej długiej są pomijane przez translator. Wewnątrz liczby mogą występować w dowolnych miejscach, lecz niedozwolone jest stawianie kropek przed pierwszą cyfrą i za ostatnią.

Translator przetwarza liczbę oktalną długą na jedno słowo długie (dwa kolejne słowa krótkie) według schematu:

1. Jeżeli ciąg składa się z mniej niż szesnastu cyfr oktalnych to uzupełnia go z przodu (do szesnastu) cyframi zero.
2. Każda z cyfr oktalnych zastępowana jest odpowiadającymi jej trzema bitami.

Bity słowa długiego będziemy numerować na dwa sposoby:

1. Traktując słowo długie jako dwa krótkie.
2. Traktując słowo długie jako jedną całość; pozycje pierwszego słowa krótkiego otrzymają numery od zera do 23, a pozycje następnego słowa krótkiego otrzymają odpowiednio numery od 24 do 47 (patrz rys.3) .

Liczba ułamkowa jest sprowadzana przez translator najpierw do postaci:

$$m \times 2^c$$

gdzie: $1/2 \leqslant |m| < 1$ oraz $0 \leqslant |c| < 256$

Następnie bezwzględna wartość liczby m przeliczana jest z postaci dziesiętnej na liczbę ułamkową binarną składającą się z 38 bitów; bezwzględną wartość liczby c translator przedstawia w postaci liczby całkowitej binarnej.

Tak otrzymane elementy liczby ułamkowej rozmieszczone są w słowie długim następująco:
W pozycji zero pierwszego słowa krótkiego umieszczana jest jedynka gdy liczba jest ujemna albo zero gdy liczba jest dodatnia, w pozycjach 1-23 pierwszego słowa krótkiego i w pozycjach 1-15 drugiego słowa krótkiego umieszczane są kolejne bity liczby m.
W pozycjach 16-23 drugiego słowa umieszczana jest bezwzględna wartość liczby c tak, aby jej pozycja jedności odpowiadała 23 pozycji słowa, znak liczby c (1 - minus, 0 - plus) umieszczany jest w zerowej pozycji słowa drugiego (patrz rys.4) .
Zero ułamkowe transponowane jest przez translator do następującej postaci:

pierwsze słowo zawiera same zera zaś w słowie drugim umieszczona jest liczba całkowita - 255 (patrz rys.5) .

Przykłady

Poprawne liczby oktalne długie:

123456701
76.54.32.10.12.34.56.70
12345670123

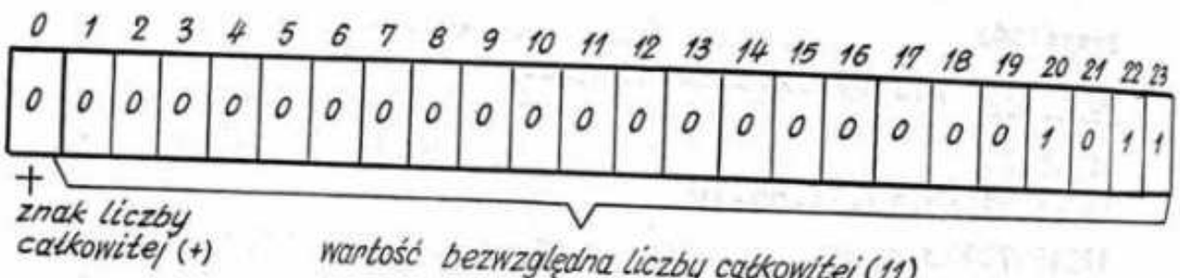
Błędne liczby oktalne długie:

12345670 (zawiera 8 cyfr)
123456789 (zawiera cyfry 8 i 9)
.12.34.56.70.12. (zawiera kropki na początku i na końcu)

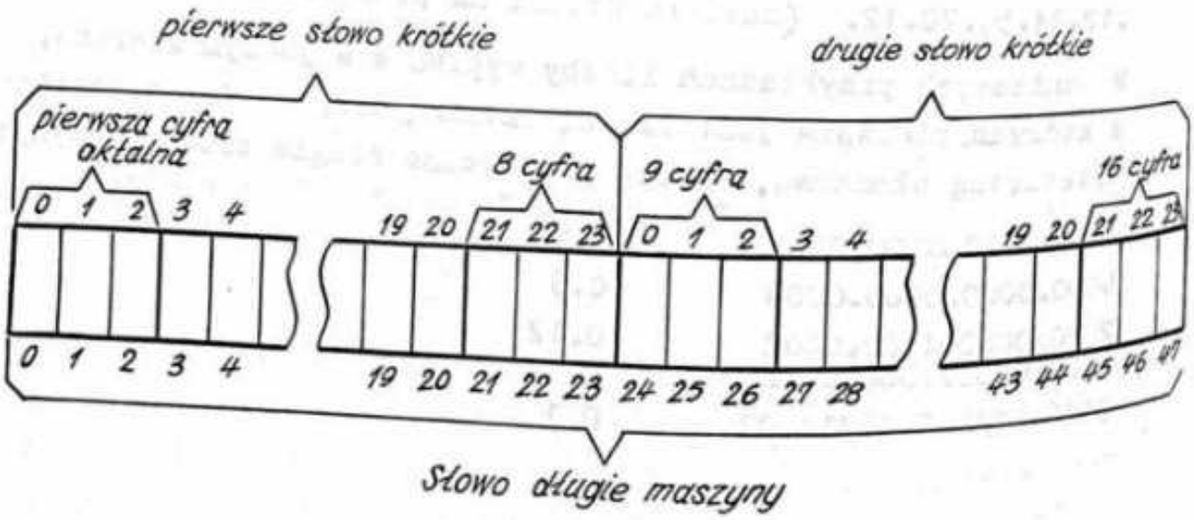
W poniższych przykładach liczby wypisane w jednym wierszu, z których pierwsza jest liczbą oktalną długą a druga liczbą dziesiętną ułamkową, tworzą identyczne długie słowa maszyny:

2000.0000.0000.0000	0.5
6000.0000.0000.0000	-0.5
2000.0000.4000.0002	0.125
2000.0000.0000.0002	2.
3146.3146.5.46314.03	0.1

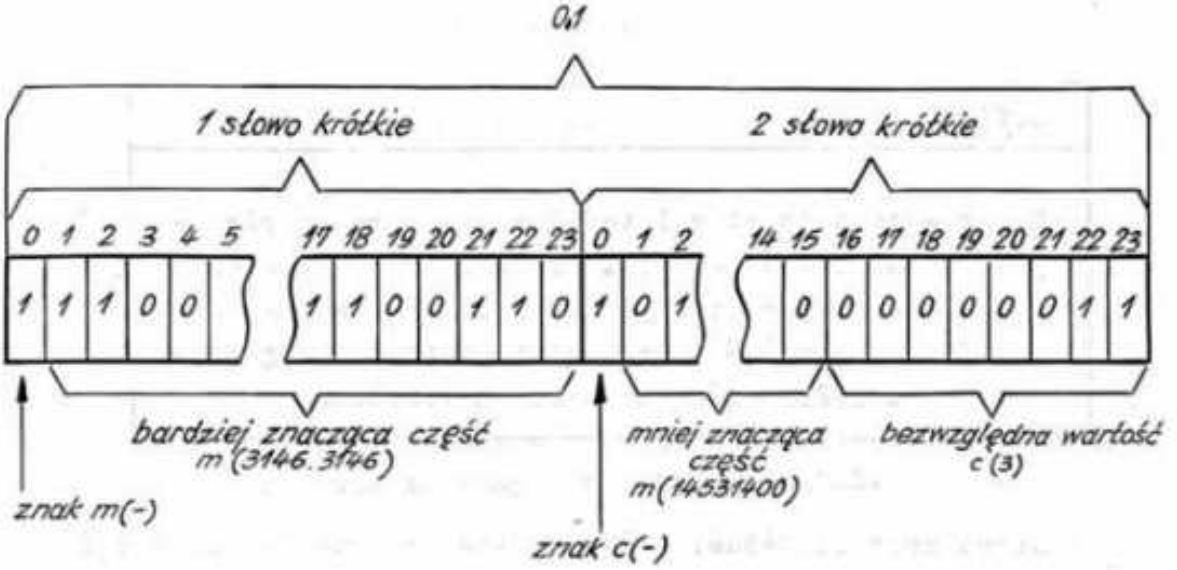
rys. 1



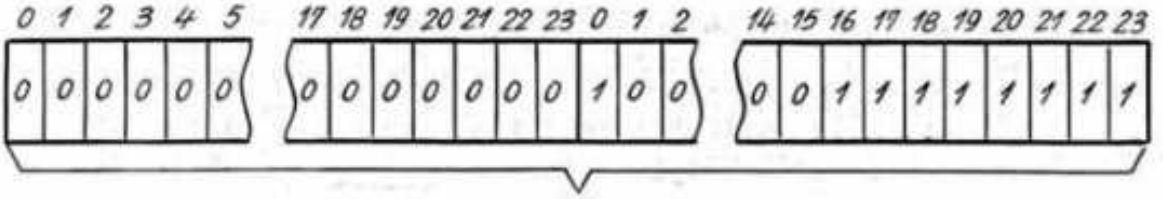
rys. 2



rys. 3


0,1
1 słowo krótkie
2 słowo krótkie
0 1 2 3 4 5 17 18 19 20 21 22 23 0 1 2 14 15 16 17 18 19 20 21 22 23
1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 1
bardziej znacząca część m (3146.3146)
mniej znacząca część m(14531400)
bezwzględna wartość c (3)
znak m(-)
znak c(-)

rys. 4


0 1 2 3 4 5 17 18 19 20 21 22 23 0 1 2 14 15 16 17 18 19 20 21 22 23
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1
+0.

rys.5

N A Z W A

Definicja
Nazwa - jest to ciąg liter i cyfr, którego pierwszym znakiem jest litera. Ciąg powinien zawierać co najmniej jeden znak. Maksymalna długość nazwy zależy tylko od tego czy zmieści się ona w jednym wierszu.

Nazwa może oznaczać:

- zmienną prostą
- blok lub zmienną indeksowaną
- tablicę
- funkcję definiowaną
- podprogram
- funkcję języka

Jeżeli nazwa oznacza zmienną prostą, zmienną indeksowaną lub blok albo tablicę i jeżeli składa się z więcej niż czterech znaków to tylko pierwsze cztery z nich będą brane pod uwagę, a pozostałe zostaną pominięte.
Jeżeli nazwa oznacza funkcję definiowaną, podprogram lub funkcję języka i składa się z więcej niż trzech znaków to tylko pierwsze trzy będą brane pod uwagę.
Nazwy w tym samym zakresie ważności, powinny być różne.

Uwaga: zwroty SAKO-wskie nie są słowami zastrzeżonymi.

Zmienna

Zmienna jest symbolem reprezentującym wartość znajdującą się w ustalonym miejscu pamięci maszyny. Wartość reprezentowana przez zmienną jest zawsze bieżącą wartością miejsca pamięci, która to zawartość może w procesie obliczeniowym być zmieniana (może się zmienić, jest zmienną).

Postać symbolu będącego zmienną zależy od typu zmiennej. Rozróżniamy dwa typy zmiennych:

- zmienne proste
- zmienne indeksowane

Deklaracja typu zmiennej

W języku SAKO występują dwa typy zmiennych prostych, a mianowicie:

- zmienne proste całkowite, które będziemy czasami nazywać zmiennymi prostymi krótkimi,
- zmienne proste ułamkowe, które niekiedy będziemy nazywać zmiennymi prostymi długimi.

Typ zmiennej prostej związany jest z wielkością, jaką sobą reprezentuje; i tak:

- zmienna prosta całkowita reprezentuje sobą liczbę całkowitą lub liczbę oktalną krótką,
- zmienna prosta ułamkowa reprezentuje sobą liczbę ułamkową lub liczbę oktalną długą.

Jeżeli zmienna prosta jest wymieniona na liście parametrów zdania CALKOWITE to będzie traktowana jako całkowita. Pozostałe zmienne proste występujące w programie będą zmiennymi ułamkowymi (zmiennych ułamkowych nie trzeba deklarować).

Przykłady

Poprawne zmienne proste:

A

ALFA

ALFABET (jest równoważna zmiennej ALFA gdyż identyfikowane są tylko pierwsze 4 znaki)

ALF A (równoważna zmiennej ALFA gdyż spacja jest nieistotna)

A1 7 B

A1

SUMA

Błędne zmienne proste:

A.B (zawiera kropkę)
8AB (zaczyna się od cyfry)

Blok

Blok, w sensie języka SAKO, jest to zbiór zmiennych posiadający wspólną nazwę. Aby odwołać się do konkretnej zmiennej w bloku możemy wskazać jej względne położenie w nim (np.: zerowa, pierwsza, druga, piąta).
Rozważmy blok o nazwie BLO, który składa się z pięciu zmiennych o wartościach: 2,7,9,0,3.

Zatem

BLO (0) ma wartość liczbową równą 2
BLO (1) - " - 7
BLO (2) - " - 9
BLO (3) - " - 0
BLO (4) - " - 3

Każda zmienna (element bloku) w powyższym bloku składa się z nazwy tego bloku po której występuje liczba całkowita zawarta w okrągłych nawiasach.
Wielkość zawartą w nawiasach będziemy nazywać indeksem zmiennej BLO.
Dla odrążnienia zmiennej z indeksem od zmiennej prostej, tę pierwszą będziemy nazywać zmienną indeksowaną. Będziemy np. mówić, że zmienna indeksowana BLO (0) ma wartość 2, zmienna indeksowana BLO (1) ma wartość 7 itd.
Zmienna indeksowana BLO (I) określa I-tą zmienną indeksowaną, gdzie I jest zmienną całkowitą, która może przyjmować wartości:
0,1,2,3 lub 4.
Rozważmy jeszcze jeden blok o nazwie MAC1 opisany przez dwa indeksy. Niech pierwszy przyjmuje wartości od 0 do 2, a drugi od 0 do 3:

	kolumna 0	kolumna 1	kolumna 2	kolumna 3
wiersz 0	2	7	25	14
wiersz 1	3	5	1	9
wiersz 2	4	8	12	27

Indeks pierwszy wskazuje numer wiersza, a indeks drugi numer kolumny.
Załóżmy teraz, że chcemy wskazać liczbę z wiersza o numerze 1 i z kolumny o numerze 3. Wtedy piszemy:

MAC 1 (1,3)

Zmienna indeksowana MAC1 (1,3) posiada wartość 9.
Normalny zapis matematyczny $MAC1_{ij}$ wskazujący na dowolny element macierzy MAC1 możemy w języku SAKO przedstawić w postaci MAC1 (I,J) , gdzie I może przyjmować wartości: 0,1 lub 2, zaś J może przyjmować wartości równe: 0,1,2 lub 3.
Zmienne indeksowane posiadające jedną współrzędną będziemy nazywać jednoindeksowymi, zmienne indeksowane posiadające dwie współrzędne wiersz i kolumnę będziemy nazywać dwuindeksowymi itd. Z praktycznego punktu widzenia ilość indeksów jest nieograniczona.

Deklaracja typu bloku

Tak samo jak zmienne proste również zmienne indeksowane mogą być typu ułamkowego lub całkowitego. Jednakże w bloku wszystkie zmienne muszą być tego samego typu tj. albo wszystkie muszą być całkowite albo też wszystkie muszą być ułamkowe. Aby zadeklarować, że blok jest całkowity, należy jego nazwę wymienić na liście parametrów zdania CALKOWITE. Jeżeli blok nie jest zadeklarowany jako całkowity to należy uważać, że jest ułamkowy.

Deklaracja wymiaru

Deklaracja wymiaru określa następujące własności bloku:

- zakresy i wymiary indeksów
- ilość indeksów bloku
- rozmiar bloku.

Zakres indeksu w bloku jest to maksymalna wartość jaką może przyjąć dany indeks. Dolna granica indeksu jest zawsze zerem. Wymiar indeksu jest to liczba wskazująca dopuszczalną ilość różnych wartości dla danego indeksu i jest równa zakresowi indeksu plus 1.

Podając zakres każdego z indeksów określamy w ten sposób ilość indeksów w bloku.
Czasami zamiast mówić, że blok jest n indeksowy będziemy mówić, że jest n wymiarowy.
Rozmiar bloku jest to liczba określająca ilość elementów bloku i równa jest iloczynowi wymiarów indeksów.
Omawiane parametry bloku wprowadzane są przy pomocy zdania BLOK.

Przydział miejsc pamięci dla bloku

Każdemu elementowi bloku przydziela się jedno miejsce pamięci gdy blok jest całkowity i dwa kolejne miejsca gdy blok jest ułamkowy. Obszar pamięci przydzielany dla bloku jest zwarty.
Dla bloków jednoindeksowych, zmiennym indeksowanym z kolejnymi wartościami indeksów, odpowiadają kolejne miejsca pamięci i z większymi wartościami indeksów związane są pamięci o większych adresach.
Weźmy blok czteroelementowy A. Jego elementy będą rozłożone w pamięci następująco:

A (0) , A (1) , A (2) , A (3)

Jednoindeksowy blok A posiada indeks o zakresie 3, wymiarze 4 i rozmiar bloku jest 4.
Dwuindeksowy blok B, którego pierwszy indeks zmienia się od 0 do 2 zaś drugi od 0 do 4 będzie zajmował w pamięci 15 (3x5) długich lub krótkich miejsc (długość zależy od deklaracji CALKOWITE) , a jego elementy będą rozmieszczone w kolejnych miejscach pamięci następująco:

B (0,0) B (0,1) B(0,2) B(0,3) B(0,4)
B (1,0) B (1,1) B(1,2) B(1,3) B(1,4)
B (2,0) B (2,1) B(2,2) B(2,3) B(2,4)

Elementowi B(0,0) odpowiada miejsce o najmniejszym względnym adresie pamięci a elementowi B(2,4) odpowiada największy adres pamięciowy.

Pnadto należy zauważyć, że bezpośrednio po elemencie B (0,4) występuje w pamięci element B (1,0) i odpowiednio po elemencie B (1,4) element B (2,0) . Macierz dwuwymiarowa umieszczona jest w pamięci kolejno wierszami.
W przypadku macierzy trójindeksowej C o zakresach indeksów: 1,2 i 3 elementy będą w pamięci rozłożone następująco:

C(0,0,0) C(0,0,1) C(0,0,2) C(0,0,3)
C(0,1,0) C(0,1,1) C(0,1,2) C(0,1,3)
C(0,2,0) C(0,2,1) C(0,2,2) C(0,2,3)
C(1,0,0) C(1,0,1) C(1,0,2) C(1,0,3)
C(1,1,0) C(1,1,1) C(1,1,2) C(1,1,3)
C(1,2,0) C(1,2,1) C(1,2,2) C(1,2,3)

Należy zauważyć, że bezpośrednio po elemencie C(0,0,3) występuje w pamięci element C(0,1,0) i odpowiednio po elemencie C(0,2,3) element C(1,0,0) . Powyższa macierz zawiera 24 elementy (2 x 3 x 4) .
Ogólnie rzecz biorąc w bloku A o n indeksach, których wymiary są odpowiednio:

$$d_1, d_2, \ldots, d_n$$

dowolny element A(I1, I2,...,In) ma względne położenie względem początku bloku ustalone według wzoru:

$$k = (\ldots ((I1 \times d_2 + I2) \times d_3 + I3) \times d_4 + \ldots) \times d_n + In \quad (1)$$

Znajomość kolejności rozmieszczenia elementów w bloku jest ważna w przypadku przygotowywania danych, które są wprowadzane liniowo.
Ilość elementów w bloku jest obliczana wg wzoru:

$$W = \prod_{i=1}^{n} d_i$$

Indeksowanie

Do tej pory podawaliśmy przykłady zmiennych indeksowanych, które miały indeksy w postaci liczb całkowitych lub zmiennych prostych całkowitych. Ogólnie, indeksem może być wyrażenie arytmetyczne o wartości całkowitej (patrz wyrażenie arytmetyczne). Indeksy oddziela się od siebie przecinkami. Ilość indeksów opisujących zmienną powinna być nie większa niż ilość indeksów zadeklarowana przy opisie bloku. Ponieważ względne położenie elementu wewnątrz bloku jest obliczane według wzoru 1, to w ogólnym przypadku opuszczenie indeksu zmiennej indeksowanej nie jest równoważne zerowej wartości tego indeksu.
Jeżeli blok nie jest formalnym blokiem w podprogramie (patrz zdanie PODPROGRAM) to przy pisaniu jego elementu, gdy wszystkie indeksy są zerami, można napisać nazwę bloku bez indeksów.
Np. zamiast pisać C(0,0,0) można napisać po prostu C.
Skrócony zapis nie tylko daje oszczędności w pisaniu, ale także upraszcza program wynikowy.
Z powyższego wynika, że zmienna prosta o tej samej nazwie co blok będzie utażsamiana z pierwszym elementem odpowiedniego bloku.

Przykłady

Poprawne zmienne indeksowane:

C (element równoważny elementowi C(0,0,0))
A (ALA) (zmienna ALA musi być całkowita)
RAK(I,J+7) (I i J są całkowite)

Błędne zmienne indeksowane:

A(1.3) (indeks jest ułamkowy)
A(1-7) (indeks jest mniejszy od zera).

Tablica

Tablica jest to blok, któremu w programie źródłowym przyporządkowano wartości początkowe. Nazwa tablicy nie może być taka sama jak nazwa bloku. Wymiary tablicy opisywane są przy pomocy zdania TABLICA lub TABLICA OKTALNA.
Zmienne indeksowane i elementy tablic mają identyczną postać zewnętrzną i takie samo znaczenie. Opis bloku i jego elementów w pełni stosuje się do tablic i ich elementów. Stąd w dalszej części opisu mówiąc o blokach i zmiennych indeksowanych, będziemy mieli na myśli również tablice i ich elementy.
Ponadto w dalszym opisie będziemy posługiwać się pojęciem zmiennej, mając na myśli zmienną prostą lub zmienną indeksowaną lub element tablicy; oczywiście tylko wtedy, gdy nie ma potrzeby wyróżniać któregokolwiek z powyższych pojęć.

Funkcja definiowana

Funkcja definiowana jest to nazwa, różna od nazwy bloku lub nazwy funkcji języka, po której występuje lista argumentów rozdzielonych od siebie przecinkami (lub jedno wyrażenie lub kropka) i zawarta w nawiasach okrągłych.
Argumentami funkcji definiowanej mogą być:

- wyrażenia arytmetyczne
- nazwy bloków
- nazwy funkcji (definiowanych i języka).

Nazwę występującą przed nawiasem będziemy nazywać nazwą funkcji definiowanej.
Wartością funkcji definiowanej jest liczba całkowita lub ułamkowa otrzymana przez wykonanie podprogramu definiującego daną funkcję.
Typ funkcji (ułamkowy czy całkowity) zależy od tego, czy nazwa funkcji jest parametrem zdania CALKOWITE.
Podprogram definiujący funkcję musi być częścią programu źródłowego. Przy opisywaniu podprogramu podaje się nazwę funkcji, ilość oraz typ i rodzaje argumentów oraz pisze się ciąg zdań, które należy wykonać, aby obliczyć wartość funkcji.

Dokładny opis funkcji i sposób definiowania podany będzie przy omawianiu zdania PODPROGRAM. Tu podajemy tylko niezbędne wiadomości do zdefiniowania wyrażeń arytmetycznych.

Przykłady

Poprawne funkcje definiowane:

TRY (A)

CZAS (DROGA, V) (identyfikujące są trzy znaki w nazwie funkcji i cztery znaki w argumencie; stąd, równoważnym jest zapis CZA (DROG, V)).

DROGA (DROGA) (pierwsza nazwa jest nazwą funkcji i jest równoważna nazwie DRO, nazwa zaś w nawiasach jest zmienną prostą)

NAZWA (NAZWA (5))

Błędne funkcje definiowane:

SINUS (1.7) (funkcja ma nazwę równoważną nazwie funkcji języka SIN)

Funkcja języka

Jest to funkcja, która różni się od funkcji definiowanej tym, że posiada z góry określoną nazwę, typ, ilości i typy argumentów oraz posiada napisany i automatycznie dołączany do programu, podprogram obliczający jej wartość (podprogram definiujący).

Oprócz wymienionych różnic pozostałe własności funkcji języka (np. postać argumentów) są identyczne jak funkcji definiowanych. Dla uproszczenia, w przyszłości będziemy posługiwać się pojęciem funkcji mając na myśli dowolną z nich tj. funkcję definiowaną lub funkcję języka.

Funkcje języka całkowite nie wymagają deklaracji typu.

Spis oraz omówienie każdej z funkcji języka podane są w DODATKU B.

Przykłady

PWK (9) - jest to funkcja języka, której wartością jest pierwiastek kwadratowy (tj. $\sqrt{9}$ =3) ; funkcja jest ułamkowa.

DIV (10,4) - jest to funkcja całkowita dwuargumentowa o argumentach całkowitych i ma wartość równą części całkowitej dzielenia argumentu pierwszego przez drugi tj. DIV (10,4) = 2

Podprogram

Nazwa podprogramu co do budowy jest identyczna jak nazwa funkcji, jednakże nie może występować w wyrażeniach arytmetycznych.
Dokładny opis podprogramów i sposób ich wywoływania podany będzie przy opisach formuł operacyjnych i zdania PODPROGRAM.

E T Y K I E T A

Definicja
Etykieta - jest to ciąg liter i cyfr zaczynający się od cyfry. Ciąg powinien zawierać przynajmniej jeden znak. Maksymalna długość etykiety zależy tylko od długości wiersza.

W etykiecie, która składa się z więcej niż czterech znaków, identyfikujące są tylko pierwsze cztery, pozostałe zaś są pomijane.
Etykiety w programie mogą występować w następujących miejscach:

1. Są parametrami zdań sterujących i wtedy określają w zależności od warunków (lub bezwarunkowo), które ze zdań ma być wykonane jako następne.
   W tym przypadku własność etykiety ma również słowo NASTEPNY. NASTEPNY zastępuje etykietę, gdy ta dotyczy zdania bezpośrednio napisanego po omawianym zdaniu sterującym.
2. Mogą występować przed zdaniami języka i wtedy wskazują miejsca programu wyszczególnione przez odpowiednie zdanie sterujące.

Etykieta (lub ciąg etykiet) może (mogą) występować w wierszu poprzedzającym zdanie lub też w tym samym wierszu co zdanie.
Niezależnie czy etykieta (etykiety) występuje(ą) w oddzielnym wierszu czy też w tym samym wierszu co odpowiednie zdanie, zawsze po każdej etykiecie musi wystąpić nawias okrągły zamykający.
Jeżeli etykieta (etykiety) występuje(ą) w tym samym wierszu co zdanie to musi (muszą) występować z lewej strony zdania.

Jeżeli występuje więcej niż jedna etykieta przed zdaniem, to wszystkie one dotyczą tego samego zdania.
Jeżeli etykieta występuje przed zdaniem, które nie tworzy części programu wynikowego(np. deklaracja CALKOWITE) , to należy uważać, że dotyczy najbliższego zdania, które tworzy ciąg rozkazów w programie wynikowym.
Niedozwolone jest pisanie etykiet przed zdaniami: ROZDZIAL i JEZYK SAKO.
Przed zdaniami można pisać etykiety, do których nie odwołują się zdania sterujące.

Przykłady

Poprawne etykiety:

```
1
1A
1ALARM      (jest równoważna etykiecie 1ALA)
1 ALA       (jest równoważna etykiecie 1ALA;
             spacja jest nieistotna)
```

Błędne etykiety:

```
1.          (zawiera kropkę)
A12         (zaczyna się od litery)
```

W Y R A Ż E N I A A R Y T M E T Y C Z N E

Wyrażenie arytmetyczne jest regułą na obliczanie wartości liczbowej. Wartość liczbową zwaną wartością wyrażenia, otrzymujemy przez wykonanie wskazanych działań - we wskazanej kolejności, na wielkościach wchodzących w skład wyrażenia.

Wartością wyrażenia arytmetycznego może być liczba całkowita albo liczba ułamkowa. Wyrażenie, którego wartością jest liczba całkowita będziemy nazywać wyrażeniem arytmetycznym całkowitym (typu całkowitego), a wyrażenie, którego wartością jest liczba ułamkowa będziemy nazywać wyrażeniem arytmetycznym ułamkowym (typu ułamkowego).

Aby dowiedzieć się, jaka będzie wartość danego wyrażenia arytmetycznego i jaki będzie jego typ (ułamkowe czy całkowite) musimy wiedzieć:

- jakie są wartości i typy argumentów,
- jak są obliczane wartości i typ działań,
- jaka jest kolejność wykonywania działań.

Pierwotne wyrażenie arytmetyczne

Najprostszym wyrażeniem arytmetycznym jest pierwotne wyrażenie arytmetyczne.

Przez pierwotne wyrażenie arytmetyczne będziemy rozumieli dowolny z następujących elementów języka: stałą dziesiętną, zmienną, funkcję wyrażenie arytmetyczne zamknięte w nawiasy okrągłe.

Wartość pierwotnego wyrażenia arytmetycznego

Wartość pierwotnego wyrażenia arytmetycznego jest taka sama jak wartość odpowiedniego elementu stanowiącego to wyrażenie. Zaś wartości odpowiednich elementów obliczane są następująco:

Wartość stałej dziesiętnej

Jest po prostu explicite podana i nie zmienia się w procesie obliczeniowym.

Wartość zmiennej prostej

Jest równa bieżącej zawartości miejsca pamięci związanego z nia (jest równa ostatnio nadanej dla niej wartości).

Wartość zmiennej indeksowanej

Wartość zmiennej indeksowanej (lub elementu tablicy) otrzymuje się przez:

1. Obliczenie wartości każdego z indeksów
2. Obliczenie wyrażenia (1) określającego względne położenie elementu w bloku (tablicy)
3. Wybranie bieżącej wartości elementu bloku.

Należy zauważyć, że indeksy są wyrażeniami arytmetycznymi całkowitymi i opisane tu reguły stosuje się do obliczania ich wartości.

Wartość funkcji

Wartość funkcji otrzymuje się przez:

1. Jeżeli argumentami są wyrażenia arytmetyczne, to są obliczane ich wartości i podstawiane do podprogramu definiującego daną funkcję,
2. Jeżeli argumentami są bloki lub nazwy innych funkcji, to podstawia się je przez nazwę,
3. Po podstawieniu argumentów wykonuje się podprogram, w wyniku czego otrzymujemy wartość funkcji.

Sposób wywoływania podprogramów będzie podany przy opisie zdań: PODPROGRAM, PODSTAW

Wartość wyrażenia arytmetycznego zamkniętego w nawiasy

Wartość wyrażenia arytmetycznego ujętego w nawiasy jest równa wartości wyrażenia arytmetycznego po usunięciu najbardziej zewnętrznych nawiasów.

Typ pierwotnego wyrażenia arytmetycznego.

Typ pierwotnego wyrażenia arytmetycznego jest taki sam jak typ odpowiedniego elementu stanowiącego to wyrażenie. I tak:

Typ stałej dziesiętnej zależy od jej zewnętrznej postaci. Typ liczby dziesiętnej całkowitej jest całkowity a typ liczby dziesiętnej ułamkowej jest ułamkowy.

Typ zmiennej prostej zależy od tego czy została zadeklarowana jako całkowita, czy też nie. Zmienna jest całkowita jeżeli została wpisana na listę parametrów zdania CALKOWITE. W przeciwnym przypadku jest ułamkowa (dokładniejsze zasady deklarowania podane są przy opisie zdania CALKOWITE).

Typ zmiennej indeksowanej zależy do tego czy odpowiedni blok (tablica) został zadeklarowany jako całkowity czy też nie.
Zmienna indeksowana jest całkowita, jeżeli nazwa odpowiedniego bloku, (tablicy), którego elementem jest dana zmienna, występuje na liście parametrów zdania CALKOWITE (patrz opis zdania CALKOWITE). W przeciwnym przypadku zmienna indeksowana jest ułamkowa.

Typ funkcji języka jest dla każdej funkcji z góry określony i podany w dodatku B.

Typ funkcji definiowanej zależy od deklaracji CALKOWITE. Jeżeli nazwa funkcji jest wymieniona na liście parametrów zdania CALKOWITE to funkcja jest całkowita, inaczej jest ułamkowa (patrz opis zdania CALKOWITE).

Typ wyrażenia arytmetycznego ujętego w nawiasy jest taki sam jak typ wyrażenia arytmetycznego otrzymanego z danego przez usunięcie najbardziej zewnętrznych nawiasów.

Dla skrócenia przykładów, aby za każdym razem nie powtarzać założeń co do typu zmiennych lub funkcji definiowanych ustalimy, że jeżeli nie jest powiedziane inaczej, to nazwy zaczynające się od I, J, K, L będą dotyczyły zmiennych lub funkcji całkowitych, zaś nazwy zaczynające się od A,B,C,D będą dotyczyły zmiennych lub funkcji ułamkowych. Natomiast typ zmiennych i funkcji o nazwach zaczynających się od pozostałych liter będzie w przykładach obojętny.

Przykłady

Poprawne pierwotne wyrażenia arytmetyczne:

123	(stała dziesiętna; typ całkowity)
1.23	(stała dziesiętna; typ ułamkowy)
A	(zmienna prosta lub zerowy element bloku typu ułamkowego)
I1	(zmienna prosta lub element zerowy bloku typu całkowitego)
B(I,7+J)	(zmienna indeksowana; typ ułamkowy. Indeksy są wyrażeniami arytmetycznymi całkowitymi)
I(2,7)	
SIN(X)	(funkcja języka; typ ułamkowy, argument obojętnego typu)
(2)	(2 jest wyrażeniem arytmetycznym, wartość jego jest 2 a typ całkowity)
(-2)	
(A+SIN(X*B))	

Błędne pierwotne wyrażenia arytmetyczne:

-2	(liczba ze znakiem nie jest stałą w sensie definicji liczby dziesiętnej, to jest wyrażenie arytmetyczne)
1A	(to nie jest zmienna; może być etykietą)
77.77.00.12	(to może być stała oktalna, w wyrażeniach mogą występować tylko stałe dziesiętne).

Działania arytmetyczne

Bardziej złożone wyrażenie arytmetyczne może być konstruowane z pierwotnych wyrażeń arytmetycznych przez wprowadzenie działań arytmetycznych.

W wyrażeniach arytmetycznych języka SAKO mogą wystąpić następujące działania:

znak działania	znaczenie
*	potęgowanie
◊	mnożenie
/	dzielenie
-	odejmowanie
+	dodawanie

Poniżej rozważymy dla kolejnych działań typy i wartości argumentów oraz typy i wartości wyników, wprowadzając następujące pojęcia: potęga, iloczyn, iloraz, różnica i suma.

Potęga

Jeżeli przez p_i dla i = 1,2,3,.... oznaczymy pierwotne wyrażenia arytmetyczne, to następujące wyrażenia będziemy nazywać potęgami:

$$p_1 * p_2$$
$$p_1 * p_2 * p_3$$
$$p_1 * p_2 * p_3 * p_4$$

itd.

Powyższe zapisy są odpowiednio równoważne zapisom:

$$p_1 * p_2$$
$$(p_1 * p_2) * p_3$$
$$((p_1 * p_2) * p_3) * p_4$$

itd.

Jak łatwo zauważyć każdy z powyższych zapisów składa się z dwóch pierwotnych wyrażeń arytmetycznych oddzielonych od siebie gwiazdką.

Stąd, wystarczy omówić znaczenie pierwszego z nich.

Kolejność obliczania wartości potęgi

1. Najpierw obliczana jest wartość lewego pierwotnego wyrażenia arytmetycznego
2. Następnie obliczana jest wartość prawego (następnego) pierwotnego wyrażenia arytmetycznego
3. Wartość pierwszego pierwotnego wyrażenia jest podnoszona do potęgi równej wartości drugiego wyrażenia, a wynik potęgowania jest wartością potęgi.

Powyższe zapisy są równoważne:

$$p_1^{p_2} \; ; \quad p_1^{p_2^{p_3}} \; ; \quad \text{itd.}$$

Typ wyniku potęgowania

Typ wyniku potęgowania podaje poniższa tabelka, gdzie w kolumnie pierwszej podany jest typ podstawy, a w pierwszym wierszu typ wykładnika.

p_1 \ p_2	Całkowity	Ułamkowy
całkowita	całkowity	ułamkowy
ułamkowa	ułamkowy	ułamkowy

Należy zauważyć, że tylko dla obydwu argumentów całkowitych potęga ma wartość całkowitą.

Zależność wyniku potęgowania od znaku argumentów

Niezależnie od znaku podstawy dla wykładnika całkowitego potęgowanie daje następujące wartości:

p_2 / p_1	$p_2 > 0$	$p_2 < 0$	$\lvert p_2 \rvert = 0$
całkowita	$\underbrace{p_1 x p_1 x \ldots}_{p_2 \text{ razy}}$	nieokreślony	1
ułamkowa	$\underbrace{p_1 x p_1 x \ldots}_{p_2 \text{ razy}}$	nieokreślony	1.

Należy zauważyć, że potęga jest określoną, gdy jednocześnie podstawa i wykładnik są zerami i wartością jest odpowiednio ułamkowa lub całkowita jedynka.

Dla wykładnika ułamkowego (niezależnie od wartości wykładnika) działanie potęgowania ma sens tylko dla podstawy większej od zera.

Przykłady

Poprawne potęgi:

I * 3 (oznacza I × I × I tj. I^3)

A * (-3.) (potęga ułamkowa; wykładnik i podstawa są ułamkowe; A musi być większe od zera)

(-7.) * I (wykładnik całkowity więc podstawa może być ujemna)

(A+B) * 2 (tj $(A + B)^2$)

A * B * 2 (tj. $(A^B)^2$)

$\underbrace{7.E-2}_{p_1} * \underbrace{W(I,2)}_{p_2} * \underbrace{(A - 3/Y + ALFA * 8)}_{p_3}$

Błędne potęgi:

A * (-3) (wykładnik całkowity ujemny)

A * - 3.0 (liczba - 3.0 nie jest pierwotnym wyrażeniem)

(-7.2) * B (podstawa ujemna przy ułamkowym wykładniku)

I l o c z y n

Niech teraz c_i (i = 1,2,3,....) oznacza dowolne poprawne wyrażenie pierwotne lub potęgę.

Powyżej określone pojęcie będziemy nazywać czynnikiem. Typ i wartość czynnika są takie same jak typ i wartość odpowiedniego wyrażenia pierwotnego lub potęgi.

Każde z poniższych wyrażeń będziemy nazywać iloczynem:

$$c_1 \Diamond c_2$$

$$c_1 \Diamond c_2 \Diamond c_3$$

$$c_1 \Diamond c_2 \Diamond c_3 \Diamond c_4$$

itd.

Powyższe zapisy są równoważne odpowiednio:

$$c_1 \Diamond c_2$$

$$(c_1 \Diamond c_2) \Diamond c_3$$

$$((c_1 \Diamond c_2) \Diamond c_3) \Diamond c_4$$

itd.

Zatem możemy rozpatrywać, nie tracąc na ogólności, iloczyn składający się z dwóch czynników oddzielonych od siebie znakiem mnożenia.

Kolejność obliczania wartości iloczynu.

Aby obliczyć wartość iloczynu liczymy najpierw wartość lewego czynnika (pierwszego z lewej) następnie liczymy wartość drugiego czynnika (następnego w kierunku prawym) i iloczyn wartości czynników jest wartością iloczynu.

Typ iloczynu.

Zależność typu iloczynu od typów argumentów podana jest w poniższej tabelce:

c_1 \ c_2	całkowity	ułamkowy
całkowity	całkowity	ułamkowy
ułamkowy	ułamkowy	ułamkowy

Tylko gdy oba argumenty są całkowite wynik jest całkowity.

Przykłady

Poprawne iloczyny:

2 ◊ 3 (tj. 2x3 = 6; wynik jest całkowity)
2 ◊ 3. 0 (tj. 2x3. 0 = 6.; wynik jest ułamkowy)
3 ◊ (-3) (znaku mnożenia opuszczać nie wolno)
A ◊ (B+C)
(A+B) ◊ (B+C)
(-2) ◊ (I+J)
4 ◊ SIN (X)
2 ◊ 3.0 ◊ A ◊ (B+C) (powyższe przykłady dotyczyły iloczynów, których czynniki były pierwotnymi wyrażeniami)
I ⋆ 3 ◊ 6 ⋆ I (oznacza liczbę całkowitą równą $I^3 \times 6^I$; najpierw liczy się wartość czynników będących potęgami)
2 ◊ A ⋆ B ⋆ 3 (tj $2(A^B)^3$; A ⋆ B ⋆ 3 jest czynnikiem)

Błędne iloczyny:

2 A	(opuszczono znak mnożenia)
(A+B) (A-B)	(również nie wolno opuszczać znaku mnożenia)
AB	(jest zmienną AB, a nie iloczynem A x B)
2 ◊ - 3	(liczba -3 nie jest czynnikiem)

I l o r a z

Jeżeli przez d_i (dla i = 1,2,3,....) oznaczymy czynnik lub iloczyn (tj. pierwotne wyrażenie arytmetyczne, potęgę lub iloczyn) to następujące wyrażenia:

$$d_1 \;/\; d_2$$

$$d_1 \;/\; d_2 \;/\; d_3$$

$$d_1 \;/\; d_2 \;/\; d_3 \;/\; d_4$$

itd.

będziemy nazywać ilorazami.

Powyższe zapisy są równoważne następującym ilorazom:

$$d_1 \;/\; d_2$$

$$(d_1 \;/\; d_2) \;/\; d_3$$

$$(d_1 \;/\; d_2) \;/\; d_3) \;/\; d_4$$

itd.

Zatem możemy rozważać iloraz jako działanie na dwóch argumentach oddzielonych od siebie znakiem dzielenia.

Kolejność obliczania wartości ilorazu

Aby obliczyć wartość ilorazu liczymy najpierw wartości dzielnej (najbardziej lewego argumentu) następnie wartość dzielnika (prawego argumentu) i wynik dzielenia dzielnej przez dzielnik będziemy nazywać wartością ilorazu.

Typ ilorazu

W wyniku dzielenia otrzymujemy zawsze liczbę ułamkową.

UWAGA! Niezależnie od typu dzielnika musi on być różny od zera.

Przykłady

Poprawne ilorazy:

4/2 (jest równe 4:2 = 2.0)

G ◊ H/F (jest równoważne (G ◊ H) / F; najpierw liczymy dzielną tj. iloczyn)

A ◊ B/C ◊ D (tj. (A ◊ B) / (C ◊ D) = $\frac{AxB}{CxD}$)

$$\underbrace{\text{OMO} \lozenge \text{SUM} * \text{COS}(\text{Y+Z} \lozenge 3)}_{d_1} \;/\; \underbrace{7.2\text{ E-2} * \text{W(I,2)} * (\text{A-3/Y+ALA} * 8)}_{d_2}$$

Aby obliczyć wartość d_1 należy najpierw obliczyć wartość iloczynu:

$$\underbrace{\text{OMO}}_{c_1} \lozenge \underbrace{\text{SUM} * \text{COS Y+Z 3}}_{c_2}$$

Aby obliczyć wartość iloczynu, przy znanej wartości OMO liczymy potęgę:

$$\underbrace{\text{SUM}}_{p_1} * \underbrace{\text{COS}(\text{Y+Z} \lozenge 3)}_{p_2}$$

Po policzeniu wartości argumentu funkcji COS i wartości funkcji liczymy wartość potęgi.

Mając wartość OMO i potęgi możemy liczyć wartość dzielnej (d_1).

Podobnie liczymy wartość dzielnika (d_2), a dopiero później obliczamy wartość ilorazu.

Błędne ilorazy:

A / B / 0.0 (dzielenie przez zero)

A / - B (- B nie należy do klasy d_1; należy napisać: A / (-B)).

R ó ż n i c a

Jeżeli r_i (dla i = 1,2,3,...) będzie oznaczało dowolne z poprzednio zdefiniowanych pojęć (pierwotne wyrażenie, potęgę, iloczyn lub iloraz) to poniższe wyrażenia:

$$r_1 - r_2$$
$$r_1 - r_2 - r_3$$
$$r_1 - r_2 - r_3 - r_4$$

itd.

będziemy nazywać różnicami. Jak poprzednio powyższe zapisy są równoważne odpowiednio:

$$r_1 - r_2$$
$$(r_1 - r_2) - r_3$$
$$((r_1 - r_2) - r_3) - r_4$$

itd.

Kolejność obliczania wartości różnicy

1. Liczymy wartość odjemnej r_1
2. Liczymy wartość odjemnika
3. Wartość różnicy jest równa różnicy wartości odjemnej i odjemnika.

Typ różnicy określa następująca tabelka:

r_1 \ r_2	całkowite	ułamkowe
całkowite	całkowite	ułamkowe
ułamkowe	ułamkowe	ułamkowe

Przykłady

Poprawne różnice:

2 - 3

$$\underbrace{G \diamond H / F}_{r_1} - \underbrace{4/3}_{r_2} \; \underbrace{-2}_{r_3}$$

$$\underbrace{OMO \diamond SUM * COS(Y-Z \diamond 3) / 7 . E+2 * W(I,2)}_{r_1} - 1$$

Błędna różnica:

- 2 - 3 (to jest wyrażenie; powinno być: (-2) -3)

S u m a

Jeżeli s_i będzie oznaczało jedno z pojęć objętych pojęciem r_i oraz różnicę, to każdy z poniższych zapisów:

$$s_1 + s_2$$
$$s_1 + s_2 + s_3$$
$$s_1 + s_2 + s_3 + s_4$$

itd.

będziemy nazywać sumą a s_i składnikami. Jak poprzednio powyższy zapis jest równoważny zapisowi:

$$s_1 + s_2$$
$$(s_1 + s_2) + s_3$$
$$((s_1 + s_2) + s_3) + s_4$$

itd.

Kolejność obliczania wartości sumy

Aby obliczyć wartość sumy najpierw liczymy wartość lewego składnika potem liczymy wartość prawego składnika i suma tych wartości jest wartością sumy.

Typ sumy obrazuje poniższa tabela:

s_1 \ s_2	całkowity	ułamkowy
całkowity	całkowity	ułamkowy
ułamkowy	ułamkowy	ułamkowy

Przykłady

Poprawne sumy:

2 + 3

$\underbrace{7 - 3}_{s_1} + \underbrace{8 - 4}_{s_2}$ (najpierw liczy się różnice tj. (7 - 3) + (8 - 4))

Wyrażenia arytmetyczne

Wreszcie dowolne z wyżej zdefiniowanych wielkości lub każde z nich poprzedzone znakiem plus lub minus będziemy nazywać wyrażeniem arytmetycznym.

Wartość i typ wyrażenia ze znakiem plus są takie same jak wartość i typ wyrażenia bez znaku. Typ wyrażenia poprzedzonego znakiem minus jest taki sam jak i typ wyrażenia bez znaku a wartość brana jest ze znakiem przeciwnym.

Przykłady wyrażeń:

Wszystkie poprawne przykłady wymienione w tym paragrafie oraz:

- 2 - 4
+ 7 + 2
- SIN (X) ◊ 2 / PI

Uwagi ogólne na temat wyrażeń arytmetycznych

Z poprzednich definicji można wyciągnąć kilka wniosków praktycznych:

1. Wynik dzielenia jest zawsze typu ułamkowego, a dla pozostałych działań wynik jest całkowity tylko, gdy oba argumenty są całkowite.

2. Niezbyt ściśle można powiedzieć, że w wyrażeniach obliczane są najpierw wyrażenia zawarte w większej ilości nawiasów; przy równej ilości nawiasów obliczane są najpierw wartości wyrażeń prostych a następnie działania według kolejności:
potęgowanie, mnożenie, dzielenie, odejmowanie, dodawanie.
Gdy występuje ciąg działań identycznych to wykonuje się je kolejno od lewej do prawej.

3. Między argumentami muszą występować zawsze znaki działań. Między znakami działań muszą występować nawiasy lub argumenty. Nadmierna ilość nawiasów nie jest błędem.

4. Wszystkie zmienne w wyrażeniu muszą mieć nadane wartości przez wprowadzenie wraz z programem (tablice) , przeczytanie lub przypisanie (patrz zdanie przypisujące) . W przeciwnym przypadku działanie będzie wykonywane na wartościach przypadkowych znajdujących się w pamięci maszyny.

5. W procesie liczenia wartości wyrażenia arytmetycznego, wszystkie wyniki pośrednie (wartości funkcji, wartości działań) muszą mieścić się w odpowiednich zakresach.
   a/ Dla wartości całkowitych zakresy są takie same jak dla liczby dziesiętnej całkowitej. Przekroczenie tego zakresu powoduje ustawienie tzw. wskaźnika nadmiaru stałoprzecinkowego. Ponadto wartość wyrażenia będzie błędna.
   Fakt zaistnienia nadmiaru stałoprzecinkowego można badać przy pomocy zdania GDY BYL NADMIAR.

b/ Przy działaniach na liczbach ułamkowych zakres pośrednich wyników jest taki sam jak dla stałej ułamkowej. Pojawienie się nadmiaru może spowodować natychmiastowe zatrzymanie programu (patrz zdanie czołówki translatora: NIWELUJ NADMIARY).

Z D A N I A J E Z Y K A S A K O

Klasyfikacja zdań i skrócone ich opisy podane są w DODATKU D. Tutaj zostaną podane względnie dokładne znaczenia zdań z objaśniającymi przykładami. Kolejność opisu zdań została dobrana tak, aby w przykładach i opisie używać jak najmniej pojęć i zdań jeszcze nie opisanych. Mówiąc o zakresie ważności deklaracji języka lub o kolejności występowania zdań w programie będziemy mieli zawsze na myśli program główny. Dopiero przy opisie zdania PODPROGRAM podane będą dokładniej zależności między programem głównym a podprogramami. Ponadto zdania lub postacie zdań, które mogą wystąpić tylko w podprogramach będą opisane po opisie zdania PODPROGRAM. Następnie przy opisie zdań dotyczących struktury rozdziałowej podane zostaną sposoby przekazywania danych z jednego rozdziału do innego.

Ponieważ zdania dotyczące działań logicznych i danych oktalnych stanowią w języku SAKO jakby oddzielną całość mało powiązaną z innymi zdaniami, to paragrafy opisujące te zdania mogą być przy czytaniu pominięte. Kolejność w jakiej program jest analizowany przez translator jest zgodna z kolejnością napisania zdań w programie źródłowym i dlatego też deklaracje opisujące zmienne, bloki, tablice itp. muszą z zasady występować wcześniej niż zostaną użyte w zdaniach tworzących części programu wynikowego (zdania wykonujące się) . Również wykonywanie się zdań (wykonywanie programu wynikowego utworzonego ze zdań) jest zgodne z kolejnością wypisania, poczynając zawsze od pierwszego zdania w rozdziale. Kolejność ta może być tylko zmieniona przez zdanie sterujące.

ZDANIE ARYTMETYCZNE PRZYPISUJACE

Ogólna postać:
a = b gdzie: a jest zmienną (prostą, indeksowaną, elementem tablicy), b jest wyrażeniem arytmetycznym.

Znaczenie zdania arytmetycznego przypisującego jest następujące: oblicz wartość wyrażenia arytmetycznego stojącego po prawej stronie znaku równości i przypisz (nadaj) tę wartość zmiennej stojącej po lewej stronie. Znak równości w zdaniu arytmetycznym przypisującym ma odmienne znaczenie od ogólnie przyjętego (np. w algebrze).

Zdanie przypisujące można czytać: a "staje się równe" b W zależności od typu zmiennej stojącej po lewej stronie znaku równości (a) oraz od typu wyrażenia arytmetycznego (b) w trakcie nadawania wartości wykonuje się jedna z operacji podanych w poniższej tabelce:

typ a \ typ b	całkowite	ułamkowe
całkowite	przypisane bez zmian	zamiana liczby ułamkowej na całkowitą
ułamkowe	zamiana liczby całkowitej na ułamkową	przypisanie bez zmian

Przy zamianie liczby całkowitej na ułamkową następuje zamiana postaci w jakiej są przedstawiane liczby w pamięci maszyny bez zmian wartości.

Przy zamianie liczby ułamkowej na całkowitą następuje najpierw zaokrąglenie do całości, a następnie pominięcie części ułamkowej i przekształcenie wyniku na postać właściwą dla liczb całkowitych. Ponieważ zakres liczby całkowitej jest mniejszy od zakresu liczby ułamkowej to w trakcie zamiany typów z ułamkowego na całkowity może nastąpić przekroczenie zakresu. W takim przypadku wartość jaką otrzyma zmienna a będzie błędna i ustawiony zostanie w stan równy "TAK" tzw. wskaźnik nadmiaru stałoprzecinkowego. Wskaźnik może być badany przy pomocy zdania sterującego GDY BYL NADMIAR.

Przykłady

Poprawne zdania przypisujące:

A = 7.6 (nadanie zmiennej A wartości równej 7.6)

I = A (ponieważ wartość A jest równa 7.6, to zmienna I otrzyma wartość równą 8)

A = I + 1 (wartość wyrażenia (I+1) jest całkowita i równa 9; nowa wartość zmiennej A (po zmianie typu) będzie równa 9.)

A = A + 7 (zwiększanie wartości A o 7.)

I1=10 * 8./10 * 6.
(po wyliczeniu wartości wyrażenia ułamkowego wynik jest zamieniany na postać całkowitą i przesyłany do I1)

X = - X (zmiana znaku wartości zmiennej)

E (I,J) = E (I,J) / 3

S = S + A (I,J) ◊ B (J,K)

I = 7.5 (I otrzyma wartość 8)

Błędne zdanie:

A=B = C+1	(B=C+1 nie jest wyrażeniem arytmetycznym; można napisać: A=C+1 A=B)
A=C+1; B=A	(każde zdanie musi zaczynać się od nowego wiersza)
A = (1+X) * 7 + (1+X) * 6 + + (1+X) * 5	(zdań arytmetycznych przypisujących nie wolno przenosić)
I = 10 * 8/102	(potęga 10^8 jest całkowita i przekracza zakres; wynik będzie niepoprawny)
M = 77.44.22.00+B	(stałe oktalne nie mogą występować w wyrażeniach arytmetycznych).

DEKLARACJA CALKOWITE

Ogólna postać:
CALKOWITE: a1,a2,.....,an gdzie: a1,a2,.....,an są elementami listy Elementami listy mogą być: nazwa zmiennej prostej, nazwa bloku poprzedzona gwiazdką, nazwa tablicy poprzedzona gwiazdką, nazwa funkcji, po której występuje para nawiasów okrągłych. Lista może być przenoszona.

Deklaracja CALKOWITE nie tworzy programu wynikowego, a tylko informuje translator, że zmienna, funkcja, bloki lub tablice, których nazwy zostały wymienione na liście są całkowite. Dotyczy to tylko tych nazw, które pojawią się w dalszej części programu.
Pozostałe zmienne, bloki, tablice, funkcje (jeżeli nie zostaną zadeklarowane jako całkowite przy pomocy następnego zdania CALKOWITE) będą ułamkowe; z wyjątkiem funkcji języka których typu nie trzeba deklarować.

Deklaracja CALKOWITE nie powoduje rezerwacji miejsc dla zmiennych, kolejność nazw na liście jest nieistotna i gwiazdka przed nazwą jest nieistotna. Dla uniknięcia pomyłek zaleca się, aby w programie deklaracja CALKOWITE występowała jako pierwsza.

Przykłady

Poprawne deklaracje:

CALKOWITE : I,I2, * A, F( ), MAT
(zmienne proste: I,I2,MAT oraz blok A i funkcję F należy uważać, że są całkowite)

```
CALKOWITE : A1,A2,A3, -
            B1,K,L, -
            Z          (listę można przenosić)
```

Błędne deklaracje:

```
CALKOWITE : A (7)   pojedynczych zmiennych indeksowanych
                    nie wolno deklarować
  ------
  ------
  ------

Z = 5               ta zmienna Z jest traktowana jako
                    ułamkowa

CALKOWITE : Z,M

M = Z + 2           (a tu jest już uważana jako całkowita
  ------            i przy liczeniu wartości wyrażenia
  ------            Z+2 wartość zmiennej Z jako całkowi-
  ------            tej nie będzie równa 5, gdyż automa-
                    tyczna zamiana typów nie nastąpi) .
```

DEKLARACJA BLOK

Postać ogólna
BLOK (a1, a2,...,an) : b1,...,bm gdzie: a1, a2,...,an są to elementy listy zakresów. Elementem listy zakresów indeksów może być liczba całkowita dodatnia lub zero, lub wyrażenie arytmetyczne, w którego skład wchodzą liczby całkowite i dopuszcza się tylko trzy działania(+,-, ◊) oraz nawiasy okrągłe. b1, b2,...,bm są to nazwy bloków. Lista zakresóe indeksów jest ujęta w nawiasy okrągłe i oddzielona od listy bloków dwukropkiem. Lista nazw bloków może być przenoszona.

Deklaracja BLOK powoduje zarezerwowanie miejsca pamięci dla każdego bloku wymienionego na liście nazw bloków oraz dostarcza następujących informacji o każdym z nich:

1) Podaje zakresy każdego z indeksów (ai) oraz wymiary (ai+1) odpowiednio a1 podaje zakres pierwszego indeksu, a2 drugiego itd.

2) Określa ilość wymiarów każdego bloku (ile indeksów może mieć zmienna) . Ilość wymiarów jest równa ilości pozycji na liście zakresów.

3) Określa ilość elementów w bloku:
   k = (a1+1) x (a2+1) x...x (an+1) .

4) Podaje ilość miejsc pamięci potrzebnej dla bloku. Dla bloków całkowitych potrzeba k miejsc, a dla bloków ułamkowych 2 x k.

Dany blok jest całkowity, gdy deklaracja BLOK opisująca go jest poprzedzona przez deklaracje CALKOWITE zawierającą na liście nazwę omawianego bloku, w przeciwnym przypadku blok jest ułamkowy.

Wśród nazw wymienionych na liście bloków mogą występować w dowolnej kolejności nazwy bloków całkowitych i ułamkowych. Aby uniknąć pomyłek zaleca się, aby zdania BLOK występowały w programie bezpośrednio po zdaniach CALKOWITE. Dla opisania bloków o różnych zakresach i /lub ilościach indeksów należy użyć różnych zdań BLOK.

Nie ma potrzeby deklarowania zmiennych prostych przy pomocy deklaracji BLOK (np. BLOK (0)) . Pierwsze wystąpienie zmiennej prostej w programie powoduje przydzielenie dla niej miejsca w pamięci maszyny (z wyjątkiem deklaracji CALKOWITE, patrz opis deklaracji CALKOWITE).

Przykłady

Poprawne użycie deklaracji BLOK:

```
1) CALKOWITE: B        (Bloki A,B,C są jednowymiarowe, jednoele-
   BLOK(0): A,B,C       mentowe o zakresie indeksu 0.
   --------------       Bloki A i C są ułamkowe, a blok B jest
   --------------       całkowity)
   --------------
   A (0) = 1.
   B = 3                (Można opuszczać indeksy przy pierw-
   A = A + B            szych elementach bloków).
2) CALKOWITE: I,J      (Blok K jest dwuindeksowy o zakresach
   BLOK(7+2,2): K       indeksów 9 i 2 o wymiarach 10 x 3,
   --------------       składa się z 10 x 3 = 30 elementów
   --------------       i zajmuje 10 x 3 x 2 = 60 miejsc pamię-
                        ci)
   K (I,J) = 14
   -------------
   -------------
```

Błędne deklaracje BLOK:

BLOK (I,6, 0.3) : C	(pierwszy zakres indeksu jest zmienną a nie liczbą całkowitą, trzeci zakres jest ułamkowy)
BLOK (-7) : B	(zakres indeksu jest ujemny)
BLOK (4) : * ALA	(nazwa bloku poprzedzona jest gwiazdką)
A = 7.7 BLOK (3) : A	(wcześniej użyto zmiennej A, a następnie zadeklarowano ją jako blok).

Rezerwacja pamięci maszyny dla bloków

Znajomość kolejności rezerwowania miejsca dla bloków jest potrzebna przy organizacji i nazywaniu pól i podpól oraz przy przekazywaniu danych z rozdziału do rozdziału i współpracy z pamięciami zewnętrznymi. Zasady zmiany organizacji pola i podpól zostaną omówione przy opisie zdania STRUKTURA. Zasady przekazywania danych między rozdziałami omówimy przy definicji zdań dotyczących rozdziałów.

Tutaj pokażemy jak jest rezerwowana pamięć dla kolejnych bloków jeżeli ich nazwy są wyszczególnione na liście jednego zdania BLOK lub na listach kolejnych zdań.

Z punktu widzenia translatora zdanie BLOK z m elementami na liście nazw bloków jest równoważen m kolejno napisanym zdaniom BLOK, z których każde ma taką samą listę zakresów indeksów i tylko jedną nazwę na liście nazw bloków.
Np. zdanie

BLOK (4) : A,B,C

jest równoważne ciągowi zdań:

BLOK (4) : A
BLOK (4) : B
BLOK (4) : C

Zasadę rezerwowania miejsca w pamięci sformułujemy dla zdań BLOK opisujących pojedyncze bloki:

Miejsce w pamięci operacyjnej rezerwuje się dla bloków w kolejności odwrotnej niż ta, w której wystąpiły ich deklaracje w programie. Jeżeli między kolejnymi zdaniami BLOK nie występują inne zdania (z wyjątkiem deklaracji CALKOWITE) to między obszarami zarezerwowanymi dla kolejnych bloków nie występują żadne luki i będziemy mówili, że taki ciąg deklaracji wyznacza pole pamięci.

Nazwą tego pola będzie nazwa bloku występującego w ostatniej deklaracji BLOK. W szczególnym przypadku deklaracja jednego bloku wyznacza pole o tej samej nazwie.

Podpolami danego pola będziemy nazywać takie pole, które można otrzymać pomijając kolejno od końca deklaracje BLOK.

Kolejne podpola można uważać za pola, które z kolei mają własne podpola itd.

Przykład

Weźmy ciąg deklaracji opisujących pole L:

```
CALKOWITE : I,J
BLOK (2)  : M,N,I
BLOK (3)  : J,K
BLOK (1,2)  : L
```

Jest on równoważny ciągowi:

```
CALKOWITE : I,J
BLOK (2)  : M
BLOK (2)  : N
BLOK (2)  : I
BLOK (3)  : J
BLOK (3)  : K
BLOK (1,2)  : L
```

Obszar pamięci dla tych bloków będzie zarezerwowany w kolejności:

L, K, J, I, N, M

tj. za ostatnią zmienną bloku L (L (1,2)) wystąpi pierwsza zmienna bloku K (K (0)) itd.

Pierwszą zmienną całego pola będzie zmienna

L (0,0) , a ostatnią M (2) .

Ilość miejsc zarezerwowanych zależy od rozmiarów i typów bloków. Rys.6 pokazuje pole zdefiniowane w powyższym przykładzie.

Pole L składa się z podpól:

K, J, I, N, M

z kolei K, które jest podpolem pola L ma następujące własne podpola:

J, I, N, M itd.

(patrz rys.6, str.59) .

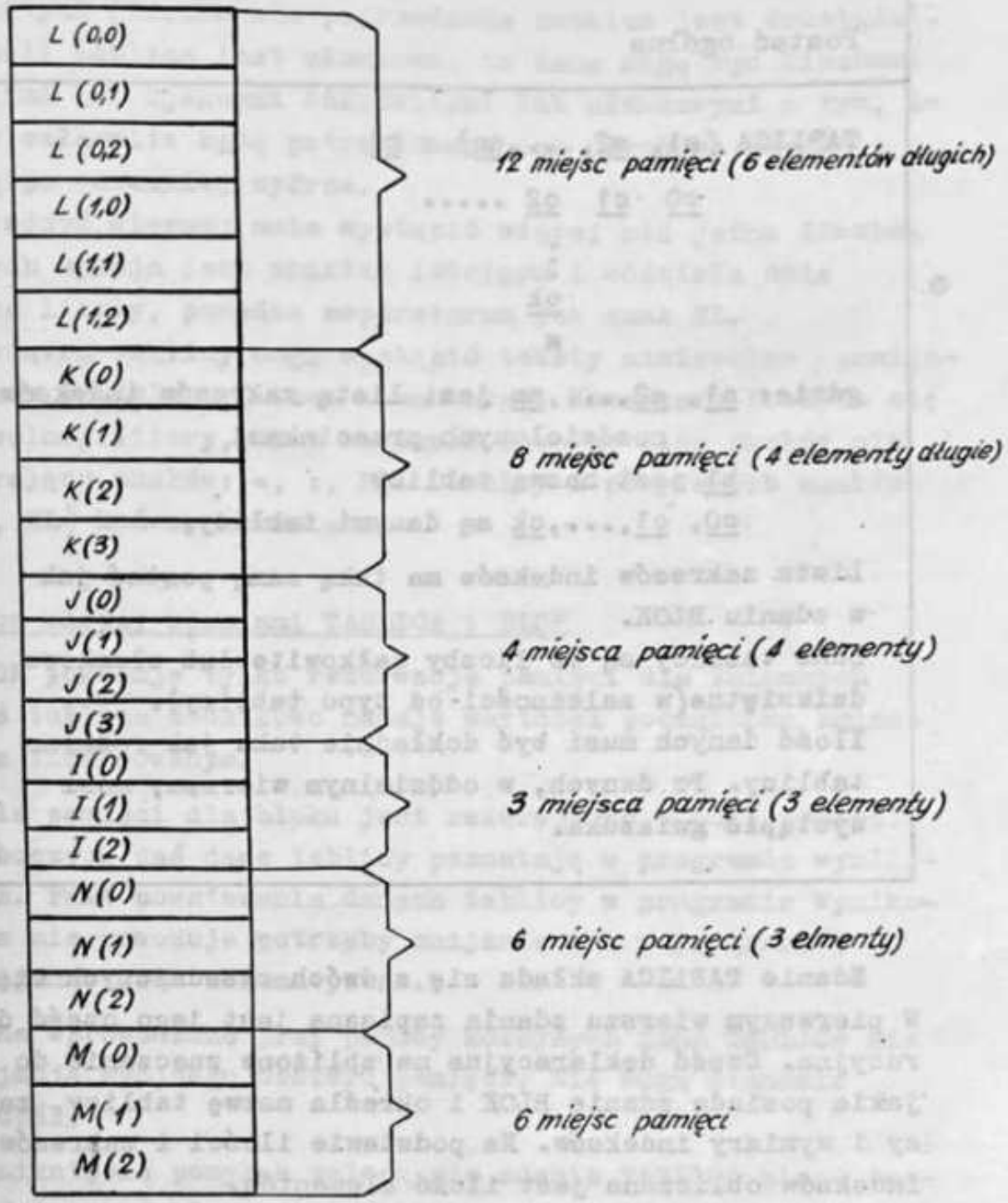
Ponieważ w języku SAKO, przy obliczaniu wartości indeksów i położenia zmiennej indeksowanej w bloku, nie jest kontrolowane przekroczenie zakresów to np.

L (2,2) ma tę samą wartość co K (2)
L (1,3) - " - K (0)

Natomiast sekwencja zdań:

1 J (0) = 5
2 J (1) = 2
3 K (4) = 12.1

spowoduje nadanie wartości zmiennym całkowitym J (0) i J (1) a następnie zniszczenie ich przez zdanie K (4) = 12.1 gdyż K (4) reprezentuje to samo miejsce pamięci co J (0) i J (1). Należy jeszcze zauważyć, że ze względu na nieparzystą ilość miejsc zajmowanych przez blok I, zmiennej K (7) odpowiada to samo miejsce pamięci co zmiennej I (2) oraz pierwszej części zmiennej N (0).


L (0,0)
L (0,1)
L (0,2)
L (1,0)
L (1,1)
L (1,2)
K (0)
K (1)
K (2)
K (3)
J (0)
J (1)
J (2)
J (3)
I (0)
I (1)
I (2)
N (0)
N (1)
N (2)
M (0)
M (1)
M (2)
12 miejsc pamięci (6 elementów długich)
8 miejsc pamięci (4 elementy długie)
4 miejsca pamięci (4 elementy)
3 miejsca pamięci (3 elementy)
6 miejsc pamięci (3 elmenty)
6 miejsc pamięci

Rys. 6

TABLICA DZIESIETNA

Postać ogólna

TABLICA (a1, a2,...,an) : b1

c0 c1 c2 .....

:

ck

*

gdzie: a1, a2,...,an jest listą zakresów indeksów rozdzielonych przecinkami

b1 jest nazwą tablicy

c0, c1,...,ck są danymi tablicy.

Lista zakresów indeksów ma taką samą postać jak w zdaniu BLOK.

Dane tablicy są to liczby całkowite lub ułamkowe dziesiętne(w zależności od typu tablicy).

Ilość danych musi być dokładnie taka jak rozmiar tablicy. Po danych, w oddzielnym wierszu, musi wystąpić gwiazdka.

Zdanie TABLICA składa się z dwóch zasadniczych części. W pierwszym wierszu zdania zapisana jest jego część deklaracyjna. Część deklaracyjna ma zbliżone znaczenie do tego, jakie posiada zdanie BLOK i określa nazwę tablicy, zakresy i wymiary indeksów. Na podstawie ilości i zakresów indeksów obliczana jest ilość elementów.

Jeżeli tablica ma być całkowita to musi być poprzedzona odpowiednią deklaracją CALKOWITE. Zdanie TABLICA musi w programie wystąpić wcześniej niż będziemy korzystać z jej elementów. W następnych wierszach stanowiących drugą część zdania zapisuje się wartości początkowe tablicy.

Postać danych tablicy

Jeżeli tablica została zadeklarowana jako całkowita to dane tablicy powinny być ujemnymi lub dodatnimi liczbami całkowitymi (liczba nie poprzedzona znakiem jest dodatnia).

Jeżeli tablica jest ułamkowa, to dane mogą być liczbami dodatnimi lub ujemnymi całkowitymi lub ułamkowymi z tym, że liczby całkowite będą potraktowane jak ułamkowe zwykłe z kropką po ostatniej cyfrze.

W jednym wierszu może wystąpić więcej niż jedna liczba. W danych spacja jest znakiem istotnym i oddziela dwie kolejne liczby, ponadto seperatorem jak znak NL.

Wewnątrz tablicy mogą wystąpić teksty nieistotne (pomijane przez translator) tzw. komentarze. Komentarz zaczyna się od dowolnej litery, może zawierać dowolny ciąg znaków nie zawierający znaków: =, :, NL. Dowolny z powyższych znaków (=, :, NL) kończy komentarz.

Różnice między zdaniami TABLICA i BLOK

1. BLOK powoduje tylko rezerwację pamięci dla zmiennych zaś tablica dodatkowo nadaje wartości początkowe zmiennym indeksowanym.
2. Pole pamięci dla bloku jest rezerwowane z puli pamięci roboczych zaś dane tablicy pozostają w programie wynikowym. Fakt powstawania danych tablicy w programie wynikowym nie powoduje potrzeby omijania ich w programie; omijanie jest automatyczne.
3. Dane wprowadzone przy pomocy kolejnych zdań TABLICA nie zajmują spójnego obszaru pamięci. Nie mogą stanowić podpól.

Dla uniknięcia pomyłek zaleca się zdania TABLICA pisać bezpośrednio po deklaracjach BLOK.

Przykłady

Poprawne użycie tablic dziesiętnych:

```
TABLICA (0) : ALA
   1.7
    *
TABLICA  1,2  : A
   3.7  +  4.2  -3.2
A (1,0) = 3.5
A (1,1) :  14
A (1,2) = -12

    *
CALKOWITE  : B
I = 3
TABLICA (3) : B
   1  TO JEST TABLICA CALKOWITA
   2
   3     4
     *
I = I + B (3)
```

(pominięte zostaną komentarze: A (1,0) =, A (1,1) : i A (1,2) = liczby całkowite są równoważne odpowiednio: 14 . i - 12. gwiazdka obowiązkowa

(po zdaniu I = 3 wykona się jako następne zdanie I = I + B (3,) tablica automatycznie zostanie ominięta)

Błędne użycie tablic dziesiętnych:

```
CALKOWITE : B
TABLICA (1) : B
   1
  -2.
C (1,1) = C (1,1) -1
TABLICA (1,1) : C
   1  2
   3  4

    *
```

(liczba ułamkowa w tablicy całkowitej i brak gwiazdki na końcu)

(wcześniej odwołano się do elementu tablicy niż została zadeklarowana)

TABLICA (4) : K

1	2	(brak jednego elementu)
3	4	

*

*) TABLICA OKTALNA

Postać ogólna

TABLICA OKTALNA (a1, a2,...,an) : b1

co c1 c2 ...

⋮

ck

*

gdzie: a1, a2,...,an jest listą zakresów indeksów oddzielonych przecinkami,

b1 jest nazwą tablicy,

co, c1 ... ck są danymi tablicy oktalnej. Lista zakresów indeksów ma taką samą postać jak w zdaniu BLOK.

Dane tablicy są liczbami oktalnymi krótkimi lub długimi. Ilość danych musi być dokładnie taka sama jak rozmiar tablicy. Po danych w oddzielnym wierszu musi wystąpić gwiazdka.

Postać danych zdania TABLICA OKTALNA

Jeżeli tablica oktalna została zadeklarowana jako całkowita, to dane tablicy powinny być liczbami oktalnymi krótkimi.

Jeżeli tablica oktalna jest ułamkowa to dane mogą być liczbami oktalnymi długimi lub krótkimi, w tym przypadku liczba oktalna krótka występująca wśród liczb oktalnych długich przekształcana jest przez translator do postaci liczby oktalnej długiej. Liczba oktalna krótka jest równoważna liczbie oktalnej długiej, której pierwsze cyfry są takie jak dana liczba krótka a ostatnie osiem cyfr są zerami.

Reguły pisania danych oktalnych dotyczące separatorów (spacji i NL) i komentarzy są takie same jak dla tablicy dziesiętnej.

Przykłady

Poprawne tablice oktalne:

```
   TABLICA OKTALNA (2) : A
A (0) = 77.77.77.00.12.34.56.70
A (1) = 77.77.77.00.1
A (2) =    77.00.77
          *
```

Dziewięciocyfrowy element A (1) (według reguł przedstawiania liczb oktalnych długich) jest równoważny liczbie oktalnej szesnastocyfrowej:

0000000.77.77.77.00.1

Natomiast element A (2) , który jest liczbą oktalną krótką (mniej niż 9 cyfr) będzie równoważny następującej liczbie oktalnej długiej:

00.77.00.77.0000.0000

gdyż najpierw utworzona została liczba oktalna krótka (00.77.00.77) a potem została uzupełniona zerami do końca.

```
TABLICA OKTALNA (0) : B
   77.77.11
      *
CALKOWITE : C
TABLICA OKTALNA (1,2) : C
   1  2  3
   4  5  66
      *
   1) A = C (1,2) + 10
```

(to jest oktalna 66 równa dziesiętnej liczbie całkowitej 6x8+6=54; należy zauważyć, że C (1,2) w zdaniu 1) będzie potraktowane jako zmienna indeksowana całkowita o wartości 54 niezależne od tego w jaki sposób nadano jej wartość)

Błędna tablica oktalna:

CALKOWITE : K

TABLICA OKTALNA (1) : K

12345678 (wystąpiła cyfra 8 w K (0) i

123456700 dziewięć cyfr w K (1))

STRUKTURA

Postać ogólna

STRUKTURA a1, a2,...,an : b1, b2,...,bn

gdzie: a1, a2,...,an są wyrażeniami arytmetycznymi całkowitymi i wchodzą w skład listy zakresów indeksów bloków,

b1, b2,...,bk są nazwami bloków i rozdzielone przecinkami stanowią listę nazw bloków.

Lista nazw bloków może być przenoszona.

STRUKTURA powoduje zmianę ilości i zakresóe indeksów każdego z bloków wymienionych na liście (dokładniej powoduje zmianę struktury pól o tych samych nazwach co wymienione na liście blok).

Ze względu na postać parametrów, zdania BLOK i STRUKTURA różnią się tylko tym, że zakresy indeksów STRUKTURY mogą być wyrażeniami arytmetycznymi. Są jednak znaczne różnice dotyczące znaczenia tych zdań. A mianowicie:

1. Deklaracja BLOK powoduje rezerwację pamięci maszyny dla zmiennych. Natomiast STRUKTURA nie rezerwuje pamięci i wymaga, aby była ona zarezerwowana wcześniej (w sensie kolejności pisania programu) przy pomocy zdania BLOK (lub TABLICA, TABLICA OKTALNA) lub też kilku zdań BLOK. Oznacza to,że każda nazwa występująca na liście bloków zdania STRUKTURA powinna wystąpić wcześniej na jednej z list zdania BLOK (lub TABLICA, TABLICA OKTALNA) albo też powinna być blokiem formalnym podprogramu (patrz opis zdania PODPROGRAM).

2. Deklaracja BLOK powoduje ustalenie zakresów indeksów statycznie (w czasie translacji) i nie tworzy elementu programu wynikowego. Natomiast STRUKTURA stanowi element programu wynikowego i może zmieniać ilość i zakresy indeksów bloku (pola) o danej nazwie w czasie wykonywania się programu wynikowego.

3. Kolejność nazw bloków na liście zdania BLOK jest istotna gdyż decyduje o kolejności rezerwowania miejsca pamięci zaś dla zdań STRUKTURA nie odgrywa roli.

4. Blok o danej nazwie może być zadeklarowany tylko raz gdy tymczasem struktura jego może być zmieniana wielokrotnie.

Do cech wspólnych zdań BLOK i STRUKTURA należy zaliczyć to, że przy wyliczaniu względnego położenia elementu w polu stosuje się ten sam wzór co i dla wyznaczania położenia w bloku.

Znaczenia pojęć: zakres indeksu pola, wymiar indeksu pola, rozmiar pola, odnoszących się do pól definiowanych przez STRUKTURE są takie same jak odpowiednie pojęcia dla bloków. Ponadto początki obszarów pamięci rezerwowanych przez zdania BLOK i początki pól definiowanych przez zdania STRUKTURA dotyczących tych samych nazw są identyczne tj. zmienna o tej samej nazwie i z wszystkimi indeksami równymi zero zawsze reprezentuje to samo miejsce pamięci niezależnie od tego czy dany blok podlegał strukturowaniu czy też nie.

Dzięki zdaniu STRUKTURA rozszerzymy pojęcie bloku: Mówiąc o blokach będziemy mieli na myśli zbiór zmiennych wyznaczony zarówno przez deklarację BLOK, zdania TABLICA lub TABLICA OKTALNA jak i zbiór zmiennych wyznaczony dynamicznie (w czasie pracy programu) przy pomocy zdania STRUKTURA.

Element pola wyznaczonego przez zdanie STRUKTURA będziemy również nazywać zmienną indeksowaną.

Przykłady

Poprawne wykorzystanie zdania STRUKTURA:

1) BLOK (99) : A (zarezerwuj 2x100 miejsc dla bloku jednowymiarowego)

A (99) = 199 (wartość ostatniego elementu jest równa 199) .

STRUKTURA (9,9) : A (niech teraz pole zarezerwowane dla bloku A będzie dwuwymiarowe o wymiarach 10x10)

Z = A (9,9) (ostatni element pola A jest równy 199.)

STRUKTURA (2,3,1) : A (od tej pory pole A będzie trójwymiarowym blokiem o rozmiarze 3x4x2 = 24)

A (1,1,1) = (1 ◇ 4 + 1) ◇ 2 + 1

2) CALKOWITE : I, J

BLOK (49) : A

I = 3

J = 4

STRUKTURA (I-1,J) : A (wyznacza pole 2x3x5 = 30 miejsc)

A (I-2,2) = 1.7

3) BLOK (1,3) : A (te dwie deklaracje wyznaczają pole

BLOK (2,2) : B B i podpole A)

STRUKTURA (3 ◇ 3 + 2 ◇ 4 -1) : B (wyznacza jednowymiarowe pole (B) o rozmiarze równym sumie rozmiarów bloków A i B)

STRUKTURA ((2+1) ◇(2+1) -1) : B (zamiana obu bloków dwuwymiarowych na jednowymiarowe)

STRUKTURA ((1+1) ◇(3+1) -1) : A

STRUTKURA (1,1) : A,B (z obu bloków wybrano pola po 2 x 2 elementy)

Błędne użycie struktury:

1) BLOK (4) : A
   STRUKTURA (Z) : A (Z nie posiada wartości oraz jest ułamkowe)

   Z = 3

2) BLOK (4) : A
   BLOK (5) : B
   STRUKTURA (5) : A (blok A jest ostatni w polu i obszar objęty przez strukturę przekracza pole)

   ----

3 CALKOWITE : I
   I = 7
   STRUKTURA (I) : A (zdanie STRUKTURA wystąpiło wcześniej niż deklaracja BLOK)
   BLOK (100) : A

Zakres ważności zdania STRUKTURA

Przy rozważaniu zakresu ważności zdania STRUKTURA ograniczymy się do programu głównego (podprogramu) . Ponadto będziemy rozważać tylko jeden zadeklarowany blok i zmiany jego struktury przy pomocy kilku zdań STRUKTURA.

Ponieważ z punktu widzenia rozważań nad zakresem oddziaływania STRUKTURY każdy blok można analizować oddzielnie, to zawężenie rozważań do jednego bloku nie uszczupla ogólności.

Rozważania poniższe będą dotyczyły kolejności napisania zdań (kolejności statycznej) , a nie kolejności wykonywania (kolejności dynamicznej) .

Najpierw podzielimy program na dwie części: Pierwszą będącą pod wpływem deklaracji BLOK i drugą będącą pod wpływem zdań STRUKTURA.

W części pierwszej po zdaniu BLOK, TABLICA lub TABLICA OKTALNA aż do najbliższego zdania STRUKTURA, blokami i zmiennymi indeksowanymi występującymi w zdaniach rządzi deklaracja BLOK. Oznacza to, że podczas translacji obliczane są wymiary wszystkich indeksów oraz rozmiar bloku na podstawie

deklaracji BLOK. Wymiary indeksów jako stałe są używane do liczenia względnego położenia zmiennej indeksowanej w bloku, a rozmiar bloku jako stała dotyczy bloków występujących w tej części programu. Zdanie STRUKTURA nie ma wpływu na tę część programu.

Zdanie STRUKTURA rządzi wszystkimi zmiennymi indeksowanymi i blokami, aż do następnego zdania STRUKTURA albo do końca programu głównego (podprogramu) . Oznacza to, że dla poprawnego obliczania względnego położenia zmiennej indeksowanej lub dla obliczania rozmiaru bloku znajdującego się w omawianym odcinku programu, należy wykonać poprzedzające je zdanie STRUKTURA. Wykonanie go spowoduje obliczenie rozmiaru bloku oraz wymiarów i zakresów wszystkich indeksów. Wartości obliczone w trakcie wykonywania danego zdania STRUKTURA są, w zarządzanym przez nie odcinku programu, tak długo ważne (używane do wyznaczenia położeń zmiennych i rozmiarów bloków) aż nie zostanie ponownie wykonane to samo zdanie STRUKTURA, niezależnie od wykonywania innych zdań STRUKTURA dotyczących tej samej nazwy.

Przykład

```
   BLOK (24-1) : A
   -------
   -------
   A (12) = 12
   -------
   -------
```

(zakres oddziaływania deklaracji BLOK)

```
1) STRUKTURA (4-1, 6-1) : A
   A (2,1) = 2 ◊ 6+1
   -------
   -------
```

(zakres oddziaływania zdania o etykiecie 1)

```
2) STRUKTURA (3-1,4-1,2-1) : A
   A (1,2,3) = 1 4 + 3/ 2 + 1
   -------
   -------
```

(zakres oddziaływania zdania 2)

SKOCZ DO

Postać ogólna
SKOCZ DO [:] e1 gdzie : e1 jest etykietą. Znak dwukropka można pominąć

Normalnie program wykonuje się zgodnie z kolejnością wypisania zdań.
Kolejność ta może być zmieniona przez zdanie sterujące (patrz DODATEK D).
Zdanie to powoduje, że zostanie przerwana normalna kolejność wykonywania programu i jako następne będzie wykonywane zdanie poprzedzone etykietą e1 (zostanie wykonany skok do e1).
Zamiast mówić: "będzie wykonane zdanie poprzedzone etykietą e1" będziemy często mówić krócej: "wykona się zdanie e1".

Przykład 1

```
--------
--------
SKOCZ DO 1A      (po wykonaniu tego zdania, jako
--------          następne wykona się zdanie A=3)
--------
1A) A=3
```

Przykład 2

```
SKOCZ DO : NASTEPNY      (to zdanie nie zmienia kolej-
--------                  ności wykonania programu)
--------
```

GDY A > B

Postać ogólna
GDY a {>,]} b : e1, INACZEJ e2 gdzie : a i b są wyrażeniami arytmetycznymi, e1 i e2 są to etykiety.

Zdanie to powoduje obliczenie wartości wyrażenia arytmetycznego a następnie obliczenie wartości wyrażenia arytmetycznego b i porównanie ich. Jeżeli wartość wyrażenia a jest większa od wartości wyrażenia b , to następnym zdaniem wykonywanym będzie zdanie poprzedzone etykietą e1, w przeciwnym przypadku będzie wykonane zdanie poprzedzone etykietą e2.
Jeżeli wyrażenia a i b są różnych typów to przed porównaniem, wartość wyrażenia całkowitego jest sprowadzana do postaci ułamkowej.
Jeżeli oba wyrażenia mają wartości równe zero, to niezależnie od znaków (zera mogą być dodatnie i ujemne) uważa się relację za niespełnioną.

Znaki > i] są równoważne i można używać dowolnego z nich.

Przykład 1

```
   -------
1) I=I+1
   -------
   GDY A > .07 : 1, INACZEJ 2
   -------
   -------
2) I=I-1
```

Przykład 2

```
    GDY SIN (X) > 0 : NASTEPNY, INACZEJ 1A
    --------
    --------
    --------
1A) --------
```

Przykład 3

Poniższy przykład wybiera maksymalną wartość z bloku A:

```
    CALKOWITE : I
    BLOK (7) : A
    ---------
    ---------
    I=0                 (ustawienie wartości początkowej)
3)  Z = A ( I )         (najpierw Z = A (0))
2)  I = I + 1           (zwiększenie wartości I o 1)
    GDY I > 7 : 1,  INACZEJ NASTEPNY
                        (zostanie wykonane następne zdanie dla
                         I=1,2,...,7)
    GDY A (I) > Z : 3, INACZEJ 2
1)  ---------           (zdanie 3 zostanie wykonane tylko wtedy,
    ---------            gdy element A (I) jest większy od
                         wszystkich poprzednich)
```

GDY A = B

Postać ogólna
GDY a = b : e1, INACZEJ e2 gdzie: a i b są wyrażeniami arytmetycznymi, e1, e2 są etykietami

Zdanie to powoduje obliczenie wartości wyrażeń arytmetycznych a i b a następnie porównanie ich.
Jeżeli wartości ich są równe to będzie wykonywane-jako następne - zdanie poprzedzone etykietą e1, jeżeli natomiast wartości tych wyrażeń będą różne, to następnym wykonywanym zdaniem będzie to, które jest poprzedzone etykietą e2.
Jeżeli wyrażenia a i b są różnych typów to wartość wyrażenia całkowitego jest sprowadzana do postaci ułamkowej.
W przypadku gdy oba wyrażenia mają wartości równe zero, relacja jest spełniona niezależnie od ich znaków.
Ze względu na akumulację błędów przy liczeniu wartości wyrażeń arytmetycznych ułamkowych, warunek a = b (dla ułamkowego a lub b) może nie być nigdy spełniony.

Przykład 1

```
   -------
   -------
   GDY A-B/C=0 : 77, INACZEJ NASTEPNY
   -------
   -------
77)-------
```

Przykład 2

```
   -------
   GDY DOKL (I,J) = EPSYLON : 1, INACZEJ 2
   -------
1) -------
2) -------
```

Przykład 3

```
   CALKOWITE : I
   BLOK (4) : A
   I=0                    (nadanie wartości początkowej)
1) A (I) = 0              (nadanie wartości zero zmiennej
                           indeksowanej)
   GDY I = 4 : 2, INACZEJ NASTEPNY
   I=I+1                  (zwiększenie wartości I o 1)
   SKOCZ DO 1             (przejście do wykonania zdania 1)
2) ----------
```

Ten przykład ilustruje jak można przy pomocy zdania GDY A=B spowodować wielokrotne wykonanie odcinka programu. Powyższy kawałek programu powoduje nadanie wartości początkowych równych zero wszystkim zmiennym bloku A (wyzerowanie bloku A).

SKOCZ WEDLUG

Postać ogólna
SKOCZ WEDLUG a : e0, e1,...,en gdzie: a jest wyrażeniem arytmetycznym całkowitym, e0, e1,...,en jest listą etykiet rozdzielonych przecinkami. Listę etykiet można przenosić.

Przyporządkujmy kolejnym etykietom na liście etykiet kolejne numery, poczynając od zera.
Jeżeli teraz, wartością wyrażenia arytmetycznego a jest liczba całkowita i to wynik wykonania zdania

SKOCZ WEDLUG a : e0, e1,...,ei,...,en

będzie równoważny wykonaniu zdania

SKOCZ DO ei

Wyrażenie arytmetyczne całkowite a musi mieć wartość nie mniejszą od zera i nie większą od ilości etykiet na liście, w przeciwnym przypadku wynik wykonania tego zdania jest nieokreślony.

Przykład

```
    ---------
1A) ---------        (dla I=1 lub dla I=6 jako następ-
    ---------         ne po SKOCZ WEDLUG wykona się
    ---------         zdanie 1A))
    SKOCZ WEDLUG I-1 : 1A, NASTEPNY, 1,2A7,-
                       NASTEPNY, 1A
 2) ---------
2A7) ---------       (dla I=2 lub I=5 wykona się
 1) ---------         zdanie 2 tj. następne)
```

POWTÓRZ

Postać ogólna

POWTORZ [OD e1] : i = j (k) l

gdzie : e1 jest etykietą,

i jest zmienną prostą tzw. kontrolującą,

j jest zmienną prostą albo liczbą bez zanku lub ze znakiem i nazywa się wartością początkową,

k jest wyrażeniem arytmetycznym i nazywa się krokiem powtarzania,

l jest wyrażeniem arytmetycznym i nazywa się granicą powtarzania.

Zwrot: OD e1 można pominąć gdyż nie ma wpływu na treść zdania.

Wskaźnik zasięgu powtarzania

Z każdym zdaniem POWTORZ związany jest tzw. wskaźnik zasięgu powtarzania, który składa się z dwóch znaków tj. gwiazdki i nawiasu zamykającego (*)); musi on wystąpić w programie wcześniej (w sensie statycznym) niż związane z nim zdanie POWTORZ.

Wskaźnik zasięgu powtarzania i zdanie POWTORZ stanowią pewien rodzaj nawiasów, obejmujących ciąg zdań podlegających powtarzaniu. Ciąg zdań zawarty między zdaniem POWTORZ i odpowiadającym mu wskaźnikiem zasięgu powtarzania, będziemy nazywać zasięgiem powtarzania. W szczególnym przypadku w zasięgu powtarzania może znaleźć się inne zdanie POWTORZ wraz ze swym wskaźnikiem zasięgu i zasięgiem powtarzania w tym przypadku będziemy mówić, że zasięgi zawierają się.

Wskaźnik zasięgu powtarzania (lub ciąg wskaźników) może być napisany w oddzielnym wierszu lub też w tym samym wierszu co zdanie. Jeżeli wskaźnik (lub ciąg wskaźników) występuje (występują) w tym samym wierszu co zdanie języka to musi (muszą) występować z lewej strony zdania.

Dodatkowo wprowadzony został skrócony zapis ciągu wskaźników i wtedy wystarczy po ciągu gwiazdek postawić jeden nawias zamykający. Np. zamiast pisać *) *) *) można napisać * * *) . Ponadto, jeżeli ciąg wskaźników powinien wystąpić po etykiecie, można go zastąpić ciągiem gwiazdek przed etykietą i wszystkie nawiasy związane ze wskaźnikiem pominąć. Np. zapis * * 1A) jest równoważny jednemu z poniższych zapisów: 1A)* * lub 1A)*) *)

Przykłady

Poprawne tworzenie zasięgów powtarzania:

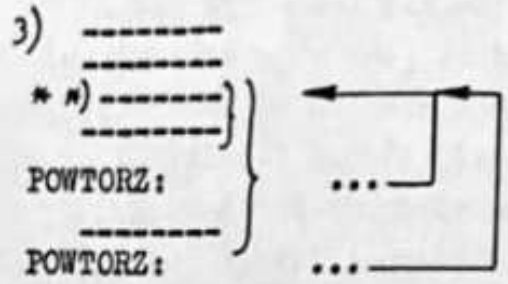
1)

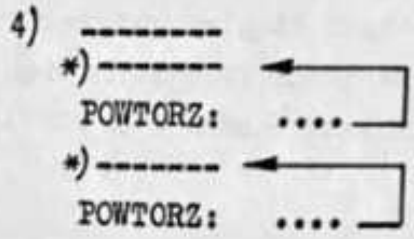
```
       -------
       -------
       -------
       *)------ }
        ------- }
        ------- }      zasięg powtarzania
        ------- }
        ------- }
POWTORZ : I = J (K) L
       -------
       -------
```

2)

```
       -------
       -------
       *)------- }
        ------- }
       *)------- }
        ------- }
        ------- }
POWTORZ: ...
       -------
POWTORZ: ...
       -------
```

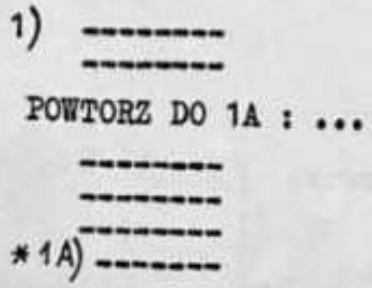
(zawierające się zasięgi powtarzania)



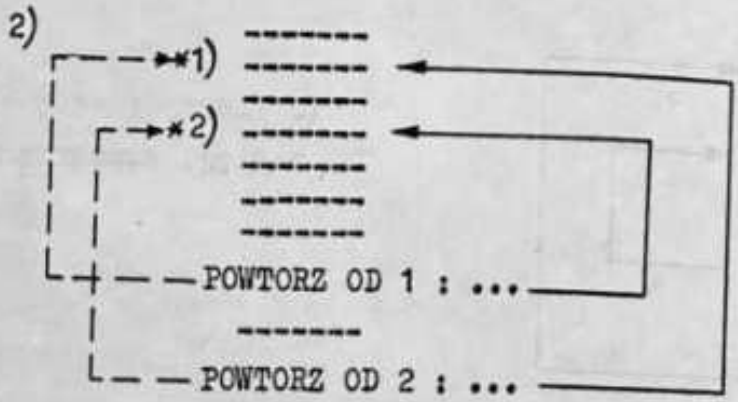


(rozłączne zasięgi powtarzania)

Błędne tworzenie zasięgów powtarzania:



(wskaźnik zasięgu winien występować wcześniej niż odpowiednie zdanie POWTORZ)



Parametry 1 i 2 w zdaniach POWTORZ są nieistotne i powstaną zasięgi podane linią ciągłą a nie tak jak sugerowałyby odpowiednie etykiety.

Zdanie POWTORZ OD e1 : i = j k l powoduje wielokrotne wykonanie co najmniej jednokrotne odcinka programu zwanego zasięgiem powtarzania. Pierwszy raz dany odcinek programu wykonuje się dla wartości i = j, każde następne wykonanie zasięgu odbywa się dla wartości i zwiększonej o wartość wyrażenia k. Gdy wartość zmiennej i "osiągnie" wartość wyrażenia L, to proces powtarzania będzie przerwany i przechodzi się do wykonania zdania napisanego bezpośrednio po POWTORZ.

W zdaniu POWTORZ wszystkie wielkości: i, j, k, l muszą być jednocześnie całkowite lub jednocześnie ułamkowe. W pierwszym przypadku będziemy mówili, że POWTORZ jest całkowite a w drugim, że ułamkowe.

Całkowite POWTORZ

Zdanie POWTORZ całkowite można dokładnie zdefiniować przy pomocy poprzednio opisanych zdań.

Odcinek programu:

```
        -------
     *) -------}
        -------}        (zasięg powtarzania)
        -------}
        POWTORZ : i = j (k) l
        -------
```

jest równoważny następującemu odcinkowi:

```
        -------
        i = j           (zamiast wskaźnika zasię-
                         gu powtarzania)
     e1) -------}
         -------}       zasięg powtarzania
         -------}
         GDY i = l : e2, INACZEJ NASTEPNY  }(zamiast
         i = i + k                         }POWTORZ)
         SKOCZ do e1                       }
     e2) -------
```

gdzie: e1 i e2 są unikalnymi etykietami w całym programie wygenerowanymi przez translator i niemożliwymi do napisania przez programistę.

Na podstawie powyższej definicji można wysnuć kilka wniosków dotyczących całkowitego POWTORZ:

1) Zasięg powtarzania musi być wykonywany przynajmniej raz.
2) Niezależnie od kroku powtarzania, jeżeli wartość początkowa jest równa wartości końcowej, to zasięg powtarzania wykonywany będzie tylko raz.
3) Jeżeli wartość zmiennej kontrolującej nie osiągnie dokładnie wartości granicy powtarzania, to zasięg powtarzania może wykonywać się bez końca.
4) Wykonywanie zdań z zasięgu powtarzania bez nadania wartości początkowej dla zmiennej kontrolującej jest niedozwolone (np. wykonanie zdania SKOCZ do wnętrza zasięgu powtarzania).
5) W przypadku zawierających się zasięgów powtarzania zmienne kontrolujące nie mogą być identyczne.

Przykłady

Poprawne użycie zdania POWTORZ całkowite:

```
CALKOWITE : I
BLOK (4) : A
*)A (I) = 0
POWTORZ : I = 0 (1) 4
-------
```

Powyższy przykład jest odcinkiem programu równoważnym odcinkowi podanemu w przykładzie 3 opisu zdania GDY A = B.

```
CALKOWITE : I, J, K, N, M
BLOK (3,3) : A,B
--------
--------          (zakładamy, że element bloku B
--------           posiadają wartości)
 S = 0
 S = S + B (I,K) ◊ B (J,K)    (S = Σ_{k=0}^{3} b_ik x b_jk  dla
 POWTORZ : K=0 (1) 3
 A (I,J) = S                   ustalonych i,j)
 A (J,I) = S                 (a_ij = S)
                             (a_ji = S)
 POWTORZ : J = 0 (1) I   (zmiana wartości J i I)
 POWTORZ : I = 0 (1) 3
 ---------
```

W rezultacie został policzony wzór

$$a_{ij} = \sum_{k=0}^{3} b_{ik} \times b_{jk}$$

Błędne użycie zdania POWTORZ całkowite:

```
1)  CALKOWITE : I
    BLOK (4) : A
    J = 4
   *) A (I) = 0
    POWTORZ : I = 0.(1) J     (J oraz 0. są ułamkowe)
    --------

2)  CALKOWITE : I
    BLOK (4) : H
    SKOCZ DO 3
    --------
    --------
*) 3) A(I) = 0
    POWTORZ : I = 0 (1) 4
```

Zapis: *)jest równoważny zdaniu I = 0, stąd SKOCZ DO 3 omija nadanie wartości początkowej dla I. Ale mielibyśmy poprawny program gdyby zamiast *) 3) było napisane * 3) co jest równoważne zapisowi 3)*)

```
3) ---------
   CALKOWITE : I
  *4) GDY I = 2 : 5A, INACZEJ 6
      5A) I = I + 1
      SKOCZ DO 4
    6) POWTORZ OD 4 : I = 0 (1) 4
   ---------
   ---------
```

Zamieszczony na poprzedniej stronie (przykład drugi) odcinek programu będzie wykonywał się bez końca, gdyż przy I = 2 wykona się zdanie 5A a następnie SKOCZ DO 4, które spowoduje wykonanie operacji I = 0 (zapis *4) jest równoważny zapisowi 4) I = 0).

Ułamkowe POWTORZ

Dokładna definicja znaczenia ułamkowego POWTORZ jest następująca:

```
odcinek programu:
---------
*)--------
---------
---------
POWTORZ : i = j (k) l
---------
```

jest równoważny odcinkowi:

```
---------
  i = j
e1) -------
    -------
    -------
GDY ABS (k/2) > ABS (i-l) : e2, INACZEJ NASTEPNY
    i = k + i
    SKOCZ DO e1
e2) --------
```

gdzie: ABS jest funkcją obliczającą wartość bezwzględną argumentu: i, j, k, l są wielkościami ułamkowymi; e1, e2 są unikalnymi etykietami.

Warunek w zdaniu GDY w powyższej definicji został wprowadzony, albowiem liczby ułamkowe w pamięci maszyny są obarczone pewnym błędem związanym z ich reprezentacją binarną i warunek użyty przy definicji całkowitego POWTORZ w ogólnym przypadku jest niemożliwy do spełnienia. Użyty zwrot "osiągnie" - w ogólnym opisie zdania POWTORZ - oznacza równość w przypadku całkowitych parametrów i spełnienie warunku wyżej omawianego w przypadku parametrów ułamkowych.

Wnioski 1,2,4 i 5 związane z całkowitym POWTORZ są również słuszne i dla ułamkowego, natomiast proces powtarzania dla parametrów ułamkowych będzie zakończony wtedy, gdy bezwzględna wartość różnicy wartości zmiennej kontrolującej i granicy powtarzania będzie mniejsza od bezwzględnej wartości połowy kroku powtarzania.

Przykłady

Poprawne użycie ułamkowego POWTORZ:

```
* 1)  --------
      --------
      POWTORZ OD 1 : A = 0.1 (A1 ◇ (7 - B)) C + 1
      --------
      --------        i   j      k        l

  2)  --------
    *)--------
      --------
      --------
      POWTORZ : Z = -3. (1.) 0.
      --------

* 3)  --------
      --------
      GDY A > B : 1A, INACZEJ NASTEPNY
      --------
      --------
      POWTORZ OD 3 : L = 0. (-1.) 2.
      --------
      --------
    1A)-------
```

Mimo, że granica powtarzania nigdy nie zostanie osiągnięta, to proces powtarzania może być przerwany gdy będzie spełniona relacja A > B.

```
*4) -------}   (ten odcinek programu będzie zawsze
    -------}    powtarzany)

    GDY A > B : 1A, INACZEJ NASTEPNY

3B) -------}   (przy A ≤ B ten odcinek programu będzie
    -------}    powtarzany)

1B) POWTORZ OD 4 : A = C (SIN (X)) C + 7
    -------
    SKOCZ DO 2B
1A) -------    (przy A > B będzie powtarzany ten
    -------     odcinek programu)
    SKOCZ DO 1B
2B) -------
```

Błędne użycie zdań POWTORZ:

```
*) -------
   POWTORZ : A = - B (C)  + 1
   --------
```

Wartością początkową nie może być zmienna ze znakiem oraz + 1 jest całkowite.

```
   SKOCZ DO 1A     (wejście do wnętrza zakresu powtarzania,
*) ----------       A nie posiada wartości początkowej)
1A) POWTORZ : A = B (C)  D
```

```
*1) --------
*2) --------
    --------
2A) POWTORZ OD 2 : A = 0. (1.) 3.
1A) POWTORZ OD 1 : A = 0. (2.) 10.
    --------
    --------
```

Przy zawierających się zasięgach, zmienne kontrolujące powinny być różne; w tym przypadku proces powtarzania nigdy się nie skończy.

ZDANIE PRZYPISUJĄCE WIELOKROTNIE

Postać ogólna
$\underline{z}\left(\left\{\begin{matrix}\underline{l1}\\ =i2\end{matrix}\right\}, \left\{\begin{matrix}\underline{l2}\\ =i2\end{matrix}\right\}, \ldots, \left\{\begin{matrix}\underline{ln}\\ =in\end{matrix}\right\}\right) = \underline{w}$ gdzie: $\underline{z}$ jest nazwą bloku (tablicy, struktury), $\underline{l1}, \underline{l2}, \ldots, \underline{ln}$ są to indeksy-wyrażenia arytmetyczne całkowite $\underline{i1}, \underline{i2}, \ldots, \underline{in}$ są to zmienne proste całkowite zwane zmiennymi kontrolującymi $\underline{w}$ jest wyrażeniem arytmetycznym

Znaczenie tego zdania można określić przy pomocy ciągu zdań przypisujących arytmetycznych oraz ciągu zdań POWTORZ. Załóżmy, że zakresy indeksów bloku $\underline{z}$ mają wartości równe odpowiednio $\underline{d1}$, $\underline{d2}, \ldots, \underline{dn}$ oraz że istnieją zmienne całkowite $\underline{r1}$, $\underline{r2}, \ldots, \underline{rn}$ o postaci niemożliwej do napisania przez programistę.

Jeżeli teraz na odpowiedniej pozycji listy indeksów występuje wyrażenie arytmetyczne to piszemy zdania przypisujące:

$$\underline{ri} = \underline{li}$$

Jeżeli zaś na odpowiedniej pozycji listy indeksów występuje zmienna kontrolująca poprzedzona znakiem równości to w kolejnym wierszu programu piszemy wskaźnik zakresu powtarzania tj.

*)

Po napisaniu takiego ciągu zdań, zdanie wielokrotnego przypisania przekształcamy do postaci:

$$\underline{z}\left(\left\{\begin{matrix}r1\\ i1\end{matrix}\right\}, \left\{\begin{matrix}r2\\ i2\end{matrix}\right\}, \ldots, \left\{\begin{matrix}rn\\ in\end{matrix}\right\}\right) = \underline{w}$$

tj. tworzymy zwykłe zdanie przypisujące arytmetyczne.

Następnie poczynając od końca listy indeksów dla wszystkich zmiennych kontrolujących piszemy zdania POWTORZ o następującej postaci:

POWTORZ : ii = di (-1) 0

Np. zdanie wielokrotnie przypisujące

z (l1, = i2, = i3) = w

jest równoważne odcinkowi programu

r1 = l1

*)
*)
z (r1, i2, i3) = w

POWTORZ : i3 = d3 (-1) 0
POWTORZ : i2 = d2 (-1) 0

Dalej, przy założeniu, że w wyrażeniach li nie występują zmienne kontrolujące (założenie może być nieco łagodniejsze) zdanie ri = li można pominąć i definicję powyższą

*) *) z (l1, i2, i3) = w

POWTORZ : i3 = d3 (-1) 0
POWTORZ : i2 = d2 (-1) 0

Dla dość dużej ilości zastosowań zdań wielokrotnego przypisywania, kolejność powtarzania i zmiany wartości zmiennych kontrolujących nie mają znaczenia. W tych przypadkach nie musimy pamiętać o kolejności zmian wartości zmiennych kontrolujących a zatem o kolejności nadawania wartości zmiennym indeksowanym.

Należy tylko pamiętać, że zmienne kontrolujące przyjmują wszystkie możliwe wartości od zera do wartości zakresów indeksów zadeklarowanych lub obliczonych przy pomocy zdań: BLOK, STRUKTURA, TABLICA.

Na podstawie definicji zdania wielokrotnego przypisania można wysnuć wnioski:

1. Zmienne kontrolujące muszą być różne gdyż w przeciwnym przypadku mamy do czynienia ze zdaniami POWTORZ

o zawierających się zakresach powtarzania i takich samych zmiennych kontrolujących (patrz opis zdania POWTORZ).

2. Jeżeli na liście indeksów występują tylko zmienne kontrolujące, to zdanie takie powoduje nadanie wartości wszystkim elementom bloku.

3. Jeżeli na liście indeksów nie występują zmienne kontrolujące, to takie zdanie jest zwykłym zdaniem arytmetycznym przypisującym.

4. Jeżeli wyrażenia arytmetyczne na liście indeksów zawierają zmienne kontrolujące, to dla określenia znaczenia należy stosować pełną definicję zdania a nie uproszczoną.

Przykłady:

Poprawne użycie zdań wielokrotnego przypisania:

1)
```
CALKOWITE: I
BLOK (4) : A
A (=I) = 0
```

Powyższy odcinek programu jest równoważny programowi

```
CALKOWITE: I
BLOK (4) : A
*)A (I) = 0
POWTORZ : I = 4 (-1)  0
```

2)
```
CALKOWITE : I,J
BLOK (3,4) : A,B,C,D
A (=I,=J) = 0          (przesłanie zer do wszystkich
                        elementów bloku A)
B (2,=J) = 1           (nadanie wszystkim elementom
                        wiersza 2 wartości równej 1)
C (=I,1)  = I          (nadanie kolejnych wartości
                        0,1,2,3 kolejnym elementom
                        kolumny 1 macierzy C).
D (=J, =I) = A (J,I) + B (J,I)
                       (każdy element macierzy D otrzyma
                        wartość równą sumie odpowiednich
                        elementów macierzy A i B)
```

3) CALKOWITE : I, J
BLOK (3,3) : A
A (= I, I) = I
Zdanie A (=I,I) = I jest równoważne odcinkowi programu:

*) r2 = I
A (I,r2) = I
POWTORZ : I = 3 (-1) 0

i podstawiając r2 = I otrzymamy odcinek programu równoważny

*) A (I,I) = I
POWTORZ : I = 3 (-1) 0

Należy zauważyć, że podstawienie było możliwe gdyż zdanie r2 = I oraz A (I,r2) = I znajdują się wewnątrz zakresu POWTORZ. (Patrz kontrprzykłady 4 i 5).

Błędne użycie zdania wielokrotnego przypisania:

4) CALKOWITE : I
BLOK (3,3) : A
A (I, =I) = I

Ponieważ zapis A (I,=I) = I jest równoważny:

r1 = I
*) A (r1,I) = I
POWTORZ : I = 3 (-1) 0

to w zdaniu r1 = I zmienna I jest nieokreślona gdyż pozostaje poza zakresem zdania POWTORZ.

5) CALKOWITE : I,J
BLOK (3,3) : A
A (=I, =I) = 0 (identyczne zmienne kontrolujące)

GDY BYL NADMIAR

Postać ogólna
GDY BYL NADMIAR : e1, INACZEJ e2 gdzie : e1 i e2 są etykietami

Zdanie powyższe powoduje zbadanie wskaźnika nadmiaru stałoprzecinkowego i jeżeli wskaźnik jest ustawiony w stan TAK, to ustawia się go w stan NIE i wykonywane jest przejście do zdania o etykiecie e1. Jeżeli wskaźnik nadmiaru stałoprzecinkowego jest ustawiony w stan NIE to zdanie e2 zostanie wykonane jako następne.
Wskaźnik nadmiaru pozostanie nadal w stanie NIE.

Należy zauważyć, że wskaźnik nadmiaru stałoprzecinkowego przed rozpoczęciem pracy programu jest zawsze ustawiony w stan NIE (nadmiar nie wystąpił) . Wskaźnik ustawiony jest w stan TAK w następujących przypadkach:

1. Przy obliczaniu wartości wyrażenia arytmetycznego całkowitego gdy ostateczny wynik obliczeń lub wyniki pośrednie przekroczyły zakres przewidziany dla stałych dziesiętnych całkowitych,
2. przy zamianie liczby ułamkowej na całkowitą gdy wynik zamiany przekracza zakres przewidziany dla liczb całkowitych,
3. przy obliczaniu wartości niektórych funkcji języka Konkretne przypadki są omówione w DODATKU B.

Jeżeli wskaźnik nadmiaru jest ustawiony w stan TAK, to ponowne przekroczenie zakresu nie powoduje zmiany stanu wskaźnika. Ustawienie wskaźnika w stan NIE jest możliwe tylko przy pomocy zdania GDY BYL NADMIAR.

Przykłady

```
1)   CALKOWITE: A,B,C
     ---------
     ---------
     ---------
     GDY BYL NADMIAR: NASTEPNY, INACZEJ NASTEPNY
     ---------        (powoduje ustawienie wskaźnika nad-
     ---------         miaru w stan NIE niezależnie w ja-
     ---------         kim był stanie poprzedni  bez zmia-
     ---------         ny kolejności wykonywania zdań
     ---------         programu)
     A = B + C
     GDY BYL NADMIAR: NASTEPNY, INACZEJ 1E
     A = B/2 + C/2    (przy liczeniu wzoru:
1E)  ---------         (B+C)/2 nastąpiłoby również przekro-
                       czenie zakresu)
2)   CALKOWITE: A,B,C
     ---------
     ---------
5)   A = B + C
6)   B = B   C
7)   C = A/B + A
8)   GDY BYL NADMIAR: 1, INACZEJ 3
     ----------
     ----------
1)   ----------
3)   ----------
```

W powyższym przykładzie wystarczy aby przy wykonywaniu jednego ze zdań 5,6,7 nastąpiło przekroczenie zakresu, żeby po zdaniu 8 wykonało się bezpośrednio zdanie 1.

GDY KLUCZ

Postać ogólna
GDY KLUCZ w : e1, INACZEJ e2 gdzie: w jest wyrażeniem arytmetycznym całkowitym, które może przyjmować wartości od zera do 23 e1, e2 są to etykiety.

Wyrażenie arytmetyczne w wskazuje numer klucza w tzw. rejestrze kluczy, który jest badany w trakcie wykonywania zdania GDY KLUCZ. Jeżeli klucz o numerze w jest w stanie 1, to jako następne po omawianym zdaniu będzie wykonywane zdanie poprzedzone etykietą e1; w przeciwnym przypadku zostanie wykonane jako następne zdanie e2.

Rejestr kluczy posiada 24 pozycje ponumerowane od 0 do 23, każda z pozycji może przyjąć stan zero lub 1. Ustawianie rejestru kluczy może odbywać się na jeden z dwóch sposobów:

- poprzez wstępne ustawienie rejestru kluczy przed rozpoczęciem wykonywania się programu wynikowego przy pomocy zdania KLUCZE opisanego w drugiej części podręcznika,
- przez operatora podczas pracy programu wynikowego za pośrednictwem monitora.

Sposób zmiany stanu rejestru kluczy będzie opisany w paragrafie pt. "Zasady Monitowania" część II podręcznika.

Zdanie GDY KLUCZ pozwala operatorowi maszyny oddziaływać na przebieg pracy programu wynikowego lub też na wcześniejsze ustalenie warunków określających sposób jego działania.

Przykłady

```
          ----------
          ----------
          ----------
          GDY KLUCZ 7 : 1A, INACZEJ 1B
          ----------
          ----------
     1A)  --------
          ----------
     1B)  --------
```

Gdy stan klucza numer 7 jest 1, to jako następne po zdaniu GDY KLUCZ zostanie wykonane zdanie 1A, w przeciwnym przypadku zostanie wykonane zdanie 1B.

```
          CALKOWITE: I
          ----------
1 KLUCZ)  ----------
          GDY KLUCZ I-1 : 1KLUCZ, INACZEJ 2KLUCZ
          ----------
          ----------
          ----------
2 KLUCZ)  ----------
          ----------
```

STOP

Postać ogólna
STOP [:] e1 gdzie : e1 jest etykietą; znak dwukropka jest nieistotny

Zdanie to musi być ostatnim wykonywanym zdaniem programu i powoduje zakończenie pracy programu.

Zdanie STOP może być używane w programie wielokrotnie. W zależności od dodatkowego życzenia, translator może tak przetłumaczyć zdanie STOP na program wynikowy, że ostateczną decyzję, czy praca programu wynikowego ma być definitywnie zakończona czy też kontynuowana, poczynając od etykiety e1, pozostawia się operatorowi. W tym przypadku następuje zawieszenie pracy programu aż do chwili interwencji operatora. To dodatkowe życzenie sygnalizowane jest przy pomocy zdania LIKWIDUJ STOP opisanego w części II podręcznika w paragrafie: Czołówka translatora.

Mimo braku żądania co do zawieszenia pracy programu wynikowego, etykieta e1 w zdaniu STOP musi wystąpić (nie musi występować po lewej stronie) , gdyż w opisie przyczyn zakończenia programu wykorzystuje się ją w celu poinformowania, które ze zdań STOP było wykonane w danym przypadku (patrz POST MORTEM - opis ZAKO; część II podręcznika).

Przykład

```
        ---------
        ---------
        ---------
        STOP 1ETY
        ---------
1ETY)   ---------
        ---------
        ---------
        ---------
        STOP NASTEPNY
        ---------
 1A)    ---------
        ---------
        STOP:1A
        ---------
        ---------
        ---------
        STOP 1A
        ---------
        ---------
        STOP NASTEPNY
        ---------
        ---------
        ---------
```

ZDANIA WEJSCIA-WYJSCIA

Do tej grupy zaliczane są zdania wprowadzające dane napisane w postaci znakowej oraz zdania wypisujące wyniki programu również w postaci ciągu znaków. Będziemy je niekiedy nazywać zdaniami współpracy z urządzeniami zewnętrznymi.

Zdania wejścia-wyjścia nie posiadają parametru określającego z jakim urządzeniem zewnętrznym pracują oraz nie określa się w nich, jaki jest kod danego urządzenia. Zakłada się, że dane zdanie czyta dane z bieżącego urządzenia symbolicznego wejściowego tzn. z urządzenia symbolicznego o numerze wybranym przez funkcję języka SEL lub z urządzenia symbolicznego o numerze zero,jeżeli funkcji SEL nie używano (patrz opis funkcji języka SEL,DOL,ODL; DODATEK B) . Ponadto zakłada się, że dane dostarczane przez urządzenie symboliczne są w kodzie KW6 (patrz tabela kodu KW6; DODATEK A) .

Podobnie zdania wyjściowe wypisują wyniki na bieżące urządzenie symboliczne wyjściowe o numerze wybranym przez funkcję SEL lub na zerowe urządzenie symboliczne wyjściowe jeżeli funcji SEL nie używano. Wyniki wypisywane są zawsze w kodzie KW6.

Zależności między numerami urządzeń symbolicznych i rzeczywistymi urządzeniami wejścia-wyjścia są ustalane w czołówce systemu. Ponadto określa się w czołówce systemu, jaki jest kod rzeczywisty danych lub jaki ma być kod rzeczywisty wyników na danym rzeczywistym urządzeniu związanym z odpowiednim symbolicznym urządzeniem. Pozwala to systemowi przekodować odpowiednie dane (wyniki) na KW6 (z KW6) ; patrz Czołówka Systemu; II część podręcznika .

W ten sposób program napisany w języku SAKO jest niezależny od urządzeń, zewnętrznych, jest uniwersalny. Wyboru urządzeń można dokonać dopiero w momencie przekazywania programu do wykonania.

DRUKUJ

Postać ogólna
DRUKUJ (a1 [,a2 [,a3]]) : b1, b2,...,bn gdzie: a1,a2,a3 są to wyrażenia arytmetyczne całkowite zwane parametrami drukowania b1,b2,...,bn są to wyrażenia arytmetyczne, których wartości należy drukować i stanowią listę wyrażeń. Listę można przenosić.

Zdanie DRUKUJ o podanych parametrach drukowania z listą zawierającą więcej niż jedno wyrażenie, jest równoważne (co do znaczenia) ciągowi zdań DRUKUJ, z których każde ma takie same parametry drukowania jak omawiane zdanie i na liście po jednym wyrażeniu arytmetycznym, tj.

DRUKUJ (a1 [,a2 [,a3]]): b1, b2,...,bn

jest równoważne ciągowi zdań:

DRUKUJ (a1 [,a2 [,a3]]): b1
DRUKUJ (a1 [,a2 [,a3]]): b2
⋮
DRUKUJ (a1 [,a2 [,a3]]): bn

Stąd, nie tracąc na ogólności rozważań, będziemy rozpatrywać zdanie z listą zawierającą tylko jedno wyrażenie, pamiętając jednak, że może być ich więcej.

Znaczenie zdania DRUKUJ zależy od ilości jego parametrów, i tak:

- jeżeli zdanie posiada jeden parametr to będziemy go nazywać drukowaniem całkowitym,
- jeżeli zdanie posiada dwa parametry to będziemy go nazywać drukowaniem ułamkowym stałoprzecinkowym,
- jeżeli zdanie posiada trzy parametry to będziemy go nazywać drukowaniem zmiennoprzecinkowym.

Drukuj całkowite

Zdanie drukowania całkowitego ma postać:

DRUKUJ (a1) : b1

i powoduje obliczenie wartości wyrażenia arytmetycznego b1 (wyrażenie to musi być całkowite) oraz wypisanie jej na bieżące urządzenie symboliczne.

Liczba drukowana ma zająć na arkuszu wydawniczym a1+1 miejsc; a1 pozycji zajmie bezwzględna wartość liczby oraz jedną pozycję rezerwuje się na znak.

Znak wypisuje się bezpośrednio przed pierwszą cyfrą znaczącą. Pisze się -(minus) dla liczb ujemnych i spację (odstęp) zamiast znaku + dla liczb dodatnich. Miejsca cyfr nieznaczących uzupełnia się z przodu liczby (przed znakiem) spacjami.

Znak wyrażenia a1 jest nieistotny a bezwzględna wartość powinna mieścić się w przedziale od 1 do 50; dla a1 większych od 50 przyjmuje się, że a1 jest równe 50.

Jeżeli liczba drukowana nie mieści się w przedziale zadanym przez wartość e1, to mimo wszystko liczba jest drukowana i drukuje się ją w minimalnej ilości potrzebnych pozycji. Fakt powyższy można wykorzystać żądając celowo zbyt małej ilości miejsc na wydrukowanie liczby, aby niezależnie od jej wielkości pozycje dla cyfr nieznaczących nie były rezerwowane.

Przykłady

Poprawne użycie drukowania całkowitego:

1) CALKOWITE : A,B

DRUKUJ (3) : 5

(zdanie to spowoduje wypisanie liczby 5 w postaci: ␣ ␣ ␣ 5 gdzie ␣ oznacza spację)

DRUKUJ (4) : 5,17,-241,-
4321,100

(zdanie to spowoduje wypisanie kolejno: ␣ ␣ ␣ ␣ 5 ␣ ␣ ␣ 17 ␣ -241 ␣ 4321 ␣ ␣ 100)

B=12

A=3

DRUKUJ (A) : 5+B

(zostanie obliczona wartość wyrażenia 5+B a następnie wydrukowana)

DRUKUJ (1) : -4, 125, 3456, - 1234

---

liczby te będą wydrukowane w postaci:
-4 ␣ 125 ␣ 3456-1234

DRUKUJ (100) : 1234

(liczba 1234 będzie poprzedzona 47 spacjami)

Błędne użycie drukowania całkowitego:

DRUKUJ (D) : 4

(D - ułamkowe)

A = 4321
DRUKUJ (5) : A (A jest ułamkowe, ale dobrze by było DRUKUJ (5) : 4321)

DRUKUJ (5) : 8000 000 + 7000 000

(wyrażenie przekracza zakres, wskaźnik nadmiaru będzie ustawiony w stan TAK a drukowany wynik będzie błędny)

Drukowanie ułamkowe stałoprzecinkowe

Drukowanie ułamkowe stałoprzecinkowe ma postać następującą:

DRUKUJ (a1, a2) : b1

i powoduje obliczenie wartości wyrażenia arytmetycznego b1 a następnie wypisanie jej na bieżące urządzenie symboliczne.

Wyrażenie arytmetyczne całkowite a1 oznacza ilość rezerwowanych pozycji arkusza wydawniczego na część całkowitą liczby drukowanej a wyrażenie arytmetyczne całkowite a2 oznacza ilość cyfr części ułamkowej. Bezpośrednio przed liczbą wypisywany jest znak liczby: - dla liczb ujemnych i + dla liczb dodatnich. Część całkowita liczby jest oddzielona kropką od części ułamkowej; przed wydrukowaniem liczby następuje zaokrąglenie. Jeżeli żąda się zera cyfr części ułamkowej (wypisywanie z dokładnością do części całkowitej) to kropka nie jest pisana i liczba zajmie na arkuszu wydawniczym a1 + 1 pozycji: w pozostałych przypadkach liczba zajmie a1 + a2 + 2 miejsca.

Zera nieznaczące w części całkowitej zastępowane są spacjami, z tym że jeżeli część całkowita jest zerem to jedno zero jest pisane.

W przypadku gdy a1 jest zerem to znak liczby występuje tuż przed kropką,

Przypadek gdy a1 i a2 są jednocześnie zerami jest niedozwolony, każda z wartości a1 i a2 może przyjmować wartości od zera do 50, jednakże sensownych cyfr znaczących w liczbie ułamkowej jest 12.

W przypadku gdy wartość wyrażenia b1 nie mieści się w żądanym przedziale określonym wartością a1 to liczba jest wypisywana w postaci ułamkowej zmiennoprzecinkowej, przy niezmienionych a1 i a2 rezerwuje się dodatkowo 4 pozycje na wykładnik (patrz drukowanie zmiennoprzecinkowe).

Jeżeli b1 jest wyrażeniem arytmetycznym całkowitym to jest sprowadzone do postaci ułamkowej.

Przykłady

Poprawne użycie drukowania stałoprzecinkowego:

1) DRUKUJ (3,2) : 24.789
(wypisane zostanie w postaci: ␣+24.79)

DRUKUJ (3,2) : -1.744
(wypisane zostanie w postaci: ␣␣-1.74)

DRUKUJ (3,2) : 0.002
(wypisane zostanie w postaci: ␣␣+0.00)

DRUKUJ (3,0) : 24.789
(wypisane zostanie w postaci: ␣+25)

DRUKUJ (0,3) : 0.02
(wypisane zostanie w postaci: ␣+.020)

DRUKUJ (20,1): -3.15
(wypisane zostanie w postaci: ␣␣....␣␣ -3.2)
19 spacji

DRUKUJ (3,2) : 7
wypisane zostanie w postaci: ␣␣+7.00

2) DRUKUJ (4,2) : (5 + PWK(5 ◇ 5 - 4 ◇ 1 ◇ 6)) /2
wartość wyrażenia będzie wypisana w postaci ␣␣␣+3.00

STOP NASTEPNY
KONIEC

Powyższe trzy zdania stanowią poprawny program napisany w języku SAKO; jest to program na obliczanie większego pierwiastka równania kwadratowego
$x^2 - 5x + 6 = 0$

3) --------

DRUKUJ (4,3) : (-B + PWK (B*2 - 4 ◇ A ◇ C))/(2 ◇ A)

(to zdanie jest bardziej ogólne od poprzedniego ale najpierw trzeba zmiennym A,B,C nadać wartości).

4) CALKOWITE : I,J
BLOK (7) : A,B

DRUKUJ (I+1,J) : A,B (3)

(drukuj wartości zmiennych A,B (3) z I+1 cyframi przed kropką i J cyframi po kropce)

Błędne użycie drukowania stałoprzecinkowego:

DRUKUJ (1,2) : 23.7
(liczba nie mieści się w zakresie; będzie wydrukowana w postaci: +2.37E+1 ⌴)

DRUKUJ (1,2) : 9.995
(jak wyżej, gdyż po zaokrągleniu argument równy będzie 10.00)

DRUKUJ (A,2) : 3.45
(zmienna A jest ułamkowa)

Drukowanie zmiennoprzecinkowe

Drukowanie zmiennoprzecinkowe ma postać:

DRUKUJ (a1, a2, a3) : b1

Znaczenie parametrów a1 i a2 jest takie same jak przy drukowaniu stałoprzecinkowym.

Sam fakt wystąpienia wyrażenia arytmetycznego całkowitego a3 określającego ilość cyfr znaczących liczby drukowanej, powoduje zarezerwowanie po liczbie, czterech dodatkowych pozycji dla wykładnika dziesiętnego.

Zatem liczba drukowana zajmie:

$\underline{a1} + \underline{a2} + 2 + 4$ (przy $\underline{a2} = 0$) lub

$\underline{a1} + 1 + 4$ (przy $\underline{a2} = 0$) pozycji

Wartość wyrażenia $\underline{a3}$ powinne spełniać nierówności:

$0 \quad \underline{a3} \quad \underline{a1} + \underline{a2}$; $\underline{a3} \quad 12$

Przy drukowaniu liczby może zaistnieć jeden z trzech przypadków:

- Liczba drukowana jest zbyt duża, żeby można było ją zmieścić w obszarze wyznaczonym parametrami a1 i a2,
- liczba drukowana jest zbyt mała, aby bez zmian można było wydrukować $\underline{a3}$ cyfr znaczących,
- liczba drukowana spełnia wszystkie warunki tj. mieści się w przedziale określonym przez $\underline{a1}$ i $\underline{a2}$ oraz posiada co najmniej a3 cyfr znaczących (może mieć więcej) lub jest zerem.

Przez pojęcie "cyfry znaczące" będziemy rozumieli wszystkie cyfry liczby poczynając od pierwszej cyfry nie będącej zerem, wszystkie zera przed pierwszą cyfrą niezerową niezależnie od położenia kropki będziemy uważać za nieznaczące.

W pierwszym przypadku, liczba jest dzielona przez minimalną potęgę dziesięciu taką, aby mogła się zmieścić w wyznaczonym przedziale i wypisywana jest w formie takiej, jak liczba ułamkowa stałoprzecinkowa a po liczbie wypisany jest wykładnik dziesiętny dodatni. W drugim przypadku liczba jest mnożona przez minimalną potęgę dziesięciu taką, aby posiadała $\underline{a3}$ cyfr znaczących i również jest wypisywana w formie stałoprzecinkowej a po liczbie wypisywany jest odpowiedni wykładnik dziesiętny ujemny.

W trzecim przypadku liczba jest wypisywana bez zmian a po liczbie drukowane są cztery spacje.

Wykładnik dziesiętny zaczyna się literą E, po której następuje znak wykładnika (+ lub -) oraz jedna lub dwie cyfry wykładnika.

W jednocyfrowym wykładniku po cyfrze występuje spacja.

Jeżeli wyrażenie b1 jest całkowite to jest przekształcane do postaci ułamkowej.

Przykłady

Poprawne użycie drukowania zmiennoprzecinkowego:

DRUKUJ (3,2,4) : 12345.67
(liczba będzie wydrukowana następująco:
+123.46E+2 ␣ ; przypadek pierwszy)

DRUKUJ (3,2,4) : 0.1234567
(wynik drukowania będzie: ␣ +12.35E-2 ␣ ;
przypadek drugi)

DRUKUJ (4,2,4) : 123.4567
(wynik drukowania będzie: ␣ +123.46 ␣ ␣ ␣ ␣;
przypadek trzeci)

DRUKUJ (3,2,4) : +12345,-12, 1
(wypisany ciąg znaków będzie następujący:
+123.45E+2␣|␣ -12.00␣ ␣ ␣ ␣|␣+10.00E-1␣)

DRUKUJ (3,2,0) : 0.002
(da wynik: ␣ ␣ ␣+0.00 ␣ ␣ ␣ ␣)

Błędnie użyte drukowanie zmiennoprzecinkowe:

DRUKUJ (4,2,7) : 4321
(7 jest większe od 4+2)

DRUKUJ (4,2,-1) : 4321
(ilość cyfr znaczących mniejsza od zera)

SPACJA

Postać ogólna
SPACJ {A, I} [:] [w]
gdzie: w jest wyrażeniem arytmetycznym całkowitym; dwukropek jest nieistotny.

Zdanie to powoduje wypisanie ciągu spacji na bieżące wyjście symboliczne programu. Ilość spacji jest równa bezwzględnej wartości wyrażenia arytmetycznego w.

Jeżeli wyrażenie w nie zostało wypisane, to zdanie takie jest równoważne zdaniu SPACJI : 1.

Zwroty SPACJA i SPACJI są równoważne.

Przykłady

Poprawne użycie zdania SPACJA:

1)
```
SPACJA        (drukuje znak ␣)
SPACJI : 1    (jak wyżej)
SPACJI  1     (jak wyżej)
SPACJA : 0    (zdanie puste)
```

2)
```
CALKOWITE : I
I = 4
DRUKUJ (1) : 123
SPACJI I+2
DRUKUJ (1) : 4567
  SPACJI -I
STOP NASTEPNY
KONIEC
```

Program zamieszczony na poprzedniej stronie (przykład 2) wypisze następujący ciąg znaków:

␣123␣␣␣␣␣␣ | ␣4567␣␣␣␣

Błędne użycie zdania SPACJA:

SPACJA A (A jest zmienną ułamkową)

LINIA

Postać ogólna
LINI $\left\{\begin{matrix} A \\ I \end{matrix}\right\}$ [:] [w] gdzie: w jest wyrażeniem arytmetycznym całkowitym; dwukropek jest nieistotny.

Zdanie to powoduje wypisanie ciągu znaków NL (przejście do początku nowej linii) na bieżące symboliczne wyjście. Ilość wypisywanych znaków NL jest równa bezwzględnej wartości wyrażenia arytmetycznego całkowitego w.

Brak argumentu w spowoduje wypisanie jednej linii. Zwroty LINIA i LINII są równoważne. Wyjątkowe znaczenie ma zdanie LINIA z parametrem równym zero np. LINIA 0. W tym przypadku wypisywany jest tzw. znak powrotu karetki (CR), bez przejścia do nowej linii i pozwala na ponowne pisanie w tym samym wierszu (na nakładanie tekstów na siebie).

Przykłady

Poprawne wykorzystanie zdania LINIA:

```
1)  CALKOWITE : I
    LINIA            (równoważne zdaniu LINII : 1)
    LINII : 1
    -------
    LINII I-4        (I musi być określone)
```

2)
```
LINIA
DRUKUJ (1) : 1234
LINIA
DRUKUJ (1) : 12345
DRUKUJ (2) : 1
STOP NASTEPNY
KONIEC
```

Powyższy program spowoduje wypisanie tekstu w poniżej podanym układzie:

```
␣1234
␣12345␣␣1
```

Błędne użycie zdania LINIA:

LINIA : A1 (A1 jest zmienną ułamkową)

STRONA

Postać ogólna
STRONA

Zdanie bezparametrowe, powoduje wypisanie znaku nowej strony na bieżące wyjście symboliczne programu.

W szczególnym przypadku gdy dane wyjście symboliczne jest związane z drukarką wierszową to zdanie STRONA spowoduje przejście do następnej strony arkusza wydawniczego.

TEKST

Postać ogólna
TEKST [:] t1 gdzie: t1 jest dowolnym ciągiem znaków wypisanym w następnym wierszu po zwrocie TEKST. Ostatnim znakiem tekstu jest NL (nowa linia). Dwukropek jest nieistotny.

Zdanie to powoduje wypisanie tekstu umieszczonego w następnym wierszu na bieżące wyjście symboliczne programu.
Przy wypisywaniu pomijane są wszystkie znaki nieistotne poczynając od początku wiersza aż do pierwszego znaku istotnego i od ostatniego znaku istotnego aż do NL włącznie.

Znaki nieistotne występujące między znakami istotnymi są wypisywane. Gdy wiersz t1 nie zawiera znaków istotnych to nie zostanie wypisany żaden znak. Wiersz pusty będący parametrem zdania TEKST nie jest pomijany.

Przykład

```
LINIA
DRUKUJ (1) : 123
TEKST
  TO␣␣JEST␣␣TE
TEKST
  KST
DRUKUJ (1) : 1234
STOP NASTEPNY
KONIEC
```

Program zamieszczony na poprzedniej stronie utworzy ciąg znaków:

123TO␣␣JEST␣␣TEKST␣1234

Błędne użycie zdania TEKST:

```
TEKST

MOWA
```

Pusty wiersz występujący bezpośrednio po zwrocie będzie uważany jako tekst do wypisania a słowo MOWA będzie potraktowane jako błędne zdanie języka SAKO.

TEKST WIERSZY

Postać ogólna
TEKST WIERSZY n [:] t1 t2 : tn gdzie: n jest liczbą całkwotą bez znaku, t1, t2,...,tn jest to n wierszy - tekstu; dwukropek jest nieistotny.

Zdanie to powoduje wypisanie (na bieżące wyjście symboliczne) n wierszy tekstu, który występuje w nowej linii bezpośrednio po zwrocie TEKST WIERSZY. Wypisywane są wszystkie znaki.
Wiersze puste są liczone na równi z niepustymi.

Należy pamiętać, że wiersz jest kończony znakiem NL a nie CR.

Zbyt mała lub zbyt duża ilość wierszy tekstu w stosunku do parametru n może prowadzić do błędów trudno wykrywalnych.

Przykłady

Poprawne użycie zdania TEKST WIERSZY:

```
LINIA
TEKST WIERSZY 5
1
2          (ta kolumna liczb zostanie
3           przepisana na urządzenie
4           wyjściowe)
5
```

TEKST WIERSZY 4 } (te dwa wiersze też są liczone)

WIERSZE PUSTE NALEŻY UWZGLEDNIC (ten wiersz również będzie wypisany)

DRUKUJ WIERSZ

Postać ogólna
DRUKUJ WIERSZ : a1, a2,...,an gdzie: a1,a2,...,an są to nazwy bloków całkowitych struktur, tablic stanowiące listę. Listę można przenosić.

Jedno zdanie z ciągiem nazw na liście jest równoważne (co do działania) ciągowi zdań DRUKUJ WIERSZ, z których każde zawiera po jednej nazwie. Zatem dla uproszczenia będziemy się zajmować zdaniami z jednym parametrem.

Zdanie DRUKUJ WIERSZ : a1 zakłada, że w kolejnych słowach bloku a1 (kolejność opisana jest w paragrafie "przydział miejsc pamięci dla bloku") zapisane są liczby całkowite.

Liczby te są traktowane przez zdanie DRUKUJ WIERSZ jako znaki zakodowane w KW6 i wypisywana są na bieżące wyjście symboliczne. Wypisywanie kończy się w momencie napotkania znaku NL (wartość liczbowa 58) . Znak NL jest również wypisywany a brak tego znaku w wypisywanym bloku prowadzi do błędów.

Przykłady

Poprawne użycie zdania DRUKUJ WIERSZ:

```
CALKOWITE: A,I
BLOK (120) : A
   LINIA
A (=I)= 48          (utworzenie bloku zawierającego
                     w każdym słowie kod cyfry 1)
A (120) = 58        (posłanie kodu NL do ostatniego
                     miejsca bloku)
DRUKUJ WIERSZ : A (wydrukowanie ciągu 120 cyfr 1
STOP NASTEPNY       oraz NL)
KONIEC
```

Błędne użycie zdań DRUKUJ WIERSZ:

```
CALKOWITE : B,I
BLOK (10) : A,B
   --------
   --------
   --------
DRUKUJ WIERSZ : A    (blok A jest ułamkowy)
DRUKUJ WIERSZ : *A,B (napisano gwiazdkę przed nazwą
                      bloku)

B (=I) = 48
DRUKUJ WIERSZ : B    (brak zanku NL kończącego wiersz)
```

CZYTAJ

Postać ogólna
CZYTAJ : a1, a2,...,an gdzie: a1, a2,...,an są to nazwy zmiennych prostych lub poprzedzone gwiazdkami nazwy bloków stanowiące listę Listę można przenosić.

Powyższe zdanie jest równoważne ciągowi zdań:

CZYTAJ : a1
CZYTAJ : a2
.
.
.
CZYTAJ : an

Stąd, będziemy rozpatrywać zdanie CZYTAJ z jednym parametrem.

CZYTAJ powoduje czytanie danych dziesiętnych z bieżącego symbolicznego wejścia programu. Jeżeli parametrem zdania jest zmienna prosta (lub zerowy element bloku) to zdanie powoduje przeczytanie jednej liczby i zmienna otrzyma wartość równą wartości liczby przeczytanej. Jeżeli parametrem zdania jest blok (struktura, tablica) tj. na liście znajduje się jego nazwa poprzedzona gwiazdką, to zdanie CZYTAJ spowoduje przeczytanie tylu liczb ile wynosi rozmiar bloku i nadanie wartości wszystkim zmiennym indeksowanym. Wartość otrzymuje najpierw zerowy element a następnie dalsze według kolejności przydziału miejsc dla bloku (patrz wzór 1).

Jeżeli zmienna (lub blok) jest całkowita, to liczba(liczby) musi być całkowita, jeżeli zmienna (blok) jest ułamkowa to liczba (liczby) może być ułamkowa lub całkowita. Liczba całkowita jest wtedy sprowadzona do postaci ułamkowej.

Przygotowanie danych liczb dla CZYTAJ

Kolejność danych wejściowych przygotowanych do czytania z odpowiedniego urządzenia wejściowego musi być taka sama jak kolejność wykonywania się zdań CZYTAJ odnoszących się do tego urządzenia. Każda liczba przewidziana jako wartość zmiennej prostej musi występować w oddzielnym wierszu, zaś dane nadające wartości elementom bloku muszą zaczynać się od nowego wiersza i mogą być rozdzielone spacjami.

Po danych stanowiących blok (w CZYTAJ wystąpiła gwiazdka przez nazwą) musi wystąpić w oddzielnym wierszu gwiazdka, niezależnie od ilości elementów w bloku. Ilość liczb stanowiących dane dla bloku musi być dokładnie równa rozmiarowi bloku (struktury, tablicy).

Wśród danych liczbowych mogą występować takie same komentarze, jak w zdaniu TABLICA; również separatorami liczb mogą być spacje, stąd też spacja nie może wystąpić między cyframi liczby ani też między znakiem i liczbą. Postać samych liczb jest zgodna z definicjami podanymi w paragrafach: "liczba ułamkowa" i "liczba całkowita".

Przykłady

Poprawne wykorzystanie CZYTAJ i przygotowania danych:

1)
```
LINIA
CZYTAJ: A,B,C
DRUKUJ  3,3  : A,B,C
STOP NASTEPNY
KONIEC
```

Program powyższy przepisze 3 liczby z wejścia na wyjście.

```
DANE PROGRAMU:          to jest komentarz
  A = 1.7               A = i C : też są komentarzami
  14
  C : -7.2E-4
```

Czytane są wartości zmiennych prostych, stąd dane muszą być napisane w "słupku".

2) CALKOWITE : I1,I2
BLOK (3) : A1,I1,A3
BLOK (0) : B
CZYTAJ : A2, * I1, * A1, I2, A3, * B

Dane dla powyższego zdania należy przygotować w/g wzoru:

DANA DLA A2 = 123.4
DANE DLA BLOKU I1:
1 -2
I (2) = 3 +4
*

DANE DLA A1 :
A1 (0) = 1.1 A1 (1) = 1.2 A1 (2) = 1.3 A1 (3) = -1.4
*

I2 = 12
A3 = 123.4
TU NIE PISALISMY GWIAZDKI GDYŻ Z PUNKTU WIDZENIA CZYTAJ A3 JEST ZMIENNA PROSTA:

B = 12 (oznacza 12.)
*
(gwiazdka jest potrzebna mimo że B jest blokiem jednoelementowym)

Błędne użycie zdania CZYTAJ:

3) CZYTAJ : A,B

DANE :
B = 7.3
A = 1.2

Niezależnie od komentarzy A otrzyma wartość 7.3 a B wartość 1.2

4) CZYTAJ : A,B

DANE :

A = 1.2 B = 7.3

C = 4.4

Dwie liczby zostały napisane w jednym wierszu, A otrzyma wartość 1.2 a B wartość 4.4, reszta wiersza po 1.2 zostanie pominięta.

5) BLOK (4) : A
CZYTAJ : * A

DANE :

1 2
3
4
*

Zbyt mało liczb dla nadania wartości wszystkim zmiennym indeksowanym.

6) BLOK (3) : A,B
CZYTAJ : * A, * B

STOP NASTEPNY
KONIEC

DANE:

1	2	3	4	
1.1	1.2	1.3	1.4	(brak gwiazdki)
*				

Jest to wyjątkowo nieprzyjemny błąd gdyż dane bloku są czytane aż do gwiazdki włącznie i sygnalizacja błędu następuje dopiero po napotkaniu jej. Tymczasem liczby przewidziane dla bloku B będą czytane do następnych miejsc po bloku A tj. poza pamięć maszyny i wcześniej będzie przerwane wykonywanie programu, zanim zostaniemy poinformowani przez program.

CZYTAJ WIERSZ

Postać ogólna
CZYTAJ WIERSZ: a1, a2,...,an gdzie : a1, a2,...,an są to nazwy bloków tablic,struktur całkowitych stanowiące listę parametrów. Listę można przenosić.

Zdanie to powoduje przeczytanie kolejnych wierszy tekstu z bieżącego symbolicznego urządzenia wejściowego do kolejnych bloków (tablic, struktur) wymienionych na liście parametrów. Ostatnim przeczytanym znakiem wiersza jest NL.

Czytanie znaków polega na nadawaniu kolejnym zmiennym indeksowanym, poczynając od elementu zerowego, wartości równych kodom (w KW6) kolejnych znaków wiersza czytanego.

Czytane są wszystkie znaki występujące w wierszu, wiersze puste nie są pomijane. Przy rezerwacji bloków dla zdania CZYTAJ WIERSZ należy uwzględnić możliwość wystąpienia dłuższych wierszy niż ilość znaków istotnych i spacji oraz należy pamiętać, że czytanie jest kończone po napotkaniu NL a nie po nadaniu wartości wszystkim elementom bloków, co z kolei może spowodować "zamazanie" innych obszarów pamięci bez sygnalizacji błędu, w szczególnym przypadku może nastąpić przerwanie pracy programu spowodowane przekroczeniem obszaru pamięci przewidzianego dla danego programu.

W przypadku, gdy dane dziesiętne dla CZYTAJ są przemieszane z tekstami pobieranymi przez CZYTAJ WIERSZ należy pamiętać, że zdanie CZYTAJ pobiera całe wiersze włącznie z NL.

Przykłady

Poprawne wykorzystanie zdania CZYTAJ WIERSZ:

```
   CALKOWITE : I,A
   BLOK (150-1) : B,A
   CZYTAJ WIERSZ : A
*) GDY A (I) = 48 : 1, INACZEJ NASTEPNY
   GDY A (I) = 58 : 2, INACZEJ NASTEPNY
   POWTORZ : I = 0 (1) 149
      ---------          (tu program przejdzie gdy ilość
      ---------           znaków w wierszu jest większa
      ---------           od 150)

1)    ---------          (tu przejdzie program gdy napotka
      ---------           pierwszą cyfrę 1 (patrz kod KW6))
      ---------

2)    ---------          (a tu program przejdzie gdy nie
      ---------           występuje w wierszu cyfra 1)
      ---------
```

Gdyby nie był zadeklarowany blok B, to przy czytaniu wiersza składającego się z więcej niż 150 znaków już w momencie wykonywania zdania CZYTAJ WIERSZ program zostałby przerwany z powodu przekroczenia zakresu pamięci (patrz: rezerwacja pól dla bloków).

Błędne wykorzystanie zdania CZYTAJ WIERSZ:

```
CALKOWITE : B
BLOK (100) : A,B
CZYTAJ WIERSZ : A        (blok A jest ułamkowy)
CZYTAJ WIERSZ : *B       (gwiazdek nie należy pisać
                          przed nazwami bloków)
```

*)CZYTAJ OKTALNIE

Postać ogólna
CZYTAJ OKTALNIE : a1,a2,...,an gdzie : a1,a2,...,an są to nazwy zmiennych prostych lub poprzedzone gwiazdkami nazwy bloków stanowiące listę. Listę można przenosić.

Zdanie to powoduje nadawanie wartości zmiennym prostym lub blokom (wyzystkim elementom wymienionego bloku,struktury lub tablicy) poprzez czytanie danych oktalnych z bieżącego symbolicznego wejścia programu.

Zdania CZYTAJ i CZYTAJ OKTALNIE różnią się tylko postacią samych danych, pozostałe uwagi odnoszące się do zdania CZYTAJ są słuszne również dla czytania oktalnego. Postać danych oktalnych i reguły zamiany oktalnych krótkich na długie są takie same jak podano w opisie danych TABLICY OKTALNEJ.

Przykład

```
CALKOWITE : I,J
BLOK (2) : J,A,C
CZYTAJ OKTALNIE : B, *J,-
                  *A,C,I
STOP NASTEPNY
KONIEC
WARTOSC B : 11. 22. 11. 1
DANE DLA J:
   1234.1234  111.222.33  1234
     *
```

```
A (0) = 1234567.012
A (1) = 1234
A (2) = 12345.12345.12345.1
        *
C DLA CZYTAJ OKTALNIE JEST ZMIENNA
PROSTA = 12345.12345670
I : 12  KONIEC DANYCH.
```

w) DRUKUJ OKTALNIE

Postać ogólna
DRUKUJ OKTALNIE : a1,a2,...,an gdzie : a1,a2,...,an są to wyrażenia arytmetyczne stanowiące listę. Listę można przenosić.

Powyższe zdanie jest równoważne ciągowi zdań:

DRUKUJ OKTALNIE : a1

DRUKUJ OKTALNIE : a2

:

DRUKUJ OKTALNIE : an

stąd możemy omawiać uproszczone zdanie, zawierające tylko jeden element na liście.

DRUKUJ OKTALNIE : a1 powoduje obliczenie wartości wyrażenia arytmetycznego a1 (nie logicznego) a następnie wypisanie jego wartości na bieżące urządzenie symboliczne.

Postać wyników zależy od typu wyrażenia:

1. Jeżeli wyrażenie a1 jest całkowite to wartość jego będzie wypisana jako liczba oktalna krótka ośmiocyfrowa z kropkami umiesczonymi co dwie cyfry.
2. Jeżeli wyrażenie a1 jest ułamkowe to wartość jego będzie wypisana jako liczba oktalna długa 16 cyfrowa z kropkami rozmieszczonymi również co dwie cyfry.

Liczba oktalna krótka zajmie na arkuszu wydawniczym 11 pozycji a liczba oktalna długa 23 pozycje.

Przykłady

Poprawne użycie zdania DRUKUJ OKTALNIE:

```
CALKOWITE : I,J
BLOK (2) : A,I
DRUKUJ OKTALNIE : (16 + 9) ◇ 2
                  (wynik drukowania będzie: 00.00.00.62
                   gdyż jest to liczba całkowita 50)
DRUKUJ OKTALNIE : 1.0
                  (wynik drukowania będzie:
---------          20.00.00.00.00.00.00.01 gdyż
---------          wyrażenie arytmetyczne ułamkowe
---------          o wartości 1.0 ma taką postać oktalną)
DRUKUJ OKTALNIE : A,I
DRUKUJ OKTALNIE : 9,2
---------         (wynik drukowania będzie:
---------          00.00.00.1100.00.00.02)
---------
```

Błędne użycie zdania DRUKUJ OKTALNIE:

```
DRUKUJ OKTALNIE : * A
                  (* A nie jest wyrażeniem)
DRUKUJ OKTALNIE :   77.77.02
                  (argument nie jest wyrażeniem
                   arytmetycznym; aby wydrukować war-
                   tość wyrażenia logicznego należy
                   najpierw nadać wartość logiczną
                   dla zmiennej np. przy pomocy zda-
                   nia przypisującego logicznie
                   (patrz następny paragraf) a na-
                   stępnie jej wartość wypisać przy
                   pomocy zdania DRUKUJ OKTALNIE).
```

*) ZDANIE PRZYPISUJACE LOGICZNE

Postać ogólna
a [b gdzie: a jest zmienną prostą, b jest wyrażeniem logicznym, znak [oznacza przypisanie wartości

Zdanie powyższe powoduje obliczenie wartości wyrażenia logicznego b i przypisanie jej zmiennej prostej a. Wyrażenie logiczne zostanie opisane poniżej w sposób rekursywny, podobny do opisu wyrażenia arytmetycznego.

Wyrażenie logiczne

Wyrażenie logiczne jest regułą na obliczanie wartości logicznej. Wartość logiczną zwaną wartością wyrażenia logicznego otrzymujemy przez wykonanie wskazanych działań we wskazanej kolejności na wielkościach wchodzących w skład wyrażenia.

Wartością wyrażenia logicznego jest ciąg 24 bitów - słowo krótkie lub ciąg 48 bitów - słowo długie. Wyrażenie logiczne, którego wartością jest słowo krótkie będziemy nazywać wyrażeniem logicznym całkowitym (typu całkowitego) a wyrażenie, którego wartością jest słowo długie, będziemy nazywać wyrażeniem logicznym ułamkowym (typu ułamkowego).

Aby dowiedzieć się jaka będzie wartość wyrażenia i jaki będzie jego typ, należy wiedzieć jakie są typy i wartości argumentów, jakie są wykonywane działania na argumentach i jakie są wartości i typy wyników działań oraz jaka jest kolejność wykonywania tych działań.

Pierwotne wyrażenie logiczne

Pierwotnym wyrażeniem logicznym jest:
- liczba oktalna
- zmienna prosta
- wyrażenie logiczne ujęte w nawiasy okrągłe.

Wartość i typ liczby oktalnej

Typ liczby oktalnej zależy od tego z ilu cyfr się ona składa (do ośmiu cyfr jest liczbą oktalną krótką, od 9 do 16 jest liczbą oktalną długą) a wartością jest ciąg bitów przez nią reprezentowany (patrz definicje liczb oktalnych).

Wartość i typ zmiennej prostej

Typ zmiennej prostej zależy od deklaracji CALKOWITE a wartość jest równa ostatnio przypisanemu jej ciągowi bitów poprzez:
- zdanie przypisujące arytmetyczne
- zdanie przypisujące logiczne
- przeczytanie przy pomocy jednego ze zdań wprowadzających dane lub czytających teksty
- nadanie przez podprogram
- przeczytanie z pamięci pomocniczych.

Wartość i typ wyrażenia logicznego ujętego w nawiasy

Wartość i typ wyrażenia ujętego w nawiasy są takie same jak wartość i typ wyrażenia zawartego w nawiasach.

Przykłady

Poprawne pierwotne wyrażenia logiczne:

77.66.55.44 (oktalna krótka)
77.66.55.44.33 (oktalna długa)
A (zmienna)
(A+(-6)◇KL◇77) (wyrażenie logiczne w nawiasach)

Błędne wyrażenia pierwotne logiczne:

19	(liczba dziesiętna; zawiera cyfrę 9)
A (I)	(zmienna indeksowana)
SIN (2)	(funkcja języka)
-A	(to jest wyrażenie logiczne a nie pierwotne wyrażenie logiczne)

Działania logiczne

Wyrażenie logiczne konstruowane jest z prostych wyrażeń logicznych i działań logicznych.

W wyrażeniach logicznych mogą wystąpić następujące działania logiczne:

znak działania	nazwa
*	przesunięcie cykliczne
◇	iloczyn logiczny
+	suma logiczna
-	negacja

Poniżej podamy: metodę konstrukcji wyrażenia, kolejność wykonywania działań logicznych, sposób wykonywania działań, typy argumentów i typy wyników.

Przesunięcie

Jeżeli przez pi (dla i = 1,2,3,...) oznaczymy jedno z wyrażeń:

1	(liczba dziesiętna całkowita bez znaku)
i	(zmienna prosta całkowita)
(-1)	(liczba dziesiętna całkowita ze znakiem minus ujęta w nawiasy)
(-i)	(zmienna prosta całkowita poprzedzona znakiem minus i ujęta w nawiasy)

oraz przez b pierwotne wyrażenie logiczne, to następujące wyrażenia logiczne będziemy nazywać przesunięciami:

b * p1
b * p1 * p2
b * p1 * p2 * p3
itd.

Powyższe wyrażenia są równoważne odpowiedno:

b * p1
(b * p1) * p2
((b * p1) * p2) * p3
itd.

W każdym z powyższych wyrażeń po lewej stronie gwiazdki występuje pierwotne wyrażenie logiczne a po prawej pojęcie uprzednio zdefiniowane. W związku z tym wystarczy omówić pierwsze z tych wyrażeń:

Aby obliczyć przesunięcie, najpierw oblicza się wartość logiczną lewego argumentu działania (wartość wyrażenia b), następnie oblicza się wartość prawego argumentu działania (prawy argument jest bardzo prostym wyrażeniem arytmetycznym całkowitym).

Typ wyniku działania

Typ wyniku działania jest taki sam jak typ pierwotnego wyrażenia logicznego (b)

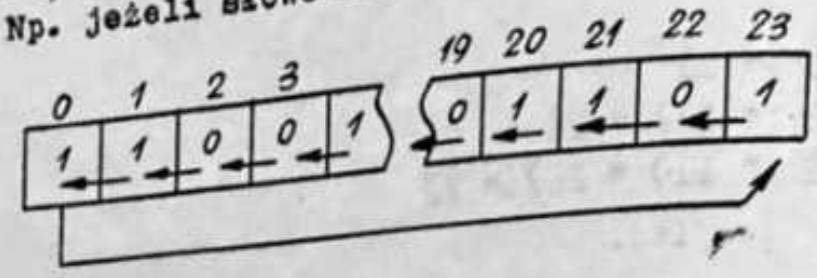
Wartość wyniku działania

Wartość otrzymujemy przez przesunięcie cykliczne wartości lewego argumentu o tyle miejsc ile wynosi bezwzględna wartość drugiego argumentu.

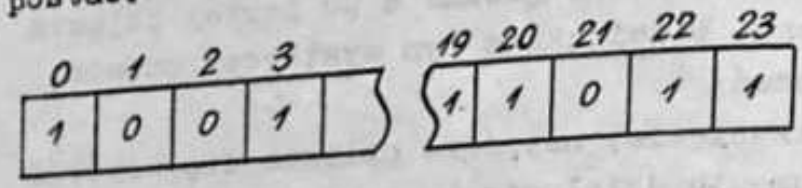
Przesuwanie następuje w lewo jeżeli drugi argument działania jest dodatni i w prawo gdy drugi argument działania jest ujemny.

Zasada przesuwania cyklicznego:

Rozpatrzmy najpierw przesunięcie cykliczne w lewo o jeden tj. operację: b * 1.
Operacja ta powoduje przesunięcie wszystkich bitów słowa (długiego lub krótkiego w zależności od typu) w lewo o jedno miejsce a bit zerowy (wypadający) przeniesiony jest na pozycję ostatnią (dwudziestą trzecią lub czterdziestą siódmą).
Np. jeżeli słowo krótkie ma postać:



to po przesunięciu o jedno miejsce w lewo będzie miało postać:



Przesunięcie cykliczne w lewo o więcej niż jedno miejsce jest równoważne wielokrotnemu wykonaniu przesunięcia cyklicznego o 1.
Przesunięcie o zero miejsc pozostawia słowo niezmienione.
Należy też zauważyć, że przesunięcie słowa krótkiego o 24 lub słowa długiego o 48 miejsc, również nie zmienia argumentu. Czas wykonania operacji zależy od tego o ile miejsc przesuwamy.

Przesunięcie cykliczne o jedno miejsce w prawo (tj. b * (-1)) powoduje przesunięcie wszystkich bitów słowa o jedno miejsce w prawo a ostatni bit (23 lub 47) jest przenoszony na pozycję zerową.
Przesunięcie w prawo o n jest równoważen n-krotnemu przesunięciu w prawo o 1.

Przykłady

Poprawne przesunięcia:

```
CALKOWITE : I
A * I
77 * 3          (wynik działania jest 770)
77 * 9 * (-9)(wynik działania będzie równy argumen-
                towi)
 I = -2
A * I           (przesunięcie cykliczne w prawo o 2
                 bo I = -2)
77 * 3 * 9 * 1  (jest równoważne 77 * 13)
```

Błędne przesunięcia:

```
A * B           (B ułamkowe)
A * 77.77.6     (drugi argument jest stałą oktalną)
```

Iloczyn logiczny koniunkcja

Jeżeli przez mi (dla i = 1,2,...) oznaczymy pierwotne wyrażenie logiczne lub przesunięcie cykliczne, to wyrażenia typu:

$$m1 \diamond m2$$

$$m1 \diamond m2 \diamond m3$$

itd.

które są równoważne wyrażeniom:

$$m1 \diamond m2$$

$$(m1 \diamond m2) \diamond m3$$

itd.

będziemy nazywać iloczynami logicznymi.

Aby obliczyć wartość iloczynu logicznego należy najpierw obliczyć wartość lewego argumentu, potem wartość prawego argumentu, a następnie wykonać operację zwaną iloczynem

logicznym. W wyniku działania otrzymamy słowo (krótkie lub długie).

Typ wyniku działania

Typ wyniku zależy od typu argumentów i zależność tę podaje poniższa tabelka:

typ m1 / typ m2	ułamkowy	całkowity
ułamkowy	ułamkowy	ułamkowy
całkowity-	ułamkowy	całkowity

Zatem wynik jest słowem krótkim tylko wtedy, gdy oba argumenty są słowami krótkimi.

Wynik działania

Aby wykonać działanie na argumentach o różnych typach, należy wartość argumentu całkowitego przedstawić w postaci długiego słowa (zmienić typ argumentu całkowitego na ułamkowy).

Zamiana typu całkowitego na ułamkowy

polega na utworzeniu słowa długiego, którego pierwszym słowem krótkim jest argument, a następne słowo krótkie składa się z samych zer.

Wynik działania na argumentach tego samego typu otrzymuje się przez obliczenie iloczynów logicznych poszczególnych bitów na tych samych pozycjach.

Poniższa tabelka podaje zależność wyników od wartości argumentów.

arg.1	arg.2	wynik
0	0	0
0	1	0
1	0	0
1	1	1

Przykłady

654321 ◇ 777777 (wynik działania jest równy pierwszemu argumentowi; jest całkowity)

765.765.40 ◇ 012.345.67
tj.

```
  111 110 101 111 110 101 100 000     (765.765.40)
  000 001 010 011 100 101 110 111     (012.345.67)
◇ -------------------------------
  000 000 000 011 100 101 100 000     (000.345.40)
```

654321 ◇ 77.77.77.77.7 (całkowita ◇ ułamkowa)

Wynik działania jest równy:

00.65.43.21.00.00.00.00 (nastąpiła najpierw zamiana liczby całkowitej na ułamkową)

```
◇00.00.00.07.77.77.77.77   (ta liczba już była ułamkową)
-----------------------
00.00.00.01.00.00.00.00
```

A ◇ B ◇ C (najpierw liczy się A ◇ B a następnie wynik koniugowany jest z C)

CALKOWITE : I
A*2◇B*3*I (najpierw liczymy A*2 i B*3*I a następnie iloczyn logiczny wyników przesunięć)

Negacja

Jeżeli przez t oznaczymy pierwotne wyrażenie logiczne lub przesunięcie cykliczne, lub iloczyn logiczny, to działanie jednoargumentowe

- t

będziemy nazywać negacją.

Aby obliczyć wartość negacji liczymy najpierw wartość wyrażenia t (tj. wyrażenia pierwotnego lub przesunięcia, lub iloczynu) a następnie negujemy jego wartość.

Typ wyniku

Typ wyniku działania jest taki sam jak typ argumentu.

Wartość wyniku

Wartość wyniku otrzymuje się przez zanegowanie każdego bitu słowa będącego argumentem. W wyniku operacji otrzymujemy słowo.

Poniższa tabelka podaje wartość wyniku w zależności od wartości bitu będącego argumentem:

wartość arg.	wynik
0	1
1	0

Przykłady

- 76.54.32.10 (wynik negacji będzie: 01234567)
- A *3 ◇ B *4 (najpierw liczymy iloczyn tj (A*3) ◇ (B*4) wcześniej licząc przesunięcia; na końcu wynik negujemy)
- -(-A *3◇B *4+C) (w nawiasie znajduje się wyrażenie logiczne)

Suma logiczna (alternatywa)

Jeżeli s1 będzie oznaczało dowolne z pojęć powyżej zdefiniowanych zaś każde z si (dla i = 2,3...) będzie pierwotnym wyrażeniem logicznym lub przesunięciem, lub iloczynem logicznym, lub negacją ujętą w nawiasy, to poniższe wyrażenia będziemy nazywać sumami logicznymi:

s1 + s2
s1 + s2 + s3
s1 + s2 + s3 + s4
itd.

Poprzednie wyrażenia są równoważne odpowiednio:

s1 + s2
(s1 + s2)+ s3
((s1 + s2)+ s3)+ s4
itd.

Stąd możemy rozpatrywać sumę jako działanie dwuargumentowe. Aby obliczyć wartość sumy logicznej należy najpierw obliczyć wartość logiczną lewego argumentu, potem wartość prawego argumentu, a następnie obliczyć sumę logiczną wyników.

Typ wyniku

Typ wyniku zależy od typów argumentów w sposób podany w poniższej tabelce:

typ arg.1 / typ arg.2	ułamkowy	całkowity
ułamkowy	ułamkowy	ułamkowy
całkowity	ułamkowy	całkowity

Tylko przy obu argumentach całkowitych wynik jest całkowity.

Wartość wyniku

Jeżeli typy argumentów są różne, to przed wykonaniem sumowania argument całkowity sprowadza się do postaci słowa długiego wg zasad opisanych przy omawianiu iloczynu logicznego.

Sumę logiczną dwóch słów otrzymuje się przez policzenie sum logicznych poszczególnych bitów.

Wartości wyników sumy logicznej dwóch bitów w zależności od ich wartości podaje poniższa tabela:

arg.1	arg.2	wynik
0	0	0
0	1	1
1	0	1
1	1	1

Przykłady

A + (-B)+ (-A * 3◇B * 4 + C) + A◇B + A + 3

Najpierw liczymy wartości A i (-B), następnie obliczamy ich sumę, dalej obliczamy wartość wyrażenia pierwotnego (-A * 3◇B * 4 + C) (w celu policzenia jego wartości należy wykonać działania w kolejności opisanej w poprzednich paragrafach) a następnie poprzednio otrzymany wynik, sumujemy logicznie z wartością ostatnio otrzymaną itd. aż do policzenia całej sumy.

Wyrażeniem logicznym będziemy nazywać dowolne z pojęć uprzednio zdefiniowanych tj. pierwotne wyrażenie logiczne, przesunięcie cykliczne, iloczyn logiczny, negację, sumę logiczną.

Przypisanie wartości w zdaniu przypisującym logicznym

Po policzeniu wartości wyrażenia b następuje przypisanie jej do zmiennej prostej stojącej po lewej stronie znaku [. Jeżeli typy zmiennej stojącej po lewej stronie znaku [i wyrażenia logicznego po prawej stronie są różne, to przed przypisaniem wartości musi nastąpić zrównanie typów.

Jeżeli a jest ułamkowe a b całkowite to wartość b sprowadza się do typu ułamkowego w sposób opisany poprzednio. Natomiast, gdy a jest całkowite a b ułamkowe, to

zmienna a otrzyma wartość równą starszej części wyniku otrzymanego z liczenia b.

Przykłady

Poprawne zdania przypisujące logiczne:

A [77.77.77.77.1 (A będzie miało wartość: 00.00.00.07.77.77.77.71 gdyż obie strony są ułamkowe)

A [77.77 (A będzie miało wartość: 00.00.77.77.00.00.00.00 bo prawa strona jest całkowita)

CALKOWITE : I

I [77 (I będzie miało wartość: 00.00.00.77 gdyż obie strony są całkowite)

I [12345670.2 (I otrzyma wartość równą 00.00.00.01 czyli starszą część liczby oktalnej długiej 00.00.00.01.23.45.67.02)

I [-A + (-B)*3 + A◇66.77.77* (-A + B)*(-1)
(minus przed jedynką oznacza, że należy (-A+B) przesunąć o jeden w prawo: nie oznacza negacji)

Błędne zdania przypisujące logiczne:

CALKOWITE : I

A (I) [12.30 (wystąpiła zmienna indeksowana)

A [19.0 (po prawej stronie znaku [jest liczba ułamkowa)

PISZ NA BEBEN

Postać ogólna
PISZ NA BEBEN OD a : b1, b2,...,bn gdzie: a jest wyrażeniem arytmetycznym całkowitym b1,b2,..,bn są to nazwy zmiennych prostych lub poprzedzone gwiazdkami nazwy bloków; nazwy stanowią listę. Listę nazw można przenosić

Zdanie to powoduje przepisanie do pamięci bębnowej wartości zmiennych prostych lub wartości bloków (wartości wszystkich zmiennych indeksowanych stanowiących blok) w kolejności występowania ich nazw na liście.

Miejsce pamięci bębnowej (adres słowa któtkiego), od którego należy rozpocząć wpisywanie jest wskazane przez wartość wyrażenia arytmetycznego całkowitego a.

Pamięć bębnowa

Pamięć bębnowa składa się ze słów krótkich, ponumerowanych kolejno, poczynając od zera. Numery kolejnych miejsc pamięci będziemy nazywać adresami. Ilość słów pamięci bębnowej dostępnych dla programu będziemy nazywać pojemnością bębna. Pojemność bębna jest zmienna, zależy od: wersji systemu operacyjnego, długości programu wynikowego i organizacji programu. Szczegółowe informacje na temat organizacji pamięci bębnowej zostaną podane w drugiej części podręcznika w paragrafie: "Zasady rozmieszczania programu na bębnie".

Rozmieszczenie danych na bębnie

Poczynając od adresu a, do kolejnych miejsc pamięci, przepisuje się wartość pierwszej zmiennej wyszczególnionej na liście nazw. Jeżeli wymieniona zmienna jest całkowita (zadeklarowana przez zdanie CALKOWITE), to zajmie jedno miejsce na bębnie, jeżeli zaś jest ułamkową to zajmie dwa miejsca. Jeżeli jest przepisywany blok (wartości z bloku), to ilość miejsc zajętych na bębnie będzie równa ilości miejsc zajmowanych przez dany blok w pamięci operacyjnej (patrz opis zdań BLOK, STRUKTURA, TABLICA).

Kolejne dane opisane w kolejnej pozycji na liście zdania PISZ NA BEBEN są zapisywane w kolejne miejsca bezpośrednio po końcu danych poprzednich.

Jeżeli na liście nazw wystąpi nazwa bloku nie poprzedzona gwiazdką to będzie potraktowana jak zmienna prosta.

Dla nowego zdania PISZ NA BEBEN programista musi sam wyliczyć adres bębnowy, pod który ma nastąpić zapis.

Ze względu na długi czas potrzebny do rozpoczęcia zapisu na bębnie (średni czas oczekiwania) zaleca się zapisywać na bębnie całe pola a nie pojedyncze bloki i zmienne proste.

Dla każdej pozycji na liście, zapis odbywa się oddzielnie.

Przykłady

Poprawne wykorzystanie zdania PISZ NA BEBEN:

1) BLOK (3) : A,B
2A) PISZ NA BEBEN OD 0 : C,*B, A

Przepisanie nastąpi odpowiednio:

C - do miejsc 0,1
B - do miejsc od 2 do 9
A - do miejsc 10-11

Pierwsze miejsce wolne na bębnie będzie dwunaste.

Jeżeli między zdaniami 1 i 2A nie zostanie nadana wartość zmiennej C i blokom A,B, to zdanie 2A przepisze na bęben

przypadkową nieokreśloną ich wartość bez sygnalizacji błędów.

2) CALKOWITE : I,J
BLOK (2) :A,I

PISZ NA BEBEN OD 1000 : *A,*I

(A zajmie 6 miejsc, I zajmie 3 miejsca, wolne miejsce będzie miało adres równy 1009)

STRUKTURA (6+3-1) : I

3) PISZ NA BEBEN OD 1000 : *I

(zdanie to przepisze pole I wraz z podpolem A na bęben, wykonując tę samą operację co poprzednie zdanie)

J = 1009
PISZ NA BEBEN OD J◇2 : *A

(Podpole A zostało przepisane poczynając od miejsca J◇2 (tj. 2018))

Błędne zdanie PISZ NA BEBEN:

PISZ NA BEBEN OD J/2 : *A (wyrażenie J/2 jest ułamkowe
PISZ NA BEBEN OD - 127 : *A (adres bębnowy ujemny)

CZYTAJ Z BEBNA

Postać ogólna
CZATAJ Z BEBNA OD a : b1, b2,...,bn gdzie: a jest wyrażeniem arytmetycznym całkowitym, b1, b2,...,bn są to nazwy zmiennych prostych lub poprzedzone gwiazdkami nazwy bloków stanowiące listę. Listy nazw można przenosić.

Zdanie to powoduje przeniesienie zawartości kolejnych miejsc pamięci bębnowej, poczynając od miejsca określonego wartością wyrażenia całkowitego a, do miejsc pamięci operacyjnej zarezerwowanych dla zmiennych i bloków, w kolejności wyszczególnienia ich na liście.
Czyli, spójny zbiór danych jest "rozdysponowywany" do różnych miejsc pamięci. W ten sposób nadawane są wartości zmiennym i blokom (strukturom, polom, tablicom).

Ilość przenoszonych miejsc pamięci bębnowej zależy od typów i rozmiarów bloków (pól, tablic) lub typów zmiennych prostych wyszczególnionych na liście.

Przykłady

Poprawne wykorzystanie zdania CZYTAJ z BEBNA:

```
BLOK (3) : A,B,D,E
   --------
   --------
   --------
1) CZYTAJ Z BEBNA OD 0 : C,* B,A
```

(zdanie to pobierze wartości zapisane

na bęben przy pomocy zdania
z przykładu 1 poprzedniego
paragrafu)

CZYTAJ Z BEBNA OD 1000: * B

(zdanie to pobierze 8 pierwszych słów zapisanych na bęben przez zdanie 3 podane w przykładach poprzedniego paragrafu).

STRUKTURA (2◇4 -1): E
CZYTAJ Z BEBNA OD 0 : * E

(za jednym dostępem do pamięci bębnowej pobraliśmy wartości do bloków E i D)

Błędne wykorzystanie zdania CZYTAJ z BEBNA:

Mogą być popełniane takie same błędy jak przy zdaniach PISZ NA BEBEN oraz może być przeczytana przypadkowa zawartość pamięci bębnowej.

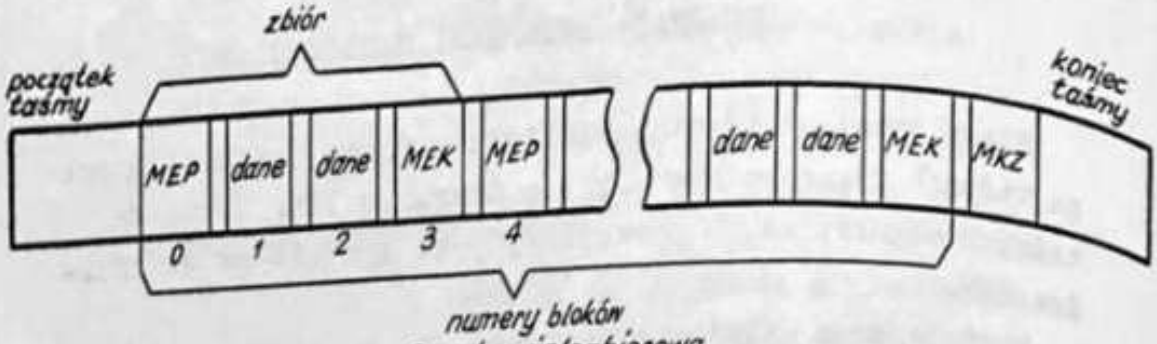
WSPÓŁPRACA Z TAŚMAMI MAGNETYCZNYMI

Niżej omawiane zdania umożliwiają magazynowanie danych na taśmach magnetycznych oraz pobieranie ich. Zapis na taśmach magnetycznych prowadzony jest zgodnie ze standardem IMM.

W niniejszym opisie ograniczymy się do przekazania niezbędnych informacji dotyczących zapisu na taśmach magnetycznych; bardziej szczegółowych danych należy szukać w opisie odpowiedniego standardu [7].

Jednostką zapisu na taśmie magnetycznej jest masyw słów 24-bitowych zwany blokiem TM. Kolejnym zapisywanym blokiem TM na taśmie przyporządkowane są kolejne liczby całkowite (poczynając od zera) zwane numerami bloków. Maksymalny blok TM zawiera 1024 słowa danych. Dodatkowo znajdują się w bloku TM takie informacje jak: suma kontrolna, numer bloku itp., które są automatycznie tworzone przez zdania języka SAKO i automatycznie pomijane przez nie. Stąd, znajomość organizacji wewnętrznej bloku TM, i rzeczywistej jego długości nie jest konieczna. Ze względu na logiczną organizację danych, taśma może zawierać jeden lub więcej zbiorów danych i taką taśmę będziemy nazywać szpulą wielozbiorową, lub też jeden zbiór danych znajdować się może na więcej niż jednej szpuli i wtedy będziemy mieli do czynienia ze zbiorem wieloszpulowym. Przez zbiór danych na szpuli wielozbiorowej będziemy rozumieli taki ciąg bloków na taśmie, który zaczyna się blokiem TM zwanym metryką początku zbioru (MEP) następnie występuje jeden lub więcej bloków TM zawierających właściwe dane oraz kończy się blokiem zwanym metryką końca zbioru (MEK) . Jeżeli na szpuli znajduje się więcej niż jeden zbiór to pomiędzy nimi nie może wystąpić na taśmie żaden zapis.

Na końcu wszystkich zbiorów na taśmie (nawet jeżeli jest tylko jeden) musi wystąpić metryka końca zapisu (MKZ) .



Szpula wielozbiorowa
rys.7

W przypadku zbiorów wieloszpulowych między blokami danych na końcu każdej (nie ostatniej) szpuli występuje metryka końca sekcji (szpuli) (MKS) a na początku następnej metryka początku zbioru MEP .

Sekcją nazywamy część zbioru znajdującą się na jednej szpuli. Sekcje są ponumerowane kolejnymi liczbami całkowitymi (poczynając od 1). Zbiór mieszczący się w całości na szpuli powinien mieć numer sekcji równy 1.

pośrednie sekcje mają taką samą budowę jak pierwsza
Zbiór wieloszpulowy
rys.8

Przy pobieraniu danych ze zbiorów wieloszpulowych i przy ich tworzeniu nie ma automatycznego przejścia ze szpuli na szpulę, stworzono jednak możliwość organizowania takiego przejścia we własnym zakresie.

Metryka początku zbioru zawiera między innymi: nazwę zbioru, numer sekcji, okres ważności.

Nazwa zbioru jest "nazwą" w sensie definicji podanej wcześniej i posiada identyfikujące pierwsze cztery znaki (dalsze, jeżeli występują są nieistotne).

Okres ważności podawany jest w dniach, poczynając od daty tworzenia zbioru.

Metryka końca zbioru zawiera między innymi sumę kontrolną wszystkich bloków danych. Metryki są tworzone przez zdania języka i przez nie rozpoznawane. Stąd znajomość dokładnej budowy metryk nie jest konieczna.

Język SAKO umożliwia jednoczesną współpracę programu wynikowego z maksimum ośmioma stacjami taśm magnetycznych. Stacjom przyporządkować można numery z zakresu od 0 do 7.

Odpowiedniość między numerami i nazwami stacji używanymi w czołówce systemu (patrz CZOŁOWKA SYSTEMU CZESCI II podręcznika) jest następująca:

0 - TMA
1 - TMB
2 - TMC
⋮
7 - TMH

Aby można było sterować programem, w zależności od poprawności działania stacji taśm lub od zawartości bloku TM (np. MEK) wprowadzono funkcję języka CHR, której argumentem jest numer stacji, o której stan pytamy, a wartością liczba całkowita określająca zaistniałe zdarzenie.

Drugą funkcją języka związaną ze współpracą z taśmami magnetycznymi jest BTM. Argumentem jej jest numer stacji, o którą pytamy, a wartością liczba całkowita, będąca numerem bloku TM, który będzie odczytany, jeżeli użyjemy zdania POBIERZ (patrz Opis funkcji języka, DODATEK B).

OTWORZ

Postać ogólna

OTWORZ a [, {* / x}] : y [, s [,w]]

gdzie: a jest wyrażeniem arytmetycznym całkowitym określającym numer stacji,
x jest nazwą zbioru podlegającego likwidacji,
y jest nazwą zbioru tworzonego,
s jest wyrażeniem arytmetycznym całkowitym określającym numer sekcji tworzenia zbioru,
w jest wyrażeniem arytmetycznym całkowitym, którego wartość podaje okres ważności tworzonego zbioru

Zdanie powyższe powoduje zapisanie na taśmie magnetycznej metryki początku zbioru lub metryki początku sekcji (MEP) Taśma, na której ma nastąpić zapis powinna znajdować się na stacji o numerze będącym wartością wyrażenia arytmetycznego całkowitego a.

Nowootwarty zbiór ma mieć nazwę postaci y, jego numer sekcji jest równy wartości wyrażenia s oraz okres ważności zbioru jest równy wartości wyrażenia w.

Drugi parametr zdania OTWORZ określa miejsce, gdzie na taśmie magnetycznej należy wpisać metrykę początku. Mogą zaistnieć trzy możliwe przypadki związane z drugim parametrem:

1. Drugi parametr nie występuje.

Omawiane zdanie ma wtedy postać:

OTWORZ a : y [,s [,w]]

Brak drugiego parametru oznacza, że metryka ma być zapisana poczynając od początku taśmy. Zakłada się wtedy, że na taśmie nie ma istotnych danych, a jeżeli były, to wszystkie ulegną zniszczeniu.

2. Drugi parametr jest nazwą zbioru.
   Zdanie OTWORZ ma wtedy postać:

   OTWORZ a, x : y [,s [,w]]

   Takie zdanie mówi, że należy odnaleźć na wskazanej szpuli (wskazuje się poprzez numer stacji) zbiór o nazwie x i zamiast jego metryki wpisać metrykę nową z nazwą y. Oznacza to, że wszystkie dane zbioru x oraz zbiory ewentualnie występujące po nim na danej taśmie zostaną zniszczone. Używając zdania w omawionym formacie, zakłada się, że na danej taśmie istnieje co najmniej metryka początku zbioru o nazwie x.

3. Zamiast drugiego parametru jest gwiazdka (*).
   Tj.

   OTWORZ a, * : y [,s [,w]]

   W tym przypadku nastąpi zapisanie metryki początku nowego zbioru po ostatnim zbiorze występującym na wskazanej taśmie. Należy tu pamiętać, że poprawność tej operacji zależy od tego czy taśma były poprawnie zbudowana (miała poprawną metrykę końca zapisu).
   Pominięcie parametru w oznacza okres ważności 0 dni zaś pominięcie parametrów s i w jednocześnie (inaczej nie można) oznacza, że nowootwierany zbiór ma numer sekcji równy 1 (standard dla zbiorów mieszczących się na jednej szpuli) i okres ważności jest 0 (zero).
   Po zapisaniu MEP urządzenie stacji gotowe jest do zapisu kolejnego bloku TM na taśmie tj. pierwszego bloku danych.

Po wykonaniu zdania OTWORZ funkcja CHR może przyjąć następujące wartości:

0 - zdanie zostało wykonane poprawnie,
1 - brak zbioru na taśmie, w miejsce którego chcemy zapisać nową metrykę,

2 - nie udało się zapisać (błędy zapisu)

3 - w trakcie poszukiwania nastąpił błąd odczytu taśmy.

Ponadto może nastąpić sygnalizacja błędów na urządzeniu wyjściowym (patrz: Opis funkcji języka, Spis błędów etapu wykonania; DODATKI B i H).

Przykłady

Poprawne użycie zdania OTWORZ:

CALKOWITE: TMA, TMB, TMD

$$\begin{cases} TMA = 0 \\ TMB = 1 \\ TMD = 3 \end{cases}$$

OTWORZ TMA: WYNIKI

(Otwórz zbiór o nazwie WYNI poczynając od początku taśmy założonej na stacji o nazwie TMA)

OTWORZ 1, * : DAL, 1, 10

(Otworz zbiór o nazwie DAL, z okresem ważności 10 dni, po ostatnim zbiorze zapisanym w/g standardu na taśmie umieszczonej na stacji TMB).

OTWORZ TMD, STARY : NOWY

(zamiast zbioru o nazwie STAR należy otworzyć zbiór o nazwie NOWY)

Błędne użycie zdań OTWORZ:

Załóżmy, że taśmy po wykonaniu powyżej podanych zdań nie zostały zdjęte, wtedy:

OTWORZ TMB, * : DAL1

(zbiór DAL nie został zakończony, brak końca zapisu)

OTWORZ TMA,ALA : NOWY

(na TMA nie ma innego zbioru niż WYNI)

ZAPISZ

Postać ogólna
ZAPISZ a [,z] : b1, b2,...,bn gdzie: a jest wyrażeniem arytmetycznym całkowitym określającym numer stacji, z jest zmienną całkowitą, b1,b2,...,bn są to nazwy bloków stanowiące listę. Listę nazw bloków można przenosić.

Zdanie to spowoduje przepisanie wartości bloków b1,b2,...,bn, na taśmę magnetyczną umieszczoną na stacji o numerze wyznaczonym przez wartość wyrażenia a. Przepisanie nastąpi w kolejności zgodnej z kolejnością rozmieszczenia nazw na liście bloków.

Zmienna z, jeżeli zostanie wyszczególniona jako drugi parametr zdania, otrzyma wartość równą numerowi pierwszego zapisanego bloku TM tj. numerowi bloku TM, w którym zaczynają się wartości bloku b1. Jeżeli zmienna z nie jest wyszczególniona, to numer pierwszego bloku nie zostanie odnotowany. Zapis bloków wyszczególnionych na liście odbywa się następująco:
Jeżeli ilość słów krótkich zajmowanych w pamięci operacyjnej przez blok jest nie większy od 1024, to jednemu blokowi (strukturze, polu, tablicy) odpowiada jeden blok TM. Jeżeli zaś blok jest większy niż 1024 słowa krótkie, to jest automatycznie dzielony i zapisywany w kolejnych blokach TM. Każdy blok TM ma wtedy po 1024 słowa, z wyjątkiem ostatniego, który może być mniejszy. Od wielkości bloków TM zależy pojemność taśmy magnetycznej oraz czas w jakim określona ilość słów zostanie zapisana. Im większe są bloki TM tym można więcej danych zmieścić na taśmie i szybkość

przesyłania określonej ilości słów będzie większa. Po wykonaniu zdania ZAPISZ, stacja jest gotowa do zapisu kolejnego bloku TM.

Jeżeli w trakcie zapisywania danych pojawi się sygnał końca taśmy, to nastąpi cofnięcie jej do położenia w jakim była przed rozpoczęciem wykonywania zdania ZAPISZ.

Na tak zapełnionej taśmie można jeszcze zapisać metrykę końca szpuli (MKS). Po wykonaniu zdania ZAPISZ fakt pojawienia się sygnału końca taśmy można stwierdzić badając wartość funkcji CHR.

Jeżeli zapis nastąpił poprawnie, to wartość CHR jest zero.

W przypadku wystąpienia sygnału końca taśmy wartość CHR jest równa 1.

Przykłady

Poprawne użycie zdania ZAPISZ:

```
CALKOWITE : TMA, I, BLOK, J
  BLOK (4) : BLOK
  BLOK (400-1,3) : A
  BLOK (199) : J,B
  TMA = 0
  OTWORZ TMA : WYNIKI
  ----------
  ----------
  ----------
  ZAPISZ TMA : B,J
               (zapisano w bloku TM nr 1 400 słów
                bloku B i w bloku TM nr 2 200 słów
                bloku J (J-całkowite))
  ZAPISZ TMA, I : A
  ----------   (I otrzyma wartość równą 3 a blok A
  ----------    będzie zapisany w blokach TM:
  ----------    3-1024 słów, 4-1024 słów,
  ----------    5-1024 słów i 6-128 słów).
```

STRUKTURA (200+100-1) : B

ZAPISZ TMA, BLOK 1 : B (BLOK 1 = 7;

Pole B łącznie z podpolem J zapisano w bloku TM numer 7)

Błędne użycie zdania ZAPISZ:

BLOK (7) : A

ZAPISZ TMA, I : * A

(TMA i I muszą być całkowite; przed nazwą bloku nie należy pisać gwiazdki; nie otwarto zbioru)

CALKOWITE : TMA

ZAPISZ TMA : B (B jest zmienną prostą)

ZAMKNIJ

Postać ogólna
ZAMKNIJ ($\left\{\begin{matrix} s \\ z \end{matrix}\right\}$) a gdzie: a jest wyrażeniem arytmetycznym całkowitym oznaczającym numer stacji.

Zdanie powyższe może mieć jedną z dwóch postaci:

1. ZAMKNIJ (S) a
   Powoduje zapisanie na taśmie, umieszczonej na stacji o numerze równym wartości wyrażenia a, metryki końca szpuli (MKS)
2. ZAMKNIJ (Z) a
   Zapisuje metrykę końca zbioru (MEK) oraz metrykę końca zapisu (MKZ) . Po zapisaniu MKZ następuje cofnięcie taśmy o jeden blok TM (ustawienie głowicy przed MKZ) . Po wykonaniu ZAMKNIJ wartość CHR jest równa zero gdy zapis nastąpił poprawnie i równa jeden, gdy metryki nie udało się zapisać.

Przykłady

Poprawne zdania ZAMKNIJ:

```
CALKOWITE : TMA
  TMA = 0
ZAMKNIJ (S) 1
ZAMKNIJ (Z) 2
ZAMKNIJ (Z) TMA
```

ZNAJDZ

Postać ogólna
ZNAJDZ a : y [,s] gdzie : a jest wyrażeniem arytmetycznym całkowitym wyznaczającym numer stacji, y jest nazwą poszukiwanego zbioru, s jest numerem sekcji zbioru.

Zdanie powyższe powoduje odszukanie zbioru o nazwie y na taśmie założonej na stacji o numerze a oraz sprawdzenie czy numer sekcji jest równy s. Jeżeli parametr zdania s jest opuszczony to oznacza, że poszukiwana sekcja ma numer równy 1.
Po znalezieniu zbioru urządzenie jest gotowe do odczytu pierwszego bloku TM znalezionego zbioru.

Po wykonaniu zdania ZNAJDZ, funkcja CHR może przyjąć jedną z wartości:

0 - zbiór został odnaleziony
1 - brak poszukiwanego zbioru na taśmie
2 - zbiór o podanej nazwie istnieje ale nie zgadza się numer sekcji
3 - w trakcie poszukiwań nastąpił błąd odczytu.

Przykłady:

```
ZNAJDZ 0 : DANE
GDY  CHR (0) = 0 : NASTEPNY, INACZEJ 1 BLEDY
---------
---------
---------
```

```
1 BLE)  GDY CHR (0) = 1 : 1BRAK, INACZEJ 2 BLEDY
  1 BRAK) TEKST WIERSZY 1
          BRAK ZBIORU NA TASMIE
             ---------
             ---------
             ---------
          ZNAJDZ 0 : DANE, 2
             ---------
             ---------
             ---------
   2 BLE) TEKST WIERSZY 1
          ZAŁOŻONO NIE TE SEKCJE LUB BLEDY ODCZYTU
          STOP NASTEPNY
          KONIEC
```

POSUN

Postać ogólna
POSUN a : n gdzie: a jest wyrażeniem arytmetycznym całkowitym wyznaczającym numer sekcji, n jest wyrażeniem arytmetycznym całkowitym wyznaczającym numer bloku TM.

Zdanie powyższe powoduje ustawienie urządzenia czytającego (piszącego) stacji przed blokiem TM o numerze n.

Ustawianie przeprowadzane jest na taśmie założonej na stacji o numerze a. Np. blok TM o żądanym numerze będzie przeczytany jako pierwszy gdy użyjemy zdania POBIERZ.

Zdanie POSUN nie zmienia wartości funkcji CHR.

Przykłady

```
   CALKOWITE : I
   BLOK (100) : A
   ---------
   ---------
   ---------
1) ZAPISZ 0, I : A
   ---------
   ---------
   POSUN 0 : I
   ---------
   ---------
```

(Urządzenie czytające piszące znajduje się przed blokiem TM zawierającym dane zapisane przez zdanie 1)

POBIERZ

Postać ogólna
POBIERZ a[, n] : b1,b2,...,bn gdzie: a jest wyrażeniem arytmetycznym całkowitym wyznaczającym nr stacji, n jest wyrażeniem arytmetycznym całkowitym określającym numer bloku TM, b1,b2,...,bn są to nazwy bloków i stanowią listę. Listę nazw bloków można przenosić.

Zdanie powyższe powoduje przeczytanie danych do pamięci operacyjnej z taśmy magnetycznej założonej na stacji o numerze a. Czytanie rozpoczyna się od bloku TM o numerze n, jeżeli parametr drugi istnieje lub od bieżącego bloku TM występującego tuż przed urządzeniem czytającym, jeżeli tego parametru brak. W procesie czytania, zawartości kolejnych bloków TM są przenoszone do obszarów pamięci wyznaczonych przez bloki.

Bloki otrzymują wartości w tej samej kolejności w jakiej ich nazwy występują na liście zdania POBIERZ. Pojemność każdego bloku musi być dokładnie taka jak suma długości bloków TM do niego wprowadzanych.

W szczególnym przypadku pojemność bloku musi być równa długości jednego bloku TM. Po wykonaniu zdania POBIERZ funkcja CHR może przyjąć jedną z poniższych wartości:

0 - zdanie zostało wykonane poprawnie

1 - w trakcie wykonywania zdania natrafiono na metrykę końca zbioru. Zawartość bloków będzie nieokreślona a głowica będzie zatrzymana przed metryką,

2 - natrafiono na metrykę końca szpuli,

3 - natrafiono na inny blok TM nie będący blokiem danych (np. metryka początku zbioru)

4 - wykryto błędy odczytu.

Przykłady

```
   CALKOWITE : I
   BLOK (100) : A,B
   BLOK (200+1) :C,D
      ---------
      ---------
      ---------
   OTWORZ 0 : ROBO
   ZAPISZ 0,I : B,A,D
   ZAMKNIJ (Z) 0
      ---------
      ---------
   ZNAJDZ 0 : ROBO  (przygotuj do odczytu blok TM
                     zawierający wartości bloku B)
   GDY CHR (0) = 0 : 1, INACZEJ NASTEPNY
   STOP NASTEPNY    (były błędy w poszukiwaniu)
1) POBIERZ 0 : D,C  (bloki TM o numerach 1 i 2 zawierające
                     wartości bloków B i A zostały odczy-
                     tane do bloku D (202◊2 słów) a
                     blok TM numer 3 przeczytany zostanie
                     do bloku C).
   POSUN 0 : 2
   POBIERZ 0 : D    (to zdanie jest błędne gdyż blok TM
                     nr 2 zawiera 202 słowa a blok TM nr 3
                     zawiera 404 słów stąd suma ich nie
                     jest równa pojemności bloku D (404)
                     a blok TM nr 2 był za mały).
```

ZMIEN

Postać ogólna
ZMIEN a gdzie: a jest wyrażeniem arytmetycznym całkowitym określającym numer stacji

Zdanie to powoduje wypisanie na monitorze następującego tekstu:

ZMIEN SZPULE NA c - dd

a następnie zawieszenie pracy programu.

c - dd jest zespołem cyfr określającym techniczny numer stacji:

c - numer kanału; jedna cyfra,

dd- numer modułu; dwie cyfry.

Po założeniu nowej szpuli operator może uruchomić dalszą pracę programu poprzez "odwieszenie" (patrz II część podręcznika, Czołówka translatora, opis zdania LIKWIDUJ STOP).

Nie ma potrzeby używania zdania ZMIEN jeżeli wiemy, że przed rozpoczęciem wykonywania się programu wynikowego na danej stacji jest na pewno założona żądana taśma.

Zdanie to musi być użyte jeżeli program współpracował z inną taśmą na tej samej stacji, w przeciwnym przypadku nowa taśma nie zostanie poprawnie odczytana (lub zapisana) Po wykonaniu omawianego zdania wartość funkcji CHR jest zawsze równa zero.

Drugie polecenie, które może być wydane operatorowi podczas pracy programu wynikowego z taśmami magnetycznymi, związane jest z blokadą mechaniczną zapisu.

Przy pierwszej próbie zapisu na danej stacji sprawdza się, czy zapis jest dozwolony. Jeżeli tak, to następuje zapis, jeżeli zaś jest zapis zabroniony, to otrzymujemy na monitorze tekst:

ODBLOKUJ c - dd

a praca programu jest zawieszona aż do "odwieszenia" przez operatora. Po "odwieszeniu" następuje ponowna próba zapisu.

Przykłady:

```
CALKOWITE : TMA
--------
--------
--------
ZMIEN O
TMA = O
--------
--------
--------
ZMIEN TMA
--------
--------
--------
```

BIBLIOTEKA

Postać ogólna
BIBLIOTEKA [:] a gdzie : a jest identyfikatorem tekstu źródłowego; dwukropek jest nieistotny

Zdanie BIBLIOTEKA powoduje przerwanie pobierania programu źródłowego z bieżącego wejścia translatora. Następnie, na taśmie magnetycznej zwanej biblioteką, następuje odszukanie tekstu o nazwie a i rozpoczęcie translacji, traktując go jako dalszy ciąg programu źródłowego napisanego w języku SAKO.

Taśmę magnetyczną oraz stację, na której taśma jest założona określa się przy pomocy zdania czołówki BIBLIOTEKA (patrz opis zdania czołówki translatora: BIBLIOTEKA CZEŚĆ II podręcznika).

a jest identyfikatorem tekstu (nazwą strony w sensie SMAD; patrz: System Magazynowania i Aktualizacji Dokumentów CZESC II podręcznika) składającym się z liter, cyfr i kropek nie więcej niż ośmiu znaków (wszystkie są identyfikujące). W identyfikatorze tekstu cyfra 0 i litera O są utożsamiane i traktowane jak cyfra.

Zdanie to nie może występować w tekście programu pobieranym z biblioteki. W bibliotece mogą znajdować się kompletne programy napisane w SAKO.

Przykłady

```
BIBLIOTEKA 12.4A
BIBLIOTEKA : RUKUT
```

PRZESTAW

Postać ogólna
PRZESTAW

Zdanie to może wystąpić w tekście źródłowym pobieranym z biblioteki i powoduje powrót do pobierania przerwanego programu źródłowego z wejścia translatora.

Zakończenie pobierania programu źródłowego z biblioteki może również nastąpić w momencie natrafienia na zdanie kończące program (KONIEC).

Przykłady

1) Program źródłowy na wejściu translatora:

```
----------
----------
A = 1
BIBLIOTEKA : WSTAWKA
B = 1
----------
----------
----------
KONIEC
```

Tekst na taśmie bibliotecznej

```
---------
---------
:: - - - -
:: - WSTAWKA -
   C = 1
---------
---------
   D = 1
PRZESTAW
:: - - - -
:: - INNE DANE -
----------
----------
```

Program równoważny:

```
---------
---------
A = 1
C = 1
---------
---------
D = 1
B = 1
---------
---------
---------
KONIEC
```

2) Program źródłowy:

```
--------
--------
--------
A = 1
BIBLIOTEKA : ALA
```

Zawartość taśmy magnetycznej:

```
---------
---------
:: - - - -
:: - ALA -
B = 1
---------
---------
KONIEC
:: - - - - -
---------
```

Tekst równoważny:

```
---------
---------
---------
A = 1
B = 1
---------
---------
KONIEC
```

KOMENTARZE

Postać ogólna
K) dowolny ciąg znaków zawarty w jednym wierszu : dowolny ciąg znaków zawarty w jednym wierszu

W języku SAKO przewiduje się dwa typy komentarzy tj. wiersze zaczynające się od dwukropka lub od dwóch znaków: K) . Są to zdania, które w całości pomija translator a mogą służyć jako objaśnienia programu lub też jako parametry dla programu SMAD. Parametr programu SMAD jest informacją potrzebną do zmagazynowania tekstu na taśmie bibliotecznej (patrz opis translatora POPR, CZESC II podręcznika).

Przykłady:

```
K)  TO JEST KOMENTARZ POMIJANY PRZEZ
K)  TRANSLATOR
:   TO TEŻ JEST KOMENTARZ
: :     - WSTAWKA -
```

JEZYK SAS

Postać ogólna

JEZYK SAS [:] [b1, b2,...,bn]

gdzie : b1, b2,...,bn są to nazwy zmiennych prostych, lub nazwy bloków; nazwy bloków mogą być poprzedzone gwiazdkami. Nazwy oddzielone przecinkami stanowią listę.

Dwukropek jest nieistotny.

Listę nazw można przenosić.

Dwa typy informacji są dostarczane translatorowi przez zdanie JEZYK SAS (zdanie to nie tworzy programu wynikowego):

1. Sam zwrot informuje translator, że dalszy ciąg programu aż do zdania JEZYK SAKO jest napisany w kodzie maszyny tzw. języku SAS (język SAS opisany jest w DODATKU F).
   Odcinek programu od zdania JEZYK SAS do JEZYK SAKO (włącznie) nazywamy wstawką w języku SAS.
2. Jeżeli dodatkowo po zwrocie występuje lista nazw zmiennych prostych lub nazw bloków (gwiazdki przed nazwami bloków mogą być pominięte; są nieistotne), to identyfikatory użyte we wszystkich dalszych wstawkach i identyczne z wymienionymi na liście, są zmiennymi języka SAKO. Pozostałe zaś identyfikatory są etykietami. Dla uniknięcia pomyłek zalecanym jest, aby deklarować wszystkie zmienne przy pierwszym wystąpieniu deklaracji JEZYK SAS.

Przykłady

```
CALKOWITE : I,J
BLOK (4) : A,B
   I = 1
   ----------
   ----------
   ----------
A (I) = 7.1
JEZYK SAS : A,I
   ----------
   ----------   } (to jest pierwsza wstawka)
   ----------
JEZYK SAKO
A(I + 1) = PWK (9.9)
JEZYK SAS
   ----------
   ----------   } (to jest następna wstawka)
   ----------
JEZYK SAKO
   STOP NASTEPNY
   KONIEC
```

FORMUŁA OPERACYJNA

Postać ogólna

([w1,w2,...,wn]) = n [([a1,a2,...,am])]

gdzie: w1,w2,...,wn stanowią listę wyników, w skład której mogą wchodzić nazwy zmiennych prostych lub poprzedzone gwiazdkami nazwy bloków,

n jest nazwą podprogramu definiującego omawiane zdanie (własność identyfikującą mają pierwsze trzy znaki nazwy).

a1,a2,...,am stanowią listę argumentów, w skład której mogą wchodzić następujące rodzaje:

- wyrażenia arytmetyczne,
- nazwy bloków (tablic,struktur); nazwa bloku poprzedzana jest gwiazdką,
- nazwy funkcji lub podprogramów; po nazwie występuje para nawiasów okrągłych,
- kropka

Obliczenia jakie są wykonywane przy pomocy omawianego zdania, zależą od operacji i obliczeń jakie zostaną wykonane przez zdanie stanowiące podprogram o nazwie n (patrz zdanie PODPROGRAM), na argumentach i wynikach formuły operacyjnej.

Wykonywanie formuły operacyjnej odbywa się w kilku poniżej omówionych fazach.

Podstawianie argumentów

Podstawianie argumentów (zwanych czasami argumentami aktualnymi) polega na przyporządkowaniu parametrom formalnym podprogramu, aktualnych wielkości występujących na liście argumentów aktualnych. O tym, jakie argumenty aktualne będą przyporządkowane, którym parametrom formalnym, decyduje tylko kolejność występowania jednych i drugich na listach (argumentów aktualnych w formule operacyjnej i argumentów formalnych w zdaniu PODPROGRAM). Pierwszemu argumentowi na liście argumentów formalnych przyporządkowany jest pierwszy argument z listy argumentów aktualnych, drugiemu drugi itd. W procesie podstawiania nie analizuje się odpowiedniości rodzajów argumentów i typów; programista powinien sam zapewnić tę odpowiedniość.

Zgodność typów argumentów polega na tym, że argument aktualny musi być tego samego typu co argument formalny; w procesie podstawiania nie następuje zmiana typów.

Zgodność rodzai polega na tym, że jeśli argumentami formalnymi są: nazwa funkcji, nazwa bloku, nazwa zmiennej prostej, to argumentami aktualnymi muszą być odpowiednio: nazwa funkcji lub nazwa podprogramu, nazwa bloku, wyrażenie arytmetyczne (o rodzaju argumentu aktualnego decyduje występowanie gwiazdek i nawiasów).

W procesie podstawiania są najpierw obliczane wartości tych argumentów, które są wyrażeniami arytmetycznymi i w tej kolejności, w jakiej występują one na liście. Następnie w zależności od typu i rodzaju argumentu aktualnego następuje właściwy proces podstawiania.

Poniżej rozpatrzymy wszystkie rodzaje argumentów:

Wyrażenie arytmetyczne

Wartość wyrażenia arytmetycznego (poprzednio policzona) jest nadana odpowiedniej zmiennej prostej formalnej w podprogramie; następuje tzw. podstawienie przez wartość. Typ podstawianej wartości jest taki jak typ wyrażenia arytmetycznego będącego argumentem.

Nazwa bloku

Podstawianie bloku polega na tym, że po podstawieniu obie nazwy (nazwa bloku formalnego i bloku aktualnego) będą reprezentowały to samo pole. Dokładniej, początki pól reprezentowanych przez nie będą takie same, natomiast struktura bloku formalnego (ilości indeksów i ich zakresy) pozostanie nadal nieokreślona i powinno się ją zadeklarować w podprogramie. Takie podstawienie będziemy nazywali podstawieniem przez nazwę.
W odróżnieniu od argumentów będących wyrażeniami arytmetycznymi w przypadku bloków wartości nie są przenoszone do podprogramu.

Nazwa funkcji lub podprogramu

Funkcje można podstawić w dwojaki sposób tj. przez wartość lub przez nazwę, zaś podprogram (podprogram definiujący formułę operacyjną) podstawiamy tylko przez nazwę.
Podstawienie funkcji przez wartość ma miejsce wtedy, gdy stanowi ona wyrażenie arytmetyczne tj. gdy w nawiasach po nazwie występują argumenty (co najmniej kropka dla funkcji bezargumentowej).
Podstawienie zaś przez nazwę następuje wtedy, gdy w nawiasach po nazwie nie występuje żaden argument.
Podstawienie funkcji (podprogramu) przez nazwę powoduje to, że obliczenie funkcji formalnej (formuły operacyjnej) w podprogramie powoduje obliczenie aktualnej funkcji formuły operacyjnej . W przypadku podstawiania funkcji (podprogramu) przez nazwę, wymagane są również zgodności typów i rodzai argumentów funkcji i podprogramów (formalnej i aktualnej); nie musi być zgodna ilość argumentów.

Uwaga: Nie wszystkie funkcje języka można podstawiać przez nazwę, ponadto przy niektórych funkcjach języka następuje automatyczna zamiana typów argumentów (patrz "Opis funkcji języka", DODATEK B).

Kropka

Na liście zamiast dowolnego z powyżej wymienionych rodzai argumentów może wystąpić kropka. Oznacza to, że dla odpowiedniej kolejnej pozycji na liście formalnych argumentów nie należy dokonywać podstawienia. Należy rozumieć, że podstawienie takie nastąpiło już poprzednio np. przy pomocy zdania PODSTAW. Jeżeli poczynając od pewnego miejsca listy, wszystkie podstawienia mają być pominięte, to pisanie kropek zamiast argumentów jest zbędne. W ten sposób lista argumentów aktualnych może być krótsza od odpowiedniej listy argumentów formalnych.
Po podstawieniu argumentów następuje tzw. podstawianie wyników.

Podstawianie wyników

Lista wyników w formule operacyjnej musi mieć takie same elementy (co do typu, rodzaju i kolejności) jak i odpowiednia lista wyników formalnych. Również długości tych list muszą być równe. Na liście wyników mogą występować tylko nazwy bloków lub nazwy zmiennych prostych. Bloki z listy wyników podstawia się przez nazwę, identycznie jak z listy argumentów. Natomiast wartości zmiennych prostych, przed wykonaniem podprogramu, nie podstawia się. Stąd, zmienne proste formalne znajdujące się na liście wyników w momencie rozpoczęcia wykonywania podprogramu nie posiadają wartości.

Po podstawieniu argumentów i wyników następuje kolejna faza wykonania formuły operacyjnej tj. przejście do wykonania podprogramu (wywołanie podprogramu). W trakcie wykonywania podprogramu mogą być zmieniane wartości bloków zarówno wymienionych na liście wyników jak i na liście argumentów. Wartości argumentów będących wyrażeniami mogą być tylko pobierane. Zmiennym prostym występującym na liście wyników można tylko nadawać wartości. Natomiast funkcje podstawiane przez nazwę mogą być wykonywane.

Po wykonaniu podprogramu następuje jeszcze ewentualne "pobranie" wartości zmiennych prostych formalnych wyszczególnionych na liście wyników formalnych i nadanie tych wartości odpowiednim zmiennym prostym wyszczególnionym na liście wyników formuły operacyjnej.

Jeżeli formuła operacyjna nie posiada listy argumentów lub nie podstawiamy żadnego argumentu to można również opuścić i nawiasy stojące za nazwą podprogramu. Czyli formy:

$$([w1,w2,\ldots,wn]) = n(\,)$$
$$([w1,w2,\ldots,wn]) = n$$
$$\text{i}\ ([w1,w2,\ldots,wn]) = n(\cdot)$$

są równoważne.

Natomiast gdy lista wyników jest pusta, to jej nawiasów opuszczać nie wolno. Zatem formuła operacyjna bezargumentowa może mieć postać:

$$(\,) = n$$

W szczególnym przypadku, gdy na liście wyników znajdują się same nazwy bloków, to formą równoważną do postaci ogólnej jest:

$$(\,) = n \quad [([a1,a2,\ldots,am,\ w1,w2,\ldots,wn])]$$

W tym przypadku do listy łącznej stosują się wszystkie reguły dotyczące listy argumentów. Forma ta jest niezbędna w przypadku stosowania zdania PODSTAW gdy zależy nam na podstawieniu wyników.

Przykłady

Poprawne formuły operacyjne

```
   BLOK (50) : B,C
     --------
     --------
     --------
1) (*C) = FORMULA (A, * B)
     --------
     --------
     --------
   (*C) = PORM (K)     (nie podstawiać wartości dla * CB)
     --------
     --------
     --------
   (*C) = PORM (., * C)  (a przy tym wykonaniu wartości
     ---------            dla AA nie podstawiać)
     ---------
     ---------
```

```
PODPROGRAM (*C1) = FOR (AA, * CB)
  STRUKTURA (40) : C1,CB
  ---------
  ---------
  ---------
```

W przykładzie 1) wartość zmiennej prostej A (wyrażenie arytmetyczne) zostanie nadana zmiennej formalnej AA, bloki B i CB będą reprezentowały to samo pole; również dotyczy to bloków C1 i C.

```
BLOK (50) : C,B
---------
---------
---------
```

( )= FOR (A, * B,* C) (ten zapis jest równoważny zapisowi 1 z poprzedniego przykładu)

( )= FOR (SIN (X), *B, * C) (tu funkcja języka SIN została podstawiona przez wartość)

( )= FOR (SIN (.), *B, * C) (tu też funkcja jest podstawiona przez wartość ale zakłada się, że jej argument został podstawiony wcześniej)

( )= F1 (SIN ( ), *B, * C) (tu funkcja SIN została podstawiona przez nazwę, takie podstawienie byłoby błędne, gdyby nastąpiło do funkcji FOR, gdyż odpowiedni parametr formalny jest zmienną prostą)

```
BLOK (0) : A,B,C
--------
--------
--------
```

( )= F (A, * B, . , * C) (bloki jednoelementowe można podstawiać przez wartość lub przez nazwę ale zgodnie z rodzajem odpowiednich argumentów formalnych)

Błędne użycie formuły operacyjnej:

```
BLOK (50) : B,C,D
CALKOWITE : Z
(.) = F1 ( ., ., *C)      (na liście wyników nie może
                           wystąpić kropka)

(A)  = FOR (*B,K)         (niezgodność rodzai argumentów)

(* C) = FOR (Z, * B)      (będzie niezgodność typów Z
                           i AA, gdyż w podprogramie brak
                           deklaracji CALKOWITE: AA)

( ) = FOR (A, * B, * D, * C)   (zbyt dużo argumentów)

PODPROGRAM (*C1) = FOR (AA, * CB)
   --------
   --------
   --------
```

LICZENIE WARTOSCI FUNKCJI

Postać ogólna funkcji
n (a1, a2,...,am) gdzie: a1, a2,...,am jest to lista argumentów aktualnych o postaci takiej samej jak lista argumentów aktualnych formuły operacyjnej

Funkcja nie jest zdaniem - jest wyrażeniem arytmetycznym, które posiada wartość.

Sposób i zasady podstawiania argumentów do funkcji są identyczne jak dla formuły operacyjnej, z wyjątkiem przypadku, gdy funkcja jest bezargumentowa lub gdy nie ma potrzeby podstawienia argumentów; wtedy w nawiasach musi wystąpić kropka.

Wyjątek stanowią też funkcje języka, gdzie podprogramu nie trzeba definiować; typy argumentów niektórych funkcji są automatycznie zmieniane, niektórych funkcji nie wolno podstawiać przez nazwę oraz typy wartości funkcji są z góry ustalone i nie trzeba ich deklarować (patrz "Opis funkcji języka", DODATEK B).

W procesie liczenia wartości funkcji, po podstawieniu argumentów następuje przejście do wykonania podprogramu o nazwie n. W wyniku wykonania podprogramu funkcja otrzymuje wartość. Typ funkcji musi być taki sam, jak typ wartości otrzymanej w wyniku wykonania podprogramu.

Oprócz tego, że funkcja otrzymuje wartość to również tak samo jak w przypadku formuł operacyjnych mogą być zmieniane wartości bloków będących argumentami funkcji.

Jeżeli podprogram definiujący funkcję zmienia wartości bloków będących argumentami aktualnymi i jeżeli na

wartości funkcji nam nie zależy, to do danego podprogramu można też odwoływać się przy pomocy formuły operacyjnej. Taka formuła operacyjna ma pustą listę wyników a po prawej stronie znaku równości występuje funkcja.

Przykłady

Poprawne użycie funkcji:

```
    CALKOWITE : F1 ( ),K
A = F(A,  ., *B, F2( ),1.7, K, F1 (X))

    ---------          (funkcję F2 podstawiono przez
    ---------           nazwę, funkcję F1 podstawiono
    ---------           przez wartość, szósty argument
    ---------           formalny podprogramu F musi
    ---------           być całkowity)

A = SI (B+2, SI (C,D))(zgodnie z regułą podstawiania
                       najpierw policzą się wartości:
                          B+2  i SI (C,D)   tj.
                       pierwszy raz wykona się funkcja
                       SI dla argumentów C i D, a do-
                       piero w następnej kolejności
                       nastąpi podstawienie argumentów
                       dla funkcji zewnętrznej. Gdyby
                       D było wyrażeniem zależnym od
                       funkcji SI to najpierw policzy-
                       łaby się jego wartość itd.)

B = SIN (A)
C = SIN (SI ( B+2, SI (C,D)))
```

Błędne użycie funkcji:

```
   BLOK (0) : B
F (*B)                (to nie jest zdanie, należy
   --------            napisać: ( ) = F (* B))
   --------

   Z = F (B)          (pominięto gwiazdkę, argument
   --------            formalny jest blokiem)

   Z = F1 (MAX( ))    (funkcji języka MAX nie wolno
   --------            podstawiać przez nazwę)
PODPROGRAM:  F  (*A)
   --------
   --------
   --------
```

PODSTAW

Postać ogólna
PODSTAW : n (a1,a2,...,am) gdzie: n jest nazwą funkcji lub podprogramu, do którego następuje podstawienie argumentów a1,a2,...,am jest to lista argumentów aktualnych o postaci takiej jak odpowiednia lista w formule operacyjnej lub funkcji.

Zdanie to powoduje podstawienie argumentów aktualnych do podprogramu definiującego funkcję lub formułę operacyjną. Zdanie PODSTAW nie powoduje wykonania podprogramu.

Podstawianie argumentów przebiega tak samo jak w formule operacyjnej. Przy pomocy PODSTAW nie można podstawiać elementów listy wyników, chyba że jest ona przedłużeniem listy argumentów (patrz: formuła operacyjna).

Przykłady

```
BLOK (1) : B,C
PODSTAW : F (A, ., *B)
          --------
          --------
          --------
(*C) = F( ., K)      (zdanie to będzie wykonane dla
          --------    następującej listy argumentów:
          --------    A,K, * B)
          --------
( )= F (., K, ., * C)
          --------   (zdanie równoważne powyższemu:
          --------    drugiej kropki nie wolno
          --------    opuszczać)
```

```
----------
(*C) = F( ., R)      (jeżeli podprogram nie zmienia
                      wartości bloku B to będzie on
----------            teraz wykonany dla listy:
----------
----------            A,R, * B)
(*C) = F (M)         (zaś teraz dla M,R, *B)
----------
----------
----------
PODSTAW : F (A,K, * B, * C)
( )= F               (te dwa zdania są równoważne
                      zdaniu:
                      (* C) = F (A,K, * B))
```

WROC

Postać ogólna
WROC

Bezparametrowe zdanie WROC może wystąpić tylko w podprogramie a wykonanie go powoduje zakończenie pracy podprogramu i powrót do programu wywołującego.

Powrót następuje do następnego zdania, jeżeli podprogram był wywoływany przy pomocy formuły operacyjnej. Natomiast powracamy do liczenia wartości wyrażenia, jeżeli wywołanie nastąpiło poprzez funkcję. W podprogramie może wystąpić więcej niż jedno zdanie WROC.

PODPROGRAM

Postać ogólna

PODPROGRAM: [([w1,w2,...,wn])]= n[([a1,a2,...,am])]

gdzie: w1,w2,...,wn jest listą wyników formalnych: elementem listy wyników formalnych może być nazwa zmiennej prostej formalnej lub nazwa bloku formalnego poprzedzona gwiazdką,

a1,a2,...,am jest to lista argumentów formalnych: argumentem formalnym może być:

- nazwa zmiennej prostej formalnej,
- nazwa bloku formalnego poprzedzona gwiazdką,
- nazwa funkcji (podprogramu), po której występuje para nawiasów,

n jest nazwą podprogramu zapoczątkowanego przez zdanie PODPROGRAM.

Nazwy zmiennych prostych formalnych, nazwy bloków formalnych i nazwy podprogramów formalnych mają taką samą postać zewnętrzną jak odpowiednio: nazwa zmiennej prostej, nazwa bloku, nazwa funkcji lub podprogramu.

Zdanie PODPROGRAM z nazwą n, jest pierwszym (w sensie kolejności występowania i w sensie kolejności wykonywania) z ciągu zdań zwanych podprogramem. Podprogram ten definiuje funkcję lub formułę operacyjną o tej samej nazwie (n). Do podprogramu należą wszystkie zdania napisane za nim aż do następnego zdania PODPROGRAM, ROZDZIAŁ lub KONIEC (wyłącznie).

Ciąg zdań od początku programu aż do pierwszego zdania PODPROGRAM (lub do zdań KONIEC, ROZDZIAL gdy nie ma podprogramów) nazywamy programem głównym.

Zdania PODPROGRAM stanowią jak gdyby punkty podziału programu na części (program główny i podprogramy). W każdej z tych części mogą występować nazwy i etykiety. Dodatkowo w podprogramach występują tzw. nazwy formalne tj. nazwy zmiennych prostych formalnych, nazwy bloków formalnych i nazwy funkcji oraz formuł operacyjnych formalnych.

Nazwy nie będące formalnymi będziemy czasami nazywać aktualnymi.

Nazwy formalne deklarowane są przy pomocy zdania PODPROGRAM i muszą być różne od nazw aktualnych występujących w rozważanym podprogramie.

W języku SAKO zakłada się, że podprogramy mogą być pisane niezależnie od siebie i od programu głównego. Stąd, wszystkie informacje dotyczące nazw (deklaracje: CALKOWITE, BLOK, TABLICA, STRUKTURA) muszą występować w każdym podprogramie i programie głównym oddzielnie. Jedynymi wspólnymi nazwami występującymi w całym programie są nazwy aktualnych funkcji języka lub podprogramów. Pozostałe nazwy występujące w różnych podprogramach lub w programie głównym, mimo że są identyczne mogą oznaczać różne pojęcia. Niezależność podprogramów jest tak dalece posunięta, że np. dla całkowitej funkcji definiowanej, mimo że jej nazwa jest wspólna dla całego programu, muszą wystąpić odpowiednie deklaracje CALKOWITE w tych podprogramach (ewentualnie też w programie głównym) w których została użyta. Etykiety w podprogramach i programie głównym mogą się powtarzać.

Do tej pory wszystkie uwagi dotyczące zdań języka SAKO związane były z przypadkiem występowania ich w programie głównym. Są one również w pełni aktualne w stosunku do zdań występujących w podprogramie, również z małymi wyjątkami stosują się do nazw formalnych. Jednakże te ostatnie należy nieco dokładniej omówić.

Wyszczególniając nazwy formalne na listach zdania PODPROGRAM określamy, które nazwy oznaczają zmienne proste, które funkcje lub formuły operacyjne, a które reprezentują

bloki. Nazwy bloków formalnych należy poprzedzić gwiazdką, a po nazwie funkcji formalnej (lub formuły operacyjnej) powinna wystąpić para nawiasów okrągłych. Następnie należy w podprogramie określić typ parametru formalnego.

Nazwy parametrów formalnych, dla których zakłada się, że ich wartości są całkowite, powinny bć wymienione na liście zdania CALKOWITE, według tych samych zasad jak dla nazw występujących w programie głównym (patrz CALKOWITE). Na liście zdania CALKOWITE, nazwy formalne mogą być przemieszane z nazwami aktualnymi.

Jeżeli parametrem formalnym jest nazwa bloku, to po zadeklarowaniu jego typu (jeżeli ma być całkowity), należy określić jego strukturę. Zakłada się, że pole pamięci dla bloku zostało zarezerwowane i należy podać ilość wymiarów i zakresy indeksów przy pomocy zdania STRUKTURA. Wszystkie uwagi podane poprzednio przy opisie zdania STRUKTURA i odnoszące się do nazw znajdujących się w programie głównym są słuszne również w stosunku do bloku formalnego, z tym, że zamiast deklaracji BLOK rezerwującej pole musi wystąpić (najlepiej na początku podprogramu) zdania STRUKTURA.

Na liście zakresów indeksów zdania STRUKTURA, mogą wystąpić w wyrażeniach arytmetycznych parametry formalne, co pozwala na przekazanie do podprogramu informacji o wymiarach podstawionego aktualnie bloku.

Pisząc zdanie PODPROGRAM zakładamy, że w momencie jego wywołania (patrz: formuła operacyjna i PODSTAW) parametry formalne będą miały podstawione wartości właściwego typu i rodzaju. Zmienne proste formalne wyszczególnione na liście wyników, po wywołaniu podprogramu, nie posiadają wartości, albowiem przewiduje się, że zostaną obliczone one w trakcie pracy podprogramu a po powrocie będą "pobrane" przez formułę operacyjną. Natomiast zmienne proste formalne wyszczególnione na liście argumentów powinny posiadać wartości i można je ewentualnie zmieniać, nie wpłynie to na zmianę wartości zmiennych aktualnych podstawianych do podprogramu.

Niezależnie, gdzie wyszczególniony jest blok formalny na liście wyników, czy na liście argumentów, to można pobierać aktualnie podstawione mu wartości lub nadawać nowe wartości przez niego reprezentowanemu blokowi.

Uwaga: W przypadku zmiennej indeksowanej będącej elementem bloku formalnego posiadającej wszystkie indeksy równe zero, listy indeksów opuszczać nie wolno.

Nazwa funkcji formalnej może wystąpić wszędzie tam, gdzie może wystąpić nazwa funkcji aktualnej i oznacza obliczenie wartości funkcji aktualnej od parametrów aktualnie podstawionych lub podstawionych uprzednio przez zdanie PODSTAW.

Nazwa formalna formuły operacyjnej może tworzyć formalne zdanie - formułę operacyjną. W trakcie wykonywania się jej, wykona się podprogram o nazwie aktualnie podstawionej z parametrami aktualnymi ostatnio podstawionymi (nie koniecznie w bieżącym podprogramie) . Wśród podprogramów wyróżniamy dwie zasadnicze grupy, które odróżnia się po postaci zdań PODPROGRAM.

1. Podprogramy definiujące funkcje, w których zdanie PODPROGRAM nie zawiera listy wyników. Postać ogólna zdania jest wtedy następująca:

$$\text{PODPROGRAM} : \underline{n}\left[\left(\left[\ \underline{a1},\underline{a2},\ldots,\underline{am}\right]\right)\right]$$

Wartość dla funkcji definiowanej jest nadawana przy pomocy zdania: PRZYPISYWANIE WARTOSCI FUNKCJI.

2. Podprogramy definiujące formuły operacyjne, w których zdania PODPROGRAM mają następującą postać ogólną:

$$\text{PODPROGRAM:}\ \left(\left[\underline{w1},\underline{w2},\ldots,\underline{wn}\right]\right) = \underline{n}\left[\left(\left[\ \underline{a1},\underline{a2},\ldots,\underline{am}\right]\right)\right]$$

Współpraca podprogramów i programu głównego

Program główny zaczyna się wykonywać zawsze poczynając od pierwszego zdania i ostatnim jego wykonywanym zdaniem z zasady powinno być zdanie STOP.

Kolejność wykonywania się zdań w programie głównym lub w podprogramie jest zgodna z kolejnością ich wypisania lub wskazywana jest przy pomocy etykiet przez zdanie sterujące.

Natomiast, aby wykonać zdanie PODPROGRAM należy wykonać formułę operacyjną z odpowiednią nazwą lub też wykonać obliczenie wyrażenia arytmetycznego, w którym występuje

odpowiednia funkcja. Ostatnim wykonującym się w podprogramie zdaniem powinno być WROC. W szczególnych przypadkach może być zdanie STOP wtedy, gdy powrót do programu głównego lub podprogramu jest zbędny.

Wywoływanie podprogramów przez program główny oraz podprogramów przez inne podprogramy nie powoduje w podprogramach wywołujących zmiany wartości innych zmiennych niż te które zostały podstawione jako wyniki lub jako bloki. Podobnie w podprogramach wywoływanych nie jesteśmy w stanie bezpośrednio zmienić wartość jakiejkolwiek zmiennej aktualnej, tylko sam fakt wykonywania się podprogramu powoduje zmiany wartości jego zmiennych aktualnych. Następnie dzięki ważności nazw funkcji i podprogramów w całym programie oraz dzięki posiadaniu aparatu PODSTAW i możliwości omijania podstawiania wszystkich argumentów (kropka jako aktualny argument) i ponieważ w podprogramach wartości podstawione dla argumentów formalnych nie ulegają zmianie jeżeli sam podprogram nie zmienia lub nie nastąpiło w międzyczasie ponowne podstawienie, to możemy budować uniwersalne podprogramy korzystające z niewielkiej ilości argumentów formalnych i używające funkcji formalnych, na które to funkcje formalne mogą być podstawione funkcje aktualne o bardzo zróżnicowanych własnościach.

Przykład ogólnej budowy programu:

```
CALKOWITE: A
BLOK (7) : A,B
----------
----------
----------
1) Z = F1 (*A)
----------
----------
----------
(*A) = F(*B)
----------
----------
----------
STOP NASTEPNY
```

program główny

```
PODPROGRAM : (*C) = F (*B)
 CALKOWITE: C
 STRUKTURA (3) : B,C
 ---------
 ---------
 ---------
1) WROC
 ---------
 ---------
 ---------
 WROC
 ---------
 ---------
 SKOCZ DO 1
```

podprogram definiujący formułę operacyjną

```
PODPROGRAM : F1 (*B)
 CALKOWITE : B
 STRUKTURA (7) : B
 ---------
 ---------
 WROC
 KONIEC
```

podprogram definiujący funkcję

Przykład <u>błędnej</u> budowy programu:

```
CALKOWITE : A
 BLOK (7) : A,B
 ---------
 ---------
 ---------
1) Z = F1 (*A)
 ---------
 ---------
 ---------
 (*A) = F (*B)
 ---------
 ---------
 ---------
 Z = 2
```

(po tym zdaniu program się nie zatrzyma ani nie wykona zdania 4)

```
PODPROGRAM: (*C) = F (*B)
 CALKOWITE : C
 STRUKTURA (3) : B,C
 ---------
 ---------
 ---------
 C (2) = 1                  (po wykonaniu tego zdania
                             nie nastąpi powrót do
                             programu głównego)
PODPROGRAM : P1 (* B)
 CALKOWITE : B
 STRUKTURA (7) : B
 ---------
 ---------
 ---------
 WROC
4) SKOCZ DO 1
 KONIEC                     (to zdanie nie należy
                             do programu głównego)
```

Przykłady zdań PODPROGRAM:

```
PODPROGRAM : ALARM          (bezargumentowa funkcja o
                             nazwie ALA)
PODPROGRAM : ALA            (jak wyżej)
PODPROGRAM AL1 (A, * B,F())(trójargumentowa funkcja
                             z parametrami formalnymi:
                             A - zmienna prosta
                             B - blok
                             F - funkcja lub podprogram
                                 podstawione przez
                                 nazwę)
PODPROGRAM : ( ) = ALARM( ) (bezargumentowa formuła ope-
                             racyjna o nazwie ALA)
PODPROGRAM : (A,* B) = F (C,* D,F1())    ( lista wyników:
                             A - zmienna prosta
                             B - blok
                                 lista argumentów:
                             C - zmienna prosta
                             D - blok
                             F1- funkcja lub podprogram)
```

Przykłady <u>błędnych</u> zdań PODPROGRAM:

PODPROGRAM : ALA (A (3)* 2) (wyrażenie nie może być parametrem formalnym)

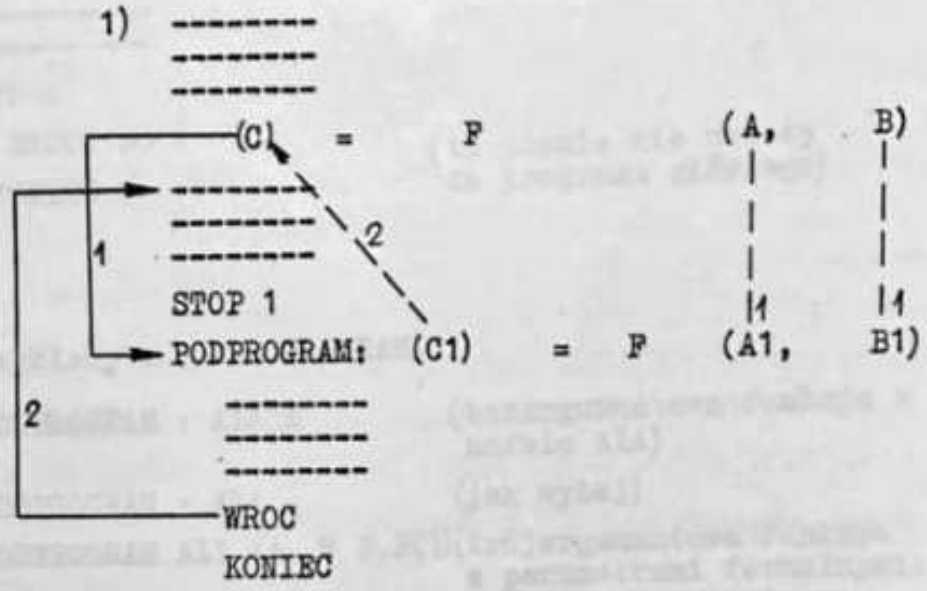
PODPROGRAM : (A,* B) = F (A,* D,F1()) (nie mogą być identyczne nazwy na liście wyników i liście argumentów)

Przykłady podstawiania i przekazywania wyników

W poniższych przykładach linie przerywane będą wskazywać możliwe kierunki przekazywania danych i wyników przy wykorzystaniu zdań PODSTAW, funkcji i formuł operacyjnych, a liczby przy liniach wskazywać będą kolejność przekazywania wartości.

Linie ciągłe (ponumerowane) będą wskazywać kolejność wykonywania się interesujących nas zdań.

Podstawienie z bezpośrednim wywołaniem



W powyższym przykładzie C otrzyma wartość równą wartości C1 dopiero po wykonaniu podprogramu, zaś wartości zmiennym formalnym A1 i B1 są nadawane przed jego wykonaniem

2)

```
---------
---------
---------
(K,   *C) = F (F1( ), *B , A)
---------
---------
---------
STOP 2
PODPROGRAM: (K,   *C1) = F (F2( ), *B3 ; A)
---------
---------
---------
WROC
PODPROGRAM: F1( )
---------
---------
---------
KONIEC
```

Kolejność 3 oznacza, że blokom aktualnym mogą być nadawane wartości w trakcie wykonywania się podprogramu.

3)

```
---------
---------
---------
A = 7 + F (A,  F1( ), *B)
---------
---------
---------
STOP 3
PODPROGRAM:  F(A1  F1( ), *B1)
---------
---------
---------
WROC
PODPROGRAM:  F1( )
---------
---------
---------
KONIEC
```

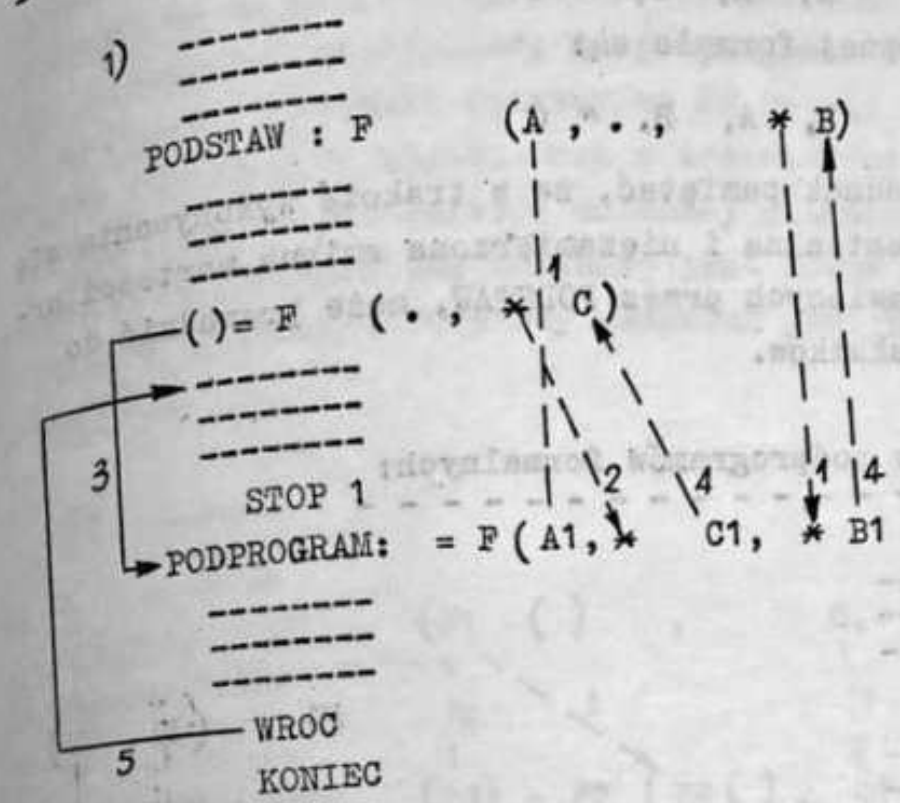
Aby obliczyć wartość funkcji, przerywa się liczenie wartości wyrażenia, wykonuje się podprogram i wraca, aby kontynuować przerwane obliczenie. Funkcja otrzymuje wartość w momencie powrotu z podprogramu patrz: przypisanie wartości dla funkcji . Należy ponadto zauważyć, że w trakcie wykonywania się podprogramu mogą być zmieniane wartości bloków aktualnych np.:

```
        --------
        --------
        --------
          B (1) = 3
    1) A = B (1) + F (*B) + B (1)
        --------
        --------
        --------
  PODPROGRAM : F (*B1)
        --------
        --------
        --------
          B1 (1) = 2
        --------
        --------
        --------
          WROC
          KONIEC
```

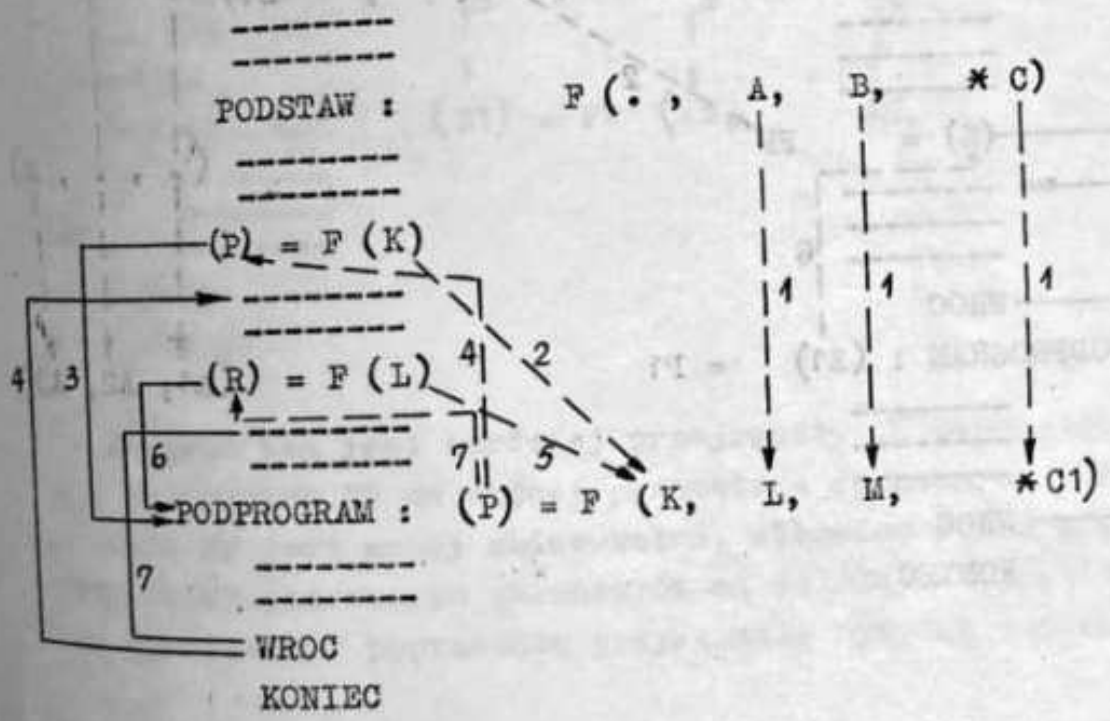
W zdaniu przypisującym 1) pierwszy składnik sumy (B (1)) ma wartość 3, ale trzeci składnik (B (1)) ma wartość równą 2, gdyż w trakcie obliczania funkcji F nastąpiła zmiana jego wartości.

Przy wykorzystywaniu powyżej opisanych efektów należy zwrócić baczną uwagę na kolejność wykonywania się działań.

Użycie zdania PODSTAW



Zdanie PODSTAW pozwala wcześniej podstawić niektóre (lub wszystkie) argumenty, w ten sposób między innymi można uniknąć zbędnych podstawień np.:



Przy pierwszym wykonaniu formuły operacyjnej argumenty aktualne będą:

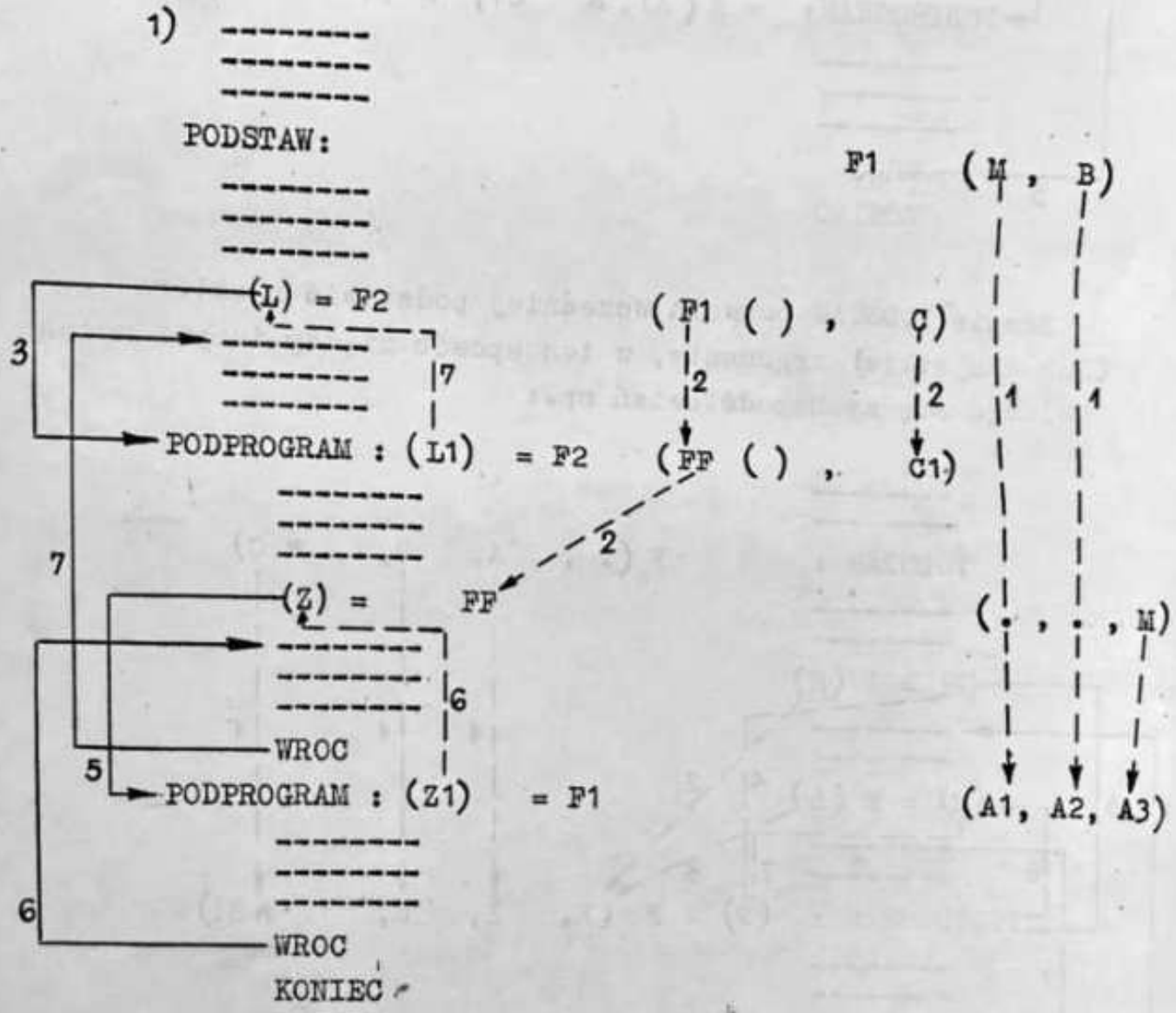
K, A, B, * C

zaś przy następnej formule są:

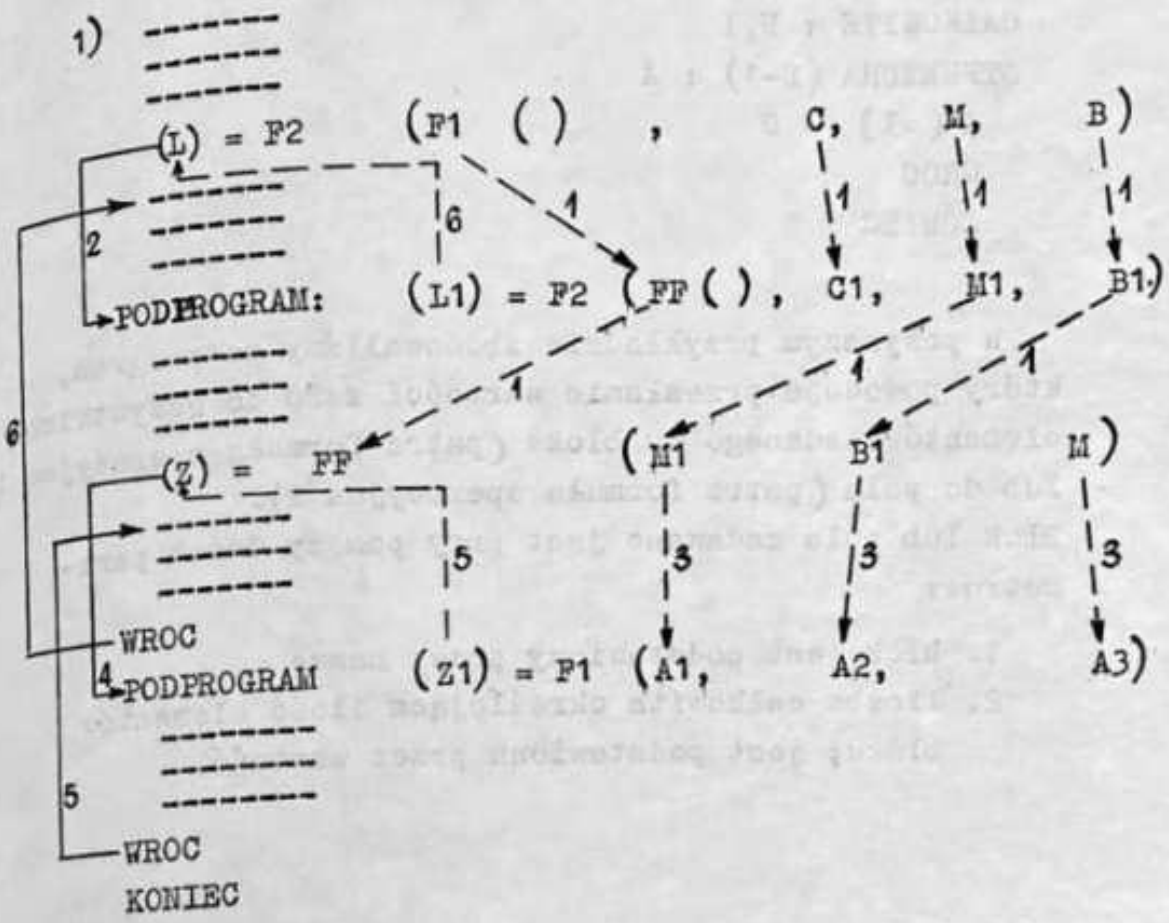
L, A, B, * C

Należy tu jednak pamiętać, że w trakcie wykonywania się podprogramu ewentualna i niezamierzona zmiana wartości argumentów podstawionych przez PODSTAW, może prowadzić do niepożądanych skutków.

Podstawianie do podprogramów formalnych:



W przykładzie umieszczonym na poprzedniej stronie wartości dwóch zmiennych aktualnych w programie głównym (M,B) przekazywane są do podprogramu F1 bezpośrednio. Następnie F1 jest podstawiane przez nazwę do podprogramu F2. W podprogramie F2 formalna formuła operacyjna FF będzie zastąpiona formułą aktualną F1 i dodatkowo w trakcie jej wykonania zostanie podstawiona wartość zmiennej aktualnej (M) do podprogramu F1. Wartość tej zmiennej jest różna od zmiennej M z programu głównego. Powyższy przykład jest równoważny programowi:



Program ten jest bardziej przejrzysty od poprzedniego ale podprogram F2 ma więcej parametrów formalnych a formalna funkcja FF jest mniej uniwersalna, albowiem rodzaje i typy dwóch pierwszych parametrów są ściśle tu określone, gdy tymczasem w poprzednim przykładzie były nieistotne.

Przykład podstawienia bloków i pól do podprogramu

```
BLOK (6,3) : A,B
  ---------
  ---------
  ---------
1) ( ) = F(*A,  7 ◇ 4)
  ---------
  ---------
  ---------
2) ( ) = F(*B,  7 ◇ 4 ◇ 2)
  ---------
  ---------
PODPROGRAM : ( ) = F(*A,N)
CALKOWITE : N,I
STRUKTURA (N-1) : A
  A (=I) = 0
  WROC
  KONIEC
```

W powyższym przykładzie zbudowaliśmy podprogram, który powoduje przesłanie wartości zero do wszystkich elementów zadanego mu bloku (patrz formuła operacyjna 1) lub do pola (patrz formuła operacyjna 2).
Blok lub pole zadawane jest przy pomocy dwóch parametrów:

1. blok jest podstawiony przez nazwę
2. liczba całkowita określająca ilość elementów bloku; jest podstawiona przez wartość.

PRZYPISANIE DZIESIETNEJ WARTOSCI DLA FUNKCJI

Postać ogólna
n ( ) = a gdzie: n jest nazwą podprogramu, w którym to zdanie występuje, a jest wyrażeniem arytmetycznym.

Zdanie powyższe powoduje obliczenie wartości wyrażenia arytmetycznego a i przypisanie jej dla funkcji, w której to zdanie się znajduje. Nazwa n musi występować ale nie niesie żadnej informacji. Zdanie to może wystąpić tylko w podprogramie.

Omawiane zdanie posiada dwie szczególne własności, o których należy pamiętać:

1. Musi występować bezpośrednio przed zdaniem WROC, gdyż w przeciwnym przypadku wartość funkcji będzie nieokreślona.
2. Przy nadawaniu wartości nazwie funkcji nie następuje zamiana typów tj. wartość nadana funkcji będzie takiego samego typu jak typ wyrażenia a niezależnie od typu funkcji.

Nie jest błędem użycie omawianego zdania w podprogramie definiującym formułę operacyjną, gdyż wartość ta zostanie pominięta.

Fakt ten może być wykorzystany przy tworzeniu uniwersalnych podprogramów, które mogą być wywoływane zarówno przez funkcję jak i przez formułę operacyjną.

Przykłady

Poprawne użycie przypisania wartości dziesiętnej dla funkcji:

```
1)  CALKOWITE : SI()
      ---------
      ---------
      ---------
    A = SI (X)      (tu zakłada się, że SI ma wartość
      ---------      typu całkowitego)
      ---------
      ---------
      STOP 1
    PODPROGRAM : SI (Z)
      CALKOWITE : A
      ---------
      ---------
      ---------
      SI( ) = A + 2  (tu wyrażenie A+2 musi być
                      całkowite)
      WROC
    KONIEC

2)  CALKOWITE : A
      ---------
      ---------
      ---------
    A = SA (X) + A   (SA jest ułamkowe)
      ---------
      ---------
      ---------
      STOP 2
    PODPROGRAM : SA (X)
    CALKOWITE : SA( )     (ta deklaracja na zdanie 1 nie
      ----------           ma wpływu)
      ----------
      ----------
    1)  SA( ) = A         (tu A musi być ułamkowe)
      WROC
      KONIEC
```

Błędne użycie zdania przypisującego wartość funkcji:

```
CALKOWITE : SI( )
  ---------
  ---------
  ---------
A = SI (X)          (zakłada się, że wartość SI jest
  ---------          typu całkowitego)
  ---------
  ---------
STOP 1
PODPROGRAM : SI (Z)
CALKOWITE : SI ( )
  ----------
  ----------
  ----------
SI ( ) = 2.0        (nie nastąpi zrównanie typów stąd
                     SI będzie miało wartość typu ułam-
                     kowego)
WROC
1) SI ( ) = 2       (między zdaniami 1 i 3 nie może
   Z = 5             wystąpić żadne zdanie)
3) WROC
   KONIEC
```

PRZYPISANIE OKTALNEJ WARTOSCI DLA FUNKCJI

Postać ogólna
n ( ) [a gdzie: n jest nazwą podprogramu zawierającego to zdanie, a jest wyrażeniem logicznym; znak [oznacza przypisanie wartości logicznej.

Zdanie to powoduje obliczenie wartości logicznej wyrażenia logicznego a (wyrażenia logiczne opisane są w paragrafie opisującym zdania przypisujące logiczne) i przypisanie obliczonej wartości nazwie funkcji (n) . Zdanie to może wystąpić tylko w podprogramie; musi występować bezpośrednio przed zdaniem WROC i przy przypisywaniu wartości nie następuje zamiana typów.

Przykład

```
A1B ( )[7.77.3 ◊ A
```

PROGRAMY WIELOROZDZIAŁOWE

Dotychczas omawialiśmy takie programy, które w całości mieszczą się w pamięci operacyjnej wraz z wartościami zmiennych i bloków zarówno programu głównego jak i podprogramów.

Istnieją jednak programy napisane w języku SAKO, które nie mieszczą się w pamięci operacyjnej i należy dzielić je na części. Każda z takich części musi z kolei mieścić się w pamięci operacyjnej wraz z zarezerwowanymi miejscami dla zmiennych i bloków. Każdą z takich części będziemy nazywać rozdziałem a program składający się z więcej niż jednego rozdziału nazywamy programem wielorozdziałowym.

Formalnie, program wielorozdziałowy jest to ciąg programów rozumianych w sensie dotychczasowym, z których każdy poprzedziliśmy zdaniem ROZDZIAL i z każdego, z wyjątkiem ostatniego, usunęliśmy zdanie KONIEC. W trakcie wykonywania się programu, w pamięci operacyjnej znajduje się tylko jeden rozdział, a pozostałe znajdują się w pamięci pomocniczej (na bębnie lub na taśmie magnetycznej).

Obecnie omówimy zdania związane z programami wielorozdziałowymi: ROZDZIAL, IDZ DO ROZDZIAŁU i KONIEC.

ROZDZIAŁ

Postać ogólna
ROZDZIAŁ [:] n gdzie: n jest liczbą całkowitą bez znaku z zakresu od zera do 32000 Dwukropek jest nieistotny

Zdanie to jest deklaracją, która nadaje rozdziałowi programu numer (n) oraz oddziela kolejne rozdziały od siebie.

Przed pierwszym rozdziałem nie musi występować zdanie ROZDZIAŁ i brak tego zdania jest równoważne zdaniu ROZDZIAŁ : 0

Każdy z rozdziałów jest niezależny i między innymi musi mieć własne podprogramy (podprogramy z innego rozdziału są niedostępne) . W jednym programie nie mogą występować rozdziały o tych samych numerach.

Przykłady programów wielorozdziałowych:

```
ROZDZIAŁ : 1
  --------
  --------
  --------
ROZDZIAŁ : 25
  --------
  --------
  --------
ROZDZIAŁ : 16
  --------
  --------
  --------
KONIEC
```

```
ROZDZIAL : 0
  --------
  --------
  --------
KONIEC

  --------
  --------
  --------
KONIEC
```

Powyższe dwa programy są szczególnymi przykładami programów wielorozdziałowych - są jednorozdziałowe i każdy z nich jest rozdziałem o numerze zero.

```
ROZDZIAL 3
  --------
  --------
  --------
A = F (X,Y)
  --------
  --------
PODPROGRAM : F (A,B)
  --------
  --------
  --------
   WROC
PODPROGRAM : F1
  --------
  --------
  --------
   WROC
ROZDZIAL : 1
  --------
  --------
A = F (X1,Y1)
  --------
  --------
PODPROGRAM : P (A,B)
  --------
  --------
  --------
   WROC
KONIEC
```

ROZDZIAŁ 3

ROZDZIAŁ 1

Podprogram P jest wykorzystywany w obu rozdziałach, stąd musi być dołączony do obu rozdziałów.

IDZ DO ROZDZIALU

Postać ogólna
IDZ DO ROZDZIALU [:] n gdzie: n jest numerem rozdziału (liczbą całkowitą bez znaku z zakresu od 0 do 32000). Dwukropek jest nieistotny.

IDZ DO ROZDZIALU jest zdaniem wykonującym się i powoduje zakończenie pracy aktualnego rozdziału. Następnie sprowadzenie do pamięci operacyjnej na jego miejsce rozdziału o numerze n i rozpoczęcie wykonywania go od pierwszego zdania programu głównego.

Należy zaznaczyć, że ponowny powrót do rozdziału, którego wykonywanie się przerwano za pomocą zdania IDZ DO ROZDZIALU, może nastąpić tylko przy pomocy innego zdania IDZ DO ROZDZIALU. A zatem tylko na jego początek. Ponowny powrót do wykonywanego poprzednio rozdziału, powoduje jego regenerację tzn. rozdział będzie miał postać taką jak przed pierwszym wykonaniem. W szczególnym przypadku wartości tablicy będą takie jak w programie źródłowym, mimo że np. w poprzednim wykonaniu były zmieniane.
Podobnie należy uważać, że argumenty formalne podprogramu nie mają wartości, mimo że nastąpiło podstawienie w poprzednim wykonaniu danego rozdziału. Ponieważ można tylko przechodzić z rozdziału do rozdziału przy pomocy zdania IDZ DO ROZDZIALU, powstaje zatem pytanie, w jaki sposób przekazywać dane między rozdziałami. Otóż, istnieją dwie metody:

1. Należy w rozdziale wywołującym zapisać dane w pamięci pomocniczej (na bębnie lub taśmie magnetycznej)

a następnie w rozdziale wywoływanym należy je pobrać. Metoda ta ma wadę, gdyż należy używać pamięci pomocniczej i dokonywać zbędnych przesłań. Ponadto adres pamięci pomocniczej musi być uzgodniony między rozdziałami.

2. W każdym rozdziale translator rezerwuje miejsca dla bloków programu głównego według tego samego algorytmu i poczynając od tego samego miejsca pamięci operacyjnej.

Podczas sprowadzania nowego rozdziału do pamięci, na miejsce poprzedniego, zawartość pamięci operacyjnej, poza potrzebną na rozdział, nie jest zmieniana. Stąd pisząc odpowiednio deklaracje BLOK w obu rozdziałach osiągamy efekt że w nowowprowadzonym do pamięci rozdziale odpowiednie bloki posiadają już wartość.

Przykład użycia zdań IDZ DO ROZDZIALU

```
ROZDZIAL : 1
  --------
  --------
  --------
IDZ DO ROZDZIALU : 2
  --------
  --------
  --------
ROZDZIAL : 2
  --------
  --------
  --------
IDZ DO ROZDZIALU : 3
  --------
  --------
  --------
ROZDZIAL : 3
  --------
  --------
  --------
IDZ DO ROZDZIALU : 1
  KONIEC
```

Na rysunku została pokazana kolejność wykonywania zdań programu wielorozdziałowego.

Przykłady przekazywania danych między rozdziałami:

```
BLOK (12) : A,B
BLOK ( 7) : C
    --------
    --------
    --------
IDZ DO ROZDZIALU : 1
ROZDZIAL : 1
BLOK (12) : A1,B1
BLOK  (7) : C1
    --------
    --------
    --------
KONIEC
```

Po przejściu z rozdziału 0 do rozdziału 1 wartości bloków A,B,C nie zostaną zniszczone i stąd w rozdziale 1 wartości bloków A1, B1, C1 w momencie rozpoczęcia wykonywania się go, będą takie same jak odpowiednio wartości bloków A,B,C w momencie wykonywania zdania IDZ DO ROZDZIALU : 1.

```
BLOK (12) : A,B
BLOK  (7) : C
    --------
    --------
    --------
IDZ DO ROZDZIALU : 1
ROZDZIAL : 1
BLOK (12 ◇ 2 + 7 + 2) : K
    --------
    --------
    --------
KONIEC
```

W powyższym przykładzie po przejściu do rozdziału 1 wartości zmiennych:

K, K (7), K (7+1), K (7+1+12+1)

będą takie same jak wartości:

C, C (7), B, A

Przykład wejścia do rozdziału nie tylko na początek.

```
CALKOWITE : A
BLOK (0) : A
  --------
  --------
  --------
  A = 2
IDZ DO ROZDZIALU : 1
ROZDZIAL : 1
CALKOWITE : I
BLOK (0) : I
SKOCZ WEDLUG I : 1A, 1C, 1D
  --------
  --------
  --------
1A) ------
    ------
    ------
1D) ------
    ------
    ------
1C) ------
    ------
    ------
KONIEC
```

Zmienna A otrzymała wartość 2 w rozdziale 0 przy pomocy powyższego układu deklaracji, zmienna I w rodzdziale 1 będzie miała tą samą wartość.

Pierwsze wykonujące się zdanie rozdziału 1 (SKOCZ WEDLUG) na podstawie wartości zmiennej I spowoduje przejście do wyznaczonych miejsc w rozdziale 1. W tym przypadku wykona się zdanie poprzedzone etykietą 1D.

KONIEC

Postać ogólna:
KONIEC [:] [n] gdzie: n jest numerem rozdziału i może przyjmować wartości od 0 do 32000. Dwukropek jest nieistotny.

Zdanie KONIEC spełnia dwie funkcje:

1. Wyznacza koniec programu,
2. Parametr zdania n informuje dodatkowo, który z rozdziałów w momencie startu programu wynikowego ma się wykonać jako pierwszy.
   Brak paramearu n oznacza, że należy wykonywać program poczynając od początku pierwszego rozdziału występującego w programie.

Przykłady użycia zdania KONIEC

1)
```
--------
--------
--------
KONIEC
```

2)
```
--------
--------
--------
KONIEC  0
```

Oba powyższe programy są jednorozdziałowe, start nastąpi w obu przypadkach poprawnie od początku programu.

```
ROZDZIAL : 2
  --------
  --------
  --------
ROZDZIAL : 3
  --------
  --------
  --------
KONIEC
```

Pierwszym wykonywanym rozdziałem będzie rozdział 2.

```
  --------
  --------
  --------
ROZDZIAL : 1
  --------
  --------
  --------
ROZDZIAL : 3
  --------
  --------
  --------
KONIEC : 3
```

Pierwszym wykonywanym rozdziałem będzie rozdział o numerze 3.

Zdania warunkowej translacji

Zdania języka SAKO mogą być poprzedzane apostrofem (') lub znakiem zapytania (?). W normalnych warunkach zdania takie nie podlegają translacji. Jednak przy pomocy zdania czołówki translatora (URUCHAMIANIE) można poinformować translator aby omawiane zdania podlegały translacji. Zdania poprzedzone apostrofem (znakiem zapytania) podlegają translacji przy określonych warunkach i mogą służyć np. w procesie uruchamiania do wypisywania pośrednich wyników.

Przykład

```
'DRUKUJ (2,3) : Z * 2
?DRUKUJ (2,4) : ALARM
```

CZĘŚĆ II

KORZYSTANIE
Z TRANSLATORA JĘZYKA SAKO

UWAGI OGÓLNE

Do zakodowania prostego zadania nie ma potrzeby korzystania z wiadomości dotyczących organizacji pracy translatora i nie są potrzebne wiadomości dotyczące organizacji programu wynikowego. Trzeba jednak wiedzieć, jak "opowiedzieć" systemowi operacyjnemu i translatorowi o tym, co należy z tym programem zrobić. Np. należy powiedzieć na jakim i na którym urządzeniu znajduje się program źródłowy, jakich usług dodatkowych życzymy sobie od translatorra, gdzie będą się znajdowały dane dla programu wynikowego itd.

W przypadku dużych i złożonych programów, trzeba znać pewne własności translatora, wiedzieć jak i gdzie jest rozmieszczany program wynikowy. Ponadto należy skorzystać z pewnego aparatu programowego dostarczanego przez translator, ułatwiającego uruchamianie programów, (usuwanie błędów) a po uruchomieniu pozwalającego na łatwą eksploatację.

Życzenia do translatora wyrażamy przy pomocy zdań języka czołówki translatora. Ciąg tych zdań stanowi czołówkę translatora. Czołówka translatora powinna zawsze występować przed programem napisanym w języku SAKO i musi zawierać przynajmniej jedno zdanie, oznaczające między innymi koniec czołówki. Czasami możemy życzyć sobie usług świadczonych przez translator, wśród których nie występuje zapotrzebowanie na translację (czołówka nie kończy się zdaniem TRANSLACJA) wtedy po czołówce program źródłowy nie występuje.

Każdy problem, który ma być rozwiązany na maszynie ZAM-41 w Systemie SO-141 musi być poprzedzony tzw. czołówką systemu tj. ciągiem zdań napisanym w języku JOM (Język Operacyjny Maszyny).

Zdania czołówki systemu definiują między innymi symboliczne wejścia i wyjścia dla translatorów i dla programu wynikowego, określają jakie translatory i w jakiej kolejności mają działać, podają czy ma być wykonywany program wynikowy lub też powołują do wykonania inne programy pomocnicze wchodzące w skład systemu.

Podsumowując powyższe można powiedzieć, że prosty problem prezentowany systemowi SO-141, np. polegający na translacji

programu napisanego w języku SAKO a następnie jego wykonaniu, składa się z:

1. czołówki systemu (napisanej w języku JOM)
2. czołówki translatora (napisanej w języku czołówki)
3. programu (napisanego w języku SAKO)
4. ewentualnych danych programu wynikowego.

W dalszych paragrafach tego opisu zostanie w miarę dokładnie omówiony język czołówki translatora i akcja podejmowana przez translator spowodowana zdaniami czołówki (patrz Czołówka Translatora). Oprócz tego podane zostaną pewne wybrane informacje o językach JOM (patrz: Czołówka Systemu) i SMAD (patrz: System Magazynowania i Aktualizacji Dokumentów) niezbędne do zrozumienia zdań języka Czołówki Translatora. Ponadto wspomniane będzie nieco o niektórych zdaniach języka JOS (Język Operacyjny Systemu).

Pełne opisy języków JOM, JOS i SMAD znajdzie czytelnik w literaturze firmowej dostarczonej ośrodkom posiadającym maszyny ZAM-41.

Z translatorem języka SAKO oraz z programem wynikowym produkowanym przez niego związany jest ściśle program zwany translatorem ZAKO.

Zadaniem tego programu jest wypisywanie informacji o programie wynikowym po zakończeniu jego działania. Postać wyników produkowanych przez ZAKO opisana jest w paragrafie zatytułowanym POST MORTEM

Do poprawnej pracy translatora języka SAKO potrzebna jest następująca konfiguracja maszyny:

- jedno urządzenie wejściowe,
- jedno urządzenie wyjściowe,
- pamięć operacyjna ok. 7 tysięcy słów (nie licząc pamięci potrzebnej dla SO-141, minimalna pamięć łączna powinna wynosić 12K słów)
- pamięć pomocnicza bębnowa minimum 8K słów (ilość słów zależy od długości i ilości rozdziałów programu źródłowego),
- jedna stacja taśm magnetycznych zawierająca taśmę systemu (w szczególnych przypadkach może nie istnieć).

Dla pełnego wykorzystania translatora, oprócz poprzednich urządzeń, konfiguracja powinna posiadać znacznie większą pamięć bębnową i nieco większą pamięć operacyjną oraz dwie dodatkowe stacje taśm magnetycznych zawierające:

- bibliotekę podprogramów i programów źródłowych,
- bibliotekę programów wynikowych wyprodukowanych przez translator języka SAKO.

CZOŁÓWKA SYSTEMU

Pełny opis czołówki systemu czytelnik znajdzie w publikacji Instytutu Maszyn Matematycznych pt.

ZAM-41, Oprogramowanie (System 141), część pierwsza, Język Operacyjny Maszyny

a spis błędów sygnalizowanych przez JOM podany jest w DODATKU J.

Poniżej ograniczymy się do omówienia sześciu podstawowych zdań języka JOM Język Operacyjny Maszyny :

WEJŚCIA PROGRAMOW
WEJŚCIA TRANSLATOROW
WYJŚCIA PROGRAMOW
WYJŚCIA TRANSLATOROW
TYTUL
PROBLEM

Ponadto w języku JOM są inne zdania z których należy wspomnieć o komentarzu. Komentarz, jest to wiersz programu zaczynający się od dwukropka (:) a kończący się znakiem LF. Wyżej wymienione zdania mają następującą postać:

- każde zdanie zaczyna się zwrotem określającym rodzaj zdania,
- zwrot oddzielony jest od parametrów znakiem dwukropka (:) ,
- zdania oddzielane są od siebie średnikami (;) ; znak nowej linii LF nie kończy zdania (nie ma potrzeby używania specjalnego znaku przeniesienia) ,
- kropka (.) kończy czołówkę systemu i może być używana zamiast (lub oprócz) średnika.

Zdania o zwrotach:

WEJŚCIA PROGRAMOW
WEJŚCIA TRANSLATOROW
WYJŚCIA PROGRAMOW
WYJŚCIA TRANSLATOROW

służą do definiowania symbolicznych wejść i wyjść translatorów i programów wykonywanych w problemie.
Jakie programy i translatory będą wykonywane w problemie określa się przy pomocy zdania o zwrocie:

PROBLEM

a zdanie o zwrocie TYTUL posiada funkcję pomocniczą.

Zdanie TYTUL

Zanim system przyjdzie do wykonywania pierwszego zdania problemu, wcześniej wypisuje na zerowe symboliczne wyjście translatora informacje o konfiguracji maszyny w jakiej dany problem jest wykonywany. W szczególności podawane są następujące informacje:

- pojemność pamięci operacyjnej (np. PAO = 15872)
- pojemność pamięci bębnowej (np. BEB = 34192)
- priorytet programu liczba po słowie ZEST .

Ponadto wypisywany jest takst będący parametrem zdania TYTUL, co np. pozwala szybko zidentyfikować do kogo należą wyniki.
Tekst będący parametrem zdania TYTUL jest zawarty między dwukropkiem (:) i średnikiem (;) lub kropką (.) .

Przykłady:

```
TYTUL : TO JEST PARAMETR
        ZDANIA TYTUL ;
```

Poniżej podajemy przykład standardowego tekstu wypisywanego przez system.

```
-----------------^-----------------------------------------------
ZAM-41 - SO 141 BCO (A) 224/404 - ZEST 0 (C)- 28.1.72 (28)/48
SUPERVISOR = SVC   PAO = 15872   BEB = 34192

TO JEST PARAMETR
ZDANIA TYTUL
```

Zdanie PROBLEM

Parametry zdania PROBLEM stanowią listę nazw translatorów lub programów powoływanych do wykonania w problemie. Nazwy oddzielone są od siebie przecinkami. Każde wywołanie będziemy nazywać zadaniem. Wśród nazw wyróżnione jest słowo PROGRAM, które oznacza wywołanie programu ostatnio przygotowanego przez translator.
Kolejność wykonywanych zdań jest zgodna z wypisaniem ich na liście.
Ilość elementów na liście zdania PROBLEM nie może przekroczyć ośmiu.

Przykłady:

1. PROBLEM : SAKO, PROGRAM ;

Powyższe zdanie informuje system, że najpierw należy powołać do działania translator języka SAKO a następnie po zakończeniu jego pracy przejść do wykonania programu wynikowego. Problem będzie zakończony w momencie zakończenia pracy programu wynikowego.

2. PROBLEM : SAKO, PROGRAM, PROGRAM, ZAKO ;

Powyższy problem składa się z następujących zdań:
- translacja przy pomocy translatora SAKO
- dwukrotne wykonanie programu wynikowego
- wywołanie translatora ZAKO, który spowoduje wypisanie informacji o przebiegu pracy programu podczas drugiego wywołania.

Definicja wejść-wyjść

Wszystkie zdania definiujące symboliczne wejścia i wyjścia posiadają identyczną budowę a różnią się tylko zwrotami. Zdania o zwrotach zaczynających się od słowa WEJŚCIA definiują urządzenia symboliczne wejściowe, a zdania zaczynające się od słowa WYJŚCIA definiują urządzenia symboliczne wyjściowe. Parametry każdego ze zdań, stanowią listę i oddzielone są od siebie parzecinkami. Jeden parametr na liście definiuje jedno urządzenie symboliczne. Definicja

polega na tym, że wskazanemu numerowi wejścia lub wyjścia przyporządkowuje się wskazane urządzenie rzeczywiste, podaje się rodzaj podprogramu obsługującego to urządzenie oraz dekoder - sposób przekodowywania znaków czytanych z danego urządzenia (lub pisanych na dane urządzenie).
W szczególnym przypadku definicja jednego urządzenia symbolicznego ma postać:

n = npp (nuz, nd)

gdzie: n - jest liczbą z przedziału 0 - 15 i oznacza numer definiowanego symbolicznego wejścia lub wyjścia,

npp - jest nazwą podprogramu obsługującego to urządzenie,

nuz - jest nazwą urządzenia rzeczywistego, związanego za pomocą tej definicji z definiowanym urządzeniem symbolicznym n,

nd - jest nazwą dekodera przekodowującego znaki.

Niektóre z parametrów definicji można opuścić, np. definicja może mieć postać:

n = npp (nuz)

W tym przypadku znaki wchodzące z urządzenia lub wyprowadzane na urządzenie nie są dekodowane.
Te same numery urządzeń symbolicznych zdefiniowane w zdaniach o różnych zwrotach oznaczają różne urządzenia symboliczne i znaczenie ich zależy tylko od definicji.
Znaczenie zdań definiujących wejścia-wyjścia są następujące:

WEJŚCIA PROGRAMOW - definiuje wejścia symboliczne programów

WYJŚCIA PROGRAMOW - definiuje wyjścia symboliczne programów

WEJŚCIA TRANSLATOROW- definiuje wejścia symboliczne translatorów

WYJŚCIA TRANSLATOROW- definiuje wyjścia symboliczne translatorów

Dla skrócenia zapisu, jeżeli symboliczne wejścia lub wyjścia translatorów i programów są identyczne (każde z tymi samymi parametrami) zostały wprowadzone zdania o zwrotach:

WEJŚCIA

i

WYJŚCIA

Każde z nich odpowiada parze zdań.

WEJŚCIA PROGRAMÓW
WEJŚCIA TRANSLATORÓW

i

WYJŚCIA PROGRAMÓW
WYJŚCIA TRANSLATORÓW

Poniżej podajemy nazwy niektórych urządzeń maszyny i związanych z nimi podprogramów obsługi oraz nazwy niektórych dekoderów.

Nazwy urządzeń

DW - drukarka wierszowa
CTP1 - czytnik taśmy papierowej nr 1
CTP2 - czytnik taśmy papierowej nr 2
PTP1 - perforator taśmy papierowej nr 1
PTP2 - perforator taśmy papierowej nr 2

TMA, TMB, ⋮, TMP } kolejne stacje taśm magnetycznych

Nazwy podprogramów

PCTP - podprogram czytania taśmy papierowej
PPTP - podprogram perforacji taśmy papierowej
PDW - podprogram obsługi drukarki wierszowej
PCTM - podprogram czytania taśmy magnetycznej zapisanej przez SMAD

Nazwy dekoderów

DM2C - dekoder czytania zamieniający znaki kodu M2 na kod KW6,

DM2P - dekoder pisania zamieniający znaki kodu KW6 na kod M2,

DFE6 - dekoder czytania zamieniający znaki kodu FERRANTI na kod KW6,

KØFE - dekoder pisania zamieniający znaki kodu KW6 na kod FERRANTI,

DDW1 - dekoder pisania zamieniający znaki kodu KW6 na zwykłe kody drukarek,

DDW2 - dekoder pisania zamieniający znaki kodu KW6 na rozszerzone kody drukarek.

Zależności między odpowiednimi kodami podane są w DODATKU A.

Przykłady czołówek systemu

WEJŚCIA : 0 = PCTP (CTP1,DM2C) ;
WYJŚCIA : 0 = PPTP (PTP1,DM2P) ;
TYTUL : PRZYKŁAD 1 ;
PROBLEM : SAKO, PROGRAM.

Powyżej opisany PROBLEM składa się z dwóch zdań: translacji programu napisanego w języku SAKO i wykonania programu wynikowego. Zdanie WYJŚCIA mówi, że wyjście translatora o numerze zero jest takie samo jak zerowe wyjście programu wynikowego, że translator i program posiadają po jednym wyjściu, i że wyjściem tym jest perforator taśmy papierowej numer 1 (PTP1) - obsługiwany przez podprogram PPTP.
Ponadto tekst na wyjściu będzie perforowany w kodzie M2 (DM2P) . Podobnie, wyjścia translatora i programu są zdefiniowane tak samo, tzn., że na tym samym wejściu tuż za programem znajdują się ewentualne dane programu wynikowego. Wejściem tym jest pierwszy czytnik taśmy papierowej (CTP1) a program i dane są zakodowane w kodzie M2 (DM2C).

Uwaga 1

Translator języka JOM zakłada, że jeżeli wejścia i wyjścia o numerach zero nie zostaną zdefiniowane, to jest to równoważne definicji podanej w powyższym przykładzie. Przykład powyższy można zatem napisać krócej:

1) TYTUL : PRZYKŁAD 1 ;
PROBLEM: SAKO, PROGRAM.

Uwaga 2

System zakłada, że czołówka systemu jest wyperforowana w kodzie M2 i jest podłożona do czytnika taśmy papierowej nr 1.

Uwaga 3

Zaleca się aby zawsze między czołówką systemu a czołówką translatora oraz między programem i danymi, pozostawić około 30 cm czystej taśmy.

W prostych przykładach najczęściej używaną czołówką systemu jest:

2) WYJŚCIA : 0 = PDW (DW, DDW1) ;
PROBLEM : SAKO, PROGRAM.

lub też

3) WYJŚCIA : 0 = PDW (DW,DDW1) ;
PROBLEM : SAKO, PROGRAM, ZAKO.

Czołówka 1) różni się od czołówki 2) tym, że w przypadku 2) wyniki translatora (teksty pisane przez translator) i wyniki programu wynikowego wypisane są na drukarce.
Dodatkowe zdanie w czołówce 3) (ZAKO) w porównaniu z czołówką 2) stosuje się, gdy program jest uruchamiany, w celu otrzymania pewnych informacji o programie (patrz opis zdań: POSTMORTEM, CHOWAJ) .

Następująca czołówka systemu:

4) WYJŚCIA PROGRAMÓW : 1 = PDW (DW,DDW1) ;
WYJŚCIA TRANSLATORÓW : 0 = PDW (DW,DDW1) ;
PROBLEM : SAKO, PROGRAM.

definiuje dwa wyjścia dla programu wynikowego o numerach zero i jeden oraz wyjście zero (domyślne) na perforator pierwszy w kodzie M2. Program wynikowy może na początku wyprowadzać wyniki na perforator (wyjście symboliczne nr zero) następnie przy pomocy funkcji języka DOL i SEL (patrz DODATEK B) można przełączyć program aby pisanie wyników odbywało się na drukarkę wierszową (wyjście symboliczne nr.1) Ilustracją powyższych uwag jest poniższa sekwencja zdań języka SAKO:

```
CALKOWITE : Z
BLOK (300) : B
Z = DOL  16+1,*B
  ---------
  ---------
  ---------
DRUKUJ ...      (wyjście na perforator)
Z = SEL (16+1)
DRUKUJ ...      (wyjście na drukarkę)
  ---------
  ---------
  ---------
Z = SEL (16)
DRUKUJ ...      (znów wyjście na perforator)
  ---------
  ---------
  ---------
```

Poniższa czołówka informuje między innymi, że urządzeniom symbolicznym wejściowym o numerach 0 i 1 przypisano urządzenia fizyczne będące taśmami magnetycznymi.

```
5)   WEJŚCIA TRANSLATORÓW : 0 = PCTM (TMA),
                            1 = PCTM (TMB)
                            2 = PCTP (CTP1,DM2C) ;
     WYJŚCIA TRANSLATORÓW : 0 = PDW (DW,DDW1) ;
     PROBLEM : POPR.
```

Dokładny opis tej czołówki czytelnik znajdzie w paragrafie zatytułowanym: SYSTEM MAGAZYNOWANIA i AKTUALIZACJI DOKUMENTÓW.

CZOŁÓWKA TRANSLATORA

Pierwszą czynnością translatora języka SAKO, po wywołaniu go przez system operacyjny, jest czytanie czołówki translatora napisanej w Języku Czołówki Translatora. W zależności od żądań wyrażonych w czołówce translatora podejmowane są odpowiednie akcje przez translator lub też odnotowywane są parametry zdań w celu późniejszego ich wykorzystania. Ogólna budowa zdań języka (Język Czołówki Translatora) jest podobna do budowy zdań języka SAKO:

- każde zdanie musi zaczynać się od nowego wiersza,
- każde zdanie zaczyna się od zwrotu identyfikującego go,
- jeżeli zdanie zawiera parametry to oddzielone są one od zwrotu znakiem dwukropek (:) ,
- znaki nieistotne występujące w zdaniu są pomijane,
- wiersze puste nie są liczone przy sygnalizacji błędów, sygnalizacja błędów syntaktycznych jest identyczna jak dla zdań języka SAKO,
- jeżeli ilość parametrów zdania jest większa niż jeden, to kolejne parametry oddziela się od siebie przecinkami,
- w niektórych zdaniach można opuścić wszystkie parametry; w takich przypadkach należy również opuścić dwukropek oddzielający je od zwrotu,
- listy parametrów można przenosić do następnego wiersza w sposób identyczny jak w języku SAKO (ostatnimi znakami w wierszu w takim przypadku muszą być kolejno przecinek i minus (,-)) ,
- w zwrotach identyfikujące są pierwsze cztery litery; pozostałe znaki zwrotu są pomijane.

Wiersz tekstu występujący wśród zdań języka czołówki i zaczynający się od dwukropka jest komentarzem.
Wśród zdań czołówki translatora wyróżniamy cztery: KONIEC, POBIERZ, ZAPISZ i TRANSLACJA. Tylko jedno z tych zdań może wystąpić w czołówce i musi być ostatnim zdaniem czołówki translatora (kończy czołówkę). Po zdaniach ZAPISZ

i TRANSLACJA powinien wystąpić program napisany w języku SAKO. Natomiast zdanie KONIEC i POBIERZ kończą czołówki, po których translator nie przechodzi do translacji programu źródłowego.

Pozostałe zdania mogą występować w czołówce translatora w dowolnej kolejności.

W dalszej części podajemy opis poszczególnych zdań; spis zdań języka czołówki translatora jest podany w DODATKU I. Kolejność opisywania zdań została przyjęta tak, aby w trakcie omawiania ich znaczenia, w miarę możliwości, jak najmniej powoływać się na informacje jeszcze nie podane.

TRANSLACJA

Postać ogólna
TRANSLACJA

Jest to zdanie bezparametrowe oznaczające koniec czołówki translatora. Ponadto informuje, że w następnych wierszach po nim występuje program źródłowy, który należy przetłumaczyć a program wynikowy pozostawić w pamięci bębnowej. Zdanie TRANSLACJA stanowi pełną i najkrótszą czołówkę translatora.

Program wynkowy pozostawiony w pamięci bębnowej jest gotów do wykonania. Jeżeli następnym zadaniem w problemie będzie wywołanie go (słowo PROGRAM) ,to system operacyjny przekaże mu sterowanie.

CHOWAJ SLOWNIKI

Postać ogólna
CHOWAJ SLOWNIKI

Zdanie to powoduje, że w trakcie translacji, oprócz przetłumaczonego programu, przechowywane są opisy rozdziałów (nazwy zmiennych, nazwy etykiet, nazwy bloków itd.) zwane słownikami i stanowią część programu wynikowego. Opisy te wykorzystywane są w trakcie pracy translatora ZAKO. Brak tego zdania w czołówce spowoduje, że słowniki nie zostaną przechowane.

POST MORTEM

Postać ogólna
POST MORTEM [: n] gdzie: n jest liczbą całkowitą określającą w ilu kolumnach będą wypisywane informacje przez translator ZAKO. Opuszczenie parametru n jest równoważne napisaniu liczby 120.

Wystąpienie tego zdania w czołówce jest niezbędne aby translator ZAKO, wywołany jako zadanie występujące po wywołaniu programu wynikowego, wypisał post mortem ostatnio wykonywanego rozdziału (patrz POST MORTEM - opis wyników translatora ZAKO).
Dodatkowo parametr zdania podaje szerokość arkusza wydawniczego przewidzianego dla wyników translatora ZAKO.
Ponadto warunkiem koniecznym efektywnego zadziałania translatora ZAKO jest, aby w skład programu wchodziły słowniki.

Przykład

Czołówka translatora:

```
POST MORTEM
CHOWAJ SLOWNIKI
TRANSLACJA
```

występująca przed programem źródłowym i poprzedzona czołówką systemu podaną w przykładzie 3) spowoduje, że po wykonaniu programu wynikowego, translator ZAKO zadziała efektywnie i wypisze post mortem (patrz opis translatora ZAKO).

URUCHAMIANIE

Postać ogólna
URUCHAMIANIE [: n] gdzie: n jest numerem rozdziału.

Zdanie to jest żądaniem do translatora aby w trakcie tłumaczenia rozdziału o numerze wskazanym przez parametr, nie były pomijane zdania warunkowej translacji poprzedzone znakiem apostrof (') lub znakiem zapytania (?) . Brak parametru oznacza, że we wszystkich rozdziałach, zdania warunkowej translacji będą tłumaczone. Natomiast brak tego zdania w czołówce oznacza, że we wszystkich rozdziałach będzie się je pomijać.

Przykłady:

1) URUCHAMIANIE
2) URUCH : 15

Poniższy odcinek programu:

3)
```
-----------
-----------
-----------
A = B + C
'DRUKUJ (2,3) : A
-----------
-----------
-----------
```

będzie równoważny:

```
-----------
-----------
-----------
A = B + C
DRUKUJ (2,3) : A
-----------
-----------
-----------
```

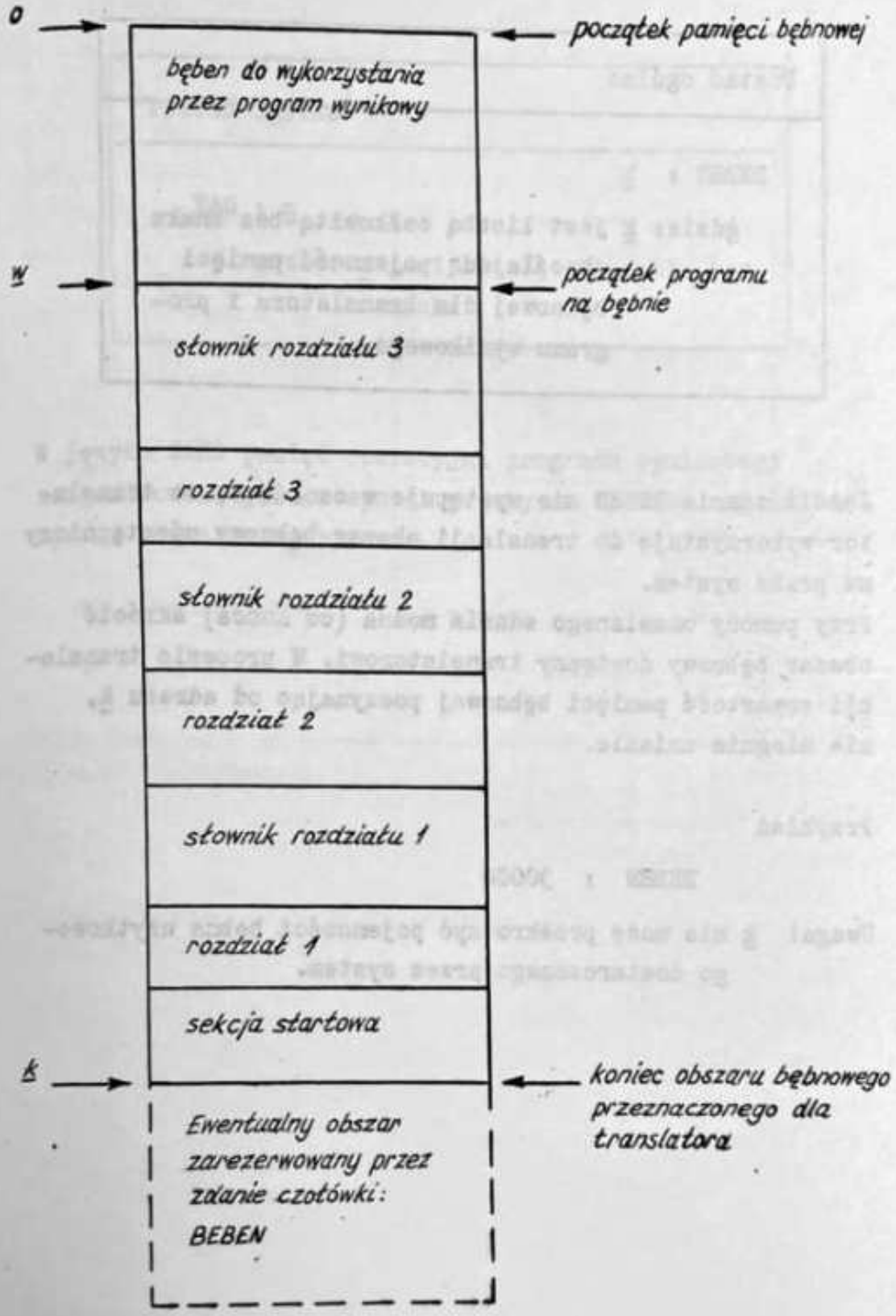
gdy w czołówce translatora zostanie użyte zdanie URUCHAMIANIE lub też będzie równoważny:

```
---------
---------
---------
A = B + C
---------
---------
---------
```

gdy z czołówki usuniemy je.

Zasady rozmieszczania programu na bębnie

Translator języka SAKO tłumaczy każdy rozdział programu oddzielnie i dopiero po całkowitym przetłumaczeniu jednego rozdziału przechodzi do tłumaczenia następnego. W przypadku gdy czołówka kończy się zdaniem TRANSLACJA, kolejne rozdziały programu wynikowego oraz ewentualne ich słowniki, są umieszczane kolejno od końca pamięci bębnowej przeznaczonej dla translatora. Reszta pamięci bębnowej , poczynając od zera, aż do najbliższego rozdziału, jest wykorzystywana podczas translacji następnych rozdziałów, a po zakończeniu translacji, może być wykorzystana przez program wynikowy (patrz PISZ NA BEBEN OD).
Wcześniej, przed rozpoczęciem translacji programu źródłowego, umieszczony jest na samym końcu pamięci bębnowej programu zwany sekcją startową. Przed startem programu wynikowego, sekcja startowa musi znajdować się na bębnie.
Rysunek na następnej stronie przedstawia sposób rozmieszczenia na bębnie programu wynikowego wraz ze słownikami, składającego się z trzech rozdziałów podlegających translacji w kolejności numerów: 1,2,3.


0
początek pamięci bębnowej
bęben do wykorzystania przez program wynikowy
W
początek programu na bębnie
słownik rozdziału 3
rozdział 3
słownik rozdziału 2
rozdział 2
słownik rozdziału 1
rozdział 1
sekcja startowa
k
koniec obszaru bębnowego przeznaczonego dla translatora
Ewentualny obszar zarezerwowany przez zdanie czołówki:
BEBEN

Rys. 1.

BEBEN

Postać ogólna
BEBEN : k gdzie: k jest liczbą całkowitą bez znaku określającą pojemność pamięci bębnowej dla translatora i programu wynikowego.

Jeżeli zdanie BEBEN nie występuje w czołówce, to translator wykorzystuje do translacji obszar bębnowy udostępniony mu przez system.
Przy pomocy omawianego zdania można (od końca) skrócić obszar bębnowy dostępny translatorowi. W procesie translacji zawartość pamięci bębnowej poczynając od adresu k, nie ulegnie zmianie.

Przykład

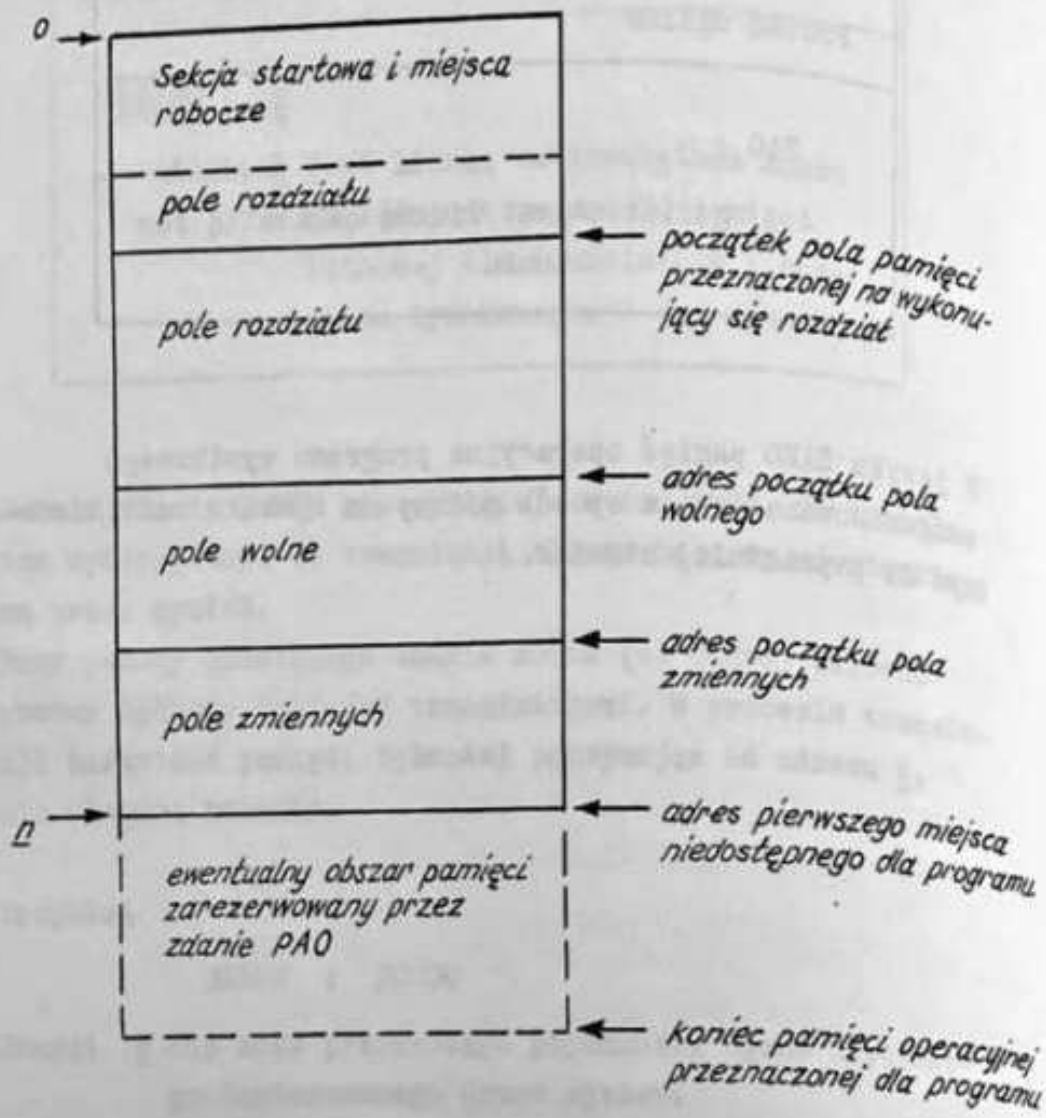
BEBEN : 30000

Uwaga! k nie może przekroczyć pojemności bębna użytkowego dostarczonego przez system.

PAO

Postać ogólna
PAO : n gdzie: n jest liczbą całkowitą bez znaku

W języku SAKO pamięć operacyjna programu wynikowego zorganizowana jest w sposób podany na rysunku zamieszczonym na poprzedniej stronie.


0
Sekcja startowa i miejsca robocze
pole rozdziału
początek pola pamięci przeznaczonej na wykonujący się rozdział
pole rozdziału
adres początku pola wolnego
pole wolne
adres początku pola zmiennych
pole zmiennych
n
adres pierwszego miejsca niedostępnego dla programu
ewentualny obszar pamięci zarezerwowany przez zdanie PAO
koniec pamięci operacyjnej przeznaczonej dla programu

Rys. 2

Wielkości i adresy początków pól wolnych, różnią się między sobą i zależą od długości rozdziałów i ilości zmiennych w nich użytych. Natomiast całkowita pamięć operacyjna przeznaczona dla rozdziałów jednego programu jest wielkością stałą wyznaczoną przez zdanie PAO,
n - jest adresem pierwszego niedostępnego słowa.
Jeżeli omawiane zdanie w czołówce translatora nie wystąpi, to przyjmuje się, że program wynikowy będzie wymagał takiej pamięci operacyjnej, jaką dysponuje w danej chwili translator.
Przy pomocy tego zdania możemy zarządać od translatora aby wyprodukował program dla maszyny o pamięci operacyjnej różnej od tej, na której odbywa się translacja.

Przykład

PAO : 12000

NAZWA

Postać ogólna
NAZWA : t gdzie: t jest identyfikatorem składającym się z liter i cyfr, zaczynającym się od litery; zawiera nie więcej niż 6 znaków.

Zdanie to przypisuje nazwę t tłumaczonemu programowi. Jeżeli zdanie NAZWA nie występuje w czołówce, to program nie jest nazwany. (ma nazwę PROGPR).
Wskazanym jest, aby pisać to zdanie przy korzystaniu ze zdania OPISZ PROGRAM i niezbędne jest przy korzystaniu ze zdania ZAPISZ. W przypadku zdania ZAPISZ, nazwa programu jest jednocześnie nazwą zbioru zawierającego ten program (patrz zdania: ZAPISZ, POBIERZ).

Przykłady

NAZWA : E7BE1
NAZWA : PRORK1

Uwaga! Nie należy używać nazw zaczynających się od POC i KON (patrz opis zdania ZAPISZ).

OPISZ PROGRAM

Postać ogólna
OPISZ PROGRAM

Zdanie to jest żądaniem wypisywania w procesie translacji informacji o programie wynikowym. Informacje wypisywane są na zerowe wyjście symboliczne translatorów. Opis programu składa się z opisów poszczególnych rozdziałów i opisu całego programu.
Opisy poszczególnych rozdziałów podają plan wykorzystania pamięci operacyjnej przez poszczególne rozdziały (rys.2)
Natomiast opis całego programu podaje:

nazwę programu, ilość rozdziałów (słowniki są również liczone), ilość błędów syntaktycznych wykrytych w programie, ilość miejsc pamięci operacyjnej wykorzystywanej przez rozdziały oraz rozkład programu na bębnie (rys.1).

IGNORUJ BLEDY

Postać ogólna
IGNORUJ BLEDY

Błędy sygnalizowane w procesie translacji, można podzielić na trzy grupy:

- błędy ostrzegawcze, które nie mają wpływu na to, czy program będzie wykonywany,
- błędy syntaktyczne, które nie powodują natychmiastowego przerwania translacji ale powodują niedopuszczenie programu do wykonania (wywołanie PROGRAM w czołówce systemu zostanie pominięte),
- błędy syntaktyczne, wykrycie każdego z nich powoduje natychmiastowe przerwanie translacji, program będzie pominięty bezwarunkowo.

Zdanie IGNORUJ BLEDY powoduje, że mimo wystąpienia błędów z grupy drugiej, program zostanie dopuszczony do wykonania.

KLUCZE

Postać ogólna
KLUCZE [: n1,n2,...,nk] gdzie : n1,n2,...,nk jest listą liczb całkowitych bez znaku, każda z liczb może przyjmować wartości z przedziału od zera do 23 włącznie.

Zdanie KLUCZE służy do wstępnego ustawienia zawartości rejestru kluczy. Liczby, występujące na liście omawianego zdania, wskazują numery kluczy, które mają być w stanie jeden (patrz opis zdania: GDY KLUCZ), pozostałe pozycje rejestru kluczy pozostają w stanie zero. Brak listy oznacza ustawienie zer w całym rejestrze.
W czasie pracy programu, zawartość rejestru kluczy może być zmieniana za pośrednictwem monitora. W celu ustawienia zera lub jedynki w odpowiedniej pozycji rejestru kluczy, należy na monitorze wypisać tekst według wzoru:

n - W - k - b

gdzie:

n - jest numerem priorytetu programu w jakim on pracuje. Priorytet programu może być: 0 lub 1, i podany jest w standardowym tekście wypisywanym przez system przed rozpoczęciem wykonywania problemu (patrz opis zdania czołówki systemu TYTUL).

k - jest numerem pozycji w rejestrze kluczy i może przyjmować wartości od zera do 23 włącznie,

b - jest wartością jaką należy wpisać do pozycji k rejestru kluczy (0 lub 1),

- znak kończący formułę, czasami na klawiaturze odpowiada mu znak zapytania (?).

Fakt zapisania odpowiedniej wartości w rejestrze kluczy jest potwierdzany przez wypisanie na monitorze zawartości rejestru kluczy (24 pozycje zer lub jedynek).

Przykłady:

0-W-17-1 *	(ustaw klucz 17 w stan 1)
0-W-16-0 *	(ustaw klucz 16 w stan 0)

LIKWIDUJ STOP

Postać ogólna
LIKWIDUJ STOP

W typowym zastosowaniu zdanie języka SAKO - STOP powoduje zakończenie pracy programu (patrz opis zdania STOP).

Zdanie LIKWIDUJ STOP powoduje, że zdanie STOP zostaje przetłumaczone przez translator na tzw. "zawieszenie" programu (stop dynamiczny), a następnie badanie czy rejestr kluczy zawiera same jedynki. Program będzie dalej kontynuowany jeżeli:

- zawartość rejestru kluczy będzie różna od samych jedynek (patrz: opis zdań GDY KLUCZ i KLUCZE) ,
- nastąpi tzw. likwidacja stanu zawieszenia.

Program będzie kontynuowany od miejsca wskazanego przez parametr zdania STOP.
Stan zawieszenia programu można zlikwidować wypisując na monitorze następującą formułę:

n - S*

gdzie n - jest numerem prioryteru programu i może przyjmować wartość 1 lub 0 (sposób ustalenia priorytetu podany jest w opisie zdania KLUCZE).

Jedynki można **załadować** do rejestru kluczy na jeden z trzech sposobów:

- przez załadowanie ich przy pomocy zdania KLUCZE, przed rozpoczęciem pracy programu,
- przez zmianę zawartości rejestru kluczy metodą podaną w opisie zdania KLUCZE w czasie pracy programu lub w stanie "zawieszenia",

- przez zastosowanie czterech następujących formuł:

n - W - 5 - 63 *

n - W - 11 - 63 *

n - W - 17 - 63 *

n - W - 23 - 63 *

napisanych na monitorze w czasie pracy programu lub w stanie zawieszenia.

Każda z powyższych metod jest czasochłonna i dlatego zaleca się aby jak najrzadziej stosować zdanie LIKWIDUJ STOP.

NIWELUJ NADMIARY

Postać ogólna
NIWELUJ NADMIARY [: {TAK / NIE}] Brak parametru oznacza TAK.

Zdanie NIWELUJ mówi, jak należy traktować w programie wynikowym, przekroczenie zakresu liczby ułamkowej:

- gdy powstanie tzw. nadmiar zmiennoprzecinkowy tj. gdy bezwzględna wartość liczby będzie większa od $(1-2^{-38}) \times 2^{255}$,
- gdy powstanie tzw. niedomiar tj. gdy bezwzględną wartość wyniku działania będzie mniejsza od 2^{-256} i różna od zera (patrz LICZBA UŁAMKOWA).

Parametr NIE mówi, że wystąpienie nadmiaru lub niedomiaru jest błędem programu wynikowego i powoduje zakończenie jego pracy (patrz spis błędów etapu wykonania DODATEK H).

Zdanie NIWELUJ z parametrem TAK mówi, że przekroczenie zakresu liczb ułamkowych w programie wynikowym nie powinno być sygnalizowane jako błąd, tylko ma następować zamiana wartości według reguł:

- w przypadku niedomiaru, wartością działania jest zero ułamkowe,
- w przypadku nadmiaru dodatniego wartością działania jest maksymalna liczba tj. $+ (1-2^{-38}) \times 2^{+255}$,
- w przypadku nadmiaru ujemnego wartością działania jest liczba minimalna $- (1-2^{-38}) \times 2^{+255}$

Brak w czołówce zdania NIWELUJ jest równoważny wystąpieniu tego zdania z parametrem NIE.

Przykłady

NIWELUJ NADMIAR

NIWELUJ NADMIAR : NIE

WYMAGANY BEBEN

Postać ogólna
WYMAGANY BEBEN [: n] gdzie : n - jest liczbą całkowitą bez znaku

Zdanie to, ma zastosowanie w dwóch przypadkach:

- gdy chcemy aby po translacji został zasygnalizowany błąd nr 210, jeżeli wolny bęben będzie miał mniej niż n słów (patrz rys.1),
- gdy będziemy stosować zdanie MINIMALIZUJ.

Brak w/w zdania lub rak parametru oznacza, że bęben wolny dla programu wynikowego nie jest wymagany; równoważne jest napisanie:

WYMAGANY BEBEN : 0

Przykład

WYMAGANY BEBEN : 17000

MINIMALIZUJ

Postać ogólna
MINIMALIZUJ

Zdanie to jest ściśle związane ze zdaniem WYMAGANY BEBEN i powoduje przesunięcie na bębnie programu wynikowego tak, aby wolny bęben był dokładnie równy obszarowi zadeklarowanemu w zdaniu WYMAGANY BEBEN.

Zdania: MINIMALIZUJ i WYMAGANY BEBEN zostały przewidziane przede wszystkim do zmniejszenia wykorzystywanego bębna przez program wynikowy w związku z włączeniem go do Biblioteki Systemu przy użyciu translatora WYBI lub BIB1 (patrz: ZAM-41, System Oprogramowania SO-141); i tylko w tych przypadkach jest sens je stosować.

BIBLIOTEKA

Postać ogólna
BIBLIOTEKA: nt, ns [,M] gdzie: nt - jest nazwą stacji taśm magnetycznych, na której jest założona szpula o numerze ns, ns - jest identyfikatorem literowo cyfrowym składającym się z co najwyżej 6 znaków.

Zdanie BIBLIOTEKA spełnia następujące funkcje:
- informuje, że w programie źródłowym występują zdania BIBLIOTEKA (patrz opis zdań BIBLIOTEKA i PRZESTAW),
- powiadamia translator, że taśma o numerze szpuli ns jest założona na stacji o nazwie nt,
- powoduje otwarcie wskazanej taśmy (do czytania).

Nazwy stacji są takie same jak w czołówce systemu tj. TMA,TMB,...,TMF. Numery szpul będą kontrolowane z chwilą wprowadzenia w SO 141 metrykowania szpul.
Parametr trzeci (litera M) jest związany z tzw. monitowaniem tj. zmianą parametrów nt i ns za pośrednictwem monitora (patrz: "Zasady monitowania"). Użycie tego parametru pozwala na współpracę z monitorem celem zmiany lub potwierdzenia ważności zadanych parametrów.
Sposób tworzenia taśmy bibliotecznej podany jest w paragrafie zatytułowanym: SYSTEM MAGAZYNOWANIA I AKTUALIZACJI DOKUMENTÓW.

Przykłady:

BIBLIOTEKA : TMB, 127
BIBLIOTEKA : TMA, P34717, M

ZAPISZ

Postać ogólna

ZAPISZ : nt, ns, {KON / POC / np} [, {M / N}]

gdzie : nt - oznacza nazwę stacji taśm magnetycznych,
ns - jest numerem szpuli,
np - jest nazwą programu już zapisanego na wskazanej szpuli

Zdanie ZAPISZ kończy czołówkę i informuje translator, że program napisany w języku SAKO i występujący po ZAPISZ ma być po przetłumaczeniu zapisany na wskazanej taśmie magnetycznej zwanej biblioteką programów wynikowych. Taśmę magnetyczną wskazują dwa pierwsze parametry tj.nazwa stacji (nt) oraz numer szpuli (ns) , która powinna być założona na tej stacji. Nazwa stacji może być: TMA, TMB,...,TMF ale musi być różna od nazwy stacji biblioteki programów źródłowych (patrz BIBLIOTEKA). Numer szpuli ma postać taką samą jak w przypadku zdania BIBLIOTEKA. Trzeci parametr zdania ZAPISZ podaje lokalizację zapisu na wskazanej taśmie:

- POC oznacza, że program ma być zapisany od początku taśmy magnetycznej. W tym przypadku zakłada się, że dotychczasowa zawartość taśmy nas nie interesuje,
- KON oznacza, że były dokonywane zapisy i że program ma być dopisany na taśmie po ostatnim już zapisanym programie,

- np jest nazwą programu już zapisanego, zamiast którego ma być wpisany nowy. W tym przypadku zakłada się, że wskazany program i ewentualne dalsze, zostaną wymagane.

Nazwę nowozapisanego programu zadaje się przy pomocy zdania NAZWA.

Ostatni parametr jest wskaźnikiem monitowania, brak parametru lub napisanie N są stanami równoważnymi i oznaczają brak monitowania. Wskaźnik monitowania równy M umożliwia zmianę pierwszych trzech parametrów (patrz: "Zasady Monitowania").

Wraz z programem przechowywany jest na taśmie jego opis: wymagany wolny bęben, wskaźnik niwelowania nadmiarów, adres końca PAO dla rozdziału, wskaźnik czy były błędy, stan początkowy kluczy, ilość rozdziałów, ewentualne słowniki.

W trakcie translacji i po jej zakończeniu na bębnie pozostaje tylko sekcja startowa, zaś kolejne rozdziały po przetłumaczeniu wysyłane są na taśmę. Po zakończeniu translacji kopia sekcji startowej jest również przesłana na taśmę.

Jeżeli w procesie translacji zostanie wykryty błąd syntaktyczny powodujący natychmiastowe przerwanie translacji, to z taśmy magnetycznej zostaną usunięte wszystkie bloki TM związane z tym programem i taśma zostanie automatycznie zamknięta. Należy tu zauważyć, że pierwszy zapis na taśmie magnetycznej jest robiony w momencie translacji zdania ZAPISZ.

Program po przetłumaczeniu jest gotów do wykonania a wywołany pracuje w ten sposób, że wskazane rozdziały są pobierane do wykonania bezpośrednio z taśmy magnetycznej.

Zdania czołówki podane w punktach 1 i 2 DODATKU I mogą występować zarówno w czołówce zakończonej zdaniem TRANSLACJA jak i w czołówce zakończonej zdaniem ZAPISZ.

Organizacja biblioteki programów wynikowych

Taśma z programami wynikowymi jest szpulą wielozbiorową zapisaną wg standardu IBM. Każdy program stanowi zbiór o nazwie nadanej przez zdanie NAZWA. Każdy rozdział programu zajmuje całkowitą ilość bloków TM po 1024 słowa z wyjątkiem ostatniego, który może być krótszy; Biblioteka może zajmować tylko jedną szpulę (może być wiele bibliotek) Ogólna organizacja szpuli jest podobna do tej, jaka została opisana w paragrafie zatytułowanym: WSPOŁPRACA Z TASMAMI MAGNETYCZNYMI.

Przykłady:

```
ZAPISZ : TMC, 124A, POC
ZAPISZ : TMA, 123, KON
ZAPISZ : TMA, 123, KRAT12, M
```

Przykład czołówki:

```
NAZWA : NOWY1
BIBLIOTEKA : TMA, 127
ZAPISZ : TMB, 128, STARY
```

W powyższej czołówce zakłada się, że będzie utworzony program o nazwie NOWY1 i zostanie zapisany w miejsce programu o nazwie STARY na szpuli 128 założonej na stacji TMB. W trakcie translacji będzie się korzystać z biblioteki programów źródłowych zapisanych na szpuli 127 i założonej na stacji TMA.

POBIERZ

Postać ogólna
POBIERZ : nt,ns,np, $\left\{\begin{matrix}N\\M\end{matrix}\right\}$ [, a1,a2,...,an] gdzie : nt - jest nazwą stacji taśm magnetycznych, ns - jest numerem szpuli, np - jest nazwą programu a1,a2,...,an - jest to lista pobieranych rozdziałów lub/i słowników; każdy element listy ma postać liczby całkowitej bez znaku poprzedzonej literą R lub S.

POBIERZ jest zdaniem kończącym czołówkę oraz przygotowującym do wykonania program znajdujący się w bibliotece programów wynikowych utworzonej przy pomocy zdań ZAPISZ.
Pierwsze dwa parametry określają gdzie znajduje się ta biblioteka (nazwa stacji i numer założonej na niej szpuli).
Trzeci parametr określa nazwę pobieranego programu.
Parametr czwarty, który musi występować, jest wskaźnikiem monitowania, może być literą N (nie przewiduje się minitowania) lub literą M (zdanie dopuszcza współpracę z monitorem, patrz: "Zasady monitowania").
Przygotowanie programu do wykonania polega na pobraniu na bęben jego sekcji startowej wraz z pewnymi informacjami zmagazynowanymi na taśmie.
Lista pobieranych rozdziałów może być pusta i w tym przypadku na bębnie znajduje się tylko sekcja startowa a wszystkie rozdziały i ewentualne ich słowniki pozostają nadal na taśmie magnetycznej.

Aby żądany rozdział był pobrany na bęben należy jego numer, poprzedzony literą R, umieścić na liście. Literą S poprzedzamy numer rozdziału, którego słownik ma być pobrany. Rozdziały i słowniki pobierane są w kolejności występowania ich na taśmie magnetycznej i umieszczane na bębnie przed sekcją startową (patrz rys.1).
W trakcie wykonywania się programu wynikowego pobranego z taśmy magnetycznej, powoływane do pamięci rozdziały (celem ich wykonania), poszukiwane są w pierwszej kolejności na bębnie.
Rozdziały, które nie zostały pobrane na bęben przez POBIERZ powoływane są do wykonania bezpośrednio z taśmy magnetycznej.
Zdania poniżej wymienione mogą wchodzić w skład czołówki zakończonej zdaniem POBIERZ:

BEBEN
POST MORTEM
NAZWA
KLUCZE
WYMAGANY BEBEN
NIWELUJ NADMIARY
IGNORUJ BLEDY

Pozostałe zdania są ignorowane lub też powodują sygnalizację błędów.
Poniżej omówimy efekty związane z występowaniem lub brakiem każdego z powyższych zdań w czołówce zakończonej zdaniem POBIERZ.

BEBEN - brak tego zdania oznacza, że program będzie umieszczony poczynając od samego końca obszaru bębnowego dostępnego w momencie pobierania. Wystąpienie tego zdania określa koniec dostępnego obszaru bębnowego (patrz: opis zdania BEBEN).

POST MORTEM - znaczenie jak podano przy jego opisie.

NAZWA - zdanie POBIERZ nie pobiera nazwy programu do pamięci operacyjnej a pobranemu programowi przypisuje standardową nazwę PROGPR. Omawiane tu zdanie może posłużyć do nadania pobranemu programowi nowej nazwy lub takiej samej jak miał na taśmie magnetycznej.

KLUCZE - zdanie to ustawia stan początkowy rejestru kluczy; brak tego zdania powoduje, że stan początkowy rejestru kluczy będzie taki, jaki był ustawiony w czołówce zakończonej zdaniem ZAPISZ w momencie tworzenia obecnie pobieranego programu. Zdanie bez parametrów oznacza żądanie ustawienia w rejestrze kluczy samych zer.

WYMAGANY BEBEN - jeżeli występuje, to ma takie samo znaczenie jak podano poprzednio. Brak tego zdania oznacza, że parametr do sprawdzenia będzie wzięty taki, jaki był zadany przez ZAPISZ.

NIWELUJ NADMIARY - ustawia nowy sposób traktowania nadmiarów. Brak tego zdania oznacza akceptację takiego wskaźnika, jaki był zadany przez ZAPISZ.

IGNORUJ BLEDY - jeżeli w czołówce nie wystąpi to program błędny zapisany na taśmie nie zostanie dopuszczony do wykonania.

Przykłady

POBIERZ : TMA,12,PR12,N	(zostanie pobrana tylko sekcja startowa)
POBIERZ : TMA,12,PR1,N,- R12,R7	(pobrane zostaną rozdziały o numerach 12 i 7)

POBIERZ : TMA,123,PR2,M,-
R12, S2, R7, S7

(M - oznacza, że można zmienić wartości pierwszych trzech parametrów; pobrane zostaną rozdziały 12 i 7 oraz słowniki rozdziałów 7 i 2)

KONIEC

Postać ogólna
KONIEC .

Zdanie powyższe należy do grupy zdań kończących czołówkę oraz kończy działanie translatora. Służy do zmiany położenia programu znajdującego się na bębnie, za pomocą zdań MINIMALIZUJ i WYMAGANY BEBEN lub też do zmiany opisu programu (za pomocą pozostałych dopuszczalnych zdań). Czołówka zakończona tym zdaniem jest dopuszczalna tylko wtedy, gdy ostatnio działającym translatorem w problemie był translator SAKO, tj. wtedy, gdy sekcja startowa znajduje się na bębnie oraz istnieje w maszynie opis programu.

W czołówce zakończonej zdaniem KONIEC mogą wystąpić następujące zdania:

OPISZ PROGRAM
POST MORTEM
NAZWA
KLUCZE
WYMAGANY BEBEN
MINIMALIZUJ
NIWELUJ NADMIAR
IGNORUJ BLEDY

Wystąpienie każdego z tych zdań powoduje podjęcie odpowiedniej akcji lub ustawienię nowego parametru. Brak odpowiedniego zdania oznacza, że odpowiedni parametr pozostanie bez zmian lub nie zostanie podjęta akcja związana z tym zdaniem

Przykłady

Jeżeli problem w czołówce systemu będzie zdefiniowany następująco:

PROBLEM : SAKO, PROGRAM, SAKO, PROGRAM, ZAKO;

i jeżeli przy pierwszym wywołaniu SAKO czołówka translatora będzie miała postać:

```
KLUCZE : 4,7
CHOWAJ SLOWNIKI
POST MORTEM
TRANSLACJA
   ---------
   ---------
   ---------
```

to po translacji pierwszy raz wywołany program będzie wykonany przy kluczach 4 i 7 w stanie jeden.
Jeżeli teraz, przy następnym wywołaniu SAKO, czołówka translatora będzie miała postać:

```
KLUCZE : 2
KONIEC
```

to po pobraniu czołówki i po jej interpretacji translator skończy pracę, a ponownie wywołany program, będzie miał w rejestrze kluczy same zera z wyjątkiem pozycji 2, która będzie zawierać jedynkę. Po zakończeniu pracy programu, translator ZAKO wypisze post-mortem, gdyż życzenie odnotowane w pierwszej czołówce nie zostanie zniszczone przez drugą czołówkę.

Zasady monitowania

Jak już było powiedziane przy opisach zdań: BIBLIOTEKA, ZAPISZ i POBIERZ, niektóre parametry tych zdań mogą być zmieniane w momencie interpretacji ich przez translator. We wszystkich w/w zdaniach może być zmieniana nazwa stacji i numer szpuli, a ponadto w dwóch pozostałach również nazwa programu. Aby można było zmienić parametry za pośrednictwem monitora muszą być spełnione dwa następujące warunki:

- wskaźnik monitowania w odpowiednim zdaniu musi mieć wartość M,
- na stoliku operatorskim musi być wciśnięty klucz nr 0. Nie wciśnięcie klucza oznacza, że zdanie zostało zaakceptowane bez zmian.

Gdy dla jakiegoś ze zdań oba powyższe warunki zostaną spełnione to na monitorze pojawi się trójliterowy napis określający typ zdania (ZAP,POB,BIB) .
Ponadto wypisane zostaną parametry podlegające zmianom, po czym następuje automatyczne przełączenie monitora w stan czytania.
Aby zmienić wartość parametru, należy go napisać w takiej samej formie, w jakiej został wypisany przez maszynę:

- jednostkę taśm magnetycznych podajemy w postaci zapisu:

  TM nm - nk *

  gdzie: nm - oznacza liczbę całkowitą będącą technicznym numerem modułu urządzenia,

  nk - oznacza liczbę całkowitą będącą numerem kanału w którym urządzenie to się znajduje

  Np.:

  TM16 - 1 * (pierwsza stacja w pierwszym kanale)

- numer szpuli zadajemy według szablonu:

  SZ ns *

gdzie: ns - jest numerem szpuli. Np:

SZ 123 *

- nazwa programu zadawana jest zgodnie z wzorcem:

= np *

gdzie: np jest nazwą programu. Np:

= POG12 *

Po każdym zadanym parametrze maszyna potwierdza jego przyjęcie, poprzez ponowne wypisanie go na monitorze i znowu ustawia monitor w stan czytania.
Wypisanie pojedynczej gwiazdki (*) oznacza, że operator zakończył pracę i uznał zadane parametry za właściwe. Translator przechodzi wtedy do realizacji odpowiedniego zdania.
Jeżeli jednak okaże się, że podane parametry są niepoprawne to pojawi się na monitorze tekst określający rodzaj błędu:

NIE OTWARTO (gdy taśma magnetyczna nie daje się otworzyć),

OTW DO CZYT (w przypadku zdania ZAPISZ, gdy taśma jest zablokowana do pisania) ,

NIEWL NR SZPULI (niewłaściwy numer szpuli) ,

BRAK MKZ (brak metryki końca zapisu na taśmie magnetycznej),

BRAK NAZWY (taśma nie zawiera zbioru o podanej nazwie),

UWAGA: 1. Można zmieniać kilka parametrów naraz oddzielając je od siebie przecinkami. Np.:

0 ; SZ123, = ALFA, TM17-2*

2. Czasami na klawiaturze dalekopisu występuje zamiast gwiazdki (*) znak zapytania (?) .

KOMPONOWANIE PROGRAMÓW SAKO Z WYBRANYCH ROZDZIAŁÓW TAŚMY BIBLIOTECZNEJ Z PROGRAMAMI WYNIKOWYMI

Wprowadza się możliwość komponowania nowych programów SAKO z rozdziałów wybranych z uprzednio przetłumaczonych i znajdujących się na tzw. TAŚMIE BIBLIOTECZNEJ programów SAKO. Taśmą biblioteczną może być każda taśma magnetyczna zawierająca programy SAKO wprowadzone tam przy zastosowaniu zdania ZAPISZ czołówki SAKO.

Powyższe funkcje zostały zrealizowane poprzez rozszerzenie zdań TRANSLACJA i ZAPISZ. W rozszerzonej postaci zdania te zawierają, dopisaną na końcu zdania, listę rozdziałów, które mają być przepisane z taśmy bibliotecznej i włączone na "początku" nowotworzonego programu.

Dla określenia na której stacji TM znajduje się taśma biblioteczna, wprowadzone zostało nowe zdanie czołówki - zdanie TMB. Zdanie to musi występować w czołówce zakończonej zdaniem TRANSLACJA bądź ZAPISZ w rozszerzonej postaci. Po takiej czołówce występuje program SAKO zapisany w języku źródłowym. Program ten zostanie przetłumaczony i jego rozdziały zostaną połączone z rozdziałami przepisanymi z taśmy bibliotecznej w jeden program wynikowy.
W szczególnym przypadku, gdy występujący po czołówce program SAKO będzie się składał z jedynego zdania KONIEC - program wynikowy zostanie utworzony z samych rozdziałów pobranych z taśmy bibliotecznej.

Ograniczenia:

Wszystkie programy wynikowe, z których wybierane są rozdziały dla skomponowania nowego programu muszą mieć identyczne z tworzonym programem następujące parametry:

a. maksymalny adres PAO rozdziału,
b. długość sekcji startowej.

Ponadto translator SAKO kontroluje czy program wynikowy, z którego pobierane są rozdziały nie zawierał błędów formalnych.

W przypadku wykrycia którejkolwiek z poprzednio omówionych niezgodności sygnalizowany jest błąd, jednak translacja nie jest przerywana i odpowiednie kopiowanie rozdziałów dokonuje się.

Zawartość taśmy bibliotecznej -

Zdanie czołówki ZAWARTOŚĆ służy do uzyskania opisu zawartości dowolnej taśmy bibliotecznej z programami SAKO. Parametry tego zdania określają taśmę, której zawartość chcemy otrzymać. Opis zawartości taśmy bibliotecznej składa się z opisów kolejnych programów znajdujących się na tej taśmie. Opis każdego programu zawiera parametry analogiczne do uzyskiwanych za pomocą zdania OPISZ PROGRAM. W szczególności, zawiera on spis rozdziałów i słowników rozdziałów z podaniem ich długości w kolejności występowania na TM.

TASMA MAGNETYCZNA BIBLIOTECZNA

Postać ogólna
TMB : nt,ns [,{M/N}]   gdzie: nt - nazwa stacji taśm magnetycznych   ns - numer szpuli

Zdanie TMB określa stację TM, na której znajduje się Taśma Biblioteczna z programami SAKO w postaci wynikowej.

Parametr nt (nazwa TM) może przyjmować wartości:

TMA,TMB,..... itd.

Parametr ns (nr szpuli) nie ma znaczenia i może przyjmować dowolne wartości typu "nazwa" (tzn. ciągi literowo-cyfrowe).

Ostatni parametr to wskaźnik monitowania - gdy jego wartością jest 'M' oraz wciśnięty jest klucz O na stoliku operatora umożliwia zmianę dwóch pierwszych parametrów za pomocą monitora.

Zdanie TMB musi występować w czołówkach zakończonych zdaniem TRANSLACJA lub ZAPISZ żądającym przepisania pewnych rozdziałów z taśmy bibliotecznej.

Przykłady:

```
TMB:TMA, 13
TMB:TMB, 1777, M
```

ZAWARTOŚĆ

Postać ogólna
ZAWARTOSC: nt,ns [,{M/N}]
gdzie: nt - nazwa stacji taśm magnetycznych ns - numer szpuli

Zdanie ZAWARTOSC powoduje wypisanie poprzez symboliczne wyjście O, opisu zawartości taśmy magnetycznej traktowanej jako taśma biblioteczna z programami wynikowymi SAKO.

Parametr nt (nazwa TM) określa stację TM, na której znajduje się taśma biblioteczna. Może on przyjmować wartości:

TMA,TMB,..... itd.

Parametr ns (nr szpuli) określa numer ewidencyjny szpuli z taśmą biblioteczną. Numer ten zostanie przepisany do tytułu opisu zawartości taśmy bibliotecznej.

Parametr trzeci - wskaźnik monitowania, w przypadku gdy jego wartość jest 'M' umożliwia zmianę dwóch pierwszych parametrów zdania za pomocą monitora. Zmian tych dokonuje się w sposób analogiczny jak w przypadku zdań BIBLIOTEKA, ZAPISZ, POBIERZ.

Uwaga:

Realizacja zdania ZAWARTOŚĆ dokonuje się natychmiast po jego rozpoznaniu w czołówce SAKO. Po wypisaniu zawartości taśma zostaje zamknięta i może być w tej samej czołówce wybrana za pomocą parametrów zdań: TMB, ZAPISZ lub POBIERZ.

Przykłady

ZAWARTOŚĆ: TMC, A845

ZAWARTOŚĆ: TMB, 23,M

TRANSLACJA

Postać ogólna
TRANSLACJA [: lista programów z wybranymi rozdziałami]

Rozszerzone zdanie TRANSLACJA kończy czołówkę SAKO i powoduje że:

- rozdziały wymienione w liście programów z wybranymi rozdziałami zostaną przepisane z taśmy bibliotecznej i umieszczone na bębnie,
- występujące po czołówce rozdziały programu SAKO zostaną przetłumaczone i rozmieszczone na bębnie, łącznie z rozdziałami pobranymi z taśmy bibliotecznej, w postaci przygotowanej do wykonania.

Lista programów z wybranymi rozdziałami może być pusta.

Opis listy programów z wybranymi rozdziałami:

Lista ta składa się z oddzielonych przecinkami elementów o jednej z następujących postaci:

(1) nazwa programu

(2) nazwa programu (lista pobieranych rozdziałów)

Postać (1) oznacza, że ze znajdującego się na taśmie bibliotecznej programu o wskazanej nazwie mają być pobrane wszystkie rozdziały i słowniki. Numery ich nie ulegają zmianie.

W postaci (2) rozdziały (słowniki), które zostaną wybrane ze wskazanego programu wymienione są na liście pobieranych rozdziałów. Określa się również numery jakie mają być im przyporządkowane w nowotworzonym programie.

"Lista pobieranych rozdziałów" składa się z elementów (oddzielonych przecinkami) o jednej z następujących postaci:

Rn/m
Sn/m
Rn
Sn

gdzie: n - nowy numer rozdziału (słownika)

m - numer wybranego rozdziału (słownika) w programie na taśmie bibliotecznej.

R - oznacza rozdział

S - oznacza słownik

Postać Rn i Sn są skróconymi zapisami postaci Rn/n i Sn/n.

Przykłady:

```
TRANS: PSAK1 (R1/15,S1/15,R2/21) ,-
       PSAK2,-
       PSAKO (R10,S10,R11)
```

W powyższym przykładzie zakłada się, że na taśmie bibliotecznej (określonej zdaniem TMB czołówki) znajdują się programy o nazwach: PSAK1, PSAK2, PSAKO. Z programu PSAK1 zostaną przepisane na bęben kolejno: rozdział 15, słownik 15 i rozdział 21. Zostaną one włączone do nowotworzonego programu jako odpowiednio: rozdział 1, słownik 1 i rozdział 2.

Z programu PSAK2 zostaną pobrane wszystkie zawarte w nim rozdziały i słowniki i włączone do tworzonego programu z zachowanymi numerami.

Z programu PSAKO zostaną przepisane: rozdział 10, słownik 10, i rozdział 11 i włączone do programu tworzonego na bębnie z zachowanymi numerami.

ZAPISZ

Postać ogólna

ZAPISZ: nt,ns, {POC / KON / np} , {M / N} [, lista programów z wybranymi rozdziałami]

gdzie: nt - nazwa stacji taśm magnetycznych
ns - numer szpuli
np - nazwa programu już zapisanego na wskazanej szpuli

Zdanie ZAPISZ w powyższej postaci kończy czołówkę i powoduje, że:

(1) rozdziały (słowniki) wymienione w "liście programów z wybranymi rozdziałami" zostaną przepisane z taśmy bibliotecznej (określonej zdaniem TMB) i zapisane na taśmie magnetycznej wyznaczonej przez 2 pierwsze parametry w miejscu wyznaczonym przez trzeci parametr,

(2) występujące po czołówce rozdziały programu SAKO zostaną przetłumaczone i umieszczone na taśmie magnetycznej (określonej przez dwa pierwsze parametry zdania) za rozdziałami pobranymi z taśmy bibliotecznej i połączone wraz z nimi w jeden program SAKO.

Wskaźnik monitowania w przypadku gdy jego wartością jest 'M' umożliwia zmianę 3 pierwszych parametrów zdania ZAPISZ za pomocą monitora.

Trzeci parametr (lokalizacja) może przyjmować wartości:

POC - co oznacza "od początku szpuli"

KON - co oznacza "za ostatnim programem już umieszczonym na danej szpuli"

np (nazwa programu) - co oznacza żądanie wpisania programu na miejsce programu o nazwie np "nazwa programu".

Szczegółowy opis "listy programów z wybranymi rozdziałami" - patrz zdanie TRANSLACJA w rozszerzonej postaci.

Przykłady

```
ZAPISZ: TMF,13, PR075, M,-
        PRB1 (R100/110, S100/110) ,-
        PRB2 (R101, R102),-
        PRB3 ,-
        PRB4 (R115, S115)
```

Przykład ilustruje zdanie ZAPISZ w postaci rozszerzonej. Trzy pierwsze parametry wskazują stację TM oraz miejsce na taśmie magnetycznej gdzie zostanie zapisany tłumaczony program. W następnych wierszach tego zdania określone są programy zawarte na Taśmie Bibliotecznej określonej w zdaniu TMB, z których pobrane będą wskazane w nawiasach rozdziały i słowniki.

Tak więc z programu PRB1 pobrane będą: rozdział 110 oraz słownik 110 i włączone do tworzonego na taśmie TMF programu jako rozdział i słownik o numerze 100; Z programu PRB2 pobrane będą rozdziały 101 i 102 i włączone pod tymi samymi numerami; Z programu PRB3 przeniesione będą wszystkie rozdziały i słowniki bez zmiany numerów oraz z programu PRB4 przeniesione będą (bez zmiany numerów) rozdział 115 i słownik 115.

Uogólnienie zdania IDZ DO ROZDZIALU

Poczynając od edycji BCS(A) systemu SO141 parametrem zdania IDZ DO ROZDZIALU może być wyrażenie arytmetyczne o wartości całkowitej. Powyższa zmiana wraz z opisanymi zasadami komponowania programów SAKO stwarza możliwość konstruowania bibliotek programów wielorozdziałowych w postaci wynikowej o symbolicznych numerach rozdziałów.

SYSTEM MAGAZYNOWANIA I AKTUALIZACJI DOKUMENTÓW

W skład systemu operacyjnego SO-141 wchodzi program ogólnego zastosowania, pracujący na prawach translatora, wywoływany w problemie jako POPR i zwie się Systemem Magazynowania i Aktualizacji Dokumentów (SMAD) [1].

SMAD służy do zapisywania tekstów na taśmie magnetycznej oraz do ich przetwarzania. Zapisany na taśmie magnetycznej tekst może być w szczególnym przypadku programem napisanym w języku SAKO.

Przy realizacji zdania BIBLIOTEKA założono, że taśma magnetyczna będąca biblioteką programów źródłowych jest zapisana według standardu SMAD tj. że ten ogólny aparat może być wykorzystany do utworzenia biblioteki.

Ogólnie, przetwarzanie tekstów przez SMAD polega na interpretacji kolejnych instrukcji pobieranych z drugiego symbolicznego wejścia translatorów. Instrukcje języka SMAD posiadają następującą budowę:

Przed zwrotem instrukcji, określającym jej nazwę musi wystąpić koniecznie sekwencja znaków: LF:: tj. dwa dwukropki muszą wystąpić na samym początku wiersza (nawet spacje nie są dozwolone).

W zwrocie indentyfikujące są tylko pierwsze dwie litery i stąd parametry instrukcji powinny być oddzielone od zwrotu dwukropkiem.

Instrukcje na wejściu są "wymieszane" z tekstem programu, stąd w tekście nie może wystąpić sekwencja LF::.

W zależności od rodzaju instrukcji używanych w ciągu wejściowym, translator potrzebuje jednej lub dwóch stacji taśm magnetycznych.

Nazwę pierwszej stacji taśm zadaje się jako parametr zerowego wejścia symbolicznego translatorów a nazwa drugiej jest parametrem pierwszego wejścia translatorów.

Dla skrócenia opisu założymy, że nazwą stacji pierwszej jest TMA a drugiej TMB.

Przy operacjach związanych z zakładaniem biblioteki, wypisywaniem jej zawartości lub odszukiwaniem tekstu na taśmie, translator współpracuje z TMA, natomiast w procesie przetwarzania, tekst pobierany jest z TMB i po uaktualnieniu zapisywany na TMA.
W celu wypisywania informacji tekstowych dokumentujących przebieg przetwarzania, translator korzysta z zerowego wyjścia translatorów.
W dalszej części paragrafu omówimy kilka przykładów wykorzystania aparatu SMAD, natomiast bardziej pełne informacje o jego możliwościach znajdzie czytelnik w pracy [1].

Zakładanie biblioteki

Dla wykonania tego zadania czołówka systemu może mieć postać:

WEJSCIA TRANSLATOROW : 0 = PCTM (TMA),
2 = PCTP (CTP1, DM2C);

WYJSCIA TRANSLATOROW : 0 = PDW (DW, DDW1);
PROBLEM : POPR.

i podaje, że będzie działał translator POPR, jego wyjściem zerowym jest drukarka wierszowa; ciąg instrukcji oraz teksty będą wprowadzane z pierwszego czytnika taśmy papierowej, wyperforowane w kodzie M2 i wszystkie instrukcje SMAD będą odnosiły się tylko do stacji TMA (nie będzie przeprowadzany proces aktualizacji).

Instrukcję pozwalającą na wpisanie tekstu na taśmę jest:

::WCZYTAJ DOKUMENT (skrót:: WC)

Po niej występuje tekst podlegający przepisaniu. Tekst na wejściu musi być podzielony na jednostki zwane stronami. Strona ma standardową budowę, składa się z nagłówka i treści.
Nagłówek składa się z dwóch wierszy o następującej postaci:

::---------

: [W] - ns -

gdzie oba napisy muszą zaczynać się od nowej linii

i od początku wiersza, ns - jest nazwą strony i może być napisem składającym się z cyfr, liter i kropek i nie dłuższym niż 8 znaków (cyfra zero i litera O są utożsamiane - patrz opis zdania BIBLIOTEKA).

Poprzednie napisy z punktu widzenia języka SAKO są komentarzami (patrz: KOMENTARZ) i są dopuszczalne w programie. Taki napis i każdy inny zaczynający się od dwóch dwukropków jest niedopuszczalny we wnętrzu strony. Strona nie może zawierać więcej niż 1000 wierszy.

Tekst należy tak podzielić, aby każda logiczna część programu zaczynała się od nowej strony. Albowiem, po zapisaniu tekstu na taśmę, może być on pobierany poczynając od początku wskazanej strony (patrz opis zdania BIBLIOTEKA).

Instrukcja ::WC działa w ten sposób, że poczynając od początku taśmy magnetycznej, zapisuje na niej kolejne strony aż do napotkania instrukcji:

::KONIEC CZYTANIA (skrót ::KO)

Instrukcja ::KO powoduje: zakończenie czytania, odwinięcie taśmy magnetycznej od początku i zakończenie pracy translatora POPR.

Standardem przy działaniu instrukcji ::WC jest drukowanie czytanego tekstu na zerowe wyjście. Przy pomocy instrukcji występujących przed ::WC :

::ARKUSZ: n (skrót: ::AR: n)
::NIE DRUKUJ (skrót: ::NI)

można: określić ilość wierszy na stronie papieru drukarkowego (np. ::AR: 66 lub ::AR: 72) , wyłączyć wypisywanie dokumentacji (::NI) Można też zażądać drukowania dokumentacji w więcej niż jednym egzemplarzu:

::DRUKUJ : n (np: ::DR :3)

Teksty programów w języku SAKO zapisane na taśmie w powyższy sposób, nie tylko mogą stanowić bibliotekę ale również, jeżeli są zapisane wraz z czołówkami translatora (czołówkę można dzielić na strony) mogą podlegać translacji.

W takim przypadku do odszukania odpowiedniego programu na taśmie służą następujące instrukcje:

::ODWIN TASME	(skrót: ::OD)
::PRZEWIN DO STRONY: i	(skrót: ::PR:i, gdzie i jest nazwą strony)
::KONIEC POSZUKIWANIA	(skrót: ::KO; kończy pracę POPR)

Przykład

Załóżmy, że na stacji TMC jest założona szpula z programem źródłowym zapisanym przy pomocy SMAD; przypuśćmy ponadto, że pierwsza strona tego programu ma nazwę PR 2.1 to w poniższy sposób opisany problem spowoduje odszukanie tego programu, jego translację i wykonanie.

```
WEJSCIA TRANSLATOROW : 0 = PCTM (TMC),
                       2 = PCTP (CTP1,DM2C);
WYJSCIA TRANSLATOROW : 0 = PDW (DW,DDW1);
PROBLEM : POPR, SAKO, PROGRAM
```

Tu musi być czysta taśma

```
::OD
::PR : PR 2.1
::KO
```

Program znajdujący się na taśmie i podlegający translacji musi posiadać czołówkę translatora.
Np. może mieć postać jak podano poniżej

```
::-------
: [W]  - PR 2.1 -
KLUCZE : 1,4
TRANSLACJA
::-------
: [W]  - PR 2.2 -

ROZDZIAL: 1
CALKOWITE : A
---------
---------
---------
: [W]  - PR 2.7 -
```

ROZDZIAŁ : 2

Przetwarzanie tekstów

Biblioteka podprogramów i programów może być wzbogacona o nowe teksty źródłowe lub też mogą być zmieniane teksty już występujące na taśmie.
W takich przypadkach często używa się czołówkę systemową o następującej postaci:

```
WEJSCIA TRANSLATOROW : 0 = PCTM (TMA),
                       1 = PCTM (TMB),
                       2 = PCTP (CTP1, DM2C);
WYJSCIA TRANSLATOROW:  0 = PDW (DW, DDW1);
PROBLEM : POPR.
```

W tym przypadku powinna wystąpić instrukcja:

::POPRAWIANIE (skrót ::PO)

która odwija obie taśmy i informuje translator, że będzie przeprowadzony proces aktualizacji. Oprócz tego można wyłączyć dokumentację zmian (::NI) lub życzyć sobie drukowania dokumentacji w więcej niż jednym egzemplarzu (np: ::DR: 3; standard jest jeden), lub też zadać wymiar arkusza (np. ::AR: 66).

Przebieg procesu aktualizacji sterowany jest instrukcjami do tego celu przeznaczonymi. Współrzędnymi określającymi miejsca zmian w starym tekście (na TMB) są nazwy stron. Ponieważ instrukcje SMAD są interpretowane w kolejności ich występowania na wejściu (nr 2) oraz tekst na TMB jest przeglądany tylko jeden raz, stąd instrukcje zmieniające stary program, powinny być ułożone w tej kolejności w jakiej występują na taśmie zmieniane przez nie fragmenty.

Operacjami odnoszącymi się do stron są:

::USUN STRONY: i - j (np. ::US: A.7 - A.12,
i,j - nazwy stron)

::WSTAW PRZED STRONE: i (np: ::WS: A.23)

::STRONA: i (np: ::ST: A.50)

W procesie aktualizacji kolejne strony tekstu przepisywane są z TMB na TMA aż do natrafienia na stronę o nazwie identycznej z nazwą będącą parametrem pierwszej instrukcji. Wtedy wykonywane są przewidziane operacje i po przeczytaniu kolejnej instrukcji przechodzi się znów do przepisywania i porównywania.
Proces trwa aż do natrafienia na instrukcję ::KO, która powoduje przepisanie pozostałych stron, odwinięcie taśm i zakończenie pracy translatora POPR.
W szczególnym przypadku dwie instrukcje występujące po sobie:

::PO

::KO

powodują utworzenie na taśmie TMA kopii taśmy TMB.
Instrukcja ::US: i - j powoduje, że ciąg stron o nazwach od i aż do j włącznie nie zostanie przepisany na taśmę wyjściową.
Instrukcja ::WS: i powoduje, że po natrafieniu na stronę o nazwie i zostanie przerwany proces przepisywania. Następnie, tekst występujący po tej instrukcji, podzielony na strony, zostanie przepisany na TMA, po czym następna instrukcja jest czytana i proces porównywania i przepisywania zaczyna się od strony i.
Instrukcja ::ST: i informuje, że będą zmieniane tylko niektóre wiersze tej strony.
Operacje odnoszące się do wierszy są następujące:

::SKRESL WIERSZE: m - n (np ::SK: 12-14)

::WYMIEN WIERSZE: m - n (np ::WY: 14-16)

::DOPISZ PO WIERSZU: n (np ::DO: 20)

gdzie: m,n - numery wierszy

Proces zmian wierszy na stronie jest podobny do procesu zmian stron. Parametry instrukcji wskazują numery wierszy na starej stronie. W kolejnych instrukcjach numery wierszy muszą być ustawione w porządku rosnącym. Do nowo-formowanej strony przepisywane są kolejno niezmieniane wiersze, następnie wykonywane są operacje przewidziane kolejną instrukcją i cykl jest powtarzany aż do natrafienia na instrukcję różną od którejś z trzech wyżej wymienionych. Wtedy przepisywana jest reszta strony i nowoutworzona strona zapisywana jest na TMA. Dalej cykl jest powtarzany na poziomie stron.

Instrukcje dotyczące wierszy:

::SK: m - n - podaje, które wiersze (włącznie) nie powinny być przepisywane do nowej strony

::WY: m - n - działa jak ::SK z tym, że po skreśleniu wymienionych wierszy tekst występujący po niej (niestronicowany) jest wpisany do nowej strony, końcem tekstu wpisywanego jest instrukcja SMAD

::DO: m - po przepisaniu m-tego wiersza dopisz do nowej strony wiersze występujące poniżej.

Dla ułatwienia, w dokumentacji produkowanej przez SMAD, wiersze są numerowane. Ponadto, mając ponumerowane wiersze, zbędnym jest pamiętanie zasady numeracji.

Czasami zachodzi potrzeba usuwania stron do końca taśmy lub też wstawienia nowych stron na końcu.

Wtedy w instrukcjach:

::US: i - j

lub

::WS: j

należy zamiast parametru j pisać gwiazdkę(*).

W przypadku uruchamiania dużych programów, zalecanym jest trzymanie ich w postaci źródłowej na taśmie, poprawianie za pomocą środków dostarczanych przez SMAD i pobieranie do translacji z taśmy magnetycznej.

Pozwala to znacznie przyśpieszyć uruchamianie.

POST MORTEM

(objaśnienie tekstów wypisywanych przez ZAKO)

Translator ZAKO jest narzędziem służącym do uruchamiania programów pisanych w języku SAKO. Korzysta z zerowego symbolicznego wyjścia translatorów, na które wypisuje informacje o poprzednio wykonywanym programie. Wypisywane informacje dzielimy na dwie grupy: przyczyn zakończenia programu i post mortem. ZAKO wywołuje się jako zadanie w problemie (patrz opis zdania PROBLEM) i wywołanie powinno występować bezpośrednio po wykonaniu programu wynikowego, gdyż tylko w tym momencie można uzyskać, od systemu SO-141, dane o przyczynach zakończenia programu oraz wartości jakie miały zmienne w momencie przerwania zakończenia pracy programu wynikowego.

Przyczyny zakończenia

Przyczyny zakończenia poprzednio działającego programu są wypisywane zawsze, niezależnie jaki program działał. Jeżeli bezpośrednio przed ZAKO działał translator to podana jest jego nazwa.
Np. PRZYCZYNA ZAKONCZENIA TRANSLATORA SAKO.
Podany w przykładzie tekst może być wypisany mimo, że wywołanie ZAKO poprzedzone było wywołaniem: PROGRAM. Dzieje się to wtedy, gdy program nie zostanie dopuszczony do działania np. były błędy w programie.
Jeżeli przed ZAKO działał program wynikowy to podaje się przyczynę zakończenia programu z nazwą translatora, który go przygotował do wykonywania.
Np. PRZYCZYNA ZAKONCZENIA PROGRAMU SAKO.
Po wyżej wymienionych zwrotach wypisywany jest numer przyczyny zakończenia (patrz przykład nr 1) np. 514.
Następne objaśnienia będą dotyczyły przypadku wywołania translatora ZAKO po działaniu programu przygotowanego przez translator SAKO.

Znaczenia numerów przyczyn zakończenia, większych od 500, podane są w DODATKU L. Znaczenia numerów 0 i 62 są podane przy okazji opisu błędów etapu wykonania, inne numery mogą pojawić się np. we wstawkach SAS (patrz przykład nr 2) Numer 63 oznacza, że wykonało się zdanie STOP i w czołówce translatora nie wystąpiło zdanie POST MORTEM. Natomiast, jeżeli było zdanie POST MORTEM i wykonało się w programie zdanie STOP, to zamiast numeru przyczyny zakończenia wypisywane jest słowo STOP z etykietą będącą jego parametrem oraz podany jest numer **ostatnio** działającego rozdziału (patrz: przykład nr 3).

W części zwanej przyczynami zakończenia, oprócz informacji opisanej powyżej, podaje się:

- adres pamięci, gdzie zostało przerwane wykonywanie programu np. LR = 1563,
- ilość jednostek niewykorzystanego czasu (jeżeli parametr ma wartość równą 8 000 000 to oznacza, że program wykorzystał mniej niż jedną jednostkę; jednostka wynosi nieco ponad jedną sekundę),
- informację, że stan wskaźnika nadmiaru stałoprzecinkowego był TAK.

Przykłady:

1/

```
=================================================================
PRZYCZYNA ZAKONCZENIA PROGRAMU SAKO  514 W ROZDZIALE ..........
0 LR = 1563.
POZOSTAŁO CZASU: 7999993
=================================================================
```

2/ **********BK

```
=================================================================
PRZYCZYNA ZAKONCZENIA PROGRAMU SAKO 62 W ROZDZIALE ............
0 LR = 2798.
POZOSTAŁO CZASU: 8000000.
=================================================================
```

Ciąg gwiazdek i BK jest tekstem sygnalizującym błąd etapu wykonania (patrz DODATEK H) i nie jest wypisywany przez ZAKO.

3/

```
==================================================================
PRZYCZYNA ZAKONCZENIA PROGRAMU SAKO STOP NASTEPNY.....
W ROZDZIALE 0 IR = 1593. POZOSTAŁO CZASU: 7999993.
==================================================================
.....
```

SŁOWNIK ETYKIET

```
7  1262  4  1289  5  1299  6  1309  1  1314  3  1503
PODPROGRAM  * STR  1594
PODPROGRAM  * PDM  1710
2  1757     1N  1763     1WR  1810
```

POST MORTEM

```
W  766    0        0        0        0
          0        7        3        0
          0        2        1        1
M  7036   12       1        2        1
          5
IL 7035   3 KH  7034    3     NR 7033        185
WY 7031   1  Z  7030   64      S 7029      1590093
J  7027   1 PTW 7025+8.926    PCH 7023    +6.409E+1
```

PODPROGRAM * STR

```
I     1596     23704     220312
WYS   1598     23704     220312
KOH   1600     23704     220312
```

PODPROGRAM	* PDM			
A	1712	7036	4201340	+2.859E+45
W	1714	7666	4201970	+2.860E+45
N	1716	70	70	
IL	1718	3	3	
KH	1720	0	0	
NR	1722	185	185	
PL	1724	1	1	
WY	1726	1	1	
Z	1728	7030	4201334	+2.859E+45
I	6814	13		
S2	6812	+3.400		

Post mortem rozdziału

Aby uzyskać więcej informacji o programie, niż przyczyny zakończenia, muszą być spełnione następujące warunki:

- do programu wynikowego powinny być włączone słowniki. Włączenie słowników dokonuje się podczas translacji na żądanie wyrażone w czołówce translatora za pośrednictwem zdania CHOWAJ SŁOWNIKI. Fakt, że są przechowane słowniki nie wystarczy, aby uzyskać post mortem,
- w czołówce translatora przygotowującej program do wykonania musi wystąpić zdanie POST MORTEM. Przy braku słowników, zdanie POST MORTEM spowoduje że w trakcie pracy ZAKO otrzymamy tylko numer ostatnio działającego rozdziału.

Jażeli powyższe warunki są spełnione i jeżeli sekcja startowa oraz słowniki, znajdujące się na bębnie, nie zostały zniszczone w trakcie pracy programu wynikowego, to po wypisaniu przyczyn zakończenia, translator ZAKO wypisze tzw. post mortem ostatnio działającego rozdziału. Translator ZAKO nie wymaga, aby słowniki były koniecznie na bębnie. W przypadku przejścia do wykonania programu po zapisaniu go na taśmie magnetycznej (patrz zdanie czołówki ZAPISZ)

lub też w przypadku pozostawienia ich na taśmie przy POBIERZ odpowiedni słownik będzie pobrany z taśmy magnetycznej.

Post mortem składa się z dwóch części zatytułowanych: SŁOWNIK ETYKIET i POST MORTEM (patrz przykład nr 3).

SŁOWNIK ETYKIET zawiera spisy etykiet użytych w danym rozdziale. Najpierw spis etykiet programu głównego a następnie spisy etykiet podprogramów. Pozycja spisu etykiet zawiera etykietę oraz jej adres pamięciowy.
Spis etykiet podprogramu poprzedzony jest słowem PODPROGRAM, po którym podane są pierwsze trzy znaki nazwy podprogramu poprzedzone gwiazdką. Ponadto podany jest adres początku podprogramu.
Wartość LR (podana w przyczynach zakończenia) oraz słownik etykiet mogą posłużyć do zlokalizowania miejsca, w którym został przerwany program.

Część tekstu zatytułowana POST MORTEM zawiera listę zmiennych użytych przez programistę. Lista podzielona jest na części. Najpierw podana jest lista zmiennych programu głównego a następnie listy zmiennych występujących w podprogramach. Każda pozycja na liście zmiennych zawiera:

- nazwę zmiennej lub bloku (tablicy),
- adres przypisany danej zmiennej; w przypadku bloku (tablicy) adres pierwszego elementu,
- wartość zmiennej; w przypadku bloku (tablicy) wypisywane są wartości kolejnych elementów (patrz wzór na obliczanie miejsca zmiennej w bloku) ale nie więcej niż piętnastu.

W zależności od typu pierwszych wartości, zmienne wypisywane są w postaci całkowitej lub ułamkowej.
Na liście w podprogramie zmienne formalne występują jako pierwsze (w podprogramach: STR wszystkie zmienne są formalne a w PDM od M do Z) i sposób wypisywania ich wartości jest następujący.

Po nazwie zmiennej formalnej podany jest adres pamięci przypisany jej. Następnie wydrukowana jest aktualnie podstawiona wartość danej zmiennej, z tym że jest drukowana w kilku postaciach:

1. jako adres
2. jako liczba całkowita
3. jako liczba ułamkowa

Gdy argument jest nieznormalizowany to postać ułamkowa jest opuszczana; drukuje się liczbę w kilku postaciach, gdyż translator ZAKO nie zna typu zmiennej formalnej. Programista analizując wyniki post mortem sam powinien zdecydować, którą formę wybrać.

- adres, gdy było podstawienie przez nazwę,
- całkowitą, gdy argument był całkowity
- ułamkową, gdy argument był ułamkowy.

Np. gdy wiemy, że A w podprogramie PDM (przykład nr 3) jest blokiem formalnym, to do analizy wyników weźmiemy postać pierwszą (adres 7036) i zauważymy, że ostatnio był podstawiony blok M (M w programie głównym ma adres równy 7036).
W przypadku funkcji formalnej po jej nazwie, wypisywany jest adres miejsc roboczych z nią związanyych a następnie drukowany jest adres podstawionej funkcji aktualnej i adres wyników funkcji aktualnej.

Jeżeli w trakcie pracy programu pewnym zmiennym formalnym nie nadawano wartości to najczęściej przypisywana jest im wartość początkowa równa 220312 (patrz wartości zmiennych formalnych podprogramu STR).
Translator ZAKO nie wypisuje słowników ani też wartości zmiennych podprogramów biblioteki MSAS wchodzących w skład programu wynikowego a dołączonych za pośrednictwem makrooperacji używanych we wstawkach języka SAS.

CZĘŚĆ III

D O D A T K I

DODATEK A

Tablica zależności między kodem wewnętrznym KW6 i kodami zewnętrznymi

KW6			M-2				FERRANTI						
N	W	I	DM2C WE	W	S	DM2P WY	DFE6 WE	W	S	D6FE WY	DW1	DW2	DW2E
NUL(NU)	0	N			L/C	NUL			L/C	NUL			
SP (SP)	1	N	SP	4	L/K	SP	SP	14	C	SP	SP	SP	SP
. (FP)	2	I	.	7	C	.	.	7	L/C	.	.	.	.
((LP)	3	I	(	30	C	(	(	20	C	(	(	(	(
+ (PL)	4	I	+	17	C	+	+	11	C	+	+	+	+
* (AS)	5	I	*	19	C	*	*	24	C	*	*	*	*
) (RP)	6	I	)	9	C	)	)	12	C	)	)	)	)
: (SE)	7	ND	:	26	C	:	→	5	C	→	:	:	:
- (MI)	8	I	-	24	C	-	-	26	C	-	-	-	-
/ (SL)	9	I	/	23	C	/	/	29	C	/	/	/	/
, (CM)	10	I	,	6	C	,	,	30	C	,	,	,	,
: (CL)	11	I	:	14	C	:	:	9	C	:	:	:	:
, (AP)	12	1	,	20	C	,	η	23	C	η	Ł	,	,
= (EQ)	13	I	=	15	C	=	=	10	C	=	=	=	=
[(LB)	14	I	[	22	C	[	≡	18	C	≡	(	[	[
] (RB)	15	2	]	11	C	]	↓	6	C	↓	)	]	]
< (LT)	16	N			C	CS			C	SP	SP	<	<
> (GT)	17	2			C	CS	>	17	C	>	SP	>	>
? (QS)	18	1			C	CS	?	23	L	?	SP	>	?
ESC(ES)	19	N			C	CS			L/C	*			
SO(SO)	20	N							C	[			
A (XA)	21	I	A	24	L	A	A	16	L	A	A	A	A
B (XB)	22	I	B	19	L	B	B	8	L	B	B	B	B
C (XC)	23	I	C	14	L	C	C	24	L	C	C	C	C
D (XD)	24	I	D	18	L	D	D	4	L	D	D	D	D
E (XE)	25	I	E	16	L	E	E	20	L	E	E	E	E

KW6			M-2				FERRANTI						
N	W	I	DM2C WE	W	S	DM2P WY	DFE6 WE	W	S	D6FE WY	DW1	DW2	DW2
F (XF)	26	I	F	22	L	F	F	12	L	F	F	F	F
G (XG)	27	I	G	11	L	G	G	28	L	G	G	G	G
H (XH)	28	I	H	5	L	H	H	2	L	H	H	H	H
I (XI)	29	I	I	12	L	I	I	18	L	I	I	I	I
J (XJ)	30	I	J	26	L	J	J	10	L	J	J	J	J
K (XK)	31	I	K	30	L	K	K	26	L	K	K	K	K
L (XL)	32	I	L	9	L	L	L	6	L	L	L	L	L
M (XM)	33	I	M	7	L	M	M	22	L	M	M	M	M
N (XN)	34	I	N	6	L	N	N	14	L	N	N	N	N
O (XO)	35	I	O	3	L	O	O	30	L	O	O	O	O
P (XP)	36	I	P	13	L	P	P	1	L	P	P	P	P
Q (XQ)	37	I	Q	29	L	Q	Q	17	L	Q	Q	Q	Q
R (XR)	38	I	R	10	L	R	R	9	L	R	R	R	R
S (XS)	39	I	S	20	L	S	S	25	L	S	S	S	S
T (XT)	40	I	T	1	L	T	T	5	L	T	T	T	T
U (XU)	41	I	U	28	L	U	U	21	L	U	U	U	U
V (XV)	42	I	V	15	L	V	V	13	L	V	V	V	V
W (XW)	43	I	W	25	L	W	W	29	L	W	W	W	W
X (XX)	44	I	X	23	L	X	X	3	L	X	X	X	X
Y (XY)	45	I	Y	21	L	Y	Y	13	L	Y	Y	Y	Y
Z (XZ)	46	I	Z	17	L	Z	Z	11	L	Z	Z	Z	Z
SI (SI)	47	N							C	]			
0 (Z0)	48	I	0	13	C	0	0	1	C	0	0	0	0
1 (Z1)	49	I	1	29	C	1	1	16	C	1	1	1	1
2 (Z2)	50	I	2	25	C	2	2	8	C	2	2	2	2
3 (Z3)	51	I	3	16	C	3	3	25	C	3	3	3	3
4 (Z4)	52	I	4	10	C	4	4	4	C	4	4	4	4
5 (Z5)	53	I	5	1	C	5	5	21	C	5	5	5	5
6 (Z6)	54	I	6	21	C	6	6	13	C	6	6	6	6

KW6			M-2				FERRANTI						
N	W	I	DM2C WE	W	S	DM2P WE	DFE6 WE	W	S	D6FE WY	DW1	DW2	DW2E
7 (Ś7)	55	I	7	28	C	7	7	28	C	7	7	7	7
8 (Ź8)	56	I	8	12	C	8	8	2	C	8	8	8	8
9 (Ż9)	57	I	9	3	C	9	9	19	C	9	9	9	9
LF (LF)	58	I	LF	8	L/C	CR,LF	LF	22	C	CR,LF	CR,LF	CR,LF	CR,LF
◊ (N6)	59	I	◊	5	C	◊	×	3	C	×	=	<	<
FF (FF)	60	N			C	CR,LF			C	CR,LF	FF	FF	FF
CS (CS)	61	N	CS	18	C	CS	Ź	15	L	Ź	SP	sl	#
CR (CR)	62	N	CR	2	L/C	CR	CR	15	C	CR	CR	CR	CR
DEL (DE)	63	N											

Tabela pomocnicza

KW6	DW1	DW2	DW2E
A ◊	↑	↑	↑
C ◊		ć	ć
E ◊	ę	ę	ę
L ◊	ł	ł	ł
N ◊		ń	ń
O ◊		ó	ó
S ◊		ś	ś
Z ◊	ż	ź	ź
!	↑		
?		SP	

Opis tablicy

Powyższa tablica zawiera trzy zasadnicze części:

1. Opis kodu wewnętrznego KW6:
Kolumna N zawiera znak kodu wewnętrznego wraz z jego dwuliterową nazwą.
Kolumna W podaje liczbę dziesiętną całkowitą będącą reprezentacją znaku w pamięci maszyny (kod znaku).
Sposób traktowania przez translator znaków kodu KW6 podany jest w kolumnie o nazwie I. Znaki w tej kolumnie oznaczają:

I - znak jest istotny
N - znak jest nieistotny i translator go pomija
ND- znak niedozwolony; w zdaniach nie może występować
1 - para znaków istotnych równoważnych
2 - znaczenie jak dla 1

2. Grupy kolumn zatytułowane M2 i FERRANTI podają sposób przekodowania znaków z w/w kodów taśmowych na kod KW6 (kolumna o nazwie WE) i w odwrotną stronę tj. z kodu KW6 na kody zewnętrzne (kolumny zatytułowane WY).
Nabory znaków zewnętrznych kodów M-2 i FERRANTI mogą się znacznie różnić dla różnych urządzeń perforujących i tabulujących.
Dlatego też, aby dokładniej określić sposób przyporządkowania znaków kodu zewnętrznego znakom kodu KW6, zostały wprowadzone kolumny o nazwach W i S określające wartość kodu zewn. oraz stronę "litery" lub cyfry" .
Podane w tablicy przyporządkowania kodów zewnętrznych kodowi wewnętrznemu realizowane są przy pomocy dekoderów DM2C, DM2P, DFE6, D6FE.
Sposób korzystania z dekoderów podany jest w opisie Czołówki Systemu.

3. Ostatnie trzy kolumny tabeli podają sposób wypisywania znaków kodu KW6 za pośrednictwem dekodera DDW1 w zależności od typu zainstalowanej drukarki.

Natomiast używając dekodera DDW2 pojedyncze znaki kodu KW6 będą wypisywane na jedną z omawianych drukarek w ten sam sposób co i przy użyciu dekodera DDW1 z wyjątkiem tego, że średnik wypisze się jako ♠ a znak zapytania jako SP. Ponadto sekwencje znaków kodu KW6 podane w kolumnie o nazwie KW6 zostaną wypisane jako pojedyncze znaki zgodnie z pomocniczą tabelką podaną powyżej.

DODATEK B

Opis funkcji języka

Sinus

Nazwa: SIN

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie zamieniany na ułamkowy

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: SIN(X)

Cosinus

Nazwa: COS

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie zamieniany na ułamkowy

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: COS(X)

Dodatkowe uwagi: funkcja COS jest realizowana przy pomocy funkcji SIN stąd przy liczeniu wartości funkcji COS jest niszczony argument poprzednio podstawiony do funkcji SIN

Tangens

Nazwa: TG

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie zamieniany na ułamkowy

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: TG(X)

Dodatkowe uwagi: wartość tej funkcji jest ilorazem funkcji SIN(X) i COS(X). Stąd, przy liczeniu wartości tangensa argumenty uprzednio podstawione do funkcji SIN i COS są niszczone. Przy liczeniu wartości tej funkcji może powstać nadmiar zmiennoprzecinkowy.

Arcus sinus

Nazwa: ASN

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie zamieniany na ułamkowy

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: ASN(X)

Dodatkowe uwagi: funkcja ASN(X) ma sens dla argumentu X spełniającego warunek

$$|X| \leqslant 1$$

Arcus cosinus

Nazwa: ACS

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest transponowany do postaci ułamkowej

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: ACS(X)

Dodatkowe uwagi: wartość funkcji obliczana jest według wzoru:

$$ACS(X) = \pi/2 - ASN(X)$$

stąd argument musi spełniać warunek:

$$|X| \leqslant 1$$

Należy też pamiętać, że przy liczeniu wartości funkcji ACS zmieniany jest argument uprzednio podstawiony do funkcji ASN

Arcus tangens

Nazwa: ATG

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie zamieniany na postać ułamkową

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: ATG(X)

Pierwiastek kwadratowy

Nazwa: PWK

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie sprowadzany do postaci ułamkowej

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: PWK(X)

Dodatkowe uwagi: funkcja ta jest określona dla dowolnej wartości argumentu i wartość jej jest liczona według wzoru

$$w = \mathrm{sgn}\, X \cdot \sqrt{|X|}$$

Pierwiastek sześcienny

Nazwa: PWS

Typ funkcji: ułamkowa

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie sprowadzany do postaci ułamkowej

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: PWS(X)

Logarytm naturalny
Nazwa: LN
Typ funkcji: ułamkowa
Sposób podstawienia: przez wartość lub przez nazwę
Typ argumentu: całkowity lub ułamkowy; argument całkowity jest automatycznie sprowadzany do postaci ułamkowej
Rodzaj argumentu: wyrażenie arytmetyczne
Przykład: LN(X)
Dodatkowe uwagi: Argument funkcji musi spełniać warunek:

$$X > 0$$

Exponent
Nazwa: EXP
Typ funkcji: ułamkowa
Sposób podstawienia: przez wartość lub przez nazwę
Typ argumentu: ułamkowy lub całkowity; argument całkowity jest automatycznie sprowadzany do postaci ułamkowej
Rodzaj argumentu: wyrażenie arytmetyczne
Przykład: EXP(X)
Dodatkowe uwagi: wartość argumentu musi być taka, aby wartość funkcji nie przekroczyła zakresu przewidzianego dla liczb ułamkowych

Arcus
Nazwa: ARC
Typ funkcji: ułamkowa
Sposób podstawienia: przez wartość lub przez nazwę
Typ argumentów: każdy z argumentów może być typu ułamkowego lub typu całkowitego; argumenty całkowite są automatycznie sprowadzane do postaci ułamkowej
Rodzaje argumentów: każdy argument może być wyrażeniem arytmetycznym
Przykład: ARC(X,Y)

Znaczenie funkcji: wartością funkcji jest liczba ułamkowa równa kątowi (wyrażonemu w mierze łukowej) zawartemu między osią x i półprostą przechodzącą przez środek układu współrzędnych prostokątnych oraz punkt A o współrzędnych X,Y

Wartość funkcji jest zawarta w następującym przedziale:

$$0 \leqslant ARC(X,Y) < 2\pi$$

Dodatkowe uwagi: funkcja ARC(0,0) jest nieokreślona.

Maksimum

Nazwa: MAX

Typ funkcji: funkcja jest całkowita, gdy wszystkie argumenty są całkowite; w pozostałych przypadkach jest typu ułamkowego

Sposób podstawienia: funkcja może być podstawiona tylko przez wartość a argumenty do tej funkcji nie mogą być podstawiane przy pomocy zdania PODSTAW

Typy i ilość argumentów: argumenty mogą być różnych typów a ilość argumentów ograniczona jest tylko długością wiersza

Rodzaj argumentów: argumenty są wyrażeniami arytmetycznymi

Przykład: MAX(A,B,X+Y)

Znaczenie funkcji: największa z liczb, będących wartościami argumentów, jest wartością funkcji

Minimum

Nazwa: MIN

Typ funkcji: funkcja jest całkowita tylko w tym przypadku gdy wszystkie argumenty są całkowite w pozostałych przypadkach jest typu ułamkowego

Sposób podstawienia: analogiczny jak w funkcji MAX

Typy i ilość argumentów: takie same jak w funkcji MAX

Rodzaj argumentów: argumenty są wyrażeniami arytmetycznymi

Przykład: MIN(A,B,X+Y)

Znaczenie funkcji: wartością funkcji jest najmniejsza z liczb będących wartościami argumentów.

Modulo

Nazwa: MOD

Typ funkcji: funkcja jest całkowita

Sposób podstawienia: patrz opis funkcji MAX

Typy i ilość argumentów: funkcja jest dwuargumentowa; argumenty muszą być całkowite

Rodzaj argumentów: argumenty są wyrażeniami arytmetycznymi całkowitymi

Przykład: MOD(X,Y)

Znaczenie: wartością funkcji jest całkowita reszta z dzielenia argumentów (ilorazu X/Y)

Dodatkowa uwaga: w przypadku gdy Y = 0 "zapalany" jest wskaźnik nadmiaru a wartość funkcji jest równa zero

Dzielenie całkowite

Nazwa: DIV

Typ funkcji: funkcja jest całkowita

Sposób podstawienia: patrz opis funkcji MAX

Typ i ilość argumentów: funkcja jest dwuargumentowa; argumenty muszą być całkowite

Rodzaj argumentów: argumenty są wyrażeniami arytmetycznymi całkowitymi

Przykład: DIV(X,Y)

Znaczenie: wartością funkcji jest część całkowita ilorazu argumentów(X/Y)

Dodatkowe uwagi: w przypadku gdy Y = 0 "zapalany" jest nadmiar a wartość funkcji jest równa zero

Signum

Nazwa: SGN

Typ funkcji: zgodny z typem pierwszego argumentu

Sposób podstawienia: patrz opis funkcji MAX

Typy i ilość argumentów: funkcja jest dwuargumentowa; każdy z argumentów może być dowolnego typu

Rodzaj argumentów: wyrażenie arytmetyczne

Przykład: SGN(X,Y)

Znaczenie: wartość funkcji oblicza się wg wzoru

$$w = |X| \times \operatorname{sgn} Y$$

Wartość bezwzględna

Nazwa: ABS

Typ funkcji: jest taki sam, jak typ argumentu

Sposób podstawienia: patrz opis funkcji MAX

Typ argumentu: całkowity lub ułamkowy

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: ABS(X).

Znaczenie: wartością funkcji jest bezwzględna wartość argumentu

Część całkowita

Nazwa: ENT

Typ funkcji: całkowita

Sposób podstawienia: patrz opis funkcji MAX

Typ argumentu: ułamkowy

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: ENT(X)

Znaczenie: wartością funkcji jest część całkowita liczby będącej wartością argumentu (bez zaokrąglenia)

Dodatkowe uwagi: w trakcie obliczania wartości funkcji może nastąpić przekroczenie zakresu liczb całkowitych. W tym przypadku będzie "zapalony" nadmiar a wartość funkcji będzie nieokreślona

Suma

Nazwa: SUM

Typ funkcji: funkcja jest całkowita jeżeli trzeci argument jest całkowity natomiast jest ułamkowa gdy trzeci argument jest ułamkowy

Sposób podstawienia: patrz opis funkcji MAX

Typy i ilość argumentów: funkcja jest trójargumentowa, pierwsze dwa argumenty muszą być typu całkowitego a trzeci argument może być typu całkowitego lub ułamkowego

Rodzaj argumentów: pierwszy argument zwany zmienną kontrolującą musi być zmienną prostą, pozostałe dwa argumenty są wyrażeniami arytmetycznymi

Przykład: SUM(I,N,W(I))

Znaczenie: wartość funkcji można zdefiniować przy pomocy następującego odcinka programu:

```
      SUM = 0
   *) SUM = SUM + W(I)
      POWTORZ : I = N(- 1)0
```

tzn. wartością funkcji jest suma

$$\sum_{i=n}^{0} w_i$$

Wyrażenie arytmetyczne W nie musi zależeć od zmiennej kontrolującej.

Jeżeli funkcja SUM jest zawarta:

- w wyrażeniu będącym argumentem innej funkcji SUM lub ILN
- w wyrażeniu będącym prawą stroną zdania wielokrotnego przypisania
- w wyrażeniu znajdującym się w zakresie powtarzania zdania POWTORZ

to jej zmienna kontrolująca powinna być różna od zmiennych kontrolujących zdań lub funkcji ją obejmujących.

Iloczyn

Nazwa: ILN

Typ funkcji: taki, jak typ trzeciego argumentu.

Sposób podstawienia: patrz funkcja MAX

Typy i ilość argumentów: funkcja jest trójargumentowa; pierwsze dwa argumenty muszą być całkowite, trzeci argument może być dowolnego typu

Rodzaj argumentów: pierwszy argument zwany zmienną kontrolującą musi być zmienną prostą, pozostałe dwa argumenty są wyrażeniami arytmetycznymi

Przykład: ILN(I,N,W(I))

Znaczenie: wartość funkcji jest taka sama jak wartość otrzymana w wyniku wykonania następującego odcinak programu:

```
        ILN = 0
*) ILN = ILN ◊ W(I)
  POWTORZ : I = N(- 1) 0
```

tzn. wartością funkcji jest iloczyn:

$$\prod_{i=n}^{0} w_i$$

Jeżeli funkcja ILN jest zawarta:

- w wyrażeniu będącym argumentem innej funkcji ILN lub SUM
- w wyrażeniu znajdującym się w zakresie zdania POWTORZ
- w wyrażeniu będącym prawą stroną zdania wielokrotnego przypisania

to jej zmienna kontrolująca powinna być różna od zmiennych kontrolujących zdań lub funkcji ją obejmujących

Scharakteryzuj wykonanie zdania dotyczącego taśm magnetycznych

Nazwa: CHR

Typ funkcji: całkowita

Sposób podstawienia: patrz opis funkcji MAX

Typ argumentu: całkowity

Rodzaj argumentu: wyrażenie arytmetyczne całkowite o wartości z zakresu 0-7 włącznie

Przykład: CHR(2)

Znaczenie: wartością funkcji jest liczba całkowita opisująca zaistniałą sytuację po ostatnio wykonanym zdaniu dotyczącym stacji taśm magnetycznych o numerze wskazanym przez argument tej funkcji Znaczenie każdej z wartości zależy od ostatnio wykonywanego zdania i opisane jest przy opisie odpowiednich zdań języka SAKO dotyczących taśm magnetycznych

Podaj numer bloku taśmy magnetycznej

Nazwa: BTM

Typ funkcji: całkowita

Sposób podstawienia: patrz opis funkcji MAX

Typ argumentu: całkowity

Rodzaj argumentu: wyrażanie arytmetyczne całkowite o wartości z zakresu 0-7 włącznie

Przykład: BTM(2)

Znaczenie: wartością funkcji jest numer bloku TM (numer bloku informacji zapisanej na taśmie magnetycznej). Blok o podanym numerze znajduje się bezpośredno przed urządzeniem odczytującym (zapisującym) stacji magnetycznej wskazanej przez liczbę będącą argumentem funkcji.

Adres

Nazwa: ADR

Typ funkcji: całkowita

Sposób podstawienia: patrz opis funkcji MAX

Typ argumentu: całkowity lub ułamkowy

Rodzaj argumentu: nazwa bloku (struktury lub tablicy) lub zmiennej prostej.
Nazwa nie może być poprzedzona gwiazdką.

Znaczenie: wartością funkcji jest suma dwóch liczb całkowitych. Pierwszym składnikiem jest stała niezależna od argumentu, drugi składnik jest adresem pamięci operacyjnej przyporządkowanej nazwie będącej argumentem funkcji.
Wartość funkcji ADR może posłużyć do podstawienia przez nazwę nie tylko bloków lecz również zmiennych prostych i elementów bloków. Na wartości funkcji można wykonywać działania, które nie zmieniają wartości stałej niezależnej.

Przykłady:

Poprawne użycie funkcji ADR:

```
CALKOWITE: I,B,A
BLOK(7) : A,B,C
   --------
   --------
   --------
( )= F (ADR (A))          (zdanie to jest równoważne zdaniu:
   --------                  ( )= F (*A))
   --------
   --------
( )= F (ADR (B) + 8)      (to zdanie jest równoważne
   --------                zdaniu powyższemu)
   --------
   --------
( )= F ( ADR (C) + 8)     (teraz podstawiliśmy przez nazwę
   --------                blok zaczynający się w pamięci
   --------                operacyjnej od miejsca C (4))
   --------
I = ADR (C) + 8
( )= F (I)                tu również następuje podstawienie
   --------               przez nazwę
   --------
   --------
PODPROGRAM : ( ) = F (*Z)
STRUKTURA (3) : Z
-----------
-----------
-----------
```

Błędne działania wykonywane na wartości funkcji ADR:

I = ADR (Z)/2 (stała niezależna została podzielona)
I = ADR (Z) ◊ ADR (A)(iloczyn stałych niezależnych)
I = 345678 - ADR (Z)(zmienił się znak stałej niezależnej)
I = ADR (A) - ADR(B)(stała niezależna została wyrugowana)

Ilość elementów bloku

Nazwa: ELM
Typ funkcji: całkowita
Sposób podstawienia: patrz opis funkcji MAX
Typ argumentu: całkowity lub ułamkowy
Rodzaj argumentu: taki jak funkcji ADR
Znaczenie: Wartością funkcji jest liczba całkowita określająca ilość elementów (nie ilość słów) bloku (tablicy, struktury) którego nazwa jest argumentem omawianej funkcji.
Gdy argumentem jest zmienna prosta, to wartość funkcji jest równa jeden.

Przykład:

```
BLOK (3) : A
I = ELM (A)
----------
----------
----------
```

W powyższym przypadku wartością funkcji jest liczba całkowita 4.

Dołącz urządzenie symboliczne

Nazwa: DOL
Typ funkcji: całkowita
Sposób podstawienia: przez wartość lub przez nazwę
Typy i ilość argumentów: funkcja jest dwuargumentowa; pierwszy argument jest całkowity a drugi może być ułamkowy lub całkowity

Rodzaj argumentów: pierwszy argument jest wyrażeniem arytmetycznym całkowitym. Drugi jest blokiem (strukturą) podstawionym przez nazwę

Przykład: DOL(16+2,* A)

Znaczenie: zostały zaprojektowane trzy funkcje: DOL,ODL i SEL pozwalające na czytanie danych (lub wypisywanie wyników) z symbolicznych urządzeń wejściowych (na symboliczne urządzenia wyjściowe) o numerach różnych od zera. Aby urządzenie symboliczne o numerze różnym od zera, mogło być wykorzystane przez program wynikowy, należy jego definicję (podprogram i dekoder) wprowadzić do pamięci operacyjnej (patrz czołówka systemu). Do sprowadzenia definicji urządzenia służy funkcja DOL. Wykonanie tej funkcji powoduje wprowadzenie do pamięci operacyjnej, na miejsce wskazane przez drugi argument, dekodera i podprogramu urządzenia symbolicznego o numerze i typie (wejściowe lub wyjściowe) wskazanym przez jej pierwszy argument. Gdy pierwszy argument ma wartość n z przedziału 0-15 to zostanie dołączona definicja urządzenia symbolicznego wejściowego o numerze n.

Natomiast gdy $16 \leqslant n \leqslant 31$ to zostanie dołączone urządzenie wyjściowe o numerze n-16.

Wartością funkcji jest liczba całkowita będąca adresem pierwszego wolnego miejsca pamięci operacyjnej za dołączoną definicją. Wartość ta może posłużyć do dołączenia bezpośredno za poprzednią definicją, nowej definicji. Np.:

```
---------
---------
---------
CALKOWITE : I
I = DOL (2, * A)
I = DOL (16+2, I)
```

Aby została poprawnie dołączona definicja urządzenia symbolicznego, muszą być spełnione następujące warunki:

- pole pamięci wskazane przez drugi argument powinno być dostatecznie duże, aby mogło pomieścić całą definicję danego urządzenia symbolicznego. Typowa definicja urządzenia symbolicznego potrzebuje nie więcej niż 300 miejsc pamięci. Dokładnych danych na ten temat należy szukać w materiałach firmowych IMM.
  Wymaganą pamięć można rezerwować np. przy pomocy deklaracji BLOK.
- urządzenie o wskazanym numerze powinno być zdefiniowane w Czołówce Systemu.
  Używanie funkcji DOL jest zbędne, gdy w programie używamy tylko urządzeń o numerach zero.

Wybierz urządzenie symboliczne

Nazwa: SEL

Typ funkcji: całkowita

Sposób podstawienia: patrz funkcja DOL

Typ argumentu: całkowity

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: I = SEL (2)

Znaczenie: zdania języka SAKO czytają dane z urządzenia (drukują wyniki na urządzenie) symbolicznego o numerze ustalonym przez wartość argumentu funkcji SEL.
Jeżeli wartość argumentu jest z przedziału 0-15 to omawiana funkcja wybiera urządzenie wejściowe dla zdań czytania danych. Natomiast gdy argument (n) jest z przedziału 16-31 to funkcja ustali urządzenie wyjściowe o numerze n-16 dla zdań drukowania wyników i pisania tekstów.

Zatem SEL pozwala na kierowanie wyników (czytanie danych) przy pomocy zdań SAKD na (z) więcej niż jedno urządzenie.
Jeżeli wybieramy nowe urządzenie wejściowe, to wartością funkcji jest numer urządzenia, z którego do tej chwili były pobierane dane. W przypadku wybierania urządzenia wyjściowego wartością funkcji jest numer starego urządzenia wyjściowego zwiększony o 16.
Przed pierwszym wybraniem urządzenia o numerze różnym od zera należy wykonać odpowiednią funkcję DOL.
Używanie tej funkcji jest zbędne gdy korzystamy tylko z urządzeń o numerach zero.

Odłącz urządzenie symboliczne

Nazwa: ODL

Typ funkcji: nieistotny

Sposób podstawienia: musi być wywołana poprzez formułę operacyjną

Typ argumentu: całkowity

Rodzaj argumentu: wyrażenie arytmetyczne

Przykład: ( ) = ODL (2)

Znaczenie: funkcja powoduje odłączenie (rozdefiniowanie) urządzenia symbolicznego o numerze wskazanym przez wartość jej argumentu. Odłączanie urządzeń jest konieczne tylko wtedy, gdy pole pamięci zajęte przez definicję omawianego urządzenia ma być wykorzystane do innych celów np. na dołączenie innego urządzenia lub jako miejsca robocze.
Numery urządzeń wyjściowych wskazywanych przez ODL muszą być zwiększane o 16

Czytaj całkowite

Nazwa: CCC

Typ funkcji: całkowita

Sposób podstawienia: przez wartość lub przez nazwę

Typ argumentu: całkowity

Rodzaj argumentu: blok całkowity jednoelementowy podstawiony przez nazwę lub brak argumentu

Przykłady: L = CCC(ADR(I)),L1 = CCC(.)

Znaczenie: wartością funkcji jest liczba całkowita przeczytana z bieżącego wejścia symbolicznego. Liczba całkowita na wejściu jest to ciąg cyfr dziesiętnych poprzedzony ewentualnie znakiem minus lub plus. W trakcie czytania pomijane są wszystkie znaki występujące przed liczbą i nie należące do liczby; czytanie kończy się na pierwszym napotkanym znaku nie należącym do liczby. Ponadto pomijane są ciągi znaków zaczynające się od dwukropka (:) i kończące się znakiem LF włącznie. Jeżeli zostanie podstawiony argument (wystarczy raz podstawić) to do wskazanej zmiennej całkowitej będzie przesłana wartość (w KW6) kodu znaku ostatnio przeczytanego (kończącego liczbę).

Jeżeli na wejściu będziemy mieli tekst:

ALA 23 ; - 2*

to po wykonaniu pierwszego ze zdań podanych w przykładzie

L będzie miało wartość 23

a

I otrzyma wartość 7 (wartość kodu KW6 przypisana średnikowi).

Dalej, po wykonaniu następnego zdania z przykładu, L1 będzie równe -2 a I otrzyma wartość 5 (kod znaku *)

Uwaga! Gdy liczba czytana przekracza zakres przewidziany dla liczb całkowitych, to "zapalany" jest nadmiar stałoprzecinkowy a wartość funkcji jest nieokreślona.

Działania w dwuprecyzji

Został zaprojektowany szereg funkcji języka przewidzianych do prowadzenia obliczeń na liczbach ułamkowych zapisanych z większą dokładnością niż zwykłe liczby ułamkowe na tzw. liczbach dwuprecyzyjnych.
Liczba dwuprecyzyjna posiada zakres taki sam jak zwykła liczba ułamkowa; zapisana jest w czterech kolejnych słowach maszyny (wykładnik binarny zajmuje ostatnie 8 bitów ostatniego słowa a znak wykładnika jest zapisany w zerowej pozycji tego słowa; znaki pierwszych trzech słów muszą być takie same i równe znakowi liczby.

Argumenty (wyniki) poniżej podanych funkcji będące liczbami dwuprecyzyjnymi muszą być podstawiane przez nazwę jako bloki ułamkowe dwuelementowe lub jako bloki całkowite czteroelementowe, lub też mogą być elementami większych bloków a adresy poszczególnych liczb mogą być wyliczane przy pomocy funkcji ADR.

Do zbioru omawianych funkcji należą:

CZD - czytanie liczb dwuprecyzyjnych
DRD - drukowanie liczb dwuprecyzyjnych
DOD - dodawanie liczb dwuprecyzyjnych
ODD - odejmowanie liczb dwuprecyzyjnych
MND - mnożenie liczb dwuprecyzyjnych
DZD - dzielenie liczb dwuprecyzyjnych
ABD - bezwzględna wartość liczb dwuprecyzyjnych
IDK - zamiana liczby ułamkowej dwuprecyzyjnej na zwykłą
IKD - zamiana liczby ułamkowe na dwuprecyzyjną.

Do wszystkich tych funkcji można argumenty podstawiać przy pomocy zdania PODSTAW a każdą z funkcji można podstawiać przez nazwę. Natomiast nie wszystkie można podstawiać przez wartość, gdyż nie wszystkie posiadają wartości.
Liczba dwuprecyzyjna jest zapisana w pamięci operacyjnej z dokładnością 25 cyfr dziesiętnych (84 pozycje binarne mantysy).

Przy wykonywaniu funkcji: DOD, ODD, MND,DZD może nastąpić przekorzcenie zakresu, przewidzianego dla liczb ułamkowych, ze wszystkimi jego konsekwencjami (patrz CZOŁÓWKA TRANSLATORA NIWELUJ NADMIARY).

Czytanie liczb dwuprecyzyjnych

Nazwa funkcji: CZD

Typ funkcji: całkowita

Sposób wywoływania: w wyrażeniu jako funkcja lub przy pomocy formuły operacyjnej

Argument funkcji: blok czterosłowowy podstawiony przez nazwę

Znaczenie: funkcja powoduje przeczytanie liczby dwuprecyzyjnej, z bieżącego wejścia symbolicznego programu, do kolejnych czterech słów maszyny poczynając od początku bloku wskazanego przez jej argument. Ponadto wartością funkcji jest liczba całkowita, informująca program o prawidłowości wykonanego czytania:

1. Czytanie odbyło się poprawnie jeżeli wartość funkcji (n) jest zawarta w przedziale: $0 < n < 64$ W tym przypadku n jest wartością dziesiętną kodu KW6, znaku kończącego liczbę.
2. Liczba była błędna jeżeli wartość funkcji jest ujemna; gdy:

   a) $|n| \geqslant 128$ - liczba była zbyt duża; $|n| - 128$ jest kodem ostatnio przeczytanego znaku.

   b) $64 \leqslant |n| < 128$ - liczba była zbyt mała; $|n| - 64$ jest kodem ostatnio przeczytanego znaku

   c) $|n| < 64$ - n jest kodem ostatnio przeczytanego znaku wystąpiły inne błędy.

Postać danych: dane czytane przez funkcję mają taką samą postać jak liczby ułamkowe występujące w programie. Separatorem liczby (znakiem kończącym liczbę) jest każdy znak nie należący do liczby (również spacja).

znaki należące do liczby są następujące:
+, -, ., E, cyfry.
Między kolejnymi liczbami czytanymi przy pomocy funkcji CZD nie mogą występować komentarze, jedynie ilość spacji i znaków NL jest nieograniczona.

Przykłady wykorzystania funkcji:

I = CZD (*A)
(*A) = CZD (w tym zdaniu zakładamy, że nie interesuje nas wartość kodu znaku kończącego liczbę)

I = CZD (*A) + CZD (*B)

Drukowanie liczby dwuprecyzyjnej

Nazwa: DRD

Typ funkcji: ułamkowy

Sposób wywoływania: przy pomocy formuły operacyjnej

Argumenty funkcji: funkcja jest pięcioargumentowa:

1 - blok podstawiony przez nazwę tj, liczba podlegająca wypisaniu,
2 - wyrażenie arytmetyczne całkowite określające ilość pozycji cyfrowych przed kropką,
3 - wyrażenie arytmetyczne całkowite określające ilość pozycji cyfrowych po kropce,
4 - wyrażenie arytmetyczne całkowite określające minimalną ilość cyfr znaczących. Pojęcie "cyfr znaczących" na takie samo znaczenie jak w przypadku zdania DRUKUJ,
5 - wyrażenie arytmetyczne ułamkowe określające rozmieszczenie spacji w liczbie drukowanej.

Ogólna postać wywołania: ( ) = DRD(* A,I,J,K,S)

Znaczenie: funkcja powoduje wydrukowanie, na bieżące symboliczne wyjście programu, liczby dwuprecyzyjnej znajdującej się w pierwszych czterech słowach bloku będącego pierwszym argumentem funkcji. Pozostałe cztery argumenty określają sposób drukowania.

Poniżej rozpatrzymy trzy przypadki wykorzystania omawianej funkcji:

1. Niech ostatnie dwa parametry mają wartości:

   K = 0

   S [0000.0000.0000.0000.

   i załóżmy, że wartości te zostały uprzednio podstawione do funkcji przy pomocy np. zdania PODSTAW.

   Wtedy odwołanie do funkcji możemy zapisać w postaci:

   ( ) = DRD(* A,I,J)

   Jeżeli wartości wyrażeń arytmetycznych I i J są dodatnie, to powyższe zdanie ma takie same własności odnośnie liczby dwuprecyzyjnej jak zdanie DRUKUJ ($\underline{a}_1$,$\underline{a}_2$) : A w stosunku do liczby ułamkowej jednoprecyzyjnej.

   Jeżeli wartości I lub/i J są ujemne, to o sposobie rozmieszczenia liczby decydują ich bezwzględne wartości, jednakże ujemność argumentów niesie dodatkowe informacje.

   a) Gdy I jest ujemne to zamiast plusa przed drukowaną liczbą dodatnią wypisywany jest znak SP (odstęp)

   b) Gdy J jest ujemne to przed wypisywaną liczbą nie rezerwuje się pozycji na jej znak, a zamiast wypełniających z przodu spacji pisze się cyfry zero.

   W przypadku drukowania liczb ujemnych znak minus jest wypisywany w środku liczby co w rezulatacie jest pewnego rodzaju sygnalizacją błędu.

2. Przy K=0 i wartości S takiej jak w poprzednim przypadku zdanie odwołujące się do funkcji DRD można zapisać w postaci:

   ()= DRD (*A,I,J,K)

   Jeżeli K>0 to znaczenie powyższego zdania jest podobne do znaczenia zdania DRUKUJ ($\underline{a}_1$,$\underline{a}_2$,$\underline{a}_3$): A Znaki parametrów I i J w tym przypadku odgrywają tę samą rolę jak w przypadku K=0. Ujemna wartość K oznacza dodatkowo, że przed dodatnim wykładnikiem zamiast znaku plus należy drukować znak spacji (SP) .

3. Argumenty I,J,K wyznaczają pewną ilość pozycji na arkuszu wydawniczym przeznaczonych dla wydrukowania liczby. Przewiduje się, że w każdej takiej pozycji będzie wydrukowany jakiś znak. Z każdą pozycją związany jest jeden bit długiego słowa podstawionego jako piąty argument. Kolejnym pozycjom odpowiadają kolejne bity poczynając do lewej strony. W procesie drukowania sprawdza się, czy danej pozycji liczby drukowanej odpowiada w słowie S jedynka czy też zero. W przypadku wystąpienia jedynki, po wydrukowaniu odpowiedniego znaku, dodatkowo drukuje się odstęp (SP) . W ten sposób można między pozycje liczby wstawiać dodatkowo pojedyncze spacje. Założenia poczynione na początku punktu 1 są spełnione ponieważ podstawienie w/w wartości jest automatyczne, w momencie rozpoczęcia wykonywania się programu wynikowego.
   Stąd argumenty K i S nie muszą być w ogóle podstawione.

Argumenty I,J,K muszą spełniać następujący warunek:

$$|K| \leqslant |I| + |J| \leqslant 40$$

Funkcja drukuje co najwyżej 27 cyfr znaczących (w sensie DRUKUJ) a pozostałe pozycje, gdy tego wymaga układ są uzupełniane zerami.

Przykłady:

Niech blok A zawiera liczbę dwuprecyzyjną o wartości 123456789.

Załóżmy ponadto, że było wykonane zdanie:

AS [5040.0000.0000.0000.0000

Poniższa tabelka podaje kilka przykładów użycia funkcji DRD.

Zdania	Wynik drukowania
( ) = DRD (*A,10,1)	␣+123456789.0
( ) = DRD (* A,-10,1)	␣␣123456789.0
( ) = DRD (* A,10,-1)	0123456789.0
( ) = DRD (* A,10,1,2)	␣+123456789.0␣␣␣␣
( ) = DRD (*A,-10,1,2)	␣␣123456789.0␣␣␣␣
( ) = DRD (*A,3,2)	+123.46E+6␣
( ) = DRD (* A,3,2,2,S)	+␣12␣3.46␣E+6␣
( ) = DRD (* A,-3,2,-2,S)	␣␣12␣3.46␣E␣6␣

Dla skrócenia zapisu i programu wynikowego, gdy pisze się różne liczby według ustalonego formatu, można jednorazowo podstawić parametry drukowania a zmieniać tylko pierwszy parametr.

Np. PODSTAW : DRD (.,-3,2,-2,S)

( ) = DRD (* A)

Dodawanie liczb dwuprecyzyjnych

Nazwa: DOD

Typ funkcji: ułamkowa

Sposób wywołania: poprzez formułę operacyjną

Argumenty funkcji: funkcja jest trójargumentowa; argumentami są bloki podstawione przez nazwę, na liście wyników umieszcza się zazwyczaj nazwę bloku, do którego należy posłać wynik działania.

Przykład: (*A) = DOD (*B, *C)

Znaczenie: funkcja oblicza sumę dwóch liczb dwuprecyzyjnych i wynik działania przesyła do bloku wskazanego na liście wyników. Zdanie podane w powyższym przykładzie ma takie samo znaczenie w stosunku do liczb dwuprecyzyjnych jak zdanie:

A = B + C

w stosunku do liczb jednoprecyzyjnych.

Zapis

( ) = DOD (*B, *C, *A)

jest równoważny powyższemu przykładowi.

Odejmowanie liczb dwuprecyzyjnych

Nazwa: ODD

Typ funkcji: ułamkowa

Sposób wywołania: poprzez formułę operacyjną

Argumenty funkcji: patrz opis funkcji DOD

Przykład: (*A) = ODD (*B, *C)

Znaczenie: funkcja oblicza różnicę liczb dwuprecyzyjnych zapisanych w blokach B i C. Wynik działania przesyła do bloku A.

Powyższe zdanie jest podobne do zdania przypisującego:

A = B - C .

Dodatkowe uwagi: funkcja ODD jest realizowana przy pomocy funkcji DOD. O tym fakcie należy pamiętać przy ewentualnym wcześniejszym podstawieniu argumentów do dwuprecyzyjnego dodawania.

Mnożenie liczb dwuprecyzyjnych

Nazwa: MND

Typ funkcji: ułamkowa

Sposób wywołania: poprzez formułę operacyjną

Argumenty: patrz opis funkcji DOD

Przykład: (*A) = MND (*B,*C)

Znaczenie: funkcja oblicza iloczyn dwóch liczb dwuprecyzyjnych zapisanych w blokach B i C; wynik obliczeń umieszcza w bloku A.
Zdanie podane w przykładzie jest podobne do zdania przypisującego:

A = B ◊ C

Dzielenie liczb dwuprecyzyjnych

Nazwa: DZD

Typ funkcji: ułamkowa

Sposób wywołania: poprzez formułę operacyjną

Argumenty: patrz opis funkcji DOD

Przykład: (*A) = DZD (*B,*C)

Znaczenie: funkcja oblicza iloraz dwóch liczb dwuprecyzyjnych zapisanych w blokach B i C. Wynik działania jest posyłany do bloku A.
Zdanie podane w przykładzie jest podobne do zdania przypisującego:

A = B/C

Zamiana liczby ułamkowej jednoprecyzyjnej na liczbę dwuprecyzyjną

Nazwa: IKD

Typ funkcji: ułamkowa

Sposób wywołania: poprzez formułę operacyjną

Argumenty funkcji: argumentem funkcji jest wyrażenie arytmetyczne ułamkowe a wynikiem blok podstawiony przez nazwę.

Przykład: (*A) = IKD (B)

Znaczenie: funkcja powoduje obliczenie wartości wyrażenia arytmetycznego będącego argumentem funkcji a następnie sprowadzenia wyniku obliczeń do

postaci liczby dwuprecyzyjnej i posłanie jej do bloku A. Sprowadzenie liczby ułamkowej zwykłej do postaci dwuprecyzyjnej polega na wydłużeniu mantysy poprzez dopisanie zer na jej końcu.

Zamiana liczby dwuprecyzyjnej na liczbę ułamkową zwykłą

Nazwa: IDK

Typ funkcji: ułamkowa

Sposób wywołania: w wyrażeniu arytmetycznym

Argument funkcji: blok podstawiony przez nazwę

Przykład: A = IDK (*A)

Znaczenie: wartością funkcji jest liczba ułamkowa utworzona z liczby dwuprecyzyjnej przez odrzucenie jej mniej znaczącej części.

Spis funkcji języka

			str.
SIN	(X)	sinus	290
COS	(X)	cosinius	290
TG	(X)	tangens	290
ASN	(X)	arcus sinus	291
ACS	(X)	arcus cosinus	291
ATG	(X)	arcus tangens	292
PWK	(X)	pierwiastek kwadratowy	292
PWS	(X)	pierwiastek sześcienny	292
LN	(X)	logarytm naturalny	293
EXP	(X)	eksponent	293
ARC	(X,Y)	arcus	293
MAX	(A,B,...,Z)	maksimum	294
MIN	(A,B,...,Z)	minimum	294
MOD	(X,Y)	modulo	295
DIV	(X,Y)	dzielenie całkowite	295
SGN	(X,Y)	signum	296
ABS	(X)	wartość bezwzględna	296
ENT	(X)	część całkowita	296
SUM	(I,N,W)	suma	297
ILN	(I,N,W)	iloczyn	298
CHR	(X)	scharakteryzuj taśmę magnetyczną	299
BTM	(X)	nr bloku taśmy magnetycznej	299
ADR	(X)	adres bloku	299
ELM	(X)	ilość elementów bloku	301
DOL	(X,*Y)	dołącz urządzenie symboliczne	301
ODL	(X)	odłącz urządzenie symboliczne	304
SEL	(X)	wybierz urządzenie symboliczne	303
CCC	(X)	czytaj całkowitą	305
CZD	(*X)	czytaj liczbę dwuprecyzyjną	307
DRD	(*A,I,J,K,S)	drukuj liczbę dwuprecyzyjną	308
DOD	(*A, *B, *C)	dodawanie liczb dwuprecyzyjnych	312
ODD	(*A, *B, *C)	odejmowanie liczb dwuprecyzyjnych	312
MND	(*A, *B, *C)	mnożenie liczb dwuprecyzyjnych	313
DZD	(*A, *B, *C)	dzielenie liczb dwuprecyzyjnych	313

		str.
IKD (A, *B)	zamiana liczby ułamkowej jednoprecyzyjnej na dwuprecyzyjną	313
IDK (*B)	zamiana liczby dwuprecyzyjnej na ułamkową jednoprecyzyjną	314

DODATEK C

Porównanie języków SAKO dla maszyn ZAM-2 i ZAM-41

Dwa zasadnicze czynniki wpłynęły na powstanie różnic między językami dla maszyn ZAM-2 i ZAM-41.

1. Język SAKO, w pierwotnej jego postaci, był w istotny sposób związany z własnościami konstrukcyjnymi maszyny ZAM-2 i wierne odtworzenie własności tego języka na maszynie ZAM-41 byłoby z praktycznego punktu widzenia nie wskazane.

2. System operacyjny maszyny ZAM-41 (SO141) dostarcza wiele udogodnień programowych (symboliczne wejścia-wyjścia, biblioteka systemu, translatory binarne, możliwość korzystania z taśm magnetycznych). Zachowując ubogi system operacyjny maszyny ZAM-2 pozbawilibyśmy programistę bogatego oprogramowania nowej maszyny.
   Podłączenie translatora języka SAKO do systemu operacyjnego SO141 oraz przystosowanie programu wynikowego do korzystania z udogodnień systemu, pociągnęły za sobą konieczność zmiany znaczenia niektórych zdań języka.
   Ponadto dla zwiększenia efektywności języka rozszerzono znacznie zakres własności poszczególnych zdań w stosunku do ich pierwotnego znaczenia oraz wprowadzono szereg nowych zdań. W szczególności wprowadzono dodatkowe zdania dotyczące współpracy z taśmami magnetycznymi. Zbiór funkcji języka został również rozszerzony o nowe funkcje.
   Dalszy ciąg niniejszego dodatku omawia dokładniej tylko te różnice występujące między omawianymi językami, na które należy zwrócić szczególną uwagę przy adaptowaniu programów napisanych dla maszyny ZAM-2 w celu wykorzystania ich w systemie SO141 maszyny ZAM-41.

Istotne różnice są następujące:

a) Języki symboliczne używane we wstawkach SAS są różne i wstawki napisane w języku SAS maszyny ZAM-2 nie są akceptowane przez translator języka SAKO maszyny ZAM-41.

b) Dane programu wynikowego maszyny ZAM-2 były pobierane w kodzie FERRANTI zaś w systemie SO141 w kodzie KW6. Ponieważ wartości tych kodów są różne (patrz DODATEK A) to programy analizujące teksty będą musiały ulec przeróbce.

c) Różnice w długości słów obu maszyn spowodowały różnice w interpretacji operacji logicznych w wyrażeniach logicznych.

d) Rejestr kluczy w maszynie ZAM-41 posiada 24 pozycje (poprzednio posiadał 36 pozycji).

e) Program napisany dla maszyny ZAM-41 musi być poprzedzony ciągiem zdań sterujących (patrz opisy czołówek systemu i translatora) i nie mogą występować żadne informacje między rozdziałami.

f) Końcem wiersza w starej wersji języka jest znak CR (karetka) zaś obecnie jest znak LF (tj. przejście do nowej linii). Różnica ta jest istotna tylko przy liczeniu wierszy będących parametrami tekstowymi zdań TEKST i TEKST WIERSZY.

g) Wprowadzenie działań zmiennoprzecinkowych spowodowało, że funkcja SKL (.) nie ma sensu i wystąpienie jej w programie jest błędem. Również z powyższych przyczyn zdania:

SKALA DZIESIETNA PARAMETROW
SKALA BINARNA PARAMETROW
USTAW SKALE DZIESIETNIE
USTAW SKALE BINARNIE
ZWIEKSZ SKALE DZIESIETNIE 0
ZWIEKSZ SKALE BINARNIE 0

straciły sens. Jednakże dla ułatwienia prac adaptacyjnych nowy translator je rozpoznaje jako zdanie poprawne i pomija - traktuje jak komentarz.

h) Zdanie KONIEC ROZDZIALU jest w nowym systemie zbędne i jest przez translator pomijane.

i) Zbiór funkcji języka został powiększony o nowe funkcje (patrz DODATEK B). Stąd podprogramy lub funkcje definiowane o takich samych nazwach jak nazwy funkcji języka, będą błędnie tłumaczone.

j) Jednostką pamięci bębnowej maszyny ZAM-2 było jedno słowo długie i niezależnie od typu liczb przesyłanych na bęben, każda informacja zajmowała jedno miejsce.
Obecnie jednostką pamięci bębnowej jest jedno słowo krótkie. Stąd dotychczasowa metoda liczenia adresów bębnowych jest poprawna tylko w przypadku przesyłania na bęben zmiennych prostych lub bloków typu całkowitego. Łatwo też jest zamienić stary adres bębnowy na nowy, gdy na bęben są posyłane informacje tylko typu ułamkowego.
Wystarczy wtedy podwoić adres bębnowy (I) występujący w zdaniach:

CZYTAJ Z BEBNA OD I :
PISZ NA BEBEN OD I :

tzn. napisać

CZYTAJ Z BEBNA OD I+I:

lub

PISZ NA BEBEN OD 2◇ I:

Sprawa jest nieco bardziej skomplikowana w przypadku przesyłania bloków o różnych typach.

Mimo tak licznych różnic między interpretacją zdań przez translatory pracujące na obu maszynach, większość programów napisanych w języku SAKO dla maszyny ZAM-2 daje się przystosować do pracy na maszynie ZAM-41, bez potrzeby analizy treści programu, postępując według formalnych reguł.

DODATEK D

Spis zdań języka SAKO

Zdania przypisujące

A = B

A - zmienna prosta lub zmienna indeksowana str. 75

B - wyrażenie arytmetyczne

A [B

A - zmienna prosta str. 128

B - wyrażenie logiczne

A(= I,...,=N)= B

A - nazwa bloku str. 87

I,...,N - zmienne proste całkowite

B - wyrażenie arytmetyczne

F( )= B

F - nazwa funkcji definiowanej str. 195

B - wyrażenie arytmetyczne

F( )[B

F - nazwa funkcji definiowanej str. 198

B - wyrażenie logiczne

PODSTAW : P(A,..., *B, F1( ))

F - nazwa funkcji definiowanej lub pod- str. 177
programu

A - wyrażenie arytmetyczne

B - nazwa bloku

F1- nazwa funkcji lub podprogramu
podstawiona przez nazwę

(*A,B,...) = P (C, *D,F1( ),...) str. 168

A,D - nazwy bloków

B - nazwa zmiennej prostej

C - wyrażenie arytmetyczne

F1- nazwa funkcji lub podprogramu
podstawionego przez nazwę

F - nazwa podprogramu

Zdania sterujące

SKOCZ DO [:] 1E str. 70

1E - etykieta lub słowo NASTEPNY

SKOCZ WEDLUG I : 1E,2E,... str. 77

I - wyrażenie arytmetyczne całkowite

1E,2E,... - etykiety lub słowo NASTEPNY

POWTORZ [OD 1E] : I = J(K) L str. 78

I - zmienna prosta

J - liczba ze znakiem lub zmienna prosta

K,L - wyrażenia arytmetyczne

GDY A > B : 1E, INACZEJ 2E str. 73

A,B - wyrażenia arytmetyczne

1E,2E - etykiety lub słowo NASTEPNY

GDY A = B : 1E, INACZEJ 2E str. 75

A,B - wyrażenia arytmetyczne

1E,2E - etykiety lub słowo NASTEPNY

GDY BYL NADMIAR : 1E, INACZEJ 2E str. 91

1E,2E - etykiety lub słowo NASTEPNY

GDY KLUCZ I : 1E, INACZEJ 2E str. 93

I - wyrażenie arytmetyczne całkowite

1E,2E - etykiety lub słowo NASTEPNY

STOP [:] 1E str. 95

1E - etykieta lub słowo NASTEPNY

WROC str.179

Deklaracje

BLOK (I1,I2,...) : A,B,... str. 54

I1,I2,... - wyrażenia całkowite liczbowe

A,B,... - nazwy bloków

CAŁKOWITE : A, *B,F( ),... str. 52

A- nazwa zmiennej prostej

B- nazwa bloku

F- nazwa funkcji definiowanej

Zdania współpracy z pamięciami pomocniczymi

PISZ NA BEBEN OD I : A,*B,... str.140

I - wyrażenie arytmetyczne całkowite

A - nazwa zmiennej prostej

B - nazwa bloku

CZYTAJ Z BEBNA OD I : A,*B,... str.143

I - wyrażenie arytmetyczne całkowite

A - nazwa zmiennej prostej

B - nazwa bloku

OTWORZ A [, {*/X}] : Y [,S [,W]] str.148

A - wyrażenie arytmetyczne całkowite
- numer stacji

X - nazwa zbioru likwidowanego

Y - nazwa zbioru tworzonego

S - wyrażenie arytmetyczne całkowite
- numer stacji

W - wyrażenie arytmetyczne całkowite
- okres ważności

ZAPISZ AA [,Z] : A,B,C,... str.151

AA - wyrażenie arytmetyczne całkowite
- numer stacji

Z - nazwa zmiennej całkowitej

A,B,C,... - nazwy bloków

ZAMKNIJ ({S/Z}) A str.154

S,Z - pojedyncze litery - typ zdania ZAMKNIJ

A - wyrażenie arytmetyczne całkowite
- numer stacji

ZNAJDZ A : Y [,S] str.155

A - wyrażenie arytmetyczne całkowite
- numer stacji

Y - nazwa zbioru

S - wyrażenie arytmetyczne całkowite
- numer sekcji

POSUN A : N str.157

A - wyrażenie arytmetyczne całkowite
- numer stacji

B - wyrażenie arytmetyczne całkowite
- numer bloku TM

POBIERZ A1 [,N] : A,B,C,... str.158

A1 - wyrażenie arytmetyczne całkowite
- numer stacji

N - wyrażenie arytmetyczne
numer bloku TM

A,B,C,... - nazwy bloków

ZMIEN A str.160

A - wyrażenie arytmetyczne całkowite
- numer stacji

Tablice i teksty

TABLICA (I1,I2,...): A str. 60

C0,C1,...

*

I1,I2,... - wyrażenia całkowite liczbowe

A -nazwa tablicy

C0,C1,... - liczby dziesiętne

TABLICA OKTALNA (I1,I2,...) : A str. 64
CO,C1,...

I1,I2,... - wyrażenia całkowite liczbowe
A - nazwa tablicy
CO,C1,... - liczby oktalne

TEKST [:] str. 111
ABC
ABC - jeden wiersz tekstu

TEKST WIERSZY I [:] str. 113
A
B
⋮
I - liczba całkowita bez znaku
A,B,... - I wierszy tekstu

Struktura

STRUKTURA (I1,I2,...) :A,B,... str. 67
I1,I2,... - wyrażenie arytmetyczne całkowite
A,B,... - nazwy bloków

Zdania opisujące podprogramy

PODPROGRAM : (A,*B,...)=F(C,*D,F1( ),...) str. 180
A,C - zmienne formalne proste
B,D - bloki formalne
F1 - funkcja formalna podstawiona przez nazwę
F - nazwa podprogramu

PODPROGRAM : F(A,*B,F1( ),...) str. 183
F - nazwa funkcji definiowanej
A - zmienna prosta formalna
B - blok formalny
F1- nazwa funkcji formalnej podstawionej przez nazwę

Zdania opisujące rozdziały

ROZDZIAL [:] N str. 200
N - liczba całkowita bez znaku - numer rozdziału

KONIEC [:][N] str. 207
N - liczba całkowita bez znaku - numer rozdziału

IDZ DO ROZDZIALU [:] N str. 203
N - zmienna całkowita lub liczba całkowita bez znaku

Zdania opisujące wstawki

JEZYK SAS [:] A,*B,... str. 166
A - nazwa zmiennej prostej
B - nazwa bloku

BIBLIOTEKA [:] A str. 162
A - nazwa tekstu w bibliotece

PRZESTAW str. 163

Komentarze

K) tekst komentarza str. 165
: tekst komentarza

Zdania wejścia-wyjścia

CZYTAJ : A,*B,... str. 117
A - nazwa zmiennej prostej
B - nazwa bloku

DRUKUJ (I1 [,I2[,I3]]): A,B,... str. 98
I1,I2,I3 - wyrażenia arytmetyczne całkowite - parametry
A,B,... - wyrażenia arytmetyczne

CZYTAJ WIERSZ : A,B,... str.122
A,B,... - nazwy bloków całkowitych

DRUKUJ WIERSZ : A,B,... str.115
A,B,... - nazwy bloków całkowitych

CZYTAJ OKTALNIE : A,*B,... str.124
A - nazwa zmiennej prostej
B - nazwa bloku

DRUKUJ OKTALNIE : A,B,... str.126
A,B,... - wyrażenia arytmetyczne

SPACJA [:] [I] str.106
I - wyrażenie arytmetyczne całkowite

LINIA [:] [I] str.108
I - wyrażenie arytmetyczne całkowite

STRONA str.110

DODATEK E

Spis nazw rozkazów dopuszczalnych we wstawkach języka SAS

Skrót nazwy	Wartość	Nazwa rozkazu
CKA	48/8+15/14	czytaj klucze do A
DOA	38/8	dodaj do akumulatora
DOB	54/8	dodaj do B
DOL	44/8	dodaj logicznie do A
DOM	40/8	dodaj do mnożnika
DOP	4/8	dodaj do pamięci
DOZ	12/8	dodaj w zmiennym przecinku
DZD	43/8	dziel AM
DZZ	15/8	dziel w zmiennym przecinku
LAR	48/8+21/14	przesuń AM w lewo arytmetycznie
LCA	48/8+25/14	przesuń A w lewo cyklicznie
LCD	48/8+17/14	przesuń AM w lewo cyklicznie
LCM	48/8+29/14	przesuń M w lewo cyklicznie
LMB	48/8+7/14	mnoż logicznie M i prześlij do B
MNL	46/8	mnoż logicznie A
MNM	42/8	mnoż M
MNZ	14/8	mnoż w zmiennym przecinku
NNR	48/8+63/14	nic nie rób
NOR	48/8+49/14	normalizuj
NZB	48/8+11/14	neguj znak B
OAM	48/8+53/14	zaokrąglij A oraz M
ODA	39/8	odejmij od akumulatora
ODB	55/8	odejmij od B
ODM	41/8	odejmij od mnożnika
ODP	5/8	odejmij od pamięci
ODZ	13/8	odejmij w zmiennym przecinku
OKZ	48/8+51/14	zaokrąglij w zmiennym
OSL	45/8	odejmij symetrycznie logicznie

Skrót nazwy	Wartość	Nazwa rozkazu
OWW	63/8+0/14	operacja wejścia-wyjścia
PAD	34/8	pamiętaj A oraz M długie
PAM	33/8	pamiętaj A oraz M
PAR	48/8+23/14	przesuń AM w prawo arytmetycznie
PCA	48/8+27/14	przesuń A w prawo cyklicznie
PCD	48/8+19/14	przesuń AM w prawo cyklicznie
PCM	48/8+31/14	przesuń M w prawo cyklicznie
PEB	48/8+1/14	przeskocz przy większym od B
PLA	48/8+13/14	pisz do rejestru lampek
PMB	51/8	pamiętaj B
PMM	50/8	pamiętaj mnożnik
POB	53/8	porównaj B
POL	47/8	porównaj logicznie A
PPB	48/8+3/14	przeskocz przy plusie B
PPM	48/8+35/14	przeskocz przy plusie M
PRB	48/8+45/14	przeskocz przy różnym B
PRM	48/8+37/14	przeskocz przy nie zerze w M
PZA	32/8	pamiętaj i zeruj akumulator
PZN	48/8+5/14	przeskocz przy zerze N
P06	6/8	rozkazy programowane
P07	7/8	
P08	8/8	
P09	9/8	
P10	10/8	
P11	11/8	
P16	16/8	
P17	17/8	
P18	18/8	
P19	19/8	
P20	20/8	
P21	21/8	
P22	22/8	
P23	23/8	
P24	24/8	

Skrót nazwy	Wartość	Nazwa rozkazu
P25	25/8	rozkazy programowane
P26	26/8	
P27	27/8	
P28	28/8	
P29	29/8	
P30	30/8	
P31	31/8	
SDY	1/8	operacja zewnętrzna
SKO	0/8	skocz
SLR	58/8	skocz pamiętając ULR
SMA	60/8	skocz przy minusie A
SOB	57/8	skocz po odjęciu od B
SSE	59/8	skocz do segmentu
SSL	2/8	skocz ze śladem
STO	48/8+33/14	stop
SUB	56/8	skocz i umieść w B
SZA	61/8	skocz przy zerze A
UAD	35/8	umieść w A oraz M długo
UAM	36/8	umieść w A oraz M
UEB	62/8	umieść adres efektywny w B
UMA	37/8	umieść w akumulatorze
UMB	52/8	umieść w B
UMM	50/8	umieść w mnożniku
WBM	48/8+39/14	wykładnik z B do M
WMB	48/8+41/14	wykładnik z M do B
WRO	3/8	wróć
ZAM	48/8+59/14	zamień A z M
ZBA	48/8+57/14	zamień B z A
ZMB	48/8+43/14	zamień M z B
ZZN	48/8+55/14	zrównaj znaki w A i M
ZZR	48/8+0/14	zmień zawartość rejestrów

DODATEK F

Wstawki w języku SAS

Uwagi wstępne

Przez wstawkę w języku SAS będziemy rozumieli odcinek programu zaczynający się od zdania JEZYK SAS i kończący się zdaniem JEZYK SAKO.

Program występujący między tymi zdaniami musi być napisany w języku symbolicznym stanowiącym podzbiór języka M-SAS (patrz "ZAM-41 oprogramowanie część II, Język SAS).

Z następujących powodów wprowadzono możliwość pisania wstawek:

1. do języka SAS została napisana duża ilość bibliotecznych funkcji języka, które tą drogą mogą być wykorzystane,
2. pewne fragmenty programu pisze się wygodniej w języku symbolicznym niż w autokodzie.

Jako język wstawek został wybrany podzbiór języka SAS z następujących przyczyn:

1. aby nie mnożyć niepotrzebnie ilości języków symbolicznych dla tej samej maszyny,
2. wybrano podzbiór, gdyż realizacja pełnego języka SAS była niemożliwa i niewskazana.

Niektóre uproszczenia tej części translatora, która realizuje kompilację wstawek, spowodowały naturalne rozszerzenie pewnych pojęć języka. Rozszerzenia te postanowiliśmy opisać, gdyż wydają się być pożyteczne.

Przy opisie wstawek zakłada się znajomość maszyny ZAM-41 [3].

Ogólna budowa wstawek

Znaki nieistotne (wymienione w DODATKU A) są we wstawkach pomijane, podobnie jak i w języku SAKO. W szczególnym przypadku znakiem nieistotnym jest spacja (SP).

Wstawka składa się z ciągu instrukcji, z których każda może być poprzedzona ciągiem etykiet.

Separatorem instrukcji (znakiem kończącym instrukcję) może być: przecinek (,) , znak nowej linii (LF) . Ponadto instrukcję może kończyć (jest separatorem) , komentarz języka SAS. Ciąg separatorów instrukcji jest równoważny jednemu separatorowi. Separatorem etykiety jest nawias okrągły zamykający. Między separatorem etykiety (nawiasem zamykającym) a następną etykietą może wystąpić separator instrukcji (LF , , , komentarz). Etykieta ma taką samą postać jak nazwa zmiennej prostej w języku SAKO (identyfikator w języku SAS); własność identyfikującą posiadają tylko pierwsze cztery znaki etykiety.

Instrukcją jest:

- rozkaz
- makrorozkaz
- dyrektywa REZE
- stała tekstowa
- liczba dziesiętna
- liczba oktalna

Komentarz

Ciąg znaków o zestandaryzowanej budowie składający się z następujących trzech części:

- dwa kolejne znaki /* stanowią początek komentarza,
- dowolny ciąg znaków nie zawierający końca komentarza,
- dwa kolejne znaki */ stanowią koniec komentarza,

jest we wstawkach pomijany i nazywa się komentarzem.

Przykład:

```
/* TO JEST JEDEN KOMENTARZ
   WE WSTAWCE SAS
   : : - - - - -
   : [s]          - 1.7-
   */
```

Uwaga

Dodatkowo dopuszcza się we wstawkach komentarz zaczynający się od dwukropka. Znaczenie tego komentarza jest takie same jak odpowiedniego komentarza w języku SAKO tj. może on występować tylko w oddzielnym wierszu (nie jest separatorem).

Liczba dziesiętna

Liczby dziesiętne dzielimy na dwa typy:

- całkowite
- ułamkowe

Liczba dziesiętna całkowita jest ciągiem cyfr dziesiętnych, który może być poprzedzony znakiem (+ lub -). Liczba całkowita powinna mieć wartość z przedziału 0 - 8388607 i zajmuje jedno miejsce w pamięci.

Liczba dziesiętna ułamkowa ma taką samą postać jak liczba ułamkowa zdefiniowana w języku SAKO. Liczba ułamkowa może być poprzedzona znakiem (+ lub -), w pamięci maszyny zajmuje dwa miejsca i jest zawsze sprowadzana przez translator do postaci zmiennoprzecinkowej.

Przykłady:

Poprawne liczby dziesiętne:

```
+7, 2456, 0.4,-13.2 E12, 0,-0432
-0      /* Bit na najwyższej pozycji słowa  */
1  E - 7, 1.E+3, 1 E3
```

Błędne liczby dziesiętne:

```
8766543      /* przekracza zakres liczby całkowitej */
1 E93        /* przekracza zakres liczby ułamkowej */
  E70        /* to nie jest liczba ułamkowa */
```

Liczba oktalna

Znaczenie liczb oktalnych, krótkich i długich występujących we wstawce jest takie samo jak znaczenie odpowiednich liczb oktalnych opisanych w języku SAKO. Różnice są jednak w zapisie tych liczb tj.:
dla odróżnienia liczby oktalnej od liczby dziesiętnej, tę pierwszą poprzedzamy znakiem równości, ponadto w liczbach oktalnych pisanych we wstawkach nie wolno pisać kropek między cyframi.

Przykłady:

Poprawnych liczb oktalnych:

```
=12345670, =00776502
=12 34 56 70      /* SPACJE SA NIEISTOTNE */
=123456701        /* LICZBA OKTALNA DŁUGA */
=000777666.       /* TO TEŻ JEST LICZBA OKTALNA DLUGA */
```

Stała tekstowa

Stała tekstowa jest specjalnym zapisem pozwalającym umieścić w słowie maszyny, od jednej do czterech liczb będących wartościami kodów znaków w KW6.
Stałą tę zapisujemy w postaci ciągu reprezentacji kodów, ujętego w apostrofy.
Reprezentacja kodu jest to zapis oznaczający liczbę będącą wartością jakiegoś znaku w KW6.

Znaki: litery, cyfry,,, +, -, :, /, ;, =, ◊ , [,] , > , ?
są prostą reprezentacją kodów znaków i oznaczają swoje wartości w kodzie KW6 (patrz DODATEK A).

Natomiast znaki: A,B,C,...,R poprzedzone kropką są złożonymi reprezentacjami kodów znaków podanych w poniższej tabelce:

Reprezentacja	Nazwa znaku	Wartość kodu	Reprezentacja	Nazwa znaku	Wartość kodu
.A	'(AP)	12	.J	? (QS)	18
.B	(SP)	1	.K	(ES)	19
.C	(LF)	58	.L	(SO)	20
.D	.(FP)	2	.M	(SI)	47
.E	(NU)	0	.N	(FF)	60
.F	[(LB)	14	.O	(CS)	61
.G	](RB)	15	.P	(CR)	62
.H	<(LT)	16	.Q	(DE)	63
.I	>(GT)	17	.R	(N6)	59

Na podstawie powyższej tabelki i listy prostych reprezentacji, można zauważyć, że niektóre liczby można kodować na jeden z dwu sposobów.

W słowie rozmieszcza się kody znaków w sposób następujący:
- pierwszą liczbę umieszczamy w pozycjach 0-5 słowa,
- drugą liczbę umieszczamy w pozycjach 6-11 słowa itd.

Gdy stała składa się z mniej niż czterech reprezentacji, to pozostałe pozycje słowa są zerami.

Przykłady:

Poprawne stałe tekstowe:

```
'TO.BJ', 'EST.B', 'CIAG', 'STAL', 'YCH.C'
    / * powyżej zapisano tekst: TO JEST CIAG STALYCH */
'[7] ', '.F7.G'
    / * powyższe dwie stałe tekstowe mają takie same
        wartości */
```

Błędne stałe tekstowe:

```
'ALAMA'     / * więcej niż cztery reprezentacje */
'ALA"       / * apostrof nie może występować jako repre-
                zentacja kodu  */
'BCK.'      / * kropka nie jest reprezentacją kodu */
```

Dyrektywa REZE

Postać ogólna
REZE n gdzie: n - jest liczbą całkowitą

Dyrektywa powoduje zarezerwowanie w programie n słów. Rezerwacja następuje w miejscu jej wystąpienia. Zawartość zarezerwowanych miejsc pamięci jest nieokreślona.

Przykład

```
--------
--------
--------
ETYKIETA) REZE 6  / * ZAREZERWUJ W PROGRAMIE 6 MIEJSC */
--------
--------
--------
```

Adres

Adresy występują w rozkazach lub w makrorozkazach, ogólnie możemy podzielić je na adresy stałe i adresy zmienne.

Adres stały

Definicja
Adres stały - jest to suma algebraiczna liczb całkowitych i/lub liczb całkowitych skalowanych. Liczba skalowana - ma postać: a/b i jest to iloczyn liczby całkowitej a przez całkowitą potęgę dwóch o wykładniku 23-b.

Adres stały jest liczbą całkowitą, która może być składową adresu zmiennego lub też jest adresem rozkazu, lub parametrem makrorozkazu

Adres stały nie może być poprzedzony znakiem, a jego wartość powinna się mieścić w słowie maszyny. Ponadto wartości poszczególnych liczb skalowanych powinna mieścić się w słowie maszyny.

Dodatkowo, b powinno spełniać warunek:

$$0 < b \leqslant 23$$

Przykłady:

Poprawne adresy stałe:

4

7 - 2

12 - 1/21 (tj. $12 - 1x2^2 = 8$)

12/21 - 12/22-4 (tj. $12x2^2 - 12x2 - 4 = 20$)

Błędne adresy stałe:

+4	(poprzedzony znakiem)
3/24	(liczba skalująca większa od 23)
123/4	(wartość przekracza zakres słowa)
1/0	(wartość przekracza zakres słowa)
1/1+1/1	(wartość przekracza zakres słowa)

Adres zmienny

Postać ogólna

$$\left\{\begin{array}{l}\underline{\text{zmienna}}\\ \underline{\text{etykieta}}\\ \underline{\text{funkcja S}}\\ *\end{array}\right\}\left[\left\{\begin{array}{l}+\\ -\end{array}\right\}\ \underline{\text{as}}\right]$$

gdzie: zmienna - jest to nazwa zmiennej wymienionej na liście zdania JEZYK SAS,

etykieta - jest to nazwa etykiety języka SAS,

funkcja S- jest to nazwa funkcji bibliotecznej języka SAS poprzedzona kropką. Postać nazwy funkcji jest taka sama jak postać zmiennej a własność identyfikującą posiadają pierwsze trzy znaki,

as - jest to adres stały.

Adres zmienny jest liczony w sposób następujący:

- najpierw obliczana jest wartość adresu stałego wchodzącego w skład adresu zmiennego. Jeżeli adres stały nie występuje, to uważamy, że jest równy zero,
- następnie obliczana jest wartość pierwszego składnika adresu,
- wreszcie liczona jest suma algebraiczna w/w wartości.

Pierwszy składnik adresu zmiennego jest adresem miejsca pamięci.

Rozpatrzmy każdy z typów adresu zmiennego.

Zmienna

Nazwa zmiennej występująca w adresie może być aktualna lub formalna.

W przypadku zmiennej aktualnej lub bloku aktualnego nazwa reprezentuje sobą adres pamięci, w której znajduje się wartość danej zmiennej prostej lub wartość zerowego elementu bloku. Typ zmiennej (całkowita lub ułamkowa) nie jest uwzględniony. W przypadku zmiennej formalnej nazwa reprezentuje sobą adres miejsca pamięci przyporządkowanego formalnej nazwie. Pod wskazanym adresem może się znajdować:

- wartość zmiennej prostej podstawionej przez wartość,
- adres bloku podstawionego przez nazwę; z dostawionym bitem modyfikacji B-rejestrem,
- miejsce na posłanie wyniku dla zmiennej prostej z listy wyników podstawionej przez wartość.

Należy tu zauważyć, że nie możemy bezpośredno we wstawkach korzystać z funkcji języka SAKO oraz z podprogramów zdefiniowanych w tym języku. Metoda pośredniego odwołania zostanie podana w paragrafie opisującym budowę podprogramów.

Etykieta

Etykieta reprezentuje sobą adres miejsca pamięci do którego translator prześle instrukcję występującą bezpośrednio po wymienionej etykiecie. Np. etykieta napisana przed dyrektywą REZE wskazuje na pierwsze słowo zarezerwowane przy pomocy tej dyrektywy.

Funkcja S

Nazwa funkcji S, gdy wystąpi w adresie po raz pierwszy, powoduje dołączenie do rozdziału, wskazanego podprogramu.

W każdym przypadku nazwa funkcji reprezentuje sobą adres wejścia do podprogramu o wskazanej nazwie.

Gwiazdka

Gwiazdka reprezentuje sobą adres instrukcji, w której występuje.

Przykłady:

Poprawne adresy zmienne:

A
A+7+2/20
B-3
.AL+5
ETYKIETA-6
* + 4

Błędne adresy zmienne:

A+B (dwa składniki zmienne w jednym adresie)
3+A (niewłaściwa kolejność składników)
A.7 (kropka nie należy do nazwy)

Rozkaz

Postać ogólna
ope [adr] [+] [*] gdzie: ope jest trójznakowym skrótem nazwy części operacyjnej rozkazu maszyny ZAM-41 reprezentującym sobą wartość liczbową operacji; dopuszczalne skróty podane są w DODATKU E, adr jest adresem rozkazu tj. wyrażeniem opisującym wartość części adresowej rozkazu maszyny.

Rozkaz jest zapisem określającym słowo maszyny, które ma wejść do programu wynikowego. Skrót nazwy części operacyjnej określa jaka operacja ma być wykonana w programie wynikowym, adres podaje na jakim argumencie; znak +, jeżeli występuje, informuje, że rozkaz w trakcie wykonywania się będzie modyfikowany zawartością B-rejestru; gwiazdka, jeżeli występuje, oznacza pośrednie adresowanie. Z rozkazu tworzy się słowo maszyny przez sumowanie logiczne: wartości części operacyjnej rozkazu, bitów określających modyfikacje i wartości adresu.

W przypadku, gdy adres w rozkazie jest pominięty lub składa się z pojedynczej gwiazdki, rozkaz może przyjąć jedną z poniżej wymienionych postaci:

ope - rozkaz z częścią adresową równą zero,

ope * - rozkaz o adresie względnym równym adresowi tego rozkazu,

ope *+* } - rozkaz o adresie względnym modyfikowany zawartością B-rejestru i z pośrednim adresem
ope **+ }

ope * + - rozkaz o adresie względnym modyfikowany zawartością B-rejestru

ope + * - rozkaz o części adresowej równej zero z pośrednim adresem i modyfikowany zawartością B-rejestru.

ope + - rozkaz o części adresowej równej zero i modyfikowany zawartością B-rejestru.

W przypadkach, gdy występuje w rozkazie adres, kolejność występowania znaków modyfikacji (+, *) jest nieistotna.

Przykłady:

Poprawne rozkazy we wstawce języka SAS:

```
  UEB 71, PMB .FUS
ETYK)UMA Z + 2 - 4/20 +    / * rozkaz modyfikowany zawar-
                             tością B-rejestru */
  PZA A7 + 3
  DOA * -12                / * adres względny * /
  LAR   4
  UMM ETYKIETA             / * rozkaz równoważny jest:
                             UMM ETYK * /
  PMM  A1 *                / * rozkaz z pośrednim adreso-
                             waniem */
  SSL .SS1                 / * odwołanie do funkcji biblio-
                             tecznej języka SAS * /
```

Błędne użycie rozkazów we wstawce:

```
  UMB + 7                  / * stałe nie są automatycznie
                             listowane */
  UMB  '.3B1'              / * teksty nie mogą występować
                             w adresie, ponadto to nie
                             jest stała tekstowa */
```

Makrorozkaz

Postać ogólna
$\underline{n}$ ($\underline{a}_1$,$\underline{a}_2$,...,$\underline{an}$) gdzie: $\underline{n}$ - jest to nazwa makrorozkazu składająca się z co najwyżej dwudziestu liter i cyfr. Pierwszy znak nazwy musi być literą. $\underline{a}_1$,$\underline{a}_2$,...$\underline{an}$ - są to adresy będące argumentami makrorozkazu. Listę argumentów można przenosić do następnych wierszy zgodnie z regułami przenoszenia list w zdaniach języka SAKO.

Makrorozkaz jest zastępowany przez translator ciągiem rozkazów tzw. makrorozwinięciem wziętym z biblioteki funkcji języka SAS.

Parametry formalne makrorozwinięcia pobieranego z biblioteki są zastępowane aktualnymi adresami wziętymi z listy argumentów makrorozkazu. Część rozwinięć makrorozkazów posiada wewnętrzne odwołania do funkcji bibliotecznych. Funkcje te są automatycznie dołączane na końcu rozdziału.

Makrorozwinięcia i funkcje biblioteczne pisane są w pełnej reprezentacji języka M-SAS (a nie w języku wstawek) , jedynym ograniczeniem jest konieczność pisania programów bibliotecznych przy użyciu identyfikatorów nie dłuższych niż czteroznakowe. Dotychczas przy pisaniu funkcji bibliotecznych nie używano identyfikatorów dłuższych niż cztery znaki.

Przykłady:

Poprawne makrorozkazy:

```
CZYTAJ4 (A)
PISZ2 (ALFA, PR, POKR, SKAL)
DRUKUJ (A,B+,C*)
PISZ2 (A,B,-
       C,D)
```

Błędne makrorozkazy:

```
DRUKUJ (+1, +5, 'Z') /* argumentami nie są adresy */
DRUKUJ (A,B * - 4)   /* trzeci parametr jest adresem
                        względnym względem rozkazu
                        występującego w rozwinięciu
                        a nie względem makrorozkazu */.
```

Budowa podprogramów języka SAKO i sposób podstawienia argumentów

Wiedza o tym, jak się podstawia argumenty i gdzie posyłane są wyniki, jest potrzebna tylko w przypadku pisania podprogramów w języku wstawek SAS. Wiadomości te podamy na reprezentatywnym przykładzie.

Zdanie:

PODPROGRAM: (*A,B) = F(*C,D,E())

niezależnie od typów argumentów tłumaczone jest na następujący ciąg rozkazów:

```
        F') +0
          SKO * + 11
        C)+0, +0
        D)+0, +0
      E ')+0, +0
  F") A)  +0, +0
      B)  +0, +0
```

Etykiety zapisane tu symbolicznie jako F′, F″ i E′ są niedostępne w języku zewnętrznym i miejsca te można osiągnąć tylko wtedy, gdy znamy wzajemne położenie argumentów i wyników.

F′ - oznacza adres początku podprogramu o nazwie F

F″ - oznacza adres początku listy wników podprogramu o nazwie F

E′ - oznacza adres informacji o funkcji podstawionej przez nazwę.

Dla każdego parametru formalnego rezerwuje się, niezależnie od jego typu, dwa słowa, w kolejności występowania parametrów, najpierw z listy argumentów a następnie z listy wyników.

Załóżmy teraz, że D1 = 12 i że nastąpiło odwołanie do naszego podprogramu za pośrednictwem zdania:

```
(*A1,B1) = F (*C1,D1,E1())
```

wtedy, po podstawieniu, omawiany początek programu będzie miał postać:

```
F′)SKO     F″ *
   SKO     * + 11
 C)SKO     C1+,    +0
 D)+12,    +0                / * DLA D1 CALKOWITEGO LUB
                                 D) +12. DLA D1
                                 ULAMKOWEGO * /
E′)SKO E1′+, SKO E1″ +
                             / * PODSTAWIONO ADRES
                                 FUNKCJI I ADRES LISTY
                                 WYNIKOW * /
F″ A) SKO A1 +, +0
   B  +0, +0                 / * B1 NIE JEST PODSTAWIO-
                                 NE * /
```

Wartość zmiennej formalnej B zostanie przesłana do B1 w momencie powrotu do programu głównego.
Powrót ma nastąpić w miejsce F".

W przykładzie na poprzedniej stronie rozważaliśmy podstawienie do podprogramu argumentów aktualnych. Podstawianie argumentów formalnych jako aktualnych odbywa się metodą bezpośredniego przenoszenia odpowiednich miejsc pamięci z jednego podprogramu do następnego.

Aby nadać wartość dla funkcji, należy przed powrotem do programu głównego, umieścić w rejestrze A liczbę całkowitą dla funkcji całkowitej lub w rejestrze A,M liczbę ułamkową dla funkcji ułamkowej.

Przykład

Załóżmy, że chcemy wypisać na wyjście adres pamięciowy początku podprogramu realizującego funkcję SIN.
Zadanie to możemy zrealizować w sposób następujący:

```
CALKOWITE : A
BLOK (0) : A
(*A) = P (SIN())
LINIA
DRUKUJ (5) : A
LINIA
STOP NASTEPNY
PODPROGRAM: (*A) = P (SI())
JEZYK SAS : A
UMA  A - 2
MNL  B
UEB  0
PZA  A*
WRO  A-4
B) = 7777
JEZYK SAKO
KONIEC
```

Biblioteka programów i podprogramów

Niżej podano formy makrorozkazów używanych przy wywołaniu odpowiedniego podprogramu dołączonego do systemu. Szczegółowy opis poszczególnych zmiennych, a także opis metody, na której dany podprogram oparto, znaleźć można w opracowaniu: "ZAM-41. Oprogramowanie. Część III - BIBLIOTEKA ".

Obliczanie funkcji trygonometrycznych: sinus, cosinus, tangens, cotangens

SCT (ARG, WYNIK, K)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

K = { 0 dla sinusa; 4 dla cosinusa; 10 dla tangensa; 16 dla cotangensa }

w makrorozkazie należy podać adres lub adres adresu wartości K

Obliczanie wartości funkcji arcsin x

ARS (ARG, WYNIK, Q)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

Q - adres lub adres adresu, pod który podprogram wraca w przypadku błędu

Obliczanie wartości funkcji arctg x

ART (ARG, WYNIK)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

Obliczanie wartości funkcji e^x i 2^x

EXP (ARG, WYNIK, K)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

$$K = \begin{cases} 0 \text{ dla } e^x \\ 4 \text{ dla } 2^x \end{cases}$$

Należy podać adres lub adres adresu wartości K

Obliczanie wartości funkcji $\ln x$

LNX (ARG, Q, WYNIK)

gdzie:

ARG - adres lub adres adresu argumentu

Q - adres lub adres adresu, pod który podprogram wraca w przypadku błędu

WYNIK - adres lub adres adresu wyniku

Obliczanie pierwiastków drugiego i trzeciego stopnia

PKS (ARG, WYNIK, Q, K)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

Q - adres lub adres adresu, pod który wraca podprogram w przypadku błędu

$$K = \begin{cases} 0 \text{ dla pierwiastka 2-go stopnia} \\ 4 \text{ dla pierwiastka 3-go stopnia} \end{cases}$$

Należy podać adres lub adres adresu wartości K

Obliczanie pierwiastków n-tego stopnia

PWN (ARG, WYNIK, N, Q)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

N - adres lub adres adresu liczby będącej stopniem pierwiastka

Q - adres lub adres adresu, pod który podprogram wraca w przypadku błędu

Funkcja Gamma

GAM (ARG, WYNIK)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

Obliczanie wartości funkcji $\gamma(x+1)$

GAX (ARG, WYNIK Q)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

Q - adres lub adres adresu, pod który podprogram wraca w przypadku błędu

Obliczanie wartości funkcji Bessel'a

BES (ARG, WYNIK, N, Q)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

N - adres lub adres adresu liczby (N=1,2,3,4) wskazującej którą z funkcji Bessel'a podprogram ma obliczyć

Q - adres lub adres adresu, pod który podprogram wraca w przypadku błędu

Obliczanie wartości funkcji $E(x) = -E_i(-x) = \int_x^{\infty} \frac{e^{-u}}{u} du$

EIX (ARG, WYNIK, Q)

gdzie:

ARG - adres lub adres adresu argumentu

WYNIK - adres lub adres adresu wyniku

Q - adres lub adres adresu, pod który podprogram wraca w przypadku błędu

Działania na liczbach zespolonych

DLZ (N, A, B, C, INF)

gdzie:

N - adres lub adres adresu liczby będącej numerem żądanego działania (N=1,2,3,4,5,6,7)

A - adres lub adres adresu pierwszego argumentu działania

B - adres lub adres adresu drugiego argumentu działania

C - adres lub adres adresu wyniku

A,B,C mogą być modyfikowane B-rejestrem

INF - adres lub adres adresu, pod który podprogram ma przesłać informacje o wykonalności działania

Sinus i cosinus argumentu zespolonego

SIZ (A,K)

gdzie:

A - adres lub adres adresu argumentu

$$K = \begin{cases} 0 \text{ dla } \sin z \\ 4 \text{ dla } \cos z \end{cases}$$

Należy podać adres lub adres adresu wartości K.

Adres A może być modyfikowany B-rejestrem.

Arcussinus i arcuscosinus argumentu zespolonego

ASZ (A, K)

gdzie:

A - adres lub adres adresu argumentu

$$K = \begin{cases} 0 \text{ dla } \arcsin z \\ 4 \text{ dla } \arccos z \end{cases}$$

Należy podać adres lub adres adresu wartości K.

Adres A może być modyfikowany B-rejestrem.

Eksponent argumentu zespolonego

EXZ (A)

gdzie:

A - adres lub adres adresu argumentu

Adres A może być modyfikowany B-rejestrem

Logarytm argumentu zespolonego

LNZ (A,Q)

gdzie:

A - adres lub adres adresu argumentu

Q - adres lub adres adresu, pod który podprogram wraca w przypadku zerowego argumentu

Adres A może być modyfikowany B-rejestrem

Dzielenie wielomianów

DZW (A,B,C,N,M, INF)

gdzie:

A - adres lub adres adresu współczynnika przy najwyższej potędze dzielnej

B - adres lub adres adresu współczynnika przy najwyższej potędzie dzielnika

C - adres lub adres adresu współczynnika przy najwyższej potędzie ilorazu

N - adres lub adres adresu liczby będącej stopniem dzielnej

M - adres lub adres adresu liczby będącej stopniem dzielnika

INF- adres lub adres adresu, pod który podprogram ma przesłać informację o wykonalności dzielenia wielomianów

Obliczanie współczynników wielomianu

AAZ (N,A,B,L)

gdzie:

N - adres lub adres adresu liczby będącej stopniem wielomianu

A - adres lub adres adresu pierwiastków wielomianu

B - adres lub adres adresu współczynników wielomianu

L - adres lub adres adresu linii roboczej

Adresy A, B, i L mogą być modyfikowane B-rejestrem

Znajdowanie pierwiastków wielomianu

NEW (N,EPS,A,B,X,INF)

gdzie:

N - adres lub adres adresu liczby oznaczającej stopień wielomianu

EPS- adres lub adres adresu żądanej dokładności

A - adres lub adres adresu współczynników wielomianu

B - adres lub adres adresu linii roboczej

X - adres lub adres adresu pierwiastków wielomianu

INF- adres lub adres adresu informacji o ilości policzonych pierwiastków

Znajdowanie pierwiastków wielomianu

FAL (N,EPS,H,A,X,INF)

gdzie:

N - adres lub adres adresu liczby oznaczającej stopień wielomianu

EPS- adres lub adres adresu żądanej dokładności

H - adres lub adres adresu parametru h

A - adres lub adres adresu współczynników wielomianu

X - adres lub adres adresu pierwiastków wielomianu

INF- adres lub adres adresu informacji o ilości policzonych pierwiastków

Znajdowanie rzeczywistych i zespolonych pierwiastków wielomianu

BRS (N,A,X,EPS,S,P,INF)

gdzie:

N - adres lub adres adresu liczby oznaczającej stopień wielomianu

A - adres lub adres adresu współczynnika przy najwyższej potędze wielomianu

X - adres lub adres adresu pierwiastków wielomianu

EPS- adres lub adres adresu żądanej dokładności

S - adres lub adres adresu parametru s

P - adres lub adres adresu parametru p

INF- adres lub adres adresu informacji o ilości policzonych pierwiastków

Obliczanie pierwiastka wielomianu metodą Bernoulli'ego

BER (N,EPS,A,INF)

gdzie:

N - adres lub adres adresu liczby oznaczającej stopień wielomianu

EPS- adres lub adres adresu żądanej dokładności

A - adres lub adres adresu współczynników wielomianu

INF- adres lub adres adresu policzonego pierwiastka lub zera zmiennoprzecinkowego w przypadku gdy pierwiastek nie został policzony

Znajdowanie pierwiastków wielomianu o współczynnikach zespolonych

MUL (N,Z,EPS,A,W,INF)

gdzie:

N - adres lub adres adresu liczby oznaczającej stopień wielomianu

Z - adres lub adres adresu wartości startowych zmiennej

EPS- adres lub adres adresu żądanej dokładności
A - adres lub adres adresu współczynnika przy najwyższej potędze wielomianu
W - adres lub adres adresu pierwiastków wielomianu
INF- adres lub adres adresu informacji o ilości policzonych pierwiastków

Mnożenie_macierzy - wersja_PAO_

MXM (N,M,L,AA,AB,AC)

gdzie:
N - adres lub adres adresu liczby będącej ilością wierszy macierzy A
M - adres lub adres adresu liczby będącej ilością kolumn macierzy A
L - adres lub adres adresu liczby będącej ilością kolumn macierzy B
AA - adres lub adres adresu pierwszego elementu macierzy A
AB - adres lub adres adresu pierwszego elementu macierzy B
AC - adres lub adres adresu pierwszego elementu macierzy C

Adresy AA, AB, AC dotyczą PAO

Mnożenie_macierzy - wersja_bębnowa_

MBM (N,M,L,AA,AB,AC,LROB)

gdzie:
N - adres lub adres adresu liczby będącej ilością wierszy macierzy A
M - adres lub adres adresu liczby będącej ilością kolumn macierzy A
L - adres lub adres adresu liczby będącej ilością kolumn macierzy B
AA - adres rzeczywisty lub adres adresu pierwszego elementu macierzy A

AB - adres rzeczywisty lub adres adresu pierwszego elementu macierzy B

AC - adres rzeczywisty lub adres adresu pierwszego elementu macierzy C

LROB- adres lub adres adresu linii roboczej

Adresy AA,AB,AC dotyczą bębna, LROB dotyczy PAO

Dodawanie, odejmowanie macierzy, mnożenie macierzy przez liczbę

PPM (N,M,L,BA,BW,BB,K,LICZ)

gdzie:

N - adres lub adres adresu liczby będącej ilością wierszy macierzy

M - adres lub adres adresu liczby będącej ilością kolumn macierzy

L - adres lub adres adresu linii roboczej

BA - adres rzeczywisty lub adres adresu pierwszego elementu macierzy A

BW - adres rzeczywisty lub adres adresu pierwszego elementu macierzy C

BB - adres rzeczywisty lub adres adresu pierwszego elementu macierzy B

K= { 0 dla mnożenia przez liczbę
5 dla dodawania macierzy
11 dla odejmowania macierzy

LICZ- adres lub adres adresu liczby, przez którą się mnoży macierz

Należy podać adres lub adres adresu wartości K

Adresy BA, BW, BB dotyczą bębna, L i LICZ dotyczą PAO

Transponowanie_macierzy - wersja_PAO_

TNN (N,M,AA,AB,L)

gdzie:

N - adres lub adres adresu liczby będącej ilością wierszy macierzy A

M - adres lub adres adresu liczby będącej ilością kolumn macierzy A

AA - adres rzeczywisty lub adres adresu pierwszego elementu macierzy A

AB - adres rzeczywisty lub adres adresu pierwszego elementu macierzy transponowanej

L - adres lub adres adresu linii roboczej

Adresy AA i AB dotyczą bębna, adres L dotyczy PAO

Transponowanie_macierzy - wersja_bębnowa_

TNM (N,M,A,L)

gdzie:

N - adres lub adres adresu liczby będącej ilością wierszy macierzy

M - adres lub adres adresu liczby będącej ilością kolumn macierzy

A - adres lub adres adresu pierwszego elementu macierzy

L - adres lub adres adresu linii roboczej

Wszystkie adresy dotyczą PAO

Transformacja macierzy_

GVN (N,AA,AT,L,Z)

gdzie:

N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

AT - adres rzeczywisty lub adres adresu początku macierzy T

L - adres lub adres adresu początku linii roboczej

Z - adres lub adres adresu liczby będącej informacją o liczeniu macierzy przekształcenia T (Z=0 lub 1)

Adresy AA i AT dotyczą bębna, pozostałe dotyczą PAO

Transformacja macierzy symetrycznej

GVS (N,AA,AT,L,Z)

gdzie:

N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

AT - adres rzeczywisty lub adres adresu początku macierzy T

L - adres lub adres adresu początku linii roboczej

Z - adres lub adres adresu liczby będącej informacją o liczeniu macierzy przekształcenia T (Z=0 lub 1)

Adresy AA i AT dotyczą bębna, pozostałe dotyczą PAO

Obliczanie wartości całki metodą Gaussa-Legandre'a

CGL (N,A,H,PRA,WYNIK)

gdzie:

N - adres lub adres adresu liczby oznaczającej ilość podprzedziałów przedziału całkowania

A - adres lub adres adresu dolnej granicy przedziału całkowania

H - adres lub adres adresu kroku całkowania

PRA - adres lub adres adresu podprogramu obliczającego wartość funkcji podcałkowej

WYNIK- adres lub adres adresu wyniku

Obliczanie wartości całki $\int_A^B f(x)dx$ metodą Simpsona

SMP (A,B,EPS,K,WYNIK,PRA)

gdzie:

A - adres lub adres adresu dolnej granicy całkowania

B - adres lub adres adresu górnej granicy całkowania

EPS - adres lub adres adresu żądanej dokładności liczenia wartości całki

K - adres lub adres adresu minimalnej ilości podziałów przedziału całkowania

WYNIK- adres lub adres adresu wyniku

PRA - adres lub adres adresu podprogramu liczącego wartość funkcji podcałkowej

Obliczanie wartości całki metodą Romberga

ROM (A,B,EPS,T,N,WYNIK,PRA)

gdzie:

A - adres lub adres adresu dolnej granicy całkowania

B - adres lub adres adresu górnej granicy całkowania

EPS - adres lub adres adresu żądanej dokładności liczenia wartości całki

T - adres lub adres adresu początku linii roboczej

N - adres lub adres adresu liczby wyznaczającej największą ilość podziałów przedziału całkowania

WYNIK- adres lub adres adresu wyniku

PRA - adres lub adres adresu podprogramu liczącego wartość funkcji podcałkowej

Rozwiązywanie układu równań różniczkowych
metodą Runge-Kutty - Gilla

RUK (N,H,X,PRA)

gdzie:

N - adres lub adres adresu liczby oznaczającej ilość równań układu

H - adres lub adres adresu kroku całkowania

X - adres lub adres adresu bloku zmiennych

PRA - adres lub adres adresu podprogramu liczącego prawe strony

Rozwiązywanie układu równań różniczkowych zwyczajnych
metoda Runge-Kutty-Gilla wersja bębnowa

RUB (N,H,X,PRA)

gdzie:

N - adres lub adres adresu liczby oznaczającej ilość równań układu

H - adres lub adres adresu kroku całkowania

X - adres rzeczywisty lub adres adresu bloku zmiennych

PRA - adres lub adres adresu podprogramu liczącego prawe strony

Adres X dotyczy bębna

Rozwiązywanie układu równań różniczkowych zwyczajnych
(metoda Adamsa)

ADS (ARG,Q,PRA, N)

gdzie:

ARG - adres lub adres adresu kroku całkowania

Q - adres lub adres adresu początku tablicy

PRA - adres lub adres adresu podprogramu obliczającego prawe strony

N - adres lub adres adresu liczby oznaczającej ilość równań układu

Dystrybuanta rozkładu normalnego

DRN (X,MI,SIG,WYNIK)

gdzie:

X - adres lub adres adresu wartości X

MI - adres lub adres adresu wartości μ

SIG - adres lub adres adresu wartości σ

WYNIK- adres lub adres adresu wyniku

Wykreślanie funkcji jednej zmiennej niezależnej

RYS (NR,N,K,M,Y,X,H,Z)

gdzie:

- NR - adres lub adres adresu liczby oznaczającej umowny numer wykresu
- N - adres lub adres adresu liczby oznaczającej ilość podprzedziałów w przedziale R
- K - adres lub adres adresu liczby oznaczającej ilość funkcji
- M - adres lub adres adresu liczby oznaczającej ilość wartości każdej z funkcji
- Y - adres lub adres adresu pierwszego elementu macierzy tablicy Y
- X - adres lub adres adresu początkowej wartości zmiennej niezależnej
- H - adres lub adres adresu przyrostu zmiennej niezależnej
- Z - adres lub adres adresu pierwszego elementu tablicy liczb, będących kodami znaków

Równania algebraiczne liniowe
(metoda Gausa bez wyboru największego elementu)

GBW (N,M,AA,L,INF)

gdzie:

N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

M - adres lub adres adresu liczby oznaczającej ilość kolumn macierzy B

L - adres lub adres adresu początku linii roboczej

INF - adres lub adres adresu informacji o istnieniu rozwiązania układu

Adres AA dotyczy bębna, pozostałe dotyczą PAO

Obliczanie wartości wyznacznika
(metoda Gaussa z wyborem największego elementu)

WAN (WYNIK,N,AA,L)

gdzie:

WYNIK- adres lub adres adresu wyniku

N - adres lub adres adresu liczby będącej wymiarem macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

L - adres lub adres adresu początku linii roboczej

Adres AA dotyczy bębna, pozostałe dotyczą PAO

Odwracanie macierzy

POM (N,AA,L,INF)

gdzie:

N - adres lub adres adresu liczby będącej wymiarem macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

L - adres lub adres adresu początku linii roboczej

INF - adres lub adres adresu informacji o istnieniu macierzy odwrotnej

Adres AA dotyczy bębna, pozostałe dotyczą PAO

Równania algebraiczne liniowe

GJR (N,M,AA,L,INF)

gdzie:

N - adres lub adres adresu liczby będącej wymiarem macierzy A

M - adres lub adres adresu liczby będącej ilością kolumn macierzy B

AA - adres rzeczywisty lub adres adresu początku macierzy A

L - adres lub adres adresu początku linii roboczej

INF - adres lub adres adresu informacji o istnieniu rozwiązania układu

Adres AA dotyczy bębna, pozostałe dotyczą PAO

Równania algebraiczne liniowe
(metoda Gaussa z wyborem największego elementu)

GWP (N,M,L,INF)

gdzie:

N - adres lub adres adresu liczby będącej wymiarem macierzy A

M - adres lub adres adresu liczby będącej ilością kolumn macierzy B

L - adres lub adres adresu początku macierzy A

INF - adres lub adres adresu informacji

Wszystkie adresy dotyczą PAO

Równania algebraiczne liniowe - wersja bębnowa
(metoda Gaussa z wyborem największego elementu)

GZW (N,M,AA,L,INF)

gdzie:

N - adres lub adres adresu liczby będącej wymiarem macierzy A

M - adres lub adres adresu liczby będącej ilością kolumn macierzy B

AA - adres rzeczywisty lub adres adresu początku macierzy A

L - adres lub adres adresu początku linii roboczej

INF - adres lub adres adresu informacji o istnieniu rozwiązania układu

Adres AA dotyczy bębna, pozostałe dotyczą PAO

Rozwiązywanie układu równań algebraicznych liniowych

HRN (N,M,AA,L,INF)

gdzie:

N - adres lub adres adresu liczby będącej wymiarem macierzy A

M - adres lub adres adresu liczby będącej ilością kolumn macierzy B

AA - adres rzeczywisty lub adres adresu początku macierzy A

L - adres lub adres adresu początku linii roboczej

INF - adres lub adres adresu informacji o istnieniu rozwiązania układu

Adres AA dotyczy bębna, pozostałe adresy dotyczą PAO

Rozwiązywanie układu równań algebraicznych liniowych o macierzy prawieprzekątniowej

GPD (N,D,M,AA,L,INF)

gdzie:

N - adres lub adres adresu liczby będącej wymiarem macierzy A

D - adres lub adres adresu liczby będącej ilością przekątnych macierzy A

M - adres lub adres adresu liczby będącej ilością kolumn macierzy B

AA - adres rzeczywisty lub adres adresu początku macierzy A

L - adres lub adres adresu początku linii roboczej

INF - adres lub adres adresu informacji o istnieniu rozwiązania układu

Adres AA dotyczy bębna, pozostałe adresy dotyczą PAO

Znajdowanie wektorów własnych i wartości własnych

JAC (N,AA,AU,L,Z,EPS)

gdzie: N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

AU - adres rzeczywisty lub adres adresu początku macierzy U

L - adres lub adres adresu początku linii roboczej

Z - adres lub adres adresu liczby będącej informacją o liczeniu wektorów własnych /Z=0,lub 1/

EPS - adres lub adres adresu żądanej dokładności

Adresy AA i AU dotyczą bębna, pozostałe dotyczą PAO

Wartości własne macierzy trójdiagonalnych symetrycznych

LRS (N,AA,EPS)

gdzie: N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

AA - adres lub adres adresu początku macierzy A

EPS - adres lub adres adresu żądanej dokładności

Wszystkie adresy dotyczą PAO

Wartości własne macierzy podtrójkątnych

LRN (N,AA,W,L,EPS)

gdzie: N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

W - adres rzeczywisty lub adres adresu wartości własnych macierzy A

L - adres lub adres adresu początku linii roboczej

EPS - adres lub adres adresu żądanej dokładności

Adresy AA i W dotyczą bębna, pozostałe dotyczą PAO

Obliczanie rzeczywistych i zespolonych wartości własnych macierzy podtrójkątnych

QRN (N,AA,W,L,EPS)

gdzie:

N - adres lub adres adresu liczby oznaczającej wymiar macierzy A

AA - adres rzeczywisty lub adres adresu początku macierzy A

W - adres rzeczywisty lub adres adresu wartości własnych macierzy A

L - adres lub adres adresu początku linii roboczej

EPS - adres lub adres adresu żądanej dokładności

Adresy AA i W dotyczą bębna, pozostałe dotyczą PAO

Aproksymacja funkcji jednej zmiennej

APX (P,N,X)

gdzie:

P - adres lub adres adresu liczby będącej stopniem wielomianu

N - adres lub adres adresu liczby będącej ilością par (X_i, Y_i)

X - adres lub adres adresu linii roboczej

Aproksymacja funkcji dwu zmiennych

APP (P,N,X,B,INF)

gdzie:

P - adres lub adres adresu liczby będącej stopniem wielomianu

N - adres lub adres adresu liczby będącej ilością trójek (X_i, Y_i, Z_i)

X - adres lub adres adresu linii roboczej w PAO

B - adres rzeczywisty lub adres adresu linii roboczej na bębnie

INF - adres lub adres adresu informacji o istnieniu rozwiązania

Rozwiązanie prostego zagadnienia simplexu

PZS (N,M,AT,AC,AW,AP,AB)

gdzie: N - adres lub adres adresu liczby będącej ilością kolumn macierzy A

M - adres lub adres adresu liczby będącej ilością wierszy macierzy A

AT - adres rzeczywisty lub adres adresu początku macierzy A

AC - adres rzeczywisty lub adres adresu wektora C

AW - adres rzeczywisty lub adres adresu wektora X

AP - adres lub adres adresu początku linii roboczej

AB - adres lub adres adresu wskaźników bazy wynikowej

Adresy AT,AC,AW dotyczą bębna, pozostałe adresy dotyczą PAO.

U W A G A:

1. Nazwy makrorozkazów są identyczne z nazwami odpowiednich podprogramów bibliotecznych.

DODATEK G

Błędy etapu translacji

W procesie translacji sprawdzana jest syntaktyczna poprawność tłumaczonego programu. W przypadku wystąpienia błędu, translator wypisuje informacje o nim, na zerowe symboliczne wyjście translatorów.

Opisy wykrytych błędów grupowane są oddzielnie dla każdego rozdziału i poprzedzone zwrotem: ROZDZIAL n (n-numer rozdziału). W ramach rozdziału, kolejność sygnalizacji błędów nie jest uporządkowana tj. kolejność sygnalizacji błędów nie musi odpowiadać kolejności występowania ich w programie.

Przy każdym błędzie podawane są cztery informacje (po dwie z każdej strony słowa BLAD):

1. etykieta, po której wystąpił błąd. Jeżeli etykieta w programie lub podprogramie jeszcze nie wystąpiła, to zamiast etykiety pisany jest identyfikator POCZ;
2. względny (względem ostatniej etykiety) numer wiersza, w którym znaleziony został błąd. Liczone są tylko wiersze niepuste (liczone są również komentarze); wierszowi, w którym występuje etykieta, przyporządkowany jest numer zero;

   Uwaga! W tekście będącym parametrem zdania TEKST lub TEKST WIERSZY liczone są również wiersze puste;
3. po słowie BLAD wypisany jest numer błędu; spis numerów błędów i ich znaczenia podajemy poniżej;
4. nazwa podprogramu-poprzedzona gwiazdką (lub funkcji M-SAS poprzedzona kreską (/)) - w którym znaleziono błąd. Jeżeli błąd został wykryty w programie głównym lub czołówce języka SAKO to zamiast nazwy podprogramu pisany jest odpowiednio identyfikator GLOW lub CZOL.

W ciągu sygnalizacji, pozycja czwarta jest wypełniana tylko wtedy, gdy to jest konieczne.

Pewne błędy mogą powodować pojawienie się innych błędów (np. błąd w zwrocie CALKOWITE spowoduje, że zdanie nie zostanie rozpoznane, co może pociągnąć sygnalizację związaną z niewłaściwymi typami wyrażeń), tzn. nie zawsze jednemu błędowi programisty odpowiada jedna informacja opisująca błąd.

Spis błędów

1. Wiersz pobrany przez translator zawiera więcej niż 120 znaków
2. Wiersz pobrany przez translator nie został rozpoznany jako zdanie języka SAKO; separatorem w SKOCZ WEDLUG nie jest przecinek; element zdania nie jest etykietą mimo, że być nią powinien
3. W procesie translacji zdania TABLICA natrafiono na informację nie należącą do odpowiedniej klasy danych
4. Liczba występująca w języku SAKO nie jest poprawna (dotyczy to również danych zdania TABLICA)
5. Nie rozpoznano końca programu
6. Brak taśmy magnetycznej przydzielonej dla realizacji zdania BIBLIOTEKA
7. Zdanie rozpoznane jako TEKST WIERSZY posiada niepoprawny parametr
8. Zdanie rozpoznane jako ROZDZIAL posiada niepoprawny parametr
9. Wielkość rozpoznana jako liczba oktalna zawiera więcej niż 16 cyfr lub posiada cyfrę 8 lub 9
10. Translator natrafił na zdanie POWTORZ dla którego nie wystąpiła gwiazdka
11. Zdanie rozpoznane jako TABLICA jest niepoprawnie zbudowane
12. Pobrana informacja nie jest nazwą części operacyjnej rozkazu we wstawce języka SAS

13. Pobrana informacja jest niepoprawną częścią adresową rozkazu we wstawce języka SAS

14. We wstawce języka SAS pobrany wiersz nie został rozpoznany przez translator

15. Informacja rozpoznana jako tekst we wstawce języka SAS jest zbyt długa lub zawiera niedozwolone znaki

16. Rozpoznana informacja nie jest poprawną makrooperacją

17. Tekst pobierany z taśmy magnetycznej zawiera niepoprawną nazwę strony

18. Tekst o zadanej nazwie nie został odnaleziony na taśmie bibliotecznej

19. Błąd odczytu bibliotecznej taśmy magnetycznej

20. Błędna składnia zdania

21. Zbyt dużo miejsc zajmuje zdanie BLOK lub TABLICA

22. Zabrakło translatorowi miejsca (w pamięci operacyjnej) na listy identyfikatorów użytych w zdaniach: BLOK, STRUKTURA, CALKOWITE, POWTORZ

23. Argument jest całkowity, a powinien być ułamkowy

24. Niewłaściwa ilość indeksów zmiennej indeksowanej

25. Przy liczeniu wyrażenia całkowitego przekroczono zakres liczb całkowitych

26. Wystąpiło w programie więcej gwiazdek niż odpowiadających im rozkazów POWTORZ

27. Błędny zakres bloku lub tablicy

28. Ilość elementów TABLICY nie jest zgodna z ilością zadeklarowną

29. Wystąpiła deklaracja JEZYK SAKO w programie napisanym w języku SAKO

30. Typ wyrażenia jest ułamkowy, a powinien być całkowity

31. Wszystkie parametry POWTORZ powinny być tego samego typu

32. Wystąpiło zdanie WROC w programie głównym

33. Niezadeklarowany BLOK lub STRUKTURA

34. Zdanie typu F( ) = wystąpiło w programie głównym

35. Za duży parametr zdania KONIEC lub IDZ DO ROZDZIAŁU

36. W tablicy typu całkowitego wystąpiła stałe typu ułamkowego

37. Element bloku formalnego wystąpił bez indeksów

38. Ułamkowa liczba przesunięć w wyrażeniu logicznym

39. Brak nawiasu zamykającego w wyrażeniu arytmetycznym lub logicznym

40., 41., 42. } Błędy związane z pobieraniem funkcji języka i nie powinny pojawiać się przy poprawnej pracy translatora i maszyny

43. Funkcje języka nie mieszczą się w pamięci operacyjnej, należy powtórzyć translację w zestawie z większą pamięcią operacyjną

44. Znaczenie takie jak numerów 40-42

46,49,51, 52,53,55, 56 } Błędna funkcja w bibliotece M-SAS lub błąd maszyny

47. Błędna składnia makrooperacji

48. Brak makrooperacji w bibliotece funkcji M-SAS

50. Za mało argumentów w makrooperacji

54. W bibliotece M-SAS brak żądanej funkcji

57. Błędna lista parametrów zdania JEZYK SAS

70. Przy prawidłowej pracy translatora i maszyny ten błąd nie powinien się pojawić

71. W tłumaczonym rozdziale użyto zbyt dużo zmiennych całkowitych, dalsze zmienne całkowite będą uznawane jako ułamkowe, translacja przebiega dalej

72. Brak miejsca w translatorze na listę identyfikatorów i listę stałych

73., 74. } Błędna funkcja w bibliotece M-SAS lub błąd maszyny informacja wiąże się z blokową strukturą funkcji M-SAS

75. Etykieta poprzedzająca zdanie już poprzednio wystąpiła przez innym zdaniem

76. Już wystąpiło zdanie PODPROGRAM ze wskazaną nazwą funkcji

77. Nazwa tablicy użyta w zdaniu TABLICA była już użyta w programie

78. Wcześniej wystąpiła nazwa, która obecnie jest deklarowana jako blok

79., 80., 81. Błędy związane ze złym uszeregowaniem listy funkcji języka znajdującej się w translatorze, mogą powstać tylko przy dołączeniu nowych funkcji języka

82,83, 84,85, Błędy te nie mogą się pojawić przy prawidłowej pracy translatora i maszyny

89. Program zawiera więcej rozdziałów niż przewiduje translator. Słowniki liczone są jako rozdziały

90. - patrz znaczenie błędów o numerze 93-94

91. Przepełniono taśmę magnetyczną z programami binarnymi

92. Brak pamięci dla translatora przy dołączeniu funkcji języka

93,94, Przy poprawnej pracy maszyny i translatora błąd nie powinien wystąpić

95. Etykieta użyta w zdaniu nie poprzedza żadnego ze zdań (brak etykiety)

96. Brak miejsca na liście stałych, powtórzyć translację w zestawie z większą pamięcią operacyjną

97. Błąd nie powinien wystąpić przy prawidłowej pracy maszyny

98. Brak deklaracji BLOK lub PODPROGRAM

99. Program wynikowy nie zmieści się w zadanej pamięci operacyjnej

100. Zabrakło miejsca na bębnie, translator nie może kontynuować pracy, zwiększyć pojemność bębna przez inne wyadaptowanie systemu lub przeprowadzać translację na taśmę magnetyczną

101. Nazwa podprogramu jest identyczna z nazwą funkcji języka

102. Po lewej stronie formuły adresowej znajduje się adres stały

200. Wskazany wiersz nie jest poprawnym zdaniem czołówki

201. Źle zakończony wiersz zdania czołówki, np. dwa teksty nałożnne na siebie

202. Wystąpiły zbędne znaki po końcu zdania czołówki

203. Błędny parametr w zdaniu czołówki

204. Koniec taśmy w tekście czołówki (nadmiar wejścia)

205. Po zdaniu kończącym czołówkę nie wystąpił znak LP

206. Niewłaściwy numer w zdaniu KLUCZE

207. Program wynikowy pobrany z taśmy magnetycznej miał błędy formalne

208. Niewłaściwa taśma magnetyczna z programem wynikowym

209. Błąd maszyny

210. Program wynikowy pobrany z taśmy magnetycznej nie mieści się na bębnie lub wolny bęben jest zbyt mały

211. Zbyt mała pamięć operacyjna dla programu wynikowego pobieranego z taśmy magnetycznej

212.
213. } Błędy zapisu na taśmie magnetycznej w procesie translacji na taśmę

214. Sekcja startowa programu wynikowego nie daje się odczytać z taśmy magnetycznej (uszkodzona taśma)

215. Błędny odczyt taśmy magnetycznej w trakcie pobierania programu wynikowego na odcinkach nie istotnych w tym konkretnym zadaniu (jest to sygnalizacja ostrzegawcza i nie liczy się jako błąd niedopuszczający programu do wykonania)

216. Ma znaczenie podobne jak numer 215

217. Nie daje się odczytać sekcja startowa programu wynikowego zapisanego na taśmie magnetycznej (niewłaściwa taśma)

218. Błędny numer rozdziału lub słownika symboli na liście parametrów zdania POBIERZ

219. Rozdział lub słownik o podanym numerze nie istnieje na taśmie magnetycznej

220. Znaleziony rozdział nie daje się odczytać z taśmy magnetycznej

221. Wystąpiło zdanie czołówki KONIEC a poprzednie zdanie w problemie nie było wywołaniem translatora SAKO

222. Zbyt duża sekcja startowa, przy poprawnej pracy translatora i maszyny błąd nie powinien wystąpić

223. W trakcie minimalizacji bębna obszar wolnego bębna jest powiększony (sygnał ostrzegawczy, patrz 215)

224. Wymagania na pamięć bębnową, postawione przy pomocy zdania WYMAGANY BEBEN, nie mogą być spełnione

225. W czołówce zakończonej zdaniem KONIEC próbowano zmniejszyć obszar pamięci przeznaczony dla programu

226. Rozdział programu wynikowego pobierany z taśmy magnetycznej nie mieści się na bębnie, pobranie nie następuje (sygnał ostrzegawczy patrz 215)

227. Błędne parametry zdania POBIERZ lub ZAPISZ

228. Zestaw maszyny użytkowej nie posiada taśmy magnetycznej o nazwie wskazanej przez ZAPISZ lub POBIERZ

229. Nie udało się otworzyć taśmy magnetycznej dla POBIERZ, ZAPISZ lub BIBLIOTEKA

230. Otwarto taśmę do czytania a powinna być otwarta do pisania (dla zdania ZAPISZ)

231. Napotkano niewłaściwy numer szpuli na taśmie magnetycznej przewidzianej dla: ZAPISZ, POBIERZ lub BIBLIOTEKA

232. Przy ZAPISZ na końcu biblioteki nie znaleziono końca

233. Przy ZAPISZ od miejsca o wskazanej nazwie - nazwy nie znaleziono

234. Przy POBIERZ znaleziono program o zadanej nazwie, ale nie znaleziono jego sekcji startowej

235. Nazwy stacji przewidziane dla ZAPISZ i BIBLIOTEKA są identyczne

236. W przypadku wykrycia błędu w "liście pobieranych rozdziałów" (zdania TRAN lub ZAPI) pomijane są wszystkie znaki aż do nawiasu zamykającego
Gdy w czasie tego pomijania przechodzimy przez znak LP sygnalizowany jest błąd 236

237. Stacja TM określona w zdaniu TMB jest identyczna ze stacją TM określoną w zdaniu ZAPISZ

238. Stacja TM określona w zdaniu BIBLIOTEKA jest identyczna ze stacją TM określoną w zdaniu TMB

239. Z taśmy bibliotecznej (określonej zdaniem TMB) nie daje się ze względów technicznych odczytać sekcja startowa programu

240. Program na taśmie bibliotecznej ma nieprawidłową strukturę bądź błędną sumę kontrolną

241. Brak programu o danej nazwie na taśmie bibliotecznej

242. Nieprawidłowa nazwa programu

243. Długość sekcji startowej programu bibliotecznego różna od długości sekcji startowej programu tworzonego

244. Maksymalny adres PAO rozdziału pobieranego z taśmy bibliotecznej nie równa się wartości tego parametru dla tworzonego programu

245. Program z taśmy bibliotecznej posiada błędy translacji

246. Błędny opis rozdziału pobieranego z taśmy bibliotecznej

247. Program na taśmie bibliotecznej nie zawiera wskazanego rozdziału lub słownika

248. Brak zdania TMB w czołówce zakończonej zdaniem TRANSLACJA lub ZAPISZ w postaci rozszerzonej

249. Rozdział nie daje się odczytać z taśmy bibliotecznej

250. Rozdział nie mieści się na bębnie (TRANSLACJA)

251. Taśma biblioteczna nieprawidłowo zbudowana lub nie daje się odczytać (ZAWARTOŚĆ)

295 } Błąd maszyny
296

DODATEK H

Błędy etypu wykonania

W trakcie wykonywania się programu wynikowego sygnalizowane są niektóre błędy popełnione przez programistę. Najczęściej są to błędy powstałe wskutek podstawiania niewłaściowych argumentów do jawnych funkcji języka lub też do funkcji ukrytych realizujących poszczególne zdania języka SAKO. Informacja o powstałym błędzie wypisywana jest na aktualne wyjście symboliczne programu w następującej postaci: dwie litery określające rodzaj błędu, poprzedzone ciągiem kilkudziesięciu gwiazdek.

Po zasygnalizowaniu błędu, mogą zaistnieć dwa przypadki:

- program liczy dalej
- program wynikowy jest zatrzymany

W opisie znaczenia każdego z błędów podaje się numer przyczyny zakończenia (patrz opis programu ZAKO) jeżeli błąd kończy pracę programu - lub wartość funkcji (czasami jest nieokreślona), jeżeli program jest kontynuowany.

Spis błędów

AA - Nieznormalizowany argument funkcji ARC

AC - Argument funkcji ACS lub ASN wykracza poza dopuszczalny przedział lub nieznormalizowany argument funkcji COS

AL - Argument funkcji LN nie jest liczbą ułamkową, prawdopodobnie wartość argumentu nie została określona. Potęga ułamkowa liczona jest przy pomocy funkcji LN, stąd błąd mógł być zasygnalizowany w trakcie liczenia potęgi. Wartość funkcji jest nieokreślona (argument nieznormalizowany)

AP - Argument funkcji PWK nie jest liczbą ułamkową. Dalsze uwagi są takie same jak w przypadku błędu CA (argument nieznormalizowany)

AN - Argument funkcji ASN nie jest liczbą ułamkową. Dalsze uwagi są takie same jak w przypadku CA (argument nieznormalizowany)

AP - Nieznormalizowany argument funkcji PWK

AR - Oba argumenty funkcji ARC są jednocześnie równe zero. Wartość funkcji w tym przypadku jest również równa zero.

AS - Argument funkcji SIN nie jest liczbą ułamkową. Dalsze uwagi są takie same jak w przypadku CA (argument nieznormalizowany)

AT - Argument funkcji TG lub ATG nie jest liczbą ułamkową. Dalsze uwagi są takie same jak w przypadku CA (argument nieznormalizowany)

AU - Argument funkcji ACS nie jest liczbą ułamkową. Dalsze uwagi są takie same jak w przypadku CA (argument nieznormalizowany)

AV - Argument funkcji PWS nie jest liczbą ułamkową. Dalsze uwagi są takie same jak w przypadku CA (argument nieznormalizowany)

AX - Argument funkcji EXP nie jest liczbą ułamkową, prawdopodobnie wartość argumentu nie została określona (argument nieznormalizowany)
Przy pomocy funkcji EXP liczona jest potęga ułamkowa, stąd błąd mógł wystąpić w potęgowaniu. Wartość funkcji jest nieokreślona

BC- Błędna postać liczby czytanej przy pomocy zdania CZYTAJ. W przypadku czytania zmiennej prostej dodatkowo sygnalizuje się NG, natomiast przy czytaniu elementu bloku następuje natychmiastowe zakończenie czytania i wtedy może być zasygnalizowany błąd IL. Wartość liczby przeczytanej jest nieokreślona

BK - Koniec danych, wykonywało się zdanie CZYTAJ
Przyczyna zakończenia - 62

CA - Argument funkcji COS nie jest liczbą ułamkową. Prawdopodobnie zmienna podstawiona do funkcji, nie otrzymała jeszcze

wartości. Wartość funkcji w tym przypadku jest nieokreślona

DL - Numer symbolicznego urządzenia dołączanego za pośrednictwem funkcji DOL nie spełnia warunku: $0 \leqslant n \leqslant 31$. Przyczyna zakończenia - 62.

EX - Wartość funkcji EXP przekracza zakres liczb ułamkowych. W tym przypadku wartość funkcji jest równa wartości argumentu

IL - W trakcie czytania danych ilość liczb występujących przed gwiazdką nie jest zgodna z ilością elementów bloku zadeklarowaną w programie, liczby najpierw czytane są aż do natrafienia na gwiazdkę a następnie sprawdzana jest ilość, po zasygnalizowaniu błędu program przechodzi do dalszego liczenia.
Przy zbyt dużej ilości danych może nastąpić przekroczenie zakresu pamięci operacyjnej i przerwanie pracy programu z sygnalizacją systemową informującą o wykonaniu nielegalnego rozkazu maszyny

LN - Argument funkcji LN jest ujemny lub zerem, wartością funkcji w tym przypadku jest wartość argumentu. Potęga ułamkowa jest liczona przy pomocy funkcji LN

NC - Koniec danych, wykonywało się zdanie:

CZYTAJ OKTALNIE

Przyczyna zakończenia - 62

NG - Przy czytaniu wartości dla zmiennej prostej pobrano gwiazdkę zamiast liczby, wartość zmiennej jest przypadkowa, program po zasygnalizowaniu błędu przechodzi do dalszego liczenia

NU - Koniec danych, wykonywało się zdanie:

CZYTAJ WIERSZ

Przyczyna zakończenia - 62

OD - Numer symbolicznego urządzenia odłączanego przy pomocy funkcji ODL nie spełnia warunku: $0 \leqslant n \leqslant 31$
Przyczyna zakończenia - 62

PA - 1. Program nie rozpoczął się wykonywać ze względu na zbyt małą pojemność pamięci operacyjnej posiadanej w danym zestawie. Przyczyna zakończenia - 0

2. Przynajmniej jeden z argumentów funkcji ARC nie jest liczbą ułamkową. W tym przypadku wartość funkcji jest nieokreślona

PC - Liczba drukowana nie jest ułamkową, prawdopodobnie zmienna drukowana nie otrzymała wartości, po sygnalizacji drukuje się zadaną wartość

QQ - Koniec danych, wykonywała się funkcji CZD. Przyczyna zakończenia - 62

RB - Rozdział programu pobierany do wykonania z taśmy magnetycznej nie dał się odczytać; przyczyna zakończenia - 3

RN - Nie istnieje w programie rozdział o numerze wskazanym przez zdanie IDZ DO ROZDZIAŁU, wykonywanie się programu zostanie przerwane, przyczyna zakończenia - 2

SE - Numer symbolicznego urządzenia wybieranego przy pomocy funkcji SEL nie spełnia warunku: $0 \leqslant n \leqslant 31$ Przyczyna zakończenia - 62

TG - Przy liczeniu funkcji TG wargość cosinusa jest równa zero, tangens nie istnieje. Wartość funkcji (po zasygnalizowaniu błędu) jest równa zero

UW - Ujemny wykładnik potęgi całkowitej o podstawie całkowitej lub ułamkowej, po zasygnalizowaniu błędu program liczy dalej. Wynik działania jest równy podstawie potęgi

XC - Patrz XZ

XP - Patrz XZ

XT - Błędny numer taśmy magnetycznej użyty w zdaniu współpracy z taśmami
Przyczyna zakończenia - 62

XZ - Błędy XC, XP, XZ sygnalizowane są w przypadku współpracy z taśmami magnetycznymi. Każda z tych sygnalizacji wiąże się z określoną wartością funkcji CHR dla określonego zdania. Zależności te podaje tabelka na następnej stronie.

Zdanie	Wartość funkcji CHR	Wypisywany rodzaj błędu
OTWORZ	1	XZ
	2	XP
	3	XC
ZAMKNIJ	1	XP
ZNAJDZ	1	XZ
	3	XC
POBIERZ	4	XC

ZN - Wystąpił nadmiar zmiennoprzecinkowy (liczba ułamkowa przekroczyła zakres) i wykonywanie programu przerwano w miejscu wystąpienia błędu; zostanie wypisana dodatkowo przyczyna zakończenia nr 77

ZP - 1. Wystąpił podmiar zmiennego przecinka; wynik działania na liczbach ułamkowych jest zbyt mały. Program przerwano w miejscu wystąpienia błędu. Błędy: ZP,ZN,ZU mogą być również spowodowane w trakcie obliczania wartości niektórych funkcji języka wykonujących działanie zmiennoprzecinkowe: np. TG,DZD,DOD itd. Przyczyna zakończenia - 77

2. W danych oktalnych czytanych przez zdanie CZYTAJ OKTALNIE, wystąpiła cyfra 8 lub 9. Cyfra ta została pominięta, a pozostałe cyfry tworzą liczbę.

ZZ - Wykonano dzielenie przez zero ułamkowe, przyczyna zakończenia programu jak dla ZN

DODATEK I

Spis zdań czołówki języka SAKO

1. Zdania związane z translacją:
   OPISZ PROGRAM
   CHOWAJ SLOWNIKI
   URUCHAMIANIE [: numer rozdziału]
   PAO : pojemność pamięci
   BEBEN : pojemność bębna
   POST MORTEM [: szerokość arkusza]
   NAZWA : nazwa
   BIBLIOTEKA: nazwa stacji, numer szpuli [, M]
   TMB : nazwa stacji, numer szpuli [, {M / N}]
   ZAWARTOŚĆ : nazwa stacji, numer szpuli [, M / N]

2. Zdania związane z wykonaniem:
   LIKWIDUJ STOP
   KLUCZE [: lista numerów kluczy]
   WYMAGANY BEBEN : pojemność
   MINIMALIZUJ
   NIWELUJ NADMIARY [: {TAK / NIE}]
   IGNORUJ BLEDY

3. Zdania kończące czołówkę:
   TRANSLACJA [: lista programów z wybranymi rozdziałami]
   ZAPISZ : nazwa stacji, numer szpuli
   lokalizacja [{, N / , M}] [, lista programów z wybranymi rozdziałami]
   POBIERZ: nazwa stacji, numer szpuli, nazwa programu, {N / M} [, lista rozdziałów lub/i słowników]

   KONIEC

DODATEK J

Spis numerów błędów sygnalizowanych przez translator języka JOM

1 - błędny zwrot zdania

2 - błędna liczba w zdaniu

Błędy w zdaniu PROBLEM:

10 - ilość zadań na liście przekracza 8

11 - błędne użycie znaków ( ), ; .

12 - brak wymienionego translatora w zestawie

13 - parametr WEOT przekracza 15

14 - błędna nazwa zadania

Błędy w zdaniach typu wejścia-wyjścia oraz zdań OGRANICZ:

20 - błędny zapis numeru wejścia lub wyjścia symbolicznego

21 - brak podprogramu lub dekodera w zestawie

22 - brak urządzeń w zestawie

23 - przy definicji symbolicznego wejścia lub wyjścia zestawiono ze sobą niewłaściwy dekoder, podprogram lub urządzenie, lub ograniczenie dotyczy urządzenia wejściowego

24 - nie zamknięta lista parametrów podprogramu

25 - więcej niż jeden parametr dodatkowy

26 - brak wartości ograniczenia

Błędy w zdaniu INFORMUJ:

30 - błędny adres początkowy lub błędna ilość słów na liście obszarów.

Błędy specjalne:

99 - błąd maszyny

BRAK KONCA CZOŁOWKI
BRAK PAO LUB BEBNA

DODATEK K

Spis numerów błędów sygnalizowanych przez translator POPR

01 - brak danych na wejściu symbolicznym nr 2

02 - urządzenie rzeczywiste podane w definicji wejścia symbolicznego 0 (lub 1 w przypadku aktualizacji) nie jest stacją taśm magnetycznych

03 - nieprawidłowy zwrot

04 - zamiast LF użyto innego znaku

05 - nieprawidłowy znacznik podziału w nagłówku strony

06 -
07 - } znaczenie jak 01

08 - nieprawidłowa budowa nagłówka strony

09 - nieprawidłowy identyfikator strony

10 - nieprawidłowy znak po identyfikatorze strony (wymieniony w nagłówku strony lub w argumencie instrukcji) lub po operatorze

12 - brak zadanej strony w dokumencie lub próba odwołania się w niewłaściwej kolejności

13 - ze względu na wadę nośnika nie udało się zapisać na taśmie magnetycznej metryki końca dokumentu

14 - niedozwolony znak w argumencie

15 - za duża liczba w argumencie

16 - próba usunięcia wiersza zerowego

17 - brak wiersza o zadanym numerze na wskazanej stronie lub próba odwołania się do wiersza w niewłaściwej kolejności

18 - próba odwołania się do wiersza w niewłaściwej kolejności

19 - znaczenie jak 01

20 - błąd maszyny

21 - po ZKT nie ma metryki końca dokumentu.

DODATEK L

Spis numerów przyczyn zakończenia programu

Znaczenia przyczyn zakończenia programu użytkowego zostały w miarę możliwości podane tak, aby dawały pewną informację programiście piszącemu program w języku SAKO.

501 - zbliżający się koniec taśmy magnetycznej wykryty w trakcie pracy PPT8 (wstawka w SAS)

502 - niezgodność sum kontrolnych przy PCT8 (wstawka w SAS)

503 - błędne parametry sekcji startowej (adres pamięci operacyjnej, adres bębnowy lub długość)

504 -
505 - } błędy nie mogą być sygnalizowane w programie SAKO
506 -

507 - dla symbolicznego wejścia lub wyjścia nie został określony dekoder

508 - próba pisania lub czytania z/na urządzenie symboliczne bez wykonania funkcji DOL

509 - wykonano funkcję ODL dla urządzenia symbolicznego, dla którego nie była wykonywana funkcja DOL

510 - próba współpracy z urządzeniem symbolicznym o nie właściwie dobranym dekoderze

511 - nielegalny rozkaz lub SDY nietypowe z błędnym parametrem w części adresowej, błąd najczęściej pojawiający się przy:
- przekroczeniu zakresu bloków, powoduje wykroczenie poza pamięć maszyny
- brak gwiazdki w bloku danych, dane wprowadzane są poza pamięć
- zbyt długi wiersz przekraczający zakres bloku

- niewłaściwe dobranie parametrów zdania POWTORZ

513 - wyczerpano czas przeznaczony dla programu

514 - program został wyrzucony przez operatora

515 - błędne parametry zdania współpracy z pamięciami pomocniczymi

516 - jak wyżej

517 - próba wykonania funkcji DOL dla urządzenia symbolicznego, która nie została zdefiniowana w czołówce systemu

518 - zdefiniowano zbyt dużo urządzeń symbolicznych (dotyczy funkcji DOL)

519 - próba wykonania DOL gdy było błędnie zdefiniowane urządzenie symboliczne

520 - próba dołączenia urządzenia symbolicznego przy pomocy funkcji DOL do obszaru wykraczającego poza pamięć maszyny, zbyt mały blok został przewidziany na dekoder i podprogram

521 - błędne parametry otwórz UZ

522 - próba rozdefiniowania przesłania będącego w toku (wstawka w SAS)

523 - błędne odwołanie do biblioteki systemu (wstawki w SAS)

524 - odwołanie się do taśmy nieotwartej lub otwartej do czytania

525 - zapętlenie adresu w SDY (wstawka w SAS)

526 - odwołanie do taśmy systemu gdy została zwolniona (błędy operatorskie)

527 - błędne parametry SDY czytaj monitor (wstawki w SAS)

529 - błędne parametry jednego ze zdań współpracy z taśmami magnetycznymi

530 - urządzenie nie jest taśmą magnetyczną (wstawki w SAS)

531 - błędne parametry "podaj definicję przesłania" (wstawka w SAS)

532 - zabrakło miejsca na liście T-etykiet dla "nadaj wartość T-etykiecie" (wstawka w SAS).

Literatura

[1] Witaszek J. - System magazynowania i aktualizacji dokumentów i jego zastosowanie w procesie uruchamiania programów. Praca Instytutu Maszyn Matematycznych, luty 1970, Warszawa

[2] Łukaszewicz L. i Mazurkiewicz A. - System automatycznego kodowania SAKO, 1966, Wrocław, Warszawa, Kraków.

[3] ZAM-41 Oprogramowanie Część pierwsza - Maszyna użytkowa, Opis eksploatacyjny, Język Operacyjny Maszyny, Instytut Maszyn Matematycznych, Warszawa.

[4] Kasprowicz A., Sawicki S. i zespół - Opis języka SAKO dla maszyny ZAM-41, Gliwice, luty 1971.

[5] Kosecka D., Łącka M., Pawlikowska E., Wrotek Z., Trzpil I, Kozłowska W., Rykowski A., Sawicki S., Swianiewicz J., Walasek J. - Opisy konstrukcyjne elementów translatora, (materiały nie publikowane) Instytut Maszyn Matematycznych, Warszawa.

[6] ZAM-41 Oprogramowanie (System 141) Tom drugi, Język SAS, Instytut Maszyn Matematycznych, Warszawa.

[7] System 141 Opisy funkcjonalne, Instytut Maszyn Matematycznych, Warszawa.

BIBLIOTEKA NARODOWA BN

13.IV.73.