POLITECHNIKA GDAŃSKA

ANDRZEJ JĘDRUCH

# PODSTAWOWY JĘZYK PROGRAMOWANIA (PJP)

## maszyny cyfrowej ZAM 41 alfa

GDAŃSK 1974

POLITECHNIKA GDAŃSKA

ANDRZEJ JĘDRUCH

# PODSTAWOWY JĘZYK PROGRAMOWANIA (PJP)

## maszyny cyfrowej ZAM 41 alfa

GDAŃSK 1974

PRZEWODNICZĄCY KOMITETU REDAKCYJNEGO
WYDAWNICTW POLITECHNIKI GDAŃSKIEJ

*Roman Kazimierczak*

REDAKTOR SKRYPTÓW

*Włodzimierz Rodziewicz*

RECENZENT:

*Tadeusz Bartkowski*

Wydano za zgodą
Rektora Politechniki Gdańskiej

PRINTED IN POLAND

Do nabycia: PP Dom Książki Nr 23, Gdańsk-Wrzeszcz, ul. Majakowskiego 11/12
(Gmach Główny Politechniki Gdańskiej) Nr kodu 80-952

WYDAWNICTWO UCZELNIANE POLITECHNIKI GDAŃSKIEJ

---

Wydanie I. Nakład 500+35+25 egz. Ark. wyd. 15,1. Ark. druku 11. Papier offset. kl. III 70 g, form. B-3. Oddano do druku 5 XII 73. Druk ukończono w lutym 1974 r. Zamówienie nr S/457/73/SC 33. Cena zł 23,-

---

Wykonano w Zakładzie Graficznym Politechniki Poznańskiej
Poznań, ul. Ogrodowa 11, tel. 554-25

SPIS TREŚCI

# Wstęp

Pomimo intensywnego rozwoju wielu klas języków programowania, języki symboliczne, których typowym przedstawicielem jest między innymi Podstawowy Język Programowania (PJP) maszyny cyfrowej ZAM 41, są nadal szeroko stosowane, ponieważ ciągle jeszcze stanowią podstawę do tworzenia translatorów języków wyższego szczebla, a także są nieodzowne w przypadkach, gdy istotne znaczenie ma minimalizacja czasu wykonywania programu. Prócz tego języki symboliczne stosowane są przy konstrukcji programów sterujących i zarządzających jak np. systemów operacyjnych czy programów sterujących przebiegiem procesów technologicznych. Dodatkowym czynnikiem zwiększającym znaczenie języka PJP jest to, że mc. ZAM 41 posiada oprócz translatora tego języka, jedynie translator języka ALGOL i w związku z tym język PJP stanowi namiastkę innych, specjalistycznych języków programowania. Wszystko to czyni niezbędną znajomość języka PJP przy rozwiązywaniu bardziej zaawansowanych zadań przy pomocy tej maszyny.

Kolejne wiersze programu w języku PJP są symbolicznymi opisami słów (rozkazów i liczb) tworzonego programu. Szczególne znaczenie ma możliwość symbolicznego zapisu rozkazu, a zwłaszcza jego części adresowej. Często bowiem, w trakcie tworzenia rozkazu nie można określić jego części adresowej, ponieważ adres komórki, na której działać będzie rozkaz, zależy od ilości dalszych rozkazów programu. W tych wypadkach przedstawienie adresu w formie symbolu – etykiety, która wskazuje komórkę pamięci o pożądanej zawartości, uwalnia programistę od żmudnych czynności adresowania rozkazów, które to czynności wykonywane są przez program tłumaczący. Program tłumaczący – translator w trakcie wczytywania programu przyporządkowuje poszczególnym etykietom wartości równe adresom komórek pamięci, które one opatrują. Następnie, po wczytaniu całego programu, translator zastępuje adresy rozkazów wyrażone przy pomocy etykiet przez wartości tych etykiet.

Dużą wygodę stanowi również możliwość literowego, mnemotechnicznego przedstawiania części operacyjnej rozkazu.

Prócz wierszy opisujących tworzone słowa w programie w języku PJP występują również dyrektywy i parametry określające sposób tłumaczenia programu, jego położenie w pamięci, maksymalny czas wykonywania itp.

Omawiając tego typu języki wspomina się zwykle, że ich wadą jest dość ścisłe ukierunkowanie maszynowe, co oznacza, że programowanie wymaga szczegółowej znajomości funkcji wykonywanych przez maszynę. Uwzględniając jednak przewidywany dydaktyczny charakter niniejszego opracowania, omawiana wada staje się zaletą,

ponieważ czytelnik będzie niejako zmuszony do dogłębnego poznania funkcji realizowanych przez maszynę. Warto dodać, że organizacja ogólna maszyny ZAM 41 jest typowa, a przez to czytelnik znający programowanie w języku PJP z łatwością opanuje programowanie w innym języku należącym do rozpatrywanej klasy.

Opracowanie niniejsze stanowi próbę opisu Podstawowego Języka Programowania (PJP) maszyny cyfrowej ZAM 41 i może służyć jako podręcznik nauki tego języka. Od czytelnika wymagana jest znajomość jedynie elementarnych zasad budowy i działania maszyn cyfrowych.

Początkowe dwa rozdziały opracowania zawierają opis hardware'u maszyny ze szczególnym uwzględnieniem potrzeb programowania. Czytelnik interesujący się wyłącznie zwykłymi programami użytkowymi może opuścić rozdział 2.

Rozdział 3 zawiera opis software'u maszyny: systemu operacyjnego i translatorów. Ponadto w rozdziale umieszczono opis obsługi operatorskiej maszyny.

Rozdział 4 zawiera szczegółowy opis rozkazów maszyny.

Dalsze rozdziały 5, 6, 7, 8 zawierają opis języka PJP. Przy pierwszym czytaniu część rozdziału 6 poświęconą podprogramom można ominąć.

Autor ma nadzieję, że niniejsze opracowanie uzupełni lukę w dokumentacji maszyny, a tym samym umożliwi opracowanie nowych, interesujących i wartościowych programów.

# 1. ORGANIZACJA OGÓLNA MASZYNY ZAM 41

Maszyna cyfrowa ZAM 41 jest jednoadresową, równoległą maszyną cyfrową. Obwody maszyny wykonane są w technice tranzystorowo-diodowej (II generacja). Maszyna jest technicznie przystosowana do pracy w systemie wieloprogramowym.

Część centralna maszyny zawiera obwody arytmometru i sterowania (procesor) oraz pamięć operacyjną o pojemności 12 288 słów 24-bitowych (3 bloki po 4096 słów).

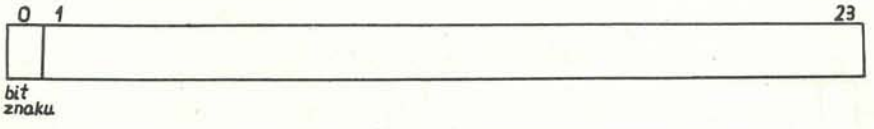
Maszyna wyposażona jest w następujące urządzenia zewnętrzne: pamięć bębnową PB-5, pamięć taśmową PT-2, czytnik kart perforowanych, monitor (dalekopis połączony z maszyną), drukarkę wierszową oraz dwa czytniki i perforatory taśmy papierowej. Pamięć taśmowa połączona jest z pamięcią operacyjną poprzez specjalny procesor (kanał) sterujący przebiegiem transmisji. Umożliwia on przeprowadzanie transmisji do/z pamięci taśmowej bez wstrzymywania wykonywania programu przez maszynę.

Maszyna charakteryzuje się średnią prędkością liczenia – wykonuje około 23 000 operacji (dodawań, przesłań itp.) w ciągu sekundy. System przerwań umożliwia równoczesną pracę wielu urządzeń zewnętrznych przy minimalnej zajętości części centralnej.

Część centralna maszyny zawiera szereg rejestrów, których znajomość jest niezbędna do opanowania programowania w języku PJP. Rejestry te omówiono w pkt. 1.1.

## 1.1. REJESTRY I WSKAŹNIKI CZĘŚCI CENTRALNEJ

*Akumulator* A – 24-bitowy rejestr o bitach ponumerowanych od 0 do 23 używany m.in. przy wykonywaniu operacji dodawania, odejmowania oraz operacji logicznych




Rys. 1.1. Rejestry A, B, M.

*Rejestr modyfikacji* B – 24-bitowy rejestr o bitach ponumerowanych od 0 do 23, używany m.in. przy modyfikacjach adresowych, a także przy niektórych operacjach arytmetycznych.

*Rejestr blokady dolnej* BD – 8-bitowy rejestr o bitach ponumerowanych od 9 do 16. Bitowi nr 16 tego rejestru przyporządkowuje się wagę $2^7 = 128$, bitowi nr 15

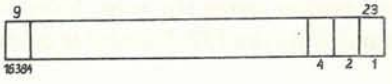
wagę $2^8 = 256$,....., bitowi nr 9 wagę $2^{14} = 16\,384$. Zawartość rejestru blokady dolnej BD określa dolną granicę obszaru pamięci przydzielonego programowi użytkowemu.


Rys. 1.2. Rejestry BD i BG.

*Rejestr blokady górnej* BG – 8-bitowy rejestr o bitach ponumerowanych od 9 do 16. Bitowi nr 16 tego rejestru przyporządkowuje się wagę $2^7 = 128$, bitowi nr 15 wagę $2^8 = 256$,....., bitowi nr 9 wagę $2^{14} = 16\,384$. Zwiększona o 127 zawartość rejestru blokady górnej BG określa górną granicę obszaru pamięci przydzielonego programowi użytkowemu.

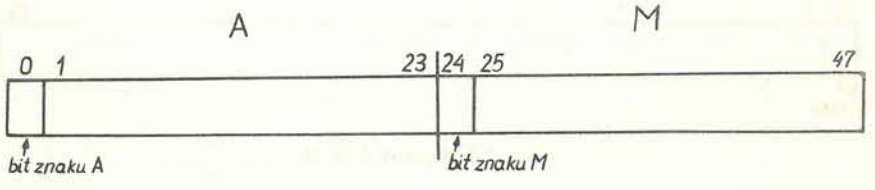
*Mnożnik* M – 24-bitowy rejestr o bitach ponumerowanych od 0 do 23 używany m.in. przy operacjach mnożenia, dzielenia oraz przesyłania do/z urządzeń zewnętrznych.

*Licznik rozkazów* LR – 15-bitowy rejestr o bitach ponumerowanych od 9 do 23. Jego zawartość po wykonaniu rozkazu wskazuje adres komórki pamięci, z której maszyna ma pobrać następny rozkaz (jeśli nie pojawił się sygnał przerwania).
Bitowi nr 23 tego rejestru przyporządkowuje się wagę $2^0 = 1$, bitowi nr 22 wagę $2^1 = 2$,....., bitowi nr 9 wagę $2^{14} = 16\,384$.




Rys. 1.3. Rejestr LR

*Akumulator – mnożnik* AM – 48-bitowy rejestr utworzony z rejestrów A i M o bitach ponumerowanych od 0 do 47.




Rys. 1.4. Rejestr AM

Prócz opisanych, część centralna maszyny zawiera także inne rejestry, które nie są jednakże dostępne bezpośrednio, przez co ich znajomość nie jest niezbędna w programowaniu maszyny.

Z tego powodu ograniczymy się tutaj do ich wymienienia:

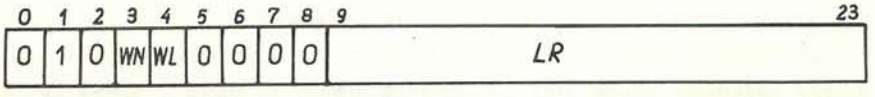
C – 24-bitowy rejestr centralny
R – 23-bitowy rejestr rozkazów
S – 23-bitowy rejestr sumatora
LK – 6-bitowy licznik kroków
RP – 16-bitowy rejestr przerwań

*Wskaźnik legalności* WL – jego stan określa zezwolenie (WL = 1), lub brak zezwolenia (WL = 0) na wykonanie rozkazu nielegalnego (rozkazy nielegalne opisano w pkt. 4.2.). Wskaźnik jest zapalany (1 → WL) w wyniku wykonania przerwania lub rozkazu SLR o adresie < 64, zaś może być zgaszony (0 → WL) rozkazem WRO.

*Wskaźnik nadmiaru* WN – do wskaźnika tego wpisywana jest 1, gdy wynik działania arytmetycznego przekracza dopuszczalną wartość.

*Wskaźnik przyjęć* WP – jego stan określa zezwolenie (WP = 1), lub brak zezwolenia (WP = 0), na przyjęcie przerwania (pkt. 1.6). Wskaźnik przyjęć jest zapalany (1 → WP) w wyniku wykonania rozkazu PWT O lub rozkazu WRO, zaś gasi go (0 → WP) rozkaz CWT O oraz przerwanie.

Ponadto, w celu uproszczenia opisu, w dalszej części niniejszego opracowania posługiwać się będziemy umownym rejestrem tzw. *uzupełnionym licznikiem rozkazów ULR.* Przez ULR rozumieć będziemy słowo 24-bitowe, którego bity reprezentują aktualną zawartość licznika rozkazów LR oraz niektórych wskaźników. Czasami uzupełniony licznik rozkazów ULR nazywany jest *śladem.*




Rys. 1.5. Uzupełniony licznik rozkazów ULR

W dalszej części niniejszego opracowania pisząc o rejestrach lub ich zawartościach, używać będziemy oznaczeń literowych zamiast podawania pełnej nazwy rejestru. Tak więc zamiast pisać np.... dodaje do zawartości rejestru B...., pisać będziemy: dodaje do B.

## 1.2. PAMIĘĆ OPERACYJNA

Pamięć operacyjna maszyny ZAM 41 zawiera 12 288 komórek o adresach od 0 do 12 287. W każdej komórce przechowuje się słowo 24-bitowe. Zawartości komórek pamięci mogą być przesyłane do obwodów arytmometru i sterowania lub poprzez kanał do pamięci taśmowej. Operacje te, zwane operacjami odczytu nie zmieniają zawartości komórek pamięci. Natomiast zmiana zawartości komórki dokonywana jest dopiero w wyniku zapisu nowego słowa do tej komórki (przesłanego z arytmometru lub pamięci taśmowej).

W dalszej części niniejszego opracowania komórkę pamięci o adresie n oznaczać będziemy: PAO[n].

## 1.3. LICZBY I ROZKAZY

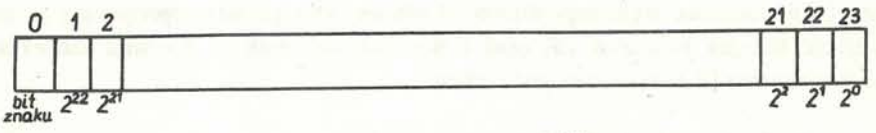
Słowa umieszczone w pamięci mogą reprezentować rozmaite informacje. Wśród nich szczególnie ważne są liczby i rozkazy, ponieważ ich struktura jest zdeterminowana przez konstrukcję maszyny; inne informacje (np. teksty) mogą być przedstawiane w dowolnej, ustalonej formie.

Na wstępie omówimy cztery zasadnicze rodzaje liczb stosowane w m.c. ZAM 41: liczby całkowite krótkie, liczby całkowite długie, liczby ułamkowe długie oraz liczby zmiennoprzecinkowe; opis rozkazów arytmetycznych zawarty w rozdziale 4 dotyczy wyłącznie ww. rodzajów liczb.

Liczby w maszynie ZAM 41 przedstawione są w systemie *znak – moduł:* zawartość wyróżnionego bitu słowa, zwanego bitem znaku określa znak liczby; gdy zawartość tego bitu wynosi 1 – liczba jest ujemna, gdy 0 – liczba jest dodatnia. Pozostałe bity określają moduł liczby.

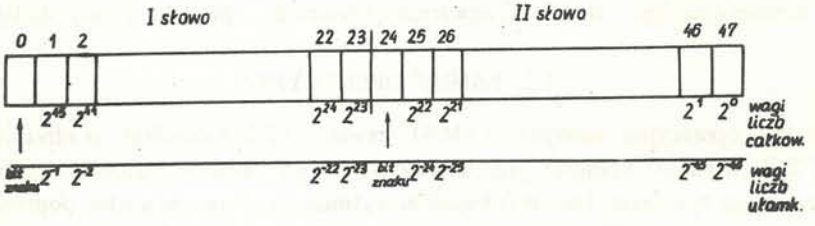
*Liczby całkowite krótkie* (rys. 1.6) reprezentowane są przez słowa 24-bitowe. Bit nr 0 określa znak liczby, zaś bity nr 1–23 moduł liczby. Bitowi nr 23 przyporządkowuje się wagę $2^0 = 1$, bitowi nr 22 wagę $2^1 = 2$,..., bitowi nr 1 wagę $2^{22} = 4\,194\,304$.
Liczba całkowita krótka może więc przyjmować wartości ź przedziału

$$<-(2^{23}-1),\quad 2^{23}-1>(2^{23}=8\,388\,608).$$




Rys. 1.6. Liczba całkowita krótka

*Liczby całkowite długie* (rys. 1.7) reprezentowane są przez 2 słowa 24-bitowe. Stosując numerację bitów podaną na rys. 1.7 bitowi nr 47 przyporządkowujemy wagi $2^0 = 1$, bitowi nr 46 wagę $2^1 = 2$,...., bitowi nr 25 wagę $2^{22} = 4\,194\,304$, bitowi nr 23 wagę $2^{23} = 8\,388\,608$,...., bitowi nr 1 wagę $2^{45} = 35\,185\,214\,088\,832$.




Rys. 1.7. Liczba długa całkowita i ułamkowa

Bity znaków mogą być jednakowe lub różne. W pierwszym przypadku mówimy o liczbie długiej *jednolitej*, w drugim o *niejednolitej*. Wartość liczby długiej niejednolitej równa się sumie zawartości obu słów tworzących liczbę wziętych wraz ze znakami i podanymi wagami.

Liczba całkowita długa może przyjmować wartości z przedziału $<-(2^{46}-1),\ 2^{46}-1>$.

*Liczby ułamkowe długie* (rys. 1.7) reprezentowane są również przez dwa słowa 24-bitowe. Od liczb całkowitych długich różnią się tylko przyporządkowaniem wag, które podane są na rys. 1.7. Liczby ułamkowe długie mogą przyjmować wartości z przedziału $(-1,+1)$.

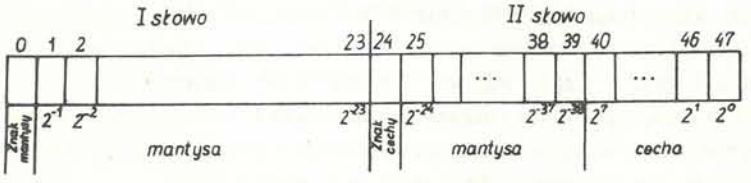
*Liczby zmiennoprzecinkowe* reprezentowane są przez 2 słowa 24-bitowe. Liczba zmiennoprzecinkowa różna od 0 ma postać

$$m \cdot 2^c$$

przy czym $0.5 \leqslant |m| < 1$, $|c| \leqslant 255$. Liczba 0 przedstawiana jest w postaci (nazywanej czasami zerem znormalizowanym)

$$m = 0 \text{ i } c = -255$$

Liczbę m nazywa się mantysą, liczbę c cechą (lub wykładnikiem).




Rys. 1.8. Liczba zmiennoprzecinkowa

Bit nr 0 określa znak mantysy, zaś bity nr 1–23, 25–39 moduł mantysy. Bit nr 24 określa znak cechy, zaś bity nr 40–47 moduł cechy.

Bitom tworzącym mantysę przyporządkowano następujące wagi: bitowi nr 1 wagę $2^{-1} = 0.5$, bitowi nr 2 wagę $2^{-2} = 0.25$,...., bitowi nr 23 wagę $2^{-23}$, bitowi nr 25 wagę $2^{-24}$,...., bitowi nr 39 wagę $2^{-38}$, zaś bitom tworzącym cechę przyporządkowano następujące wagi: bitowi nr 47 wagę $2^0 = 1$, bitowi nr 46 wagę $2^1 = 2$,...., bitowi nr 40 wagę $2^7 = 128$.

Uwzględniając powyższe przyporządkowanie i poprzednio podane założenia można stwierdzić, że liczby zmiennoprzecinkowe mogą przyjmować wartości z następujących przedziałów:

$$(-2^{255},\ -2^{-256}>,\ 0,\ <2^{-256},\ 2^{255})$$

przy czym $2^{255} \cong 10^{77}$.

Przejdziemy obecnie do omówienia reprezentacji rozkazów w m.c. ZAM 41.

Rozkaz zapisany jest przy pomocy jednego słowa 24-bitowego. W rozkazie wyróżniamy trzy części:

– część modyfikacyjną na bitach nr 0, 1, 2

- część operacyjną na bitach nr 3–8; gdy zawartość tych bitów wynosi 55 lub 63, to część operacyjna wydłużona jest o dalsze trzy bity
- część adresową na bitach nr 9–23 lub na bitach nr 12–23, gdy część operacyjna jest wydłużona


Ryc. 1.9. Struktura rozkazu

Część modyfikacyjna, opisana szczegółowo w pkt. 1.6, określa rodzaj modyfikacji adresowych, które mają być wykonane w celu obliczenia adresu efektywnego (końcowego) rozkazu. Część operacyjna określa czynności, które mają być wykonane w czasie realizowania rozkazu. Część adresowa, która może zawierać liczby 0–32 767 lub 0–4095 (gdy część operacyjna rozkazu jest wydłużona), określa adres komórki pamięci, która weźmie udział w czynnościach wykonywanych przez rozkaz, lub ilość przesunięć, lub liczbę, która ma być wpisana do licznika rozkazów LR itd.

## 1.4. KODY ALFANUMERYCZNE

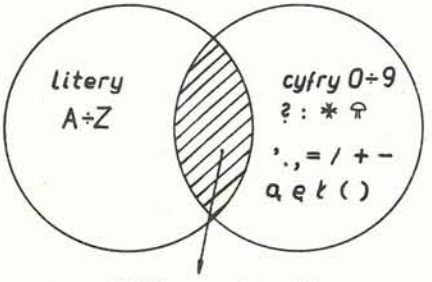
Poniższe tablice zawierają stosowane w maszynie ZAM 41 kody alfanumeryczne:

- M2 międzynarodowy drugi, podstawowy kod maszyny
- DW kod drukarki wierszowej
- ZAM 41, mający charakter umowny, wprowadzony w celu ujednolicenia niektórych programów (np. sortowania tekstów)

W 5-kanałowym kodzie M2 w celu zwiększenia ilości możliwych kombinacji ($2^5 = 32$) wprowadzono dwa znaki sterujące „litery" LS i „cyfry" FS. Znak „litery" LS działa w ten sposób, że następujące po nim kombinacje są interpretowane przez dalekopis jako litery, aż do momentu, gdy nadejdzie znak „cyfry" FS. Wówczas następne kombinacje interpretowane są jako cyfry, znaki przestankowe itp., aż do nadejścia znaku „litery" LS, itd.

Innymi słowy w dalekopisie znajduje się przełącznik mający dwa położenia stabilne L i C. Przełączenie następuje w wyniku przeczytania przez dalekopis znaku LS lub FS. W zależności od położenia przełącznika dalekopis interpretuje nadchodzące znaki jako litery, gdy przełącznik znajduje się w pozycji L lub jako cyfry, gdy przełącznik znajduje się w pozycji C.

Opis znaków kodu M2 podany w tablicy 1.1 zawiera także ustawienie przełącznika dalekopisu podczas drukowania znaku. W maszynie cyfrowej taki przełącznik realizowany jest zwykle przy pomocy odpowiedniej sekwencji rozkazów.
Uwaga: znaki sterujące „litery" LS, „cyfry" FS, powrót karetki CR, nowa linia LF, spacja (odstęp) SP, pusty BL powodują wykonanie przez dalekopis odpowiednich czynności, niezależnie od położenia przełącznika cyfry/litery. Ilustruje to rys. 1.10.




Rys. 1.10. Zbiory znaków kodu M2

Tablica 1.1

Kody M2, DW, ZAM 41

| Znak | Kod M2 | | | Kod DW | | Kod ZAM 41 | |
|---|---|---|---|---|---|---|---|
| | dzies. | okt. | przesuw | dzies. | okt. | dzies. | okt. |
| A | 24 | 30 | L | 34 | 42 | 34 | 42 |
| Ą | 22 | 26 | C | 43 | 53 | 43 | 53 |
| B | 19 | 23 | L | 18 | 22 | 18 | 22 |
| C | 14 | 16 | L | 50 | 62 | 50 | 62 |
| Ć | | | | | | 8 | 10 |
| D | 18 | 22 | L | 10 | 12 | 10 | 12 |
| E | 16 | 20 | L | 27 | 33 | 27 | 33 |
| Ę | 11 | 13 | C | 42 | 52 | 42 | 52 |
| F | 22 | 26 | L | 26 | 32 | 26 | 32 |
| G | 11 | 13 | L | 58 | 72 | 58 | 72 |
| H | 5 | 5 | L | 6 | 6 | 6 | 6 |
| I | 12 | 14 | L | 38 | 46 | 38 | 46 |
| J | 26 | 32 | L | 22 | 26 | 22 | 26 |
| K | 30 | 36 | L | 54 | 66 | 54 | 66 |
| L | 9 | 11 | L | 14 | 16 | 14 | 16 |
| Ł | 5 | 5 | C | 59 | 73 | 59 | 73 |
| M | 7 | 7 | L | 46 | 56 | 46 | 56 |
| N | 6 | 6 | L | 30 | 36 | 30 | 36 |

C.d. Tablicy 1.1

| Znak | kod M2 | | | kod DW | | kod ZAM 41 | |
|---|---|---|---|---|---|---|---|
| | dzies. | okt. | przesuw | dzies. | okt. | dzies. | okt. |
| Ń | | | | | | 12 | 14 |
| O | 3 | 3 | L | 62 | 76 | 62 | 76 |
| Ó | | | | | | 16 | 20 |
| P | 13 | 15 | L | 1 | 1 | 1 | 1 |
| Q | 29 | 35 | L | 33 | 41 | 33 | 41 |
| R | 10 | 12 | L | 17 | 21 | 17 | 21 |
| S | 20 | 24 | L | 49 | 61 | 49 | 61 |
| Ś | | | | | | 20 | 24 |
| T | 1 | 1 | L | 9 | 11 | 9 | 11 |
| U | 28 | 34 | L | 41 | 51 | 41 | 51 |
| V | 15 | 17 | L | 25 | 31 | 25 | 31 |
| W | 25 | 31 | L | 57 | 71 | 57 | 71 |
| X | 23 | 27 | L | 5 | 5 | 5 | 5 |
| Y | 21 | 25 | L | 37 | 45 | 37 | 45 |
| Z | 17 | 21 | L | 21 | 25 | 21 | 25 |
| Ż | | | | 60 | 74 | 60 | 74 |
| Ź | | | | | | 24 | 30 |
| O | 13 | 15 | C | 53 | 65 | 53 | 65 |
| 1 | 29 | 35 | C | 13 | 15 | 13 | 15 |
| 2 | 25 | 31 | C | 45 | 55 | 45 | 55 |
| 3 | 16 | 20 | C | 29 | 35 | 29 | 35 |
| 4 | 10 | 12 | C | 61 | 75 | 61 | 75 |
| 5 | 1 | 1 | C | 3 | 3 | 3 | 3 |
| 6 | 21 | 25 | C | 35 | 43 | 35 | 43 |
| 7 | 28 | 34 | C | 19 | 23 | 19 | 23 |
| 8 | 12 | 14 | C | 51 | 63 | 51 | 63 |
| 9 | 3 | 3 | C | 11 | 13 | 11 | 13 |
| * | 7 | 7 | C | 47 | 57 | 47 | 57 |
| , | 6 | 6 | C | 44 | 54 | 44 | 54 |
| ' | 20 | 24 | C | | | 32 | 40 |
| : | 14 | 16 | C | 31 | 37 | 31 | 37 |
| = | 15 | 17 | C | 28 | 34 | 28 | 34 |
| ? | 19 | 23 | C | | | 36 | 44 |
| * | | | | 2 | 2 | 2 | 2 |
| ↑ | 26 | 32 | C | | | 40 | 50 |
| * | 18 | 22 | C | | | 48 | 60 |
| – | 24 | 30 | C | 55 | 67 | 55 | 67 |
| + | 17 | 21 | C | 15 | 17 | 15 | 17 |
| / | 23 | 27 | C | 23 | 27 | 23 | 27 |
| ) | 9 | 11 | C | 7 | 7 | 7 | 7 |
| ( | 30 | 36 | C | 39 | 47 | 39 | 47 |
| SP | 4 | 4 | | 4 | 4 | 4 | 4 |
| CR | 2 | 2 | | | | | |

C.d. Tablicy 1.1

| Znak | kod M2 | | | kod DW | | kod ZAM 41 | |
|---|---|---|---|---|---|---|---|
| | dzies. | okt. | przesuw | dzies. | okt. | dzies. | okt. |
| LF | 8 | 10 | | | | 63 | 77 |
| LS | 31 | 37 | | | | | |
| FS | 27 | 33 | | | | | |
| BL | 0 | 0 | | | | | |

Oznaczenia:

SP – spacja (odstęp)
CR – powrót karetki
LF – nowa linia
LS – znak liter L
FS – znak cyfr C
BL – znak pusty (blanka)

Tablica 1.2

Przyporządkowanie znaków kodów M2, DW, ZAM 41

| M2→ | C' | N' | O' | S' | Z' | Z" | ? | ' | ↑ | * | CR | LF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| → DW | | | | | Ż | | • | = | * | SP | | |
| → ZAM | Ć | Ń | Ó | Ś | Ż | Ź | ? | ' | ↑ | * | | LF |

| ZAM→ | Ć | Ń | Ó | Ś | Ż | Ź | ? | ' | * | ↑ | * | LF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| → M2 | C' | N' | O' | S' | Z' | Z" | ? | ' | ↑ | ↑ | * | CR LF |
| → DW | C | N | O | S | Ż | Z | • | = | * | * | SP | |

oznaczenia: patrz tablica 1.1.

Tablica 1.1. podaje wartości liczbowe przyporządkowane przez kody poszczególnym znakom. Puste miejsce oznacza brak przyporządkowania.

Tablica 1.2 podaje wzajemne przyporządkowanie znaków z poszczególnych kodów, przy czym znaki posiadające przyporządkowania identyczne w trzech rozpatrywanych kodach pominięto.

## 1.5. CYKL ROZKAZOWY MASZYNY

Maszyna cyfrowa ZAM 41 wykonuje program w sposób typowy dla maszyn jednoadresowych: po wykonaniu rozkazu, zawartość licznika rozkazów LR wskazuje adres

komórki pamięci, w której znajduje się następny rozkaz. Rozkazy działań arytmetycznych, przesłań, przesunięć, zwiększają na ogół zawartość LR o 1, zaś rozkazy sterujące zmieniają LR w inny sposób.

Wykonanie rozkazu składa się z kilku faz.

W pierwszej fazie rozkaz zostaje pobrany z komórki pamięci wskazanej przez zawartość LR i przesłany do rejestru rozkazów R. Następnie, w drugiej fazie, obliczany jest adres efektywny rozkazu (pkt. 1.6), zaś w ostatniej, trzeciej fazie, zostają wykonane poszczególne czynności rozkazu.

Przedstawione zasady zmieniają się nieco, w chwili, gdy maszyna przyjmuje przerwanie (pkt. 1.7 i 2.1) – następuje wówczas zapamiętanie aktualnego ULR, zmiana zawartości LR, oraz szereg innych czynności opisanych w pkt. 2.1.

## 1.6. MODYFIKACJE ADRESOWE

Wykonanie rozkazu, jak już wspomnieliśmy, poprzedzone jest obliczeniem jego adresu efektywnego (końcowego). Adres efektywny rozkazu, o bitach ponumerowanych od 9–23, powstaje przez odpowiednie przekształcenie (modyfikację) adresu umieszczonego na 15 ostatnich bitach rozkazu.

W maszynie ZAM 41 stosuje się trzy rodzaje modyfikacji adresowych: D-modyfikację, P-modyfikację, oraz B-modyfikację. Żądanie wykonania którejkolwiek z nich wskazane jest przez zawartość trzech pierwszych bitów rozkazu: bit nr 0 wskazuje D-modyfikację, bit nr 1 P-modyfikację, zaś bit nr 2 B-modyfikację.

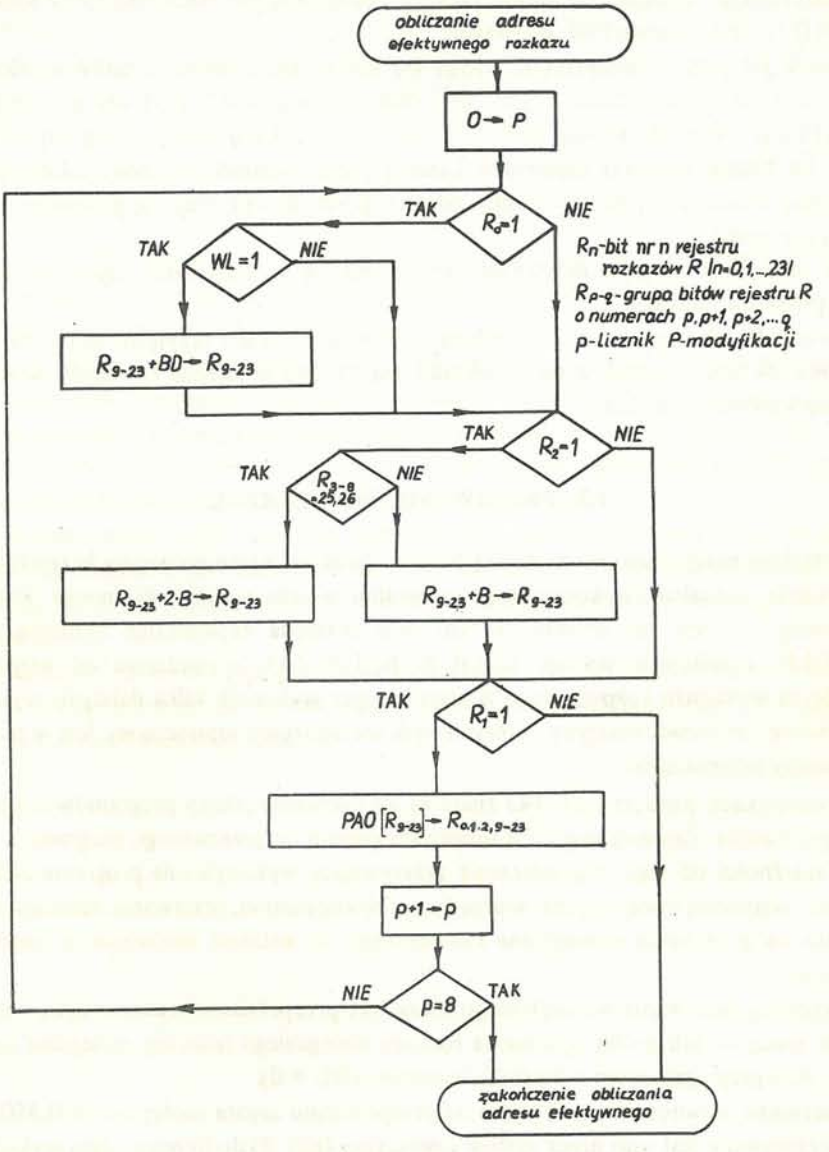
*D-modyfikacja* polega na dodaniu do adresu rozkazu zawartości rejestru blokady dolnej BD. Modyfikacja ta wykonywana jest wyłącznie przy zapalonym wskaźniku legalności WL (WL = 1), toteż w programach w języku PJP nie ma ona żadnego znaczenia, bowiem programy te wykonywane są przy WL = 0. D-modyfikacja przedłuża czas wykonywania rozkazu o 6$\mu$s.

*P-modyfikacja,* czyli adresowanie pośrednie wskazuje, że adres efektywny rozkazu znajduje się w komórce pamięci określonej przez adres rozkazu, jeśli zawartość trzech pierwszych bitów tej komórki pamięci wynosi 0. W przeciwnym razie, adres umieszczony na 15 ostatnich bitach rozpatrywanej komórki pamięci jest adresem pośrednim, na podstawie którego wykonywane są dalsze modyfikacje określone przez 3 pierwsze bity tej komórki pamięci.

W szczególności może się pojawić ponownie P-modyfikacj, w wyniku której opisane czynności powtórzą się. Ze względu na to, że w czasie wykonywania rozkazu maszyna nie może przyjmować przerwań ograniczono ilość P-modyfikacji przy obliczaniu adresu efektywnego rozkazu do 7. Adres wyznaczony w wyniku wykonania 7 P-modyfikacji uważa się za adres efektywny, niezależnie od ewentualnych żądań dalszych modyfikacji. Jednokrotna P-modyfikacja przedłuża czas wykonania rozkazu o 22$\mu$s.

*B-modyfikacja* polega na dodaniu do adresu rozkazu zawartości B-rejestru z uwzględnieniem znaku, przy czym liczbę umieszczoną w B-rejestrze uważa się za liczbę całkowitą krótką, oraz pomija się 9 pierwszych bitów wyniku dodawania. B-modyfikacja przedłuża czas wykonania rozkazu o 6$\mu$s.

Opisane modyfikacje mogą występować w rozkazie jednocześnie – wówczas tryb obliczania adresu efektywnego najlepiej pokazać przy pomocy wykresu operacyjnego (rys. 1.11).




Rys. 1.11. Obliczanie adresu efektywnego rozkazu

Powyższy wykres operacyjny przedstawiający sposób obliczania adresu efektywnego wymaga kilka wyjaśnień. Obliczanie adresu efektywnego rozpoczyna się od sprawdzania, czy w rozkazie występuje D-modyfikacja. Jeśli tak i jeśli wskaźnik legalności jest

zapalony (WL = 1), to do adresu rozkazu zostaje dodana zawartość rejestru blokady dolnej BD, przy czym bierze się zawsze pod uwagę tylko ostatnie 15 bitów wyniku. Następnie wykonuje się badanie bitu nr 2 rozkazu i jeśli jest on równy 1, to wykonuje się B-modyfikację dodając do adresu rozkazu pojedynczą lub podwójną (przy rozkazach 25 UAD i 26 PAD) zawartość B-rejestru.

Podobnie, jak przy D-modyfikacji, bierze się pod uwagę ostatnie 15 bitów wyniku.

Z kolei następuje badanie, czy bit nr 1 rozkazu równy jest 1, czyli czy występuje P-modyfikacja. Jeśli tak, to bity nr 0, 1, 2 oraz 9–23 rozkazu zastępuje się odpowiadającymi im bitami komórki pamięci wskazanej przez ostatnio obliczony adres rozkazu (tj. liczbę umieszczoną na 15 ostatnich bitach), po czym wykonuje się ponownie opisane wyżej czynności.

Po obliczaniu adresu efektywnego rozpoczyna się wykonywanie czynności wskazanych przez kod rozkazu.

Uwaga: W celu uproszczenia niniejszego opisu modyfikacji przyjęto, że rejestr rozkazów ma 24 bity, chociaż w rzeczywistości ma 23. Nie zmniejsza to jednak w żadnym stopniu ścisłości rozważań.

## 1.7. PRZERWANIA WEWNĘTRZNE

W trakcie pracy maszyny zachodzą pewne zdarzenia, które wymagają bezzwłocznego przerwania aktualnie wykonywanego programu w celu wykonania innego programu związanego z tymi zdarzeniami. W tym celu maszyna zapamiętuje aktualną zawartość ULR, a następnie wpisuje do LR liczbę 129–143, w zależności od urządzenia, w którym wystąpiło rozpatrywane zdarzenie, oraz wykonuje kilka dalszych czynności. Omówione czynności maszyny, których opis szczegółowy umieszczony jest w pkt. 2.1, nazywamy *przerwaniem.*

W komórkach pamięci 129–143 znajdują się pierwsze rozkazy programów obsługujących przerwania. Zapamiętany ULR umożliwia powrót do przerwanego programu.

W zależności od tego, czy zdarzenie przerywające wykonywanie programu zachodzi w części centralnej maszyny, czy w urządzeniu zewnętrznym, przerwania dzielimy, odpowiednio na przerwania wewnętrzne i zewnętrzne. Te ostatnie omówiono szczegółowo w pkt. 2.1.

Przyczyną przerwania wewnętrznego może być przepełnienie rejestru zegara wewnętrznego maszyny lub próba wykonania rozkazu nielegalnego (rozkazy nielegalne opisane w pkt. 4.2) przy zgaszonym wskaźniku legalności (WL = 0).

Przerwanie wewnętrzne, polegające na przepełnieniu zegara następuje co 0,3508 sek. Wykorzystywane jest ono przez system operacyjny (pkt. 3) do liczenia czasu wykonywania programu, a także „przypomina" systemowi operacyjnemu o konieczności zbadania, czy niektóre jego czynności czekające na realizację, mogą już być wykonane. Dokładną zawartość rejestru zegara wewnętrznego maszyny można odczytać wykonując rozkaz CML 524, który przesyła 12-bitową zawartość rejestru zegara do rejestru M.

## 1.8. REŻIMY PRACY MASZYNY

Wspomnieliśmy już, że maszyna ZAM 41 jest przystosowana konstrukcyjnie do pracy w systemie wieloprogramowym. Cechą takich systemów jest m.in. przejęcie przez specjalny program nadzorujący wykonywanie innych programów (użytkowych) – system operacyjny (pkt. 3), funkcji związanych ze sterowaniem pracą urządzeń zewnętrznych. Ponadto, w celu zapobiegnięcia przypadkowemu zniszczeniu zawartości komórek pamięci (np. wskutek błędów w programie) zawierających inny program użytkowy lub system operacyjny, stosuje się również system ochrony pamięci – w maszynie cyfrowej ZAM 41 wszelkie rozkazy zmieniające zawartość komórek pamięci o adresach leżących poza obszarem wyznaczonym przez zawartość rejestrów BD i BG+127, są rozkazami nielegalnymi. Nielegalne, w myśl przedstawionych zasad, są także rozkazy współpracy z urządzeniami zewnętrznymi, bowiem program użytkowy mógłby swoimi błędnymi poleceniami zakłócić pracę tych urządzeń.

Innymi słowy, można powiedzieć, że wykonywanie pewnych operacji w programie użytkowym jest zabronione. Operacje te są zastrzeżone dla systemu operacyjnego. Tak więc, w maszynie cyfrowej musi znajdować się wskaźnik określający, czy rozkazy nielegalne mogą być wykonywane; czyli wskaźnik określający rodzaj programu wykonywanego aktualnie przez maszynę: użytkowy czy system operacyjny.

Tym wskaźnikiem jest wskaźnik legalności WL: podczas wykonywania programu użytkowego wskaźnik jest zgaszony (WL = 0), co oznacza, że wykonywanie rozkazów nielegalnych jest zabronione. Próba wykonania rozkazu nielegalnego powoduje wówczas przerwanie.

Reasumując: maszyna może pracować w dwóch stanach, określonych przez WL. Gdy WL = 0 maszyna nie może wykonywać rozkazów nielegalnych, gdy WL = 1 maszyna może wykonywać wszystkie rozkazy.

# 2. URZĄDZENIA ZEWNĘTRZNE I ICH WSPÓŁPRACA Z CZĘŚCIĄ CENTRALNĄ

## 2.1. SYSTEM PRZERWAŃ Z URZĄDZEŃ ZEWNĘTRZNYCH

Urządzenie zewnętrzne sygnalizuje zakończenie wykonywania wydanego mu wcześniej polecenia poprzez wysłanie sygnału przerwania do części centralnej maszyny. Sygnał ten powoduje zapalenie w rejestrze przerwań bitu odpowiadającego rozpatrywanemu urządzeniu zewnętrznemu.

Zdarza się również, że wydane polecenie nie może być wykonane ze względu na pojawienie się błędu, który polegać może np. na niezgodności bitu parzystości w odczytanym rządku taśmy magnetycznej. Wówczas wysyłany jest również sygnał przerwania, który zapala w rejestrze przerwań bit (wspólny dla wszystkich urządzeń zewnętrznych) błędu. Ten sam bit rejestru przerwań przyjmuje również sygnały przerwań wywołane naciśnięciem przycisku „Zgłoszenie operatora" ZO przy urządzeniu zewnętrznym. W rejestrze przerwań znajduje się także bit rejestrujący przerwania z zegara wewnętrznego maszyny oraz bit rejestrujący przerwania spowodowane próbą wykonania rozkazu nielegalnego przy zgaszonym wskaźniku WL.

Warunkiem wykonania przez maszynę dalszych czynności związanych z przyjęciem przerwania jest stan wskaźnika przyjęć WP = 1 lub nadejście przerwania spowodowanego przez nielegalny rozkaz. Czynności te rozpoczynają się po zakończeniu wykonywania rozkazu.

Przyjęcie przerwania polega na wykonaniu przez część centralną maszyny następujących czynności:

a) odjęcie 1 od zawartości licznika rozkazów LR, a następnie zapamiętanie uzupełnionego licznika rozkazów ULR w PAO [BD] (tj. komórce, której adres wskazuje zawartość rejestru blokady dolnej BD).
b) wpisanie 0 do wskaźnika przyjęć WP
c) wpisanie 1 do wskaźnika legalności WL
d) wyzerowanie odpowiedniego bitu w rejestrze przerwań (nie dotyczy przerwania spowodowanego błędami lub naciśnięciem przycisku „Zgłoszenie operatora" ZO.).
e) wpisanie do licznika rozkazów LR liczby określonej przez tablicę 2.1.

Podane w tabeli priorytety przerwań określają kolejność przyjmowania przerwań, jeśli więcej niż jedno z nich oczekuje na załatwienie. Jak łatwo zauważyć, urządzenia szybsze mają wyższe priorytety (1 – najwyższy, 15 – najniższy). Najwyższy priorytet pamięci bębnowej wynika ze szczególnego sposobu współpracy z nią, jaki zastosowano w m.c. ZAM 41: sygnał przerwania z pamięci bębnowej wskazuje, że słowo, którego adres

ustawiono wcześniej w rejestrze adresu pamięci bębnowej za 600 $\mu$s będzie mogło być odczytane lub zapisane. Jakiekolwiek opóźnienie rozpoczęcia operacji zapisu lub odczytu powodowałoby konieczność czekania przez czas jednego obrotu bębna (około 40 ms.).

Tablica 2.1.

Przerwania

| Przyczyna przerwania | zawartość LR po przyjęciu przerwania | priorytet przerwania |
|---|---|---|
| Zbliżanie się do głowicy zapisu/odczytu pamięci bębnowej słowa, którego adres został ustawiony w rejestrze adresów pamięci bębnowej. | 129 | 1 |
| Zakończenie czytania kolumny karty perforowanej. | 130 | 2 |
| Zakończenie czytania znaku przez czytnik taśmy II | 131 | 3 |
| Przepełnienie zegara | 132 | 4 |
| Zakończenie czytania karty perforowanej. | 133 | 5 |
| Zakończenie perforowania znaku przez perforator II. | 134 | 6 |
| Zakończenie czytania znaku przez czytnik taśmy I. | 135 | 7 |
| Zakończenie drukowania wiersza i przesuwu papieru przez drukarkę wierszową. | 136 | 8 |
| Zakończenie perforowania znaku przez perforator I | 137 | 9 |
| Zakończenie drukowania znaku lub przyjęcie znaku przez monitor. | 138 | 10 |
| Zakończenie wykonywania polecenia wydanego kanałowi taśmy magnetycznej. | 140 | 12 |
| Rozkaz nielegalny | 142 | 14 |
| Naciśnięcie przycisku ZO (zgłoszenie operatora) lub błędy. | 143 | 15 |

Ustawienie licznika rozkazów na wartość 143 wskazuje na pojawienie się błędu w jakimś urządzeniu lub na naciśnięcie przycisku „Zgłoszenie operatora" ZO, przy czym nie można oczywiście jeszcze wskazać konkretnej przyczyny tego przerwania. W celu identyfikacji stosuje się w tym wypadku sekwencję rozkazów czytania wskaźników błędu i wskaźników naciśnięcia przycisku „Zgłoszenie operatora" poszczególnych urządzeń zewnętrznych aż do chwili znalezienia wskaźnika zapalonego. Odczyt wskaźnika powoduje jednoczesne wyzerowanie jego zawartości, oraz jeśli była tylko jedna przyczyna tego przerwania, wyzerowanie odpowiedniego bitu w rejestrze przerwań. W przypadku, gdy omawiane przerwanie miało kilka przyczyn, wyzerowanie bitu w rejestrze przerwań następuje po odczycie wszystkich zapalonych wskaźników błędów i wskaźników „Zgłoszenia operatora" ZO.

Rozkazy służące do odczytywania zawartości wskaźników urządzeń zewnętrznych podane są w tablicy 2.2.

## 2.2. URZĄDZENIA WEJŚCIA – WYJŚCIA

Zestaw urządzeń wejścia – wyjścia m.c. ZAM 41 składa się z 2 stolików operatora, z których każdy zawiera czytnik i perforator, drukarki wierszowej, monitora oraz czytnika kart.

Tablica 2.2.

Rozkazy odczytu zawartości rejestrów kluczy i wskaźników zgłoszenia operatora oraz rozkazy wpisywania do rejestrów lampek

| Typ urządzenia | Czytaj wskaźnik ZO | Czytaj rejestr kluczy do mnożnika M | | Pisz mnożnik M do rejestru lampek | |
|---|---|---|---|---|---|
| | | I sposób | II sposób | I sposób | II sposób |
| Stolik operatora I | CWT 64 | CMP 68 | CML 68 | PML 580 | PMP 580 |
| Stolik operatora II | CWT 128 | CMP 132 | CML 132 | PML 644 | PMP 644 |
| Monitor | CWT 672 | CMP 676 | CML 676 | PML 1188 | PMP 1188 |
| Czytnik kart | CWT 1280 | CMP 772 | CML 772 | PML 1284 | PMP 1284 |
| Drukarka wierszowa | CWT 608 | CMP 1636 | CML 1636 | PML 1124 | PMP 1124 |

Tablica 2.3.

Działanie rozkazów odczytu rejestru kluczy i ładowania rejestru lampek

| | | | | | |
|---|---|---|---|---|---|
| Stan rejestru kluczy przed odczytem | | ⊗ | ○ | ⊗ | ⊗ |
| Zawartość mnożnika po odczycie rejestru kluczy lub przed przesłaniem zawartości do rejestru lampek. | I sposobem | 1 1 0 1 | | | |
| | | 0 1 2 3 4 . . . (bity M) | | | |
| | II sposobem | | | | 1 0 1 1 |
| | | | | | (bity M) . . . 19 20 21 22 23 |
| Stan rejestru lampek po załadowaniu | | ⊗ | ○ | ⊗ | ⊗ |

○ — lampka zgaszona lub klucz wyciśnięty

⊗ – lampka zapalona lub klucz wciśnięty

Dodatkowo, każde z tych urządzeń wyposażone jest w pulpit sterujący zawierający rejestr kluczy (4), rejestr lampek (4) oraz przycisk niestabilny „Zgłoszenie operatora" ZO. Naciśnięcie przycisku ZO powoduje zapalenie odpowiedniego wskaźnika naciśnięcia przycisku ZO oraz wysłanie sygnału przerwania. Wskaźnik naciśnięcia przycisku ZO można odczytać rozkazem CWT, którego postać dla poszczególnych urządzeń podana jest w tablicy 2.2.

Tablica 2.2 zawiera również rozkazy odczytu zawartości rejestru kluczy oraz zapisu rejestru lampek. Zakończenie zapisu w rejestrze lampek lub odczytu rejestru kluczy nie jest sygnalizowane przerwaniem; jeżeli w trakcie takiej operacji pojawi się rozkaz dotyczący rejestru mnożnika M, to jego wykonanie zostaje wstrzymane aż do zakończenia operacji przesłania. Działanie tych rozkazów zilustrowane jest przykładem podanym w tabeli 2.3.

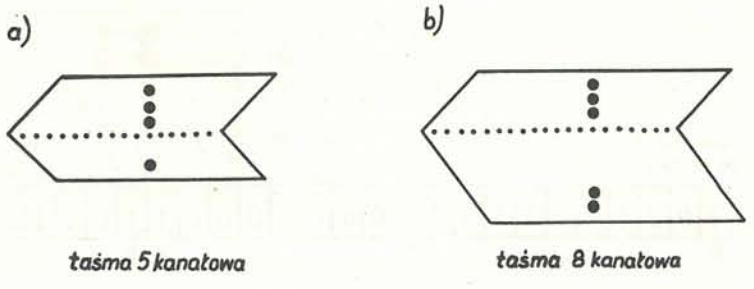
### 2.2.1. Czytniki i perforatory

Zarówno czytniki jak i perforatory przystosowane są do pracy na taśmie 5, 6, 7 lub 8 – kanałowej.

Czytnik zainstalowany na stoliku I, typu CT–1001, czyta taśmę perforowaną z prędkością 1000 rządków/sek., zaś czytnik zainstalowany na stoliku II, typu FERRANTI czyta taśmę z prędkością 300 rządków/sek.

Oba perforatory, typu FACIT, perforują taśmę z prędkością 150 rządków/sek.

Rozpatrzmy na wstępie rozkazy czytania rządka taśmy perforowanej. Czytanie rozpoczyna się zawsze od rozkazu PWT 1088 (dla stolika I) lub PWT 1152 (dla stolika II). Rozkaz ten zapala wskaźnik czytania oraz powoduje rozpoczęcie czytania rządka taśmy, przy czym zakończenie czytania rządka sygnalizowane jest przerwaniem. Z kolei rozkaz CMP lub CML (dokładna postać podana jest w tablicy 2.4) powoduje przesłanie zawartości bufora czytnika do rejestru mnożnika M, a następnie, jeżeli wskaźnik czytania jest zapalony, rozpoczęcie czytania następnego rządka taśmy. Zakończenie czytania grupy rządków taśmy poprzedza zwykle rozkaz CWT 1088 (dla stolika I) lub CWT 1152 (dla stolika II). Rozkaz ten gasi wskaźnik czytania, tak że rozkaz CMP lub CML, który się ewentualnie pojawi spowoduje jedynie przesłanie zawartości bufora czytnika do M, nie powodując rozpoczęcia czytania następnego rządka. Wykonanie omawianego rozka-




Rys. 2.1. Rządek taśmy perforowanej czytany lub perforowany rozkazami z tablicy 2.4.

zu CWT w czasie odczytu rządka (tj. w czasie od chwili wykonania rozkazu PWT lub CMP lub CML do chwili nadejścia przerwania z czytnika) powoduje zatrzymanie czytania rządka, co ma szczególne znaczenie w przypadku zacięcia się czytnika lub natrafienia na koniec taśmy perforowanej.

Tablica 2.4

Rozkazy współpracy z czytnikami i perforatorami

| | Sposób I | | Sposób II | |
|---|---|---|---|---|
| | stolik I | stolik II | stolik I | stolik II |
| czytaj znak | PWT 1088<br>CMP 1096 | PWT 1152<br>CMP 1160 | PWT 1088<br>CML 1096 | PWT 1152<br>CML 1160 |

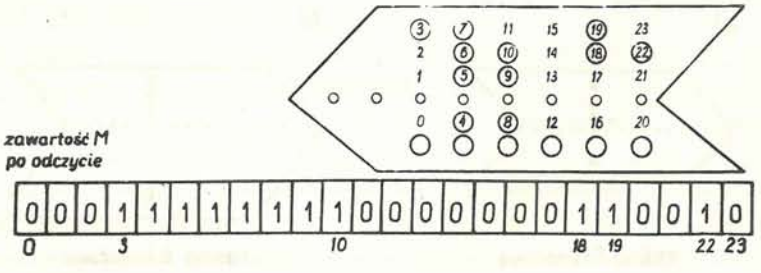
zawartość M po odczycie lub przed perforowaniem rządka z rys. 2.1.a.

| 1 | 1 | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... | (bity M) | ... | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | | |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | | | | | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | | |

zawartość M po odczycie lub przed perforowaniem rządka z rys. 2.1.b.

| Perforuj znak | PML 1608 | PML 1672 | PMP 1608 | PMP 1672 |
|---|---|---|---|---|

Prócz omówionego czytania „po rządku" na stoliku I istnieje również możliwość czytania tzw. 5-kanałowej taśmy oktalnej. Podczas czytania tej taśmy nie są wysyłane sygnały przerwań. Skrajny (dolny na rys. 2.2) bit rządka taśmy jest bitem prowadzącym – rządki taśmy bez bitu prowadzącego są pomijane. Pozostałe bity w rządku są bitami znaczącymi. Rozkaz „czytaj oktalnie" PWT 1600 powoduje przeczytanie 6 rządków taśmy i utworzenie z nich w rejestrze mnożnika M słowa 24-bitowego. Działanie tego rozkazu wyjaśnia rys. 2.2.




Rys. 2.2. Czytanie oktalne

Jeżeli w trakcie odczytu taśmy perforowanej wystąpi błędne działanie sterowania czytnika, to wówczas do wskaźnika błędu czytnika wpisywana jest 1, a w ślad za tym wykonywane jest przerwanie. Zawartość tego wskaźnika można odczytać rozkazem CWT 576 (w stoliku I) lub CWT 640 (w stoliku II).

Rozkazy perforowania rządków taśmy papierowej dla obu perforatorów podane są również w tablicy 2.4. Rozpoczęcie perforowania następuje w wyniku wykonania rozkazu PMP lub PML, zaś zakończenie perforowania rządka taśmy sygnalizowane jest przerwaniem.

### 2.2.2. Drukarka wierszowa

Drukarka wierszowa drukuje 120 znaków w wierszu z prędkością 10 wierszy/sek. Drukarka posiada wbudowaną pamięć buforową o pojemności 120 znaków 6-bitowych.

Drukowanie znaków na drukarce poprzedza zapełnienie pamięci buforowej drukarki – wykonuje się to rozkazem PMZ 96, który przesyła z rejestru mnożnika M do pamięci buforowej 4 znaki 6-bitowe w kodzie drukarki wierszowej DW (pkt. 1.3). Wcelu wypełnienia całej pamięci buforowej należy, oczywiście, rozkaz PMZ powtórzyć 30-krotnie, przy czym znaki wprowadzone przez pierwszy rozkaz PMZ drukowane będą z lewej strony arkusza itd.

W celu wydrukowania wprowadzonych znaków do 6-bitowego rejestru ilości linii drukarki należy wprowadzić z mnożnika M liczbę określającą ilość linii przesuwu papieru po wydruku (realizuje to rozkaz PMP 614), a następnie zapalić wskaźnik wydruku rozkazem PWT 96. Jeżeli do rejestru przesuwu papieru wprowadzić liczbę 0, to po wydrukowaniu papier nie przesunie się – następny wiersz będzie drukowany na poprzednim. Zakończenie drukowania i przesuwu papieru (również o 0 linii) sygnalizowane jest przerwaniem – LR przyjmuje wartość 136.

Jeżeli w drukarce wierszowej wystąpi błąd podczas drukowania, to wówczas do wskaźnika błędu drukarki wpisywana jest jedynka, a w ślad za tym wykonywane jest przerwanie. Wskaźnik błędu drukarki odczytuje się (i zeruje) rozkazem CWT 1120.

### 2.2.3. Monitor

Monitor jest urządzeniem technicznym m.c. ZAM 41 służącym do bezpośredniej, dwukierunkowej komunikacji operatora z wykonywanym programem (praca konwersacyjna). Zasadniczą częścią monitora jest dalekopis pracujący w kodzie M2. Prędkość drukowania dalekopisu wynosi 5 znaków/sek. Wiersz dalekopisu może zawierać najwyżej 68 znaków.

Rozpatrzmy drukowanie znaków na monitorze przesyłanych z rejestru mnożnika M. W tym celu należy zapalić wskaźnik pisania monitora rozkazem PWT 160 (do wskaźnika tego dołączona jest czerwona lampka „PISANIE" umieszczona na pulpicie sterującym monitora). Następnie należy, drukowany znak w kodzie M2, umieszczony na pierwszych 5 bitach M, przesłać do monitora rozkazem PML 165. Po wydrukowaniu znaku monitor wyśle sygnał przerwania – LR przyjmie wartość 138.

Czytanie znaków z klawiatury dalekopisu i przesyłanie ich do mnożnika M odbywa się ze zgaszonym wskaźnikiem pisania i zapalonym wskaźnikiem czytania. Odczyt (i zerowanie) wskaźnika pisania rozkazem CWT 160 powoduje jednoczesne zapalenie wskaźnika czytania. Przy zapalonym wskaźniku czytania naciśnięcie „klawisza" dalekopisu powoduje wpisanie kodu znaku do rejestru monitora, zgaszenie wskaźnika czytania oraz wysłanie sygnału przerwania (LR przyjmuje wartość 138). Przyjęcie przerwania zapala wskaźnik czytania, jeśli nie był zapalony wskaźnik pisania.

Przesłanie kodu znaku z rejestru monitora do M realizowane jest przy pomocy rozkazu CML 165.

### 2.2.4. Czytnik kart

Czytnik kart umożliwia czytanie standardowych kart 80-kolumnowych.

Rozkaz PWT 156 powoduje rozpoczęcie operacji czytania. Po przeczytaniu każdej kolumny wysyłany jest sygnał przerwania (LR = 130), zaś po przeczytaniu całej karty wysyłany jest również sygnał przerwania, z tym, że LR przyjmuje wartość 133. Po każdym przerwaniu – sygnalizacji przeczytania kolumny rozkaz CML 268 powoduje przesłanie zawartości bufora czytnika kart do rejestru mnożnika M.

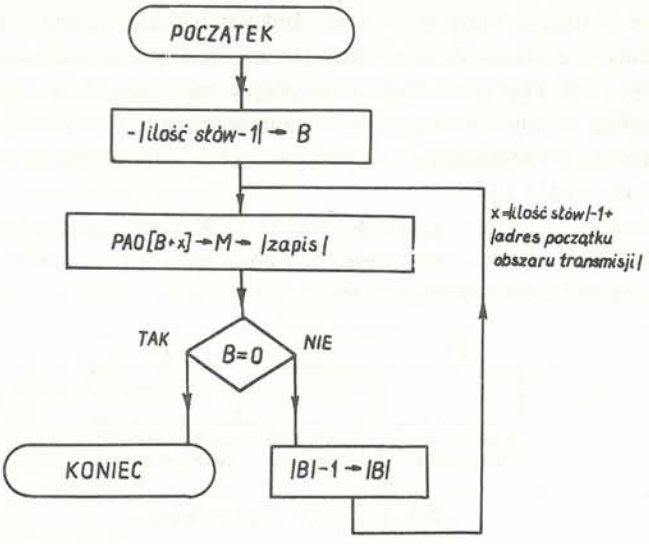
Jeśli wydane polecenie czytania nie może być wykonane ze względu na brak kart lub nastąpiło zacięcie czytnika, to wówczas do wskaźnika błędu czytnika kart wpisywana jest jedynka, a w ślad wysyłany jest sygnał przerwania LR = 143. Wskaźnik błędu czytnika kart można przeczytać (i wyzerować) rozkazem CWT 768.

## 2.3. PAMIĘCI ZEWNĘTRZNE

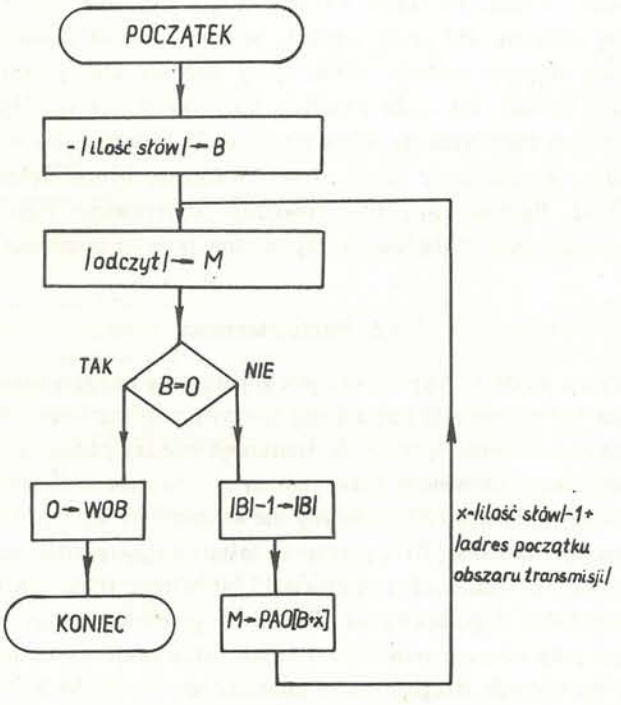
### 2.3.1. Pamięć bębnowa

Pojemność pamięci bębnowej PB-5 wynosi 32 768 słów 24-bitowych, zapisanych na 128 ścieżkach, z których każda zawiera 256 słów. Średni czas dostępu do pamięci bębnowej wynosi 20 msek.

Omówimy poniżej zespół czynności realizujących operacje zapisu i odczytu pamięci bębnowej. Na początku należy, ustawiony w M adres bębnowy (0–32 767) przesłać do rejestru adresu bębna rozkazem PMP 50, po czym zapalić wskaźnik odczytu WOB rozkazem PWT 544 (lub zapisu WZB rozkazem PWT 1056). W chwili, gdy słowo bębnowe wskazane przez zawartość rejestru adresu bębna zbliży się na odległość 4-słów bębnowych do głowicy odczytu/zapisu wysyłany jest sygnał przerwania (oznacza to, że za 600 $\mu$s należy rozpocząć odczyt lub zapis na bęben). W trakcie odczytu lub zapisu kolejne słowa pojawiać się będą co około 150 $\mu$s. Czas ten w zupełności wystarcza na zapisanie w pamięci operacyjnej przeczytanego słowa (lub przesłanie do M i dalej do pamięci bębnowej) oraz wykonanie zmiany modyfikatora. Zwykle sekwencja rozkazów zapisu i odczytu ma postać pokazaną na rys. 2.3 i 2.4.



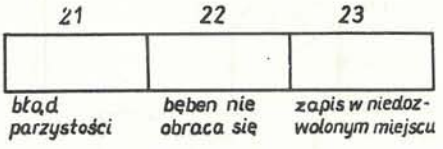

Rys. 2.3. Realizacja zapisu na bęben




Rys. 2.4. Realizacja odczytu z bębna

Przesłanie zawartości mnożnika M do pamięci bębnowej realizuje rozkaz PMZ 544, zaś przesłanie z pamięci bębnowej do mnożnika M realizuje rozkaz CMZ 544. Oba te rozkazy inicjują przesłania do/z mnożnika M nie blokując wykonywania następnych rozkazów chyba, że pojawi się rozkaz dotyczący mnożnika M. Z tego powodu przy odczycie z bębna stosuje się kolejność rozkazów podaną na rys. 2.4 – rozkaz SOB, badający zawartość B i odejmujący 1 od modułu jego zawartości jest wykonywany jednocześnie z rozkazem CMZ 544.

Po wykonaniu operacji czytania rozkaz CWT 544 gasi wskaźnik odczytu z bębna WOB. Gaszenie wskaźnika zapisu na bęben jest zbędne, ponieważ gaśnie on po około 200$\mu$s od nadejścia ostatniego rozkazu zapisu.




Rys. 2.5. Rejestr błędów bębna

Pamięć bębnowa wyposażona jest także w 3-bitowy rejestr błędów, którego pozycje ponumerowane są od 21 do 23. Wpisanie jedynki do którejkolwiek pozycji tego rejestru powoduje zapalenie wskaźnika błędu, a w ślad za tym wysłanie sygnału przerwania do części centralnej maszyny. Bit nr 23 zapalany jest przy próbie zapisania informacji na bębnie w miejscu objętym blokadą zapisu (przy pomocy kluczy znajdujących się na tablicy sterującej bębna). Bit nr 22 zapalany jest przy próbie współpracy z pamięcią bębnową, gdy bęben nie obraca się. Wreszcie bit nr 21 zapalany jest w wyniku niespełnienia parzystości w odczytanej części słowa. Wskaźnik błędu bębna odczytuje się rozkazem CWT 32. Rozkaz CML 1059 powoduje wyzerowanie tego wskaźnika oraz przesłanie zawartości rejestru błędów na trzy ostatnie pozycje mnożnika M.

### 2.3.2. Pamięć taśmowa

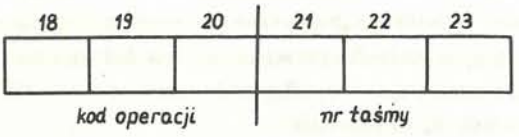
Maszyna cyfrowa ZAM 41 wyposażona jest w jednostkę pamięci taśmowej typu PT-2, która dołączona jest do pamięci operacyjnej maszyny poprzez kanał taśmy magnetycznej. Tego rodzaju połączenie sprawia, że transmisje między pamięcią taśmową i operacyjną nie wstrzymują normalnej pracy maszyny, bowiem odbywają się w czasie, w którym arytmometr i sterowanie maszyny nie komunikują się z pamięcią operacyjną.

Jednostka pamięci taśmowej PT-2 pracuje na taśmie magnetycznej szerokości 1/2 cala. Zapis wykonuje się na 8 ścieżkach z gęstością 12 bitów/mm, (6 ścieżek informacyjnych, 1 kontroli parzystości, 1 pomocnicza). Prędkość przesuwu taśmy wynosi normalnie 2 m/sek., zaś przy operacji odwijania 4 m/sek. Informacje na taśmie zapisywane są w formie bloków, których długość może zmieniać się od 20 do 8192 słów. Długość przerwy międzyblokowej wynosi 60 mm. Szybkość przesyłania informacji wynosi

6000 słów 24-bitowych/sek. Na typowej szpuli taśmy o długości 800 m można więc zapamiętać około 2 mln słów 24-bitowych.

Kanał taśmy magnetycznej jest przystosowany do dołączenia 3 jednostek taśmy magnetycznej, przy czym w danej chwili może pracować tylko jedna jednostka (nie dotyczy to odwijania, które może się odbywać jednocześnie we wszystkich jednostkach). Nie ma to jednakże znaczenia wobec aktualnego wyposażenia m.c. ZAM 41, w jedną jednostkę taśmy, oznaczoną numerem 1.

Kanał zawiera trzy rejestry sterujące: adresu, ilości, operacji i numeru taśmy oraz zawiera rejestr wskaźników służący do sygnalizacji błędów oraz początku i końca taśmy. Rejestr (rys. 2.6) operacji i numeru taśmy (6-bitowy), o bitach ponumerowanych od 18 do 23 zawiera na trzech ostatnich bitach numer taśmy (nr 1 dla jednostki taśmy m.c. ZAM 41), zaś na bitach 18–20 kod operacji. Kody operacji podane są w tablicy 2.5.




Rys. 2.6. Rejestr operacji i numer taśmy

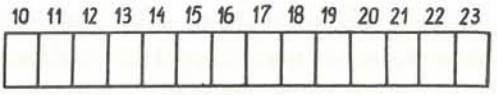
Tablica 2.5

Operacje taśmy magnetycznej

| Kod | Operacja |
|---|---|
| 0 | nic nie rób |
| 1 | odczytaj blok z taśmy magnetycznej zawierający ilość słów podaną w rejestrze ilości i umieść go w pamięci operacyjnej począwszy od komórki wskazanej przez zawartość rejestru adresu. |
| 3 | pisz blok na taśmę magnetyczną zawierający ilość słów podaną w rejestrze ilości pobierając kolejne słowa z pamięci operacyjnej począwszy od komórki wskazanej przez zawartość rejestru adresu. |
| 4 | kasuj odcinek taśmy magnetycznej, którego długość podana jest w rejestrze ilości. |
| 5 | odwiń taśmę magnetyczną do tyłu o ilość bloków podaną w rejestrze ilości. |
| 6 | odwiń taśmę magnetyczną do przodu o ilość bloków podaną w rejestrze ilości |
| 7 | odwiń taśmę magnetyczną do początku. |

Rejestr operacji i numeru taśmy ładuje się rozkazem PMP 710, który przesyła do niego 6 ostatnich bitów mnożnika M.

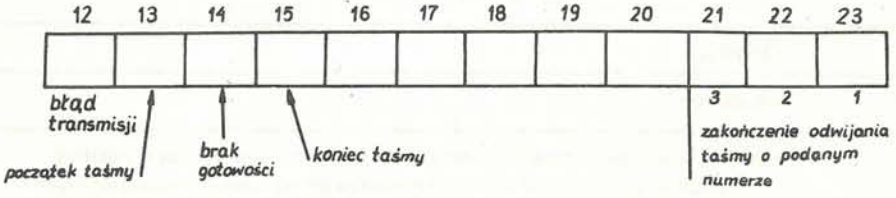
Rejestr ilości (rys. 2.7) o bitach ponumerowanych od 10 do 23 określa ilość przesyłanych słów przy operacjach zapisu/odczytu oraz ilość bloków przy operacjach odwijania taśmy do tyłu i do przodu, zaś przy operacji kasowania określa długość kasowanego odcinka taśmy. Zawartość tego rejestru nie ma wpływu na przebieg operacji odwijania taśmy do początku. Załadowanie rejestru ilości kanału następuje w wyniku wykonania rozkazu PMP 1742, który przesyła do niego 14 ostatnich bitów mnożnika M.




Rys. 2.7. Rejestry kanału: ilości i adresu

Rejestr adresu (rys. 2.7), o bitach ponumerowanych od 10 do 23 określa adres pamięci operacyjnej począwszy, od którego przesyłane są słowa do/z pamięci taśmowej. Zawartość tego rejestru przy pozostałych operacjach nie ma żadnego znaczenia. Załadowanie rejestru adresu kanału następuje w wyniku wykonania rozkazu PMP 1230, który przesyła do niego 14 ostatnich bitów mnożnika.

Rejestr wskaźników kanału taśmy magnetycznej (rys. 2.8) zawiera 12 bitów ponumerowanych od 12 do 23. Wpisanie 1 do którejkolwiek pozycji tego rejestru powoduje zapalenie wskaźnika błędu kanału taśmy magnetycznej (chociaż nie zawsze oznacza to błąd), a w ślad za tym wysłanie do części centralnej maszyny sygnału przerwania.




Rys. 2.8. Rejestr wskaźników taśmy magnetycznej

Odczyt rejestru wskaźników rozkazem CML 204 zeruje ten rejestr oraz wskaźnik błędu kanału taśmy magnetycznej (wskaźnik błędu odczytuje się rozkazem CWT 192).

Pozycje rejestru wskaźników ładowane są wskutek następujących zdarzeń:

bit nr 12 – wykrycie błędu parzystości podczas odczytu z taśmy lub błędny zapis

bit nr 13 – wydanie polecenia odwijania od tyłu (kody 5 lub 7), gdy taśma jest odwinięta do początku

bit nr 14 – wydanie polecenia, gdy taśma nie jest włączona lub odpowiedź na polecenie odwijania do początku, informujące o rozpoczęciu wykonywania odwijania

bit nr 15 – wydanie polecenia posuwu taśmy do przodu, gdy taśma została odwinięta do końca (operacje czytania, pisania, kasowania i odwijania do przodu)

bit nr 21 – zakończenie odwijania do początku taśmy nr 3

bit nr 22 – zakończenie odwijania do początku taśmy nr 2

bit nr 23 – zakończenie odwijania do początku taśmy nr 1.

Omówimy teraz wykonywanie poszczególnych operacji taśmy magnetycznej.

Operacja czytania z taśmy lub pisania na taśmę wymaga załadowania, kolejno rejestrów ilości, adresu oraz operacji i numeru taśmy. Zakończenie operacji czytania lub pisania w przypadku, gdy przy odczycie lub zapisie nie zaobserwowano błędu lub zakończenia operacji w obszarze końca taśmy (co nie jest błędem) powoduje przerwanie, przy czym LR przyjmuje wartość 140. Zakończenie omawianych operacji w obszarze końca taśmy lub wykrycie błędu podczas ich wykonywania powoduje wpisanie jedynki do odpowiednich bitów rejestru wskaźników, przerwanie, przy czym LR przyjmuje wartość 143. Odczytując wskaźnik błędu rozkazem CWT 192 oraz rejestr wskaźników rozkazem CML 204 można zidentyfikować konkretną przyczynę tego przerwania.

Operacja kasowania wykonywana jest podobnie jak operacja zapisu – nie wymaga jednak ładowania rejestru adresu. Jej zakończenie jest takie samo jak przy operacji zapisu.

Operacje odwijania do tyłu i do przodu wymagają załadowania, kolejno, rejestru ilości i rejestru operacji i numeru taśmy. Poprawne zakończenie wykonywania tych operacji sygnalizowane jest przerwaniem, przy czym LR przyjmuje wartość 143. Zakończenie lub przerwanie operacji odwijania do przodu wskutek natrafienia na koniec taśmy powoduje wpisanie 1 do pozycji 15 rejestru wskaźników, przerwanie, przy czym LR przyjmuje wartość 143. Podobnie, przy operacji odwijania do tyłu natrafienie na początek taśmy sygnalizowane wpisaniem jedynki do bitu 13 rejestru wskaźników, co powoduje przerwanie (LR = 143).

Próba zainicjowania wykonywania omówionych operacji, w przypadku, gdy taśma magnetyczna jest wyłączona, spowoduje wpisanie jedynki do pozycji nr 14 rejestru wskaźników, a w ślad za tym przerwanie (LR = 143).

Operacja odwijania taśmy do początku nie wymaga ładowania rejestrów adresu i ilości. Jeśli taśma znajduje się na początku, to nie jest oczywiście odwijana, a na polecenie odwinięcia taśmy do początku, wysyłana jest odpowiedź w postaci wpisania 1 do bitu nr 13 rejestru wskaźników („początek taśmy"), czemu towarzyszy przerwanie (LR = 143). Gdy taśma nie znajduje się na początku, odpowiedzią na polecenie jest wpisanie 1 do bitu nr 14 rejestru wskaźników (co powoduje przerwanie, LR = 143) i rozpoczęcie operacji odwijania do początku. Zakończenie odwijania sygnalizowane jest przez wpisanie jedynki do jednej z pozycji 21–23 rejestru wskaźników, co z kolei powoduje przerwanie (LR = 143).

# 3. SYSTEM OPERACYJNY

## 3.1. POJĘCIE I PRZEZNACZENIE SYSTEMU OPERACYJNEGO

Pojęcie systemu operacyjnego nie jest ściśle sprecyzowane. Treść jego formułowana jest odmiennie przez rozmaitych autorów w zależności od kontekstu, w którym ono występuje. Można jednak powiedzieć, że system operacyjny to zestaw programów zarządzających systemem liczącym (1). Z jednej strony może to oznaczać, że system operacyjny ułatwia operowanie maszyną, z drugiej zaś strony może to oznaczać, że system operacyjny organizuje proces przetwarzania, a ponadto, co jest zagadnieniem dodatkowym, dąży do jak najpełniejszego wykorzystania rozmaitych urządzeń maszyny. Systemowi operacyjnemu m.c. ZAM 41 można przypisać obie te cechy. Objaśnimy to na przykładzie.

W pkt. 2.2.1 omówiono zasady współpracy maszyny z czytnikiem taśmy papierowej. Zawarte tam informacje wskazują, że realizacja tej współpracy jest dość skomplikowana. Tak więc należy najpierw zapalić wskaźnik czytania i po nadejściu przerwania z czytnika przesłać przeczytany znak z bufora czytnika do mnożnika M. Jednocześnie należy mierzyć czas odczytu, bowiem zbyt długi czas odczytu znaku może wskazywać na zacięcie się czytnika, co wymaga bezzwłocznego wyłączenia. Ponadto, wskazane jest czytanie znaków przez czytnik z pewnym wyprzedzeniem i umieszczanie ich w określonym obszarze pamięci operacyjnej. Ma to na celu takie przygotowanie danych (przeczytanych rządków taśmy) do programu, ażeby w trakcie ich obróbki przez program (np. tworzenie liczby binarnej na podstawie wczytywanych cyfr dziesiętnych), nie potrzebował on czekać na wczytanie znaku. Funkcje takie realizuje fragment systemu operacyjnego opisany w pkt. 3.3.

Jak widać, sterowanie pracą czytnika taśmy papierowej jest dość złożone, przy czym sterowanie wymaga znajomości działania czytnika. Stąd też, należało skonstruować taki program, stanowiący integralną część wyposażenia maszyny, który to program sterowałby pracą czytnika na podstawie otrzymywanych krótkich, prostych zleceń.

Z kolei, przy większej ilości tego typu urządzeń, potrzebny jest jeszcze program nadzorujący pracę poszczególnych programów sterujących tymi urządzeniami, oraz nadzorujący wykonywanie programu (programów) użytkowego. W trakcie pracy tych programów występują bowiem długie okresy bezczynności (np. oczekiwanie na przeczytanie znaku przez czytnik), po których wymagana jest szybka reakcja systemu. Ponieważ urządzeń tych jest zazwyczaj kilka, program sterujący jednego z nich wykonuje się w czasie oczekiwania na zakończenie pracy pozostałych. W pozostałym zaś, wolnym

czasie, wykonuje się program użytkowy. Jeżeli jedno z tych urządzeń zgłosi, w czasie wykonywania programu użytkowego lub sterującego, zakończenie pracy (w formie przerwania), to zazwyczaj należy niezwłocznie przerwać aktualnie wykonywany program, aby podjąć pewne czynności w stosunku do urządzenia, które właśnie sygnalizowało koniec pracy, po czym można powrócić do przerwanego programu.

Proces ten można więc nazwać dyrygowaniem programami sterującymi i programem użytkowym, zaś sam program nadzorujący – *dyrygentem.* Funkcje dyrygenta zostaną omówione nieco szerzej w pkt. 3.2.

Tak więc przez system operacyjny rozumieć będziemy zbiór programów sterujących pracą urządzeń zewnętrznych wraz z programem „dyrygent".

## 3.2. DYRYGENT

O funkcjach programu dyrygent wspomnieliśmy już nieco w pkt. 3.1. Obecnie omówione funkcje dyrygenta podamy systematycznie. Dyrygent:

a) nadzoruje pracę programów sterujących urządzeniami zewnętrznymi, tworząc kolejki operacji do wykonania
b) kontroluje czas wykonywania się niektórych operacji w urządzeniach zewnętrznych
c) kontroluje czas wykonywania się programu użytkowego
d) steruje rozpoczęciem i zakończeniem wykonywania programu użytkowego
e) przyjmuje przerwania zegara wewnętrznego.

## 3.3. BLOKI WSPÓŁPRACY Z URZĄDZENIAMI ZEWNĘTRZNYMI

### 3.3.1. Bloki współpracy z czytnikami

Ponieważ bloki obu czytników działają identycznie, rozpatrzymy tylko blok współpracy z czytnikiem na stoliku I.

Program sterujący pracą czytnika zawiera dwa bufory o pojemności 16 słów oznaczone przez A i B, umieszczone w określonym miejscu pamięci operacyjnej. Po uruchomieniu programu „System operacyjny" rozpoczyna się operacja czytania 32 rządków taśmy, w wyniku której oba bufory zostają zapełnione. Tak więc, jeżeli nawet nie zostało wydane żadne polecenie czytania znaku (pkt. 4.1.2), to program sterujący, na polecenie dyrygenta, spowoduje przeczytanie 32 rządków taśmy.

Jeżeli później, w trakcie wykonywania programu pojawi się polecenie czytania znaku (w formie rozkazu SLR 1), to polecenie to nie spowoduje fizycznego ruchu taśmy związanego z odczytem, zaś jedynie przesłanie z bufora A do rejestru B (zgodnie z określeniem rozkazu SLR 1) pierwszego z wczytanych znaków. Dalsze rozkazy SLR 1 powodować będą również tylko przesyłanie kolejnych znaków z bufora A do rejestru B aż do wyczerpania zawartości bufora tj. pobrania z niego 16-znaków. Wówczas nazwy obu buforów zostają zamienione (przez zamianę nazw należy tu rozumieć zamianę adresów obu buforów) i ewentualne dalsze rozkazy SLR 1 powodować będą pobieranie znaków

z bufora A (po zamianie nazwy). Z chwilą opróżnienia jednego z buforów uruchamiany jest, niezależnie od ewentualnych rozkazów czytania znaku SLR 1, program sterujący czytnikiem, należący do systemu operacyjnego. Rezultatem działania tego programu, o którym zakłada się, że nie może trwać dłużej niż 1 sek., jest wczytanie do pustego bufora dalszych 16 rządków taśmy. Jeżeli program nie zakończy się w przewidzianym czasie 1 sek., co wskazuje na zacięcie czytnika lub koniec taśmy, czytnik jest wyłączany, zaś w nieuzupełnione miejsca bufora wpisuje się zera.

Zdarza się również, że program główny szybciej pobiera znaki z buforów, niż program sterujący czytnikiem jest w stanie je zapełnić. Wówczas wykonywanie programu głównego jest zawieszone, aż do czasu wypełnienia jednego z buforów.

Program sterujący czytnikiem każdorazowo przed uruchomieniem czytnika bada stan jego wskaźników programowych, których stan wskazuje na zezwolenie, lub nie, czytania taśmy. Zmiana tych wskaźników następuje w wyniku naciśnięcia przycisku „Zgłoszenie operatora" ZO na stoliku operatora.

Wyczerpanie zawartości jednego z buforów czytnika II przy braku zezwolenia na czytanie powoduje zapalenie lampki alarmu LA.

### 3.3.2. Bloki współpracy z perforatorami

Podobnie jak przy czytnikach, bloki sterujące perforatorami działają identycznie, wobec tego podamy opis bloku sterowania perforatorem 1.

Działanie bloku perforowania stanowi jakby lustrzane odbicie programu sterowania czytnikiem, opisanym w punkcie 3.3.1. Tak więc program ten zawiera również w pamięci operacyjnej dwa bufory 16-słowowe oznaczone przez A i B. Na początku pracy systemu operacyjnego oba bufory zostają wyzerowane.

W tej sytuacji pojawienie się rozkazu drukowania rządka taśmy (SLR2) spowoduje jedynie zapamiętanie rządka (czyli zawartości mnożnika M) w buforze A. Następne rozkazy SLR2 powodować będą również zapamiętywanie kolejnych rządków w buforze A, aż do wypełnienia bufora. Wówczas następuje zamiana nazw buforów i dalsze znaki wpisywane są również do bufora A (po zmianie nazwy). Jednocześnie, w chwili zapełnienia jednego z buforów uruchamiany jest program drukowania rządków taśmy zarejestrowanych w tym buforze. Program drukowania bada każdorazowo stan klucza KP na stoliku operatora – jego wciśnięcie stanowi zezwolenie na perforowanie. Brak takiego zezwolenia przy wypełnionym jednym z buforów powoduje zapalenie lampki alarmu LA.

Jeśli program główny powoduje szybsze wypełnienie buforów niż trwa ich opróżnienie związane z drukowaniem to, podobnie jak przy czytaniu, wykonywanie programu użytkowego zostaje zawieszone aż do wydrukowania jednego z buforów.

### 3.3.3. Blok współpracy z drukarką wierszową

Struktura tego bloku jest stosunkowo prosta. Wprowadzane znaki w kodzie DW (znaki w kodzie M2 są zamieniane na kod DW) ładowane do 30-słowowego (120 znaków 6-bito-

wych) bufora w pamięci operacyjnej. Właściwym poleceniem drukowania jest rozkaz posuwu papieru (również o 0 linii) lub próba umieszczenia w buforze więcej niż 120 znaków.

Przed wykonaniem tego rozkazu zawartość bufora w pamięci operacyjnej przesyłana jest do pamięci buforowej drukarki.

Przed każdym wydrukiem lub posuwem papieru badany jest wskaźnik programowy zezwolenia na drukowanie, ustawiany poprzez naciśnięcie przycisku „Zgłoszenie operatora" ZO na drukarce. Żądanie drukowania wiersza lub wysuwu papieru przy braku zezwolenia na drukowanie powoduje zapalenie się lampki alarmu LA na stoliku operatora I.

### 3.3.4. Blok współpracy z monitorem

W odróżnieniu od czytnika, perforatora czy drukarki, sterowanie monitorem, ze względu na jego zadania związane z drukowaniem informacji o przebiegu wykonywania programu oraz ewentualnym przyjmowaniu instrukcji od operatora, musi być tak zorganizowane, by polecenie drukowania choćby jednego znaku powodowało niezwłoczne jego wydrukowanie. Wynika stąd konieczność rezygnacji z tworzenia buforów, używanych w blokach sterowania czytnikami i perforatorami, co oczywiście nie wyklucza w ogóle stosowania buforów przy współpracy z monitorem, niemniej jednak sterowanie wypełnianiem buforów należeć musi do programu użytkowego.

Blok współpracy z monitorem realizuje dwie podstawowe transmisje: przesyłanie znaków z monitora do określonego obszaru pamięci operacyjnej oraz przesyłanie innego obszaru pamięci do monitora w celu wydrukowania. Odpowiednie polecenia (rozkazy SLR 8, SLR 18, SLR 9) inicjują jedynie rozpoczęcie przesyłania, które sterowane jest przez blok współpracy z monitorem. Program czytania, tj. przesyłanie z monitora do pamięci, posiada również ciekawą własność, która polega na możliwości zainicjowania procesu czytania dopiero po napisaniu przez operatora znaku – jeśli to nie nastąpiło, to operacja nie została zainicjowana, a tym samym monitor może przystąpić np. do drukowania.

Blok współpracy z monitorem interpretuje również rozkaz oczekiwania na zakończenie transmisji (SLR 20) oraz rozkaz badający, czy zainicjowana transmisja została już zakończona (SLR 21).

Wykonanie każdej operacji poprzedzone jest badaniem programowego wskaźnika gotowości monitora. Jeżeli monitor (dalekopis) nie jest przygotowany do pracy, to operacja zostaje zawieszona aż do przygotowania monitora (co jest sygnalizowane przez operatora naciśnięciem przycisku ZO na pulpicie sterującym monitora – naciśnięcie powoduje zanegowanie stanu programowego wskaźnika gotowości). Jednocześnie zapala się lampka alarmu LA na stoliku I.

### 3.3.5. Blok współpracy z czytnikiem kart

Blok ten inicjuje proces czytania karty, przyjmuje przerwania nadchodzące po wczytaniu kolejnych kolumn, przesyła odczytane kolumny do pamięci a także przyjmuje przer-

wania spowodowane błędnym odczytem. Blok posiada szczególnie prostą budowę, co wynika m.in. z tego, że wykonywanie rozkazu czytania karty (SLR 15) kończy się dopiero po wczytaniu karty, a nie po zainicjowaniu operacji, co ma miejsce np. przy monitorze.

Przed rozpoczęciem czytania każdej karty badany jest programowy wskaźnik gotowości czytnika kart (każde naciśnięcie przycisku „Zgłoszenie operatora" ZO neguje zawartość tego wskaźnika). Jeżeli czytnik kart nie jest przygotowany do pracy to operacja zostaje zawieszona aż do przygotowania czytnika kart. Jednocześnie zapala się lampka alarmu LA na stoliku I.

### 3.3.6. Blok współpracy z pamięcią bębnową

Program współpracy z pamięcią bębnową steruje transmisjami z pamięci bębnowej do operacyjnej i na odwrót. Transmisje te, w odróżnieniu od transmisji do/z pamięci taśmowej, realizowane są przez część centralną maszyny, co uniemożliwia jednoczesne wykonywanie innego programu.

Polecenie wykonania transmisji (SLR 3) powoduje przygotowanie odpowiedniej sekwencji rozkazów realizujących przesyłanie (pkt. 2.3.1), zapalenie wskaźnika pisania lub czytania bębna, a następnie powrót do programu użytkowego. Na 600$\mu$s przed rozpoczęciem transmisji pamięć bębnowa wysyła przerwanie do części centralnej, co powoduje ponowne wejście do bloku współpracy z pamięcią bębnową i rozpoczęcie przesyłania.

Blok nie interpretuje przerwania spowodowanego błędem bębna – w takim wypadku maszyna jest zatrzymywana.

### 3.3.7. Blok współpracy z pamięcią taśmową

Zadaniem bloku jest inicjowanie transmisji do/z pamięci taśmowej oraz inicjowanie pozostałych operacji taśmowych. Ze względu na połączenie pamięci taśmowej z operacyjną poprzez kanał pamięci taśmowej struktura omawianego bloku różni się w istotny sposób od pozostałych bloków. Istota zagadnienia polega na tym, że kanał wysyła sygnał przerwania po wykonaniu całej transmisji, nie zaś jej fragmentu, jak to ma miejsce w pozostałych urządzeniach. Jednakże, z uwagi na dość duży stopień skomplikowania pamięci taśmowej oraz ilość czynności, które może ona wykonywać, blok współpracy z pamięcią taśmową jest dość obszerny.

Jak już wspomniano w pkt. 2.3.2 kanał taśmy magnetycznej jest przystosowany do dołączenia trzech jednostek taśmy megnetycznej (a przy pewnych modyfikacjach nawet 7). Stąd również blok współpracy z pamięcią taśmową jest przystosowany do sterowania pracą 7 taśm, przy czym w danej chwili może pracować tylko jedna jednostka. Sprawa ta nie jest istotna, ponieważ m.c. ZAM 41 posiada tylko 1 jednostkę taśmy magnetycznej.

Zainicjowanie operacji taśmy (rozkazem SLR 7) powoduje wpisanie do odpowiednich rejestrów kanału parametrów operacji, po czym sterowanie zostaje oddane programowi użytkowemu.

W przypadku poprawnego zakończenia transmisji blok współpracy z pamięcią taśmową może zainicjować następną transmisję. Przyjmuje się, że czas operacji odwijania taśmy do początku nie powinien przekroczyć 300 sek – przekroczenie tego czasu wskazuje na uszkodzenie jednostki taśmy magnetycznej.

Niepoprawne zakończenie operacji zapisu lub odczytu powoduje trzykrotne jej powtórzenie, i jeśli nie da to wyniku pozytywnego, to w przypadku operacji czytania rezygnuje sisię z dalszych prób odczytu, zapalając tylko odpowiedni wskaźnik programowy błędu, zaś w przypadku operacji pisania kasuje się pewien odcinek taśmy, a następnie za odcinkiem skasowanym, dokonuje się ponownie trzykrotnej próby zapisu itd.

Jeżeli w odpowiedzi na wydane polecenie kanał poinformuje blok współpracy z pamięcią taśmową o braku gotowości taśmy, to dla wszystkich operacji, z wyjątkiem odwijania do początku, odpowiedź jest interpretowana jako uszkodzenie jednostki taśmy magnetycznej.

Po sygnalizacji końca taśmy, blok współpracy dopuszcza jeszcze możliwość zapisu lub odczytu 1 bloku taśmowego. Niedozwolone są natomiast operacje: kasuj i odwiń do przodu.

Identyfikacja uszkodzenia taśmy magnetycznej nie powoduje przerwania aktualnie wykonywanego programu użytkowego. Przeciwnie, jest on wykonywany dopóty, dopóki program użytkowy nie zapyta o zakończenie poprzednio zainicjowanej operacji lub spróbuje zainicjować nową operację. Dopiero wówczas wykonywanie programu użytkowego zostaje zakonczone, przy czym drukowany jest tekst AWARIA TAŚMY.

## 3.4. TRANSLATORY M.C. ZAM 41

Maszyna cyfrowa ZAM 41 wyposażona jest w dwa translatory języków: PJP, który stanowi przedmiot niniejszego opracowania, oraz ZAM–Algolu. Translatory te zgrupowane są w dwóch zestawach różniących się wielkością zajmowanej pamięci bębnowej, systemem operacyjnym i niektórymi parametrami.

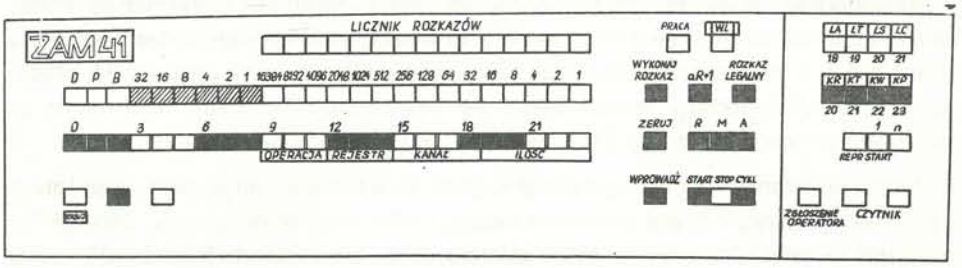
Zestaw I zawiera skrócony system operacyjny (bez bloków współpracy z pamięcią taśmową, monitorem, czytnikiem kart i stolikiem operatora II), translator języka PJP oraz translator języka ZAM–Algol. Zestaw ten zajmuje na bębnie 16 384 słowa.

Zestaw II zawiera pełny system operacyjny oraz translator języka PJP. Zestaw ten zajmuje na bębnie 8192 słowa.

Prócz tego każdy z zestawów zawiera także podprogramy standardowe definiujące operacje drukowania liczb, tekstów itp., czytania oraz operacje arytmetyczne zmiennoprzecinkowe. Opis tych rozkazów podano w pkt. 4.1.3.

## 3.5. CZYNNOŚCI OPERATORSKIE PRZY MASZYNIE

W niniejszym podrozdziale przedstawimy podstawowe czynności operatorskie przy maszynie, przy czym pominiemy, z nielicznymi, uzasadnionymi wyjątkami, czynności techniczne, jak np. włączanie zasilania urządzeń. Z tego powodu umieszczone w dalszej części tego podrozdziału czynności typu: włącz lub wyłącz urządzenie (np. perforator) oznaczać będą wyłącznie sygnalizację gotowości urządzenia do pracy, przekazywaną przez operatora systemowi operacyjnemu.




Rys. 3.1. Pulpit sterujący stolika operatora I.

Szczególną rolę w sterowaniu przebiegiem wykonywania programów odgrywają klucze, przyciski i lampki stolika operatora I przedstawione na rys. 3.1. Znaczenie poszczególnych przycisków i kluczy zostanie omówione w dalszej części niniejszego podrozdziału.

### 3.5.1. Uruchomienie systemu operacyjnego

System operacyjny wraz z translatorami przechowywany jest normalnie w zablokowanej części pamięci bębnowej i zajmuje w zależności od zestawu 8 do 16 k pamięci. Uruchomienie systemu operacyjnego wymaga przepisania jego fragmentu do pamięci operacyjnej. Służy to tego krótki, 10 rozkazowy program wprowadzany do PAO przy pomocy przycisku WPROWADŹ znajdującego się na stoliku operatora I. W celu wprowadzenia tego programu, zwanego popularnie „ściągaczką", należy wykonać następujące czynności:

a) nacisnąć klucz R a następnie przycisk ZERUJ
b) wpisać do rejestru R przy pomocy przycisków stolika I liczbę 2048+1024+512
c) nacisnąć przycisk WYKONAJ ROZKAZ
d) podłożyć taśmę z programem pod czytnik
e) nacisnąć przycisk WPROWADŹ 10 razy
f) nacisnąć przycisk START
g) po kilku sekundach nacisnąć przycisk KASUJ
h) nacisnąć przycisk START.

W wyniku wykonania powyższych czynności system operacyjny zostanie wprowadzony do PAO i uruchomiony. Objawem poprawnej pracy systemu po jego uruchomieniu jest palenie się lampek LT i LS stolika operatora I.

Jednocześnie w liczniku rozkazów LR pojawia się liczba 385 (256+128+1).

### 3.5.2. Regeneracja systemu operacyjnego

W pracy systemu operacyjnego mogą czasami wystąpić zakłócenia spowodowane chwilowymi błędami pracy maszyny. Ponadto, włączanie urządzeń zewnętrznych nie może się odbywać w trakcie pracy systemu operacyjnego, ponieważ powstające przy tym zakłócenia powodują na ogół jego uszkodzenie. W takich wypadkach można zregenerować system operacyjny wykonując następujące czynności:

a) nacisnąć klucz KR
b) nacisnąć przycisk ZO1

Nastąpi wówczas przepisanie fragmentu systemu operacyjnego z pamięci bębnowej do operacyjnej, po czym maszyna zatrzyma się. Można wówczas wykonać pewne prace przy urządzeniach zewnętrznych jak np. zmianę papieru w drukarce wierszowej (podczas wymiany papieru silnik musi być wyłączony).

c) nacisnąć przycisk KASUJ
d) nacisnąć przycisk START.

Czasami wykonanie podanych czynności nie uruchamia systemu operacyjnego. Można wówczas zastosować nieco inny sposób, który polega na wykonaniu następujących czynności:

a) nacisnąć przycisk KASUJ
b) nacisnąć klucz R
c) nacisnąć przycisk ZERUJ
d) wpisać do rejestru R przy pomocy przycisków stolika I liczbę 1138 (I zestaw z Algolem ) lub 2148 (II zestaw bez Algolu)
e) nacisnąć przycisk WYKONAJ ROZKAZ
f) nacisnąć przycisk START
g) po kilku sekundach nacisnąć przycisk KASUJ
h) nacisnąć przycisk START.

Jeśli wykonanie tych czynności również nie uruchomi systemu operacyjnego, to trzeba wówczas wykonać czynności opisane w pkt. 3.5.1.

### 3.5.3. Sterowanie wykonywaniem programów

Klucze, lampki i przyciski stolika operatora I, prócz funkcji związanych z włączaniem i wyłączaniem czytnika i perforatora, pełnią również funkcje sterujące i sygnalizujące przebieg wykonywania się programu.

Rozpatrzmy na wstępie przebieg sterowania programem znajdującym się na jednej taśmie. Palenie się lampki LS oznacza, że system może w każdej chwili rozpocząć czytanie programu. Należy wówczas podłożyć taśmę z programem pod czytnik, wycisnąć klucze KR i KW, a następnie nacisnąć przycisk ZO stolika I. Nastąpi wówczas przeczytanie czołówki programu zawierającej tytuł programu oraz nazwę translatora, a następnie sterowanie dalszym odczytem programu zostanie przekazane odpowiedniemu translatorowi.

Jednocześnie tytuł programu zostanie wydrukowany na drukarce, monitorze lub jednym z perforatorów (w zależności od położenia kluczy zamiany perforatorów opisanych w pkt. 3.5.4). Po odczytaniu czołówki programu gaśnie lampka LS. Jeśli w czołówce programu podano błędną nazwę translatora to zostanie wydrukowany tekst „JAKI TRANSLATOR", po czym nastąpi przejście do stanu oczekiwania na następny program.

Po wczytaniu całego programu następuje jego translacja, która w przypadku programów w języku PJP nie trwa zwykle dłużej niż kilka sekund. Po translacji programu w języku PJP drukowany jest słownik etykiet. Jeśli translator wykryje błędy w programie, to po ich wypisaniu następuje przejście w stan oczekiwania na wprowadzenie następnego programu. Gdy program jest poprawny to następuje wypisanie tekstu STA < adres początku programu > i rozpoczęcie jego wykonywania, podczas którego mogą pracować rozmaite urządzenia zewnętrzne.

Na ogół obsługa tych urządzeń, z wyjątkiem czytników taśmy i kart, jest konwencjonalna i jej wpływ na przebieg wykonywania programu polegać może jedynie na chwilowym zatrzymaniu wykonywania programu poprzez programowe wyłączenie (tj. przez naciśnięcie ZO) któregoś urządzenia.

Sterowanie przebiegiem czytania może w pewnych sytuacjach zakończyć wykonywanie programu. I tak, gdy program lub dane umieszczone są na kilku taśmach lub w kilku grupach kart perforowanych, to podczas wczytywania kolejnych taśm lub grup kart klucz KT (nr 21) poszczególnych urządzeń musi być wciśnięty. Po wczytaniu ostatniej taśmy lub grupy kart klucz KT należy wycisnąć. Jeśli wówczas program żąda dalszego ciągu taśmy lub następnej grupy kart, to naciśnięcie przycisku ZO na pulpicie sterującym stolika operatora I, II lub czytnika kart powoduje zakończenie wykonywania programu, czemu towarzyszy wydruk (objaśniony w pkt. 4.1.2.):

LR = < licznik rozkazów > WYCZERPANO WEJŚCIE (KART)
POZOSTAŁO CYKLI < ilość cykli >.

Przyczyną zakończenia wykonywania programu może być:

a) przekroczenie zadeklarowanego czasu wykonywania programu
b) próba wykonania rozkazu nielegalnego
c) brak dalszego ciągu taśmy lub kart perforowanych
d) wykonanie rozkazu SLR 40 (zakończenie poprawne)
e) wykonanie rozkazu SLR o adresie mniejszym od BD i różnym od opisanych w pkt. 4.1.2
f) wykonanie rozkazu SLR spośród opisanych w pkt. 4.1.2 z niepoprawnymi parametrami
g) przekroczenie zadeklarowanej wielkości wydruku dla perforatorów lub drukarki
h) uszkodzenie pamięci taśmowej
i) wykonanie niektórych rozkazów taśmy magnetycznej, gdy ta znajduje się na początku lub końcu.

Zdarza się również, że program w widoczny sposób działa niepoprawnie, a nie następuje jego zakończenie wskutek ww. przyczyn. Wówczas operator może zakończyć

wykonywanie programu poprzez wciśnięcie klucza KW i naciśnięcie przycisku ZO stolika I. Towarzyszy temu wydruk (objaśniony w pkt. 4.1.2):

LR = < licznik rozkazów 1 > LUB < licznik rozkazów 2 >
LUB < licznik rozkazów 3 > PROGRAM WYRZUCONO
POZOSTAŁO CYKLI < ilość cykli >
B = < zawartość B-rejestru >

Każda z wymienionych przyczyn zakończenia programu drukowana jest wraz z parametrami określającymi miejsce przerwania wykonywania programu, ilość czasu, która pozostała w stosunku do zaplanowanej itp. Dokładna postać wydruków towarzyszących podanym zakończeniom wykonywania programu podana jest przy opisie rozkazów (pkt. 4.1.2) oraz dyrektyw i parametrów programu w języku PJP (pkt. 5.3).

Omówimy teraz lampki stolika operatora I (rys. 3.1). Lampka LS sygnalizuje stan oczekiwania maszyny na rozpoczęcie czytania programu. Lampka LA sygnalizuje alarm, spowodowany najczęściej odłączeniem urządzenia od systemu operacyjnego. Operator powinien wówczas bezzwłocznie ustalić przyczynę alarmu i usunąć ją, poprzez przygotowanie urządzenia do pracy i naciśnięcie przycisku ZO (lub klucza 23, gdy przyczyną alarmu jest perforator) na pulpicie sterującym odłączonego urządzenia. Przyczyny alarmów zostaną szczegółowo omówione przy opisie obsługi operatorskiej poszczególnych urządzeń. Pozostałe lampki stolika operatora I zostaną omówione w pkt. 3.5.5.

Zestawienie czynności wykonywanych przez system operacyjny wskutek naciśnięcia przycisku ZO na stoliku I podaje tablica 3.1.

Tablica3.1

Czynności systemu operacyjnego po naciśnięciu przycisku „Zgłoszenie operatora" ZO stolika I

<table>
<tr><td colspan="6">Klucz KR</td></tr>
<tr><td rowspan="5">wciśnięty</td><td colspan="5">wyciśnięty</td></tr>
<tr><td colspan="5">Klucz KW</td></tr>
<tr><td rowspan="3">wciśnięty</td><td colspan="4">wyciśnięty</td></tr>
<tr><td colspan="4">lampki</td></tr>
<tr><td>LC = 1</td><td>LT=LC=0</td><td colspan="2">LT = 1<br>Klucz KT</td></tr>
<tr><td colspan="4"></td><td>wciśnięty</td><td>wyciśnięty</td></tr>
<tr><td>regeneracja systemu operacyjnego</td><td>wyrzucenie programu</td><td>odłączenie czytnika od systemu</td><td colspan="2">dołączenie czytnika do systemu</td><td>zakończenie wykonywania programu</td></tr>
</table>

Na zakończenie wspomnimy jeszcze, że w trakcie wykonywania lub translacji programu pojawiają się czasami zakłócenia, w wyniku których maszyna zatrzymuje się. Czasami udaje się ponownie uruchomić program od miejsca, w którym nastąpiło zatrzymanie maszyny. Takie możliwości istnieją, gdy po zatrzymaniu maszyny LR = 320 – oznacza to, że maszyna przyjęła przerwanie typu „zgłoszenie operatora lub błędy" pochodzące od wskaźnika, którego zawartość nie jest odczytywana i interpretowana przez system operacyjny. System operacyjny bowiem, nie odczytuje wskaźników błędów czytników taśmy perforowanej oraz wskaźnika błędu drukarki wierszowej; ponadto I zestaw systemu (z Algolem) nie odczytuje wskaźników związanych z taśmą magnetyczną, czytnikiem kart, monitorem i stolikiem operatora II. W celu ponownego uruchomienia programu należy do rejestru rozkazów, w celu zgaszenia zapalonego wskaźnika, wpisywać i wykonywać rozkazy CWT czytające wskaźniki rozmaitych urządzeń. Następnie należy do rejestru rozkazów wpisać i wykonać rozkaz WRO (52) z adresem równym aktualnej zawartości rejestru blokady dolnej BD, po czym nacisnąć przycisk START.

Ustawienie LR = 325 (I zestaw z Algolem) lub LR = 333 (II zestaw bez Algolu) wskazuje na pojawienie się błędu podczas transmisji bębnowej. Można wówczas ponownie uruchomić zatrzymany program wykonując kolejno rozkazy: CML 1059, CWT 32 i rozkaz WRO, tak jak opisano wyżej. Oczywiście wyniki programu mogą być w takim przypadku niepoprawne.

### 3.5.4. Klucze zamiany perforatorów

Translator języka PJP drukuje informacje o wprowadzonym programie na perforatorze I (tj. wykonuje rozkaz SLR 2). Często wygodniej jest informacje te wyprowadzać poprzez inne urządzenia jak np. drukarkę wierszową czy monitor. Można to zrealizować ustawiając odpowiednio klucze zamiany perforatorów, umieszczone na pulpicie drukarki wierszowej.

Tablica 3.2

Klucze zamiany perforatorów

| Perforator II | | | Perforator I | | |
|---|---|---|---|---|---|
| Położenie kluczy | | zostaje zamieniony na | Położenie kluczy | | zostaje zamieniony na |
| 20 | 21 | | 22 | 23 | |
| 0 | 0 | monitor | 0 | 0 | monitor |
| 0 | 1 | perfòrator I | 0 | 1 | perforator I |
| 1 | 0 | perforator II | 1 | 0 | perforator II |
| 1 | 1 | drukarkę wierszową | 1 | 1 | drukarkę wierszową |

0 – klucz wyciśnięty
1 – klucz wciśnięty

Klucze oznaczone numerami 20 i 21 służą do zamiany perforatora II na perforator I, drukarkę lub monitor, zaś klucze 22 i 23 służą do zamiany perforatora 1. Klucze należy ustawić przed rozpoczęciem wprowadzania programu do maszyny. Zamiana perforatorów obowiązuje aż do zakończenia wykonywania programu.

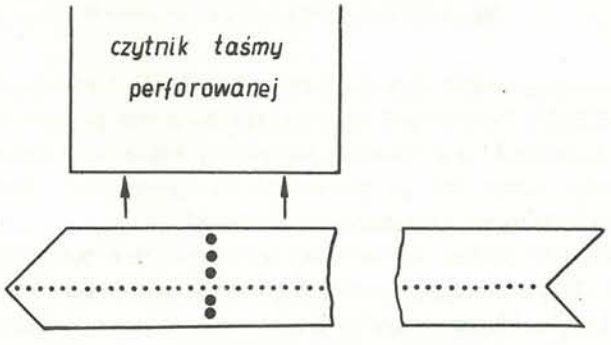
### 3.5.5. Obsługa czytnika taśmy papierowej

Zarówno czytnik jak i system operacyjny, przystosowane są do czytania taśmy perforowanej 5, 6, 7 i 8-kanałowej. W celu przygotowania czytnika do czytania taśmy określonego typu należy przełącznik znajdujący się na przedniej ścianie czytnika ustawić w odpowiedniej pozycji (zwykle przełącznik ten ustawiony jest na pozycji „5" – czytanie taśmy 5-kanałowej). Podczas obracania tego przełącznika należy jednocześnie naciskać dźwignię czytnika.

Taśma z programem lub danymi musi posiadać na początku i końcu 20 cm odcinki czyste (blanki). Dopuszcza się również sklejenia taśmy – muszą one jednakże być wykonane starannie, tak by nie powodowały zatrzymania wczytywania taśmy (tego rodzaju zatrzymania system operacyjny interpretuje jako brak taśmy w czytniku, co jest sygnalizowane zapaleniem się lampki LT; w celu ponownego uruchomienia czytnika, często wystarcza tylko włączenie czytnika przyciskiem ZO, przy czym klucz KT musi być wciśnięty). Taśma musi być zakończona trójkątnym wycięciem, którego wierzchołek znajduje się przy rządku prowadzącym.

Przygotowanie czytnika do czytania taśmy perforowanej wymaga wykonania następujących czynności:

a) upewnić się, czy typ wczytywanej taśmy jest zgodny z ustawionym na czytniku
b) upewnić się, czy czytnik jest odłączony od systemu operacyjnego (lampka LC nie pali się); w przeciwnym razie nacisnąć przycisk ZO
c) włączyć czytnik do sieci kluczem „WŁĄCZENIE CZYTNIKA"
d) ustawić taśmę wg rys. 3.2 i podłożyć ją pod czytnik. Dla każdego typu taśmy obowiązuje reguła, polegająca na takim ustawieniu taśmy, by rządek prowadzący odległy był od brzegu taśmy, bliższego czytnikowi, o 3 dziurki.



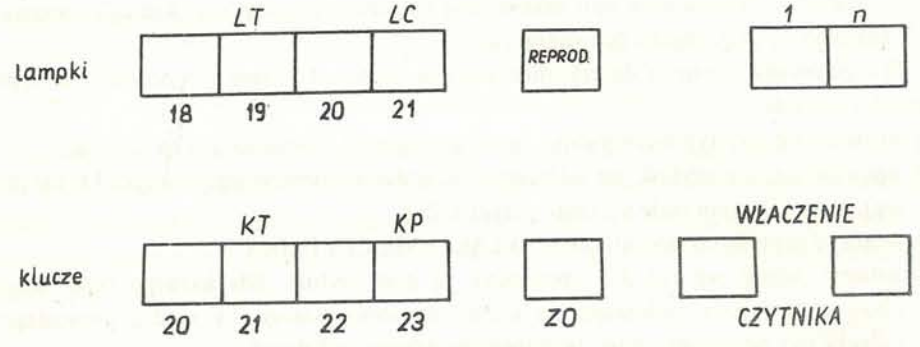

Rys. 3.2. Załadowanie taśmy perforowanej

e) nacisnąć zielony przycisk czytnika – taśma przesunie się o 1 rządek
f) nacisnąć czerwony przycisk czytnika
g) sprawdzić czy klucze KR i KW stolika I znajdują się w położeniu „wyciśnięty", zaleca się wcisnąć klucz KT
h) włączyć czytnik do systemu naciskając przycisk „Zgłoszenie operatora" ZO.

Po wykonaniu tych czynności czytnik może, na polecenie systemu operacyjnego („inspirowanego" przez translator lub program użytkowy), rozpocząć czytanie taśmy.

Jeżeli program (lub dane) składa się z grupy taśm, to po wczytaniu każdej z nich system operacyjny będzie sygnalizował koniec taśmy zapalając lampkę LT. Wówczas należy następny fragment programu przygotować do czytania, wykonując czynności omówione w pkt. d-h.

Jak już wspomniano w pkt. 3.3.1 znaki z czytników wczytywane są z pewnym wyprzedzeniem w stosunku do żądań programu. Należy o tym pamiętać przy zakładaniu taśm z danymi. Zdarza się bowiem dość często, że dane do programu umieszczone są na taśmie tuż za nim (bez separującego odcinka pustej taśmy). Oczywiście, dla tych danych poprawny program da poprawne wyniki. Jednakże, gdy operator wymieni dane, wyłączając czytnik przed końcem translacji i podkładając taśmę z innymi danymi, uzyskane wyniki są zwykle błędne, co spowodowane jest wczytaniem fragmentu poprzednich danych.




Rys. 3.3. Pulpit sterujący stolika operatora II

Włączaniem i wyłączaniem czytnika steruje przycisk ZO. Stan czytnika sygnalizowany jest lampkami LT, LC. Czytnik jest włączony (programowo) gdy pali się lampka LC, jest wyłączony gdy lampka LC jest zgaszona. Lampka LT wskazuje na wykrycie przez system operacyjny końca taśmy (lub jej ztrzymania na sklejeniu itp.). Wszelkie prace przy czytniku można wykonywać przy zgaszonej lampce LC.

Przedstawiany opis obsługi operatorskiej czytników taśmy perforowanej dotyczy obu stolików I i II. Pulpit sterujący stolika II (rys. 3.3) jest znacznie mniejszy, bowiem posiada tylko lampki, klucze i przyciski niezbędne do sterowania czytnikiem II i perforatorem II. Należy jeszcze dodać, że żądanie czytania przez czytnik II, w przypadku gdy czytnik jest odłączony (LC = 0), powoduje zapalenie lampki alarmu LA na stoliku I.

### 3.5.6. Obsługa perforatora taśmy papierowej

Podobnie jak czytniki, perforatory są również przystosowane do perforowania taśmy 5, 6, 7 i 8-kanałowej. Przełączenie wymaga jednak zmiany blokady mechanicznej perforatora, która to czynność może być wykonana tylko przez obsługę maszyny.

Taśmę papierową zakłada się do perforatora w następujący sposób:

a) otworzyć zewnętrzną osłonę perforatora
b) nacisnąć dźwignię przy mechanizmie perforacji
c) wyciągnąć poprzednio wprowadzoną taśmę
d) założyć krążek taśmy i przeciągać papier pod mechanizmem perforującym
e) zamknąć mechanizm perforowania i zewnętrzną obudowę
f) nacisnąć na kilka sek. klucz na perforatorze
g) oderwać wyperforowaną taśmę
h) wypuścić jeszcze kilka centymetrów taśmy.

Wciśnięcie klucza KP powoduje dołączenie (programowe) perforatora do systemu operacyjnego, zaś wyciśnięcie powoduje odłączenie i ewentualną sygnalizację alarmu na stoliku I (lampka LA), jeżeli perforator otrzymał polecenie drukowania.

Taśma papierowa znajdująca się przy końcu krążka jest zwykle w wyraźny sposób zabarwiona. Pojawienie się takiej taśmy wskazuje na potrzebę założenia nowego krążka. W tym celu należy zatrzymać perforowanie (poprzez wyciśnięcie klucza KP, przy czym nie należy zwracać uwagi na alarm, który zostanie natychmiast sygnalizowany), wypuścić około 20 cm czystej taśmy, oderwać taśmę i założyć nowy krążek, tak jak to opisano wyżej, następnie wcisnąć ponownie klucz KP.

### 3.5.7. Obsługa drukarki wierszowej

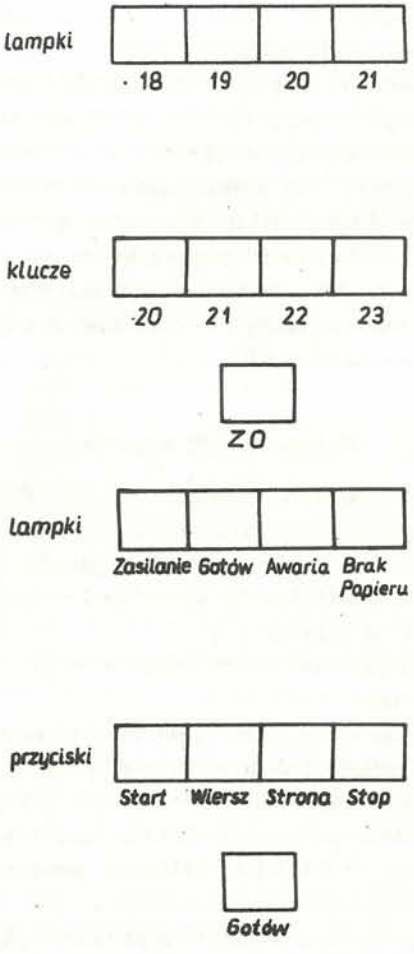
Podobnie jak w innych urządzeniach, naciśnięcie przycisku ZO na pulpicie drukarki powoduje dołączenie lub odłączenie (programowe) drukarki do/od systemu operacyjnego. Sygnalizuje to lampka 21 na pulpicie drukarki (lampka zapalona – drukarka dołączona, zgaszona – odłączona). Odłączenie drukarki od systemu w trakcie drukowania sygnalizowane jest alarmem na stoliku I.

Klucze umieszczone na pulpicie sterującym drukarki (nr 20–23) służą do zamiany perforatorów, która została opisana w pkt. 3.5.4.

Zakładanie papieru do drukarki nie może się odbywać w trakcie wykonywania programu, ponieważ w trakcie wymiany silnik drukarki musi być wyłączony, zaś z kolei jego włączenie powoduje powstanie zakłóceń, które na ogół przerywają wykonywanie programu. Tak więc zakładaniu papieru do drukarki musi towarzyszyć regeneracja systemu operacyjnego opisana w pkt. 3.5.2. Założenie papieru wymaga wykonania następujących czynności:

a) zainicjować regenerację systemu operacyjnego (czynności a, b pkt. 3.5.2)
b) wyłączyć silnik drukarki naciskając przycisk „STOP" na pulpicie drukarki
c) odchylić górną i dolną osłonę mechanizmu drukarki

d) odblokować i odchylić mechanizm drukujący
e) przesunąć papier, wsuwając go z tyłu drukarki
f) założyć papier na ząbki prowadzące
g) zamknąć i zablokować mechanizm drukarki
h) włączyć silnik drukarki przyciskiem START
i) nacisnąć kilka razy przycisk „STRONA", co powoduje pewien przesuw papieru
j) podłożyć papier pod osłonę plastikową drukarki
k) zamknąć dolną i górną osłonę drukarki
l) nacisnąć przycisk GOTÓW, w wyniku czego powinna się zapalić lampka ZASILANIE
ł) zakończyć regenerację systemu operacyjnego (czynności c, d, pkt. 3.5.2).

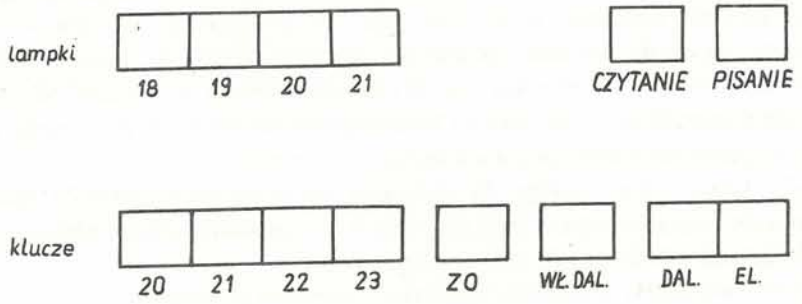



**Rys. 3.4. Pulpit sterujący drukarki wierszowej**

### 3.5.8. Obsługa monitora

Monitor umożliwia przesyłanie informacji z maszyny (pisanie), jak i do maszyny (czytanie). O typie przesyłania decyduje program użytkowy.

Operator steruje pracą monitora przy pomocy pulpitu sterującego, przedstawionego na rys. 3.5.




Rys. 3.5. Pulpit sterujący monitora

Dołączanie lub odłączanie (programowe) monitora następuje w wyniku naciśnięcia przycisku ZO, przy czym dołączenie monitora do systemu sygnalizowane jest paleniem się lampki 21. Ponadto, rozpoczynając operację czytania przed naciśnięciem przycisku ZO trzeba włączyć silnik dalekopisu poprzez naciśnięcie przycisku WŁĄCZENIE DALEKOPISU.

Proces *pisania* na monitorze inicjowany jest przez program użytkowy lub translator. Zapala się wówczas czerwona lampka PISANIE i dalekopis rozpoczyna pisanie tekstu. Niedopuszczalne są wówczas jakiekolwiek manipulacje przy dalekopisie. Jeżeli zachodzi potrzeba wymiany papieru lub taśmy barwiącej, to wstrzymujemy operację naciskając przycisk ZO. Gaśnie wówczas lampka 21, zaś zapala się lampka alarmu LA na stoliku I. Wymiana papieru na dalekopisie jest prosta – wymaga jednak wyłączenia go z sieci. Po założeniu papieru włączamy dalekopis do sieci, naciskamy przycisk WŁĄCZENIE DALEKOPISU, a w końcu przycisk ZO, co powoduje ponowne dołączenie dalekopisu do systemu, zapalenie się lampki 21 i kontynuację drukowania.

Proces *czytania* z monitora może przebiegać w trybie czytania normalnego (klucz 23 wyciśnięty) lub wymuszonego (klucz 23 wciśnięty). W trybie czytania normalnego operacja czytania rozpoczyna się z chwilą napisania przez operatora pierwszego znaku. Jeżeli to nie nastąpi, operację czytania uważa się za niezainicjowaną, a tym samym, można wykonywać dalsze rozkazy dotyczące monitora (np. pisanie).

Jeśli natomiast operator naciśnie choćby jeden klawisz dalekopisu, to zakończenie operacji czytania nastąpi dopiero po napisaniu znaku ↑ , w myśl reguł operacji czytania, które podamy dalej. W trybie czytania normalnego program użytkowy nie może więc bezzwłocznie po zainicjowaniu operacji czytania pytać o jej zakończenie, ponieważ operator musi mieć co najmniej kilkanaście sekund na zastanowienie się i ewentualne rozpoczęcie przesyłania informacji.

W trybie czytania wymuszonego każda zainicjowana operacja czytania kończy się dopiero z chwilą naciśnięcia przez operatora klawisza ↑ na dalekopisie.

Dobrze zorganizowany program inicjuje zwykle operację czytania na długo przedtem, nim informacje przesłane z monitora do maszyny będą wykorzystane. W ten sposób operator ma dostatecznie dużo czasu na przesłanie informacji. Jednakże w praktyce wymaganie to jest dość trudne do spełnienia, toteż często, wkrótce po zainicjowaniu operacji czytania większość programów czeka na jej zakończenie. Wówczas system operacyjny sygnalizuje ten stan operatorowi, zapalając lampkę 20, przez co informuje operatora, że wykonywanie programu użytkowego zostało zawieszone aż do czasu zakończenia operacji czytania. Stanowi to wskazówkę dla operatora, że powinien szybko zakończyć przesyłanie informacji do maszyny.

Należy także zwrócić uwagę, że wciśnięcie lub wyciśnięcie klucza 23 spowoduje zmianę trybu czytania dopiero przy ponownym zainicjowaniu operacji czytania. Zaleca się tryb czytania ustalać na cały czas pracy programu.

Podamy teraz reguły przesyłania informacji z monitora do maszyny:

a) operatorowi wolno przesyłać informacje do maszyny tylko i tylko wtedy, gdy maszyna mu na to zezwala tj. świecą się lampki: CZYTANIE, nr 19 i nr 21,
b) przesyłane informacje nie mogą być dłuższe niż 1 wiersz, który musi się zaczynać znakiem cyfr lub liter i kończyć się znakiem ↑
c) do maszyny przesyłane są wszystkie, dostępne z klawiatury znaki kodu M2. Niektóre z nich, jak znaki puste BL (blanki), znaki powrotu karetki CR i nowej linii LF są w efekcie pomijane,
d) jeśli operator pomylił się w przesyłanym do maszyny tekście – może błąd naprawić przenosząc się do nowego wiersza tabulogramu, czyli wypisać sekwencję znaków: powrót karetki CR i nowa linia LF. W tym przypadku znak ↑ należy umieścić na końcu tekstu poprawnego. Cały tekst wypisany w poprzednim wierszu ulega skasowaniu w pamięci maszyny,
e) wszelkie manipulacje z monitorem należy zawsze poprzedzać wyłączeniem monitora.

System operacyjny sygnalizuje alarm, jeśli nie może zainicjować operacji monitora ze względu na odłączenie go od systemu. Przyczyną alarmu może być także żądanie końca operacji czytania (sygnalizują to lampki 20 monitora i LA stolika I.).

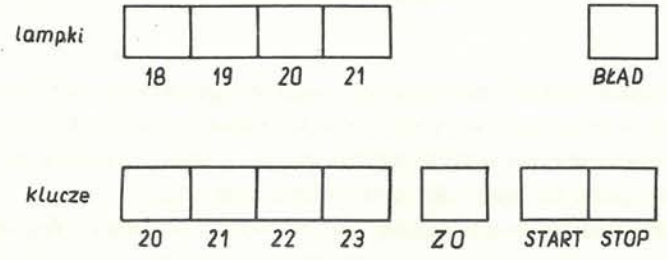
### 3.5.9. Obsługa czytnika kart

W celu uruchomienia czytnika należy wykonać następujące czynności:

a) załadować karty
b) włączyć silnik czytnika przełącznikiem na jego obudowie
c) nacisnąć przycisk START na pulpicie sterującym
d) nacisnąć przycisk „Zgłoszenie operatora" ZO na pulpicie sterującym.

Po wykonaniu tych czynności czytnik może w każdej chwili, na polecenie programu użytkowego, rozpocząć czytanie.

Jeśli w trakcie czytania nastąpi zacięcie czytnika lub zabraknie kart, to zapalają się lampki: nr 19 i BŁĄD na pulpicie sterującym oraz lampka alarmu LA na stoliku I. Należy wówczas nacisnąć klucz STOP, ponownie założyć karty, przy czym zaleca się dodanie karty pustej, a następnie nacisnąć klucz 21 oraz przyciski START i ZO – gasną wówczas lampki BŁĄD i nr 19, zapala się natomiast lampka nr 21, która sygnalizuje dołączenie czytnika kart do systemu.




Rys. 3.6. Pulpit sterujący czytnika kart

Przycisk ZO na pulpicie sterującym czytnika kart pełni taką samą funkcję jak przy innych, dotychczas omówionych, urządzeniach: jego naciśnięcie powoduje dołączenie lub odłączenie czytnika do/od systemu.

Jeśli w trakcie pracy programu nastąpi wyczerpanie kart, stanowiących dane, a program użytkowy żąda podłożenia dalszych kart sygnalizując to lampką nr 19, to należy wówczas wycisnąć klucz 21, a następnie nacisnąć przycisk ZO, co spowoduje zakończenie wykonywania programu.

### 3.5.10. Obsługa pamięci bębnowej i taśmowej

Pamięć bębnowa nie wymaga w zasadzie obsługi. Jedynie, po nagraniu jednego z zestawów systemu operacyjnego wraz z translatorami należy zablokować część pamięci bębnowej przy pomocy kluczy, umieszczonych na tablicy sterującej pamięci bębnowej. Blokowanie przy pomocy kluczy stosuje się także po zapisaniu w pamięci bębnowej większych zbiorów, które będą poddane przetwarzaniu dopiero po pewnym czasie (np. po kilku godzinach).

Wszelkie prace przy pamięci taśmowej nie mogą być wykonywane podczas pracy systemu operacyjnego, czy programu użytkowego. Tak więc zakładanie taśmy magnetycznej wymaga regeneracji systemu operacyjnego opisanego w pkt. 3.5.2.

Założenie szpuli taśmy magnetycznej wymaga wykonania następujących czynności:

a) rozpocząć regenerację systemu operacyjnego (czynność a, b pkt. 3.5.2)
b) wyłączyć pamięć taśmową
c) otworzyć szafkę pamięci
d) założyć szpulę z taśmą
e) włączyć pamięć taśmową
f) nawinąć taśmę magnetyczną na drugą szpulę i przeprowadzić ją przez rolki prowadzące
g) zamknąć szafkę taśmy
h) zakończyć regenerację systemu operacyjnego (czynności c, d).

# 4. OPIS ROZKAZÓW

Niniejszy rozdział zawiera zarówno opis rozkazów podstawowych (elementarnych), wykonywanych w jednym cyklu rozkazowym, jak i opis rozkazów definiowanych przez podprogramy utworzone z rozkazów podstawowych. Z grupy rozkazów podstawowych wyodrębniono rozkazy nielegalne, które omówiono w pkt. 4.2.

Opis rozkazu podstawowego podany jest w dwóch kolumnach. W lewej kolumnie podano techniczny opis rozkazu, zaś w prawej opis rozkazu sformułowany w sposób przyjęty w podręcznikach programowania. Obie kolumny zawierają więc tę samą treść wyrażoną w odmienny sposób. Także kod rozkazu wyrażony jest dwojako: w lewej kolumnie podana jest wartość dziesiętna kodu oraz w nawiasie wartość ósemkowa kodu, zaś w prawej jego skrót literowy stosowany w języku PJP.

Uwaga: rozkazy z grupy 55 i 63 mają wydłużoną o 3 bity część operacyjną, toteż kody tych rozkazów podano przy pomocy dwóch liczb, z których pierwsza określa zawartość bitów 3–8, zaś druga 9–11 rozkazu. Podane czasy wykonywania rozkazów dotyczą rozkazów nie zawierających modyfikacji adresowych.

## 4.1. Rozkazy legalne

### 4.1.1. Rozkazy podstawowe

#### 4.1.1.1. Rozkazy sterujące

SKOCZ

| 1(1) | SKO |
|---|---|
| Wpisz adres efektywny rozkazu do licznika rozkazów LR. | Skocz do miejsca wskazanego przez adres efektywny rozkazu. |

Czas wykonania rozkazu 22 $\mu$s

SKOCZ PAMIĘTAJĄC LICZNIK ROZKAZÓW

| 2(2) | SLR |
|---|---|
| Zapamiętaj uzupełniony licznik rozkazów ULR w komórce PAO [1+BD], a następnie adres efektywny rozkazu wpisz do licznika rozkazów LR. Jeżeli adres efektywny rozkazu jest mniejszy | Jeżeli adres efektywny rozkazu jest większy od 63, to zapamiętaj uzupełniony licznik rozkazów ULR w komórce PAO [1+BD], a następnie skocz do miejsca wskazanego przez adres efek- |

od 64, to wpisz 1 do wskaźnika legalności WL.

tywny rozkazu. Rozkazy SLR o adresach efektywnych mniejszych od 64 opisane są w pkt. 4.1.2.

Czas wykonania rozkazu 44 $\mu$ s

## SKOCZ ZE ŚLADEM

3(3) SKS

Zapamiętaj uzupełniony licznik rozkazów ULR w komórce pamięci wskazanej przez adres efektywny rozkazu, a następnie zwiększ adres efektywny rozkazu o 1 i wpisz go do licznika rozkazów LR.

Zapamiętaj uzupełniony licznik rozkazów ULR w komórce pamięci wskazanej przez adres efektywny rozkazu, a następnie skocz do miejsca określonego przez zwiększony o 1 adres efektywny rozkazu.

Czas wykonania rozkazu 44 $\mu$ s

## WYKONAJ ROZKAZ PROGRAMOWANY

4–15 (4–17) PO4 – P15

Zapamiętaj uzupełniony licznik rozkazów ULR w komórce PAO [2+BD], a następnie wyzeruj wskaźnik nadmiaru WN i wpisz do licznika rozkazów LR kod rozkazu zwiększony o zawartość rejestru blokady dolnej BD. Część adresowa i modyfikacji nie ma wpływu na sposób wykonywania się tych rozkazów.

Zapamiętaj uzupełniony licznik rozkazów ULR w komórce PAO [2+BD], a następnie wyzeruj wskaźnik nadmiaru WN i skocz do miejsca określonego przez sumę zawartości rejestru blokady dolnej BD i kodu rozkazu.
Rozkazy służą do wywoływania podprogramów opisanych w pkt. 4.1.3.

Czas wykonania rozkazów 44 $\mu$ s

## WYKONAJ ROZKAZ PROGRAMOWANY

16–22 (20–26) P16–P22

Zapamiętaj uzupełniony licznik rozkazów ULR w komórce PAO [3+BD], a następnie wyzeruj wskaźnik nadmiaru WN i wpisz do licznika rozkazów LR kod rozkazu zwiększony o zawartość rejestru blokady dolnej BD.
Część adresowa i modyfikacji nie ma wpływu na sposób wykonywania się tych rozkazów.

Zapamiętaj uzupełniony licznik rozkazów ULR w komórce PAO [3+BD], a następnie wyzeruj wskaźnik nadmiaru WN i skocz do miejsca określonego przez sumę zawartości rejestru blokady dolnej BD i kodu rozkazu.
Rozkazy służą do wywoływania podprogramów opisanych w pkt. 4.1.3.

Czas wykonania rozkazów 44 $\mu$ s

## PORÓWNAJ LOGICZNIE Z AKUMULATOREM

39(47) POL

Porównaj kolejne 24 bity zawartości A z odpowiadającymi im bitami komórki pamięci, określonej przez adres efektywny rozkazu.

Zawartość akumulatora jest „mniejsza logicznie' od zawartości komórki pamięci, jeżeli w najwyższej pozycji, w której bit A jest różny od bitu w miejscu pamięci, w A jest bit O, a w komórce pamięci bit 1.

W tym przypadku do zawartości licznika rozkazów dodaj 1

W tym przypadku przejdź do następnego rozkazu.

Zawartość A jest „większa logicznie" od zawartości komórki pamięci, jeśli w najwyższej pozycji, w której bit akumulatora jest różny od bitu w komórce pamięci, w A jest bit 1, a w komórce pamięci bit O.

W tym przypadku do zawartości licznika rozkazów dodaj 2.

W tym przypadku przeskocz jeden rozkaz.

Zawartość A i zawartość komórki pamięci są równe logicznie, jeśli są identyczne.

W tym przypadku do licznika rozkazów dodaj 3.

W tym przypadku przeskocz dwa rozkazy.

Czas wykonania rozkazu 60 $\mu$ s

## SKOCZ PRZY ZERZE AKUMULATORA

48(60) SZA

Jeżeli bity nr 1–23 akumulatora A są równe zero, to wpisz adres efektywny rozkazu do licznika rozkazów LR, w przeciwnym razie LR zwiększ o 1.

Jeżeli bity nr 1–23 akumulatora A są równe zero, to skocz do miejsca wskazanego przez adres efektywny rozkazu, w przeciwnym razie przejdź do następnego rozkazu.

Czas wykonania rozkazu 22 $\mu$ s

## SKOCZ PRZY MINUSIE AKUMULATORA

49(61) SMA

Jeżeli bit nr O akumulatora A równa się 1, to wpisz adres efektywny rozkazu do licznika rozkazów LR, w przeciwnym razie LR zwiększ o 1.

Jeżeli bit nr O akumulatora A równa się 1, to skocz do miejsca wskazanego przez adres efektywny rozkazu, w przeciwnym razie przejdź do następnego rozkazu.

Czas wykonania rozkazu 22 $\mu$ s

## SKOCZ PRZY NADMIARZE

50(62) SNA

Jeżeli wskaźnik nadmiaru WN = 1,. to wyzeruj WN i wpisz do licznika rozkazów LR adres efektywny rozkazu, w przeciwnym razie zwiększ LR o 1.

Jeżeli wskaźnik nadmiaru WN = 1, to wyzeruj WN i skocz do miejsca wskazanego przez adres efektywny rozkazu, w przeciwnym razie przejdź do następnego rozkazu.

Czas wykonania rozkazu 22 $\mu$ s

## SKOCZ I UMIEŚĆ W B-REJESTRZE

51 SUB

Prześlij do B-rejestru zawartość komórki PAO [3+BD], a następnie wpisz adres efektywny rozkazu do licznika rozkazów LR.

Prześlij do B-rejestru zawartość komórki PAO [3+BD], a następnie skocz do miejsca wskazanego przez adres efektywny rozkazu.

Czas wykonania rozkazu 46 $\mu$ s

## WRÓĆ

52(64) WRO

Rozkaz ten odtwarza stan maszyny na podstawie uzupełnionego licznika rozkazów ULR, zapamiętanego w komórce pamięci, wskazanej przez adres efektywny rozkazu. Rozkaz ten wykonuje następujące czynności:

a) wpisz do licznika rozkazów LR zwiększoną o 1 liczbę umieszczoną na 15 ostatnich bitach rozpatrywanej komórki.

a) skocz do miejsca wskazanego przez zwiększoną o 1 liczbę umieszczoną na 15 ostatnich bitach rozpatrywanej komórki,

b) dodaj logicznie do wskaźnika nadmiaru WN zawartość bitu nr 3 rozpatrywanej komórki pamięci,

c) wpisz do wskaźnika legalności WL iloczyn logiczny WL i bitu nr 4 rozpatrywanej komórki,

d) wpisz jedynkę do wskaźnika przyjęć WP.

Czas wykonania rozkazu 43 $\mu$ s

## SKOCZ PO ODJĘCIU 1 OD B-REJESTRU

56(70) SOB lub SRB

Jeżeli B = O to zwiększ licznik rozkazów o 1, w przeciwnym razie odejmij 1 od

Jeżeli B = O przejdź do następnego rozkazu, w przeciwnym razie odejmij 1 od

modułu B i wpisz do licznika rozkazów LR adres efektywny rozkazu.

modułu B i skocz do miejsca wskazanego przez adres efektywny rozkazu.

Czas wykonania rozkazu 33 $\mu$ s

PORÓWNAJ Z B-REJESTREM

61(75) POB

Porównaj algebraicznie zawartość B-rejestru z zawartością komórki pamięci określonej przez adres efektywny rozkazu.

Jeżeli zawartość B jest mniejsza od zawartości komórki pamięci to do licznika rozkazów dodaj 1, jeśli większa to dodaj 2, gdy zaś równa dodaj 3.

Jeśli zawartość B jest mniejsza od zawartości komórki pamięci to przejdź do następnego rozkazu, jeśli większa to przeskocz jeden rozkaz, gdy zaś równa to przeskocz dwa rozkazy.

Przy porównaniu przyjęto, że liczba +O jest większa od liczby -O.

Czas wykonania rozkazu 60 $\mu$ s

4.1.1.2. Rozkazy przesyłania

Po wykonaniu rozkazu z tej grupy zawartość licznika rozkazów LR zwiększa się o 1.

Po wykonaniu rozkazu z tej grupy maszyna przechodzi do wykonania następnego rozkazu.

UMIEŚĆ W AM DŁUGO

25(31) UAD

UMIEŚĆ W AM

27(33) UAM

Umieść w mnożniku M zawartość komórki pamięci wskazanej przez adres efektywny rozkazu, zaś w akumulatorze A umieść zawartość komórki pamięci wskazanej przez zwiększony o 1 adres efektywny rozkazu. Oba rozkazy wykonują się tak samo w przypadku braku B-modyfikacji, zaś w przypadku jej występowania do adresu rozkazu 25 UAD zostaje dodana podwojona zawartość B-rejestru, tak jak to opisano w pkt. 1.6.

Czas wykonania rozkazu 78$\mu$ s

PAMIĘTAJ AM DŁUGO

26(32) PAD

PAMIĘTAJ AM

28(34) PAM

Zapamiętaj mnożnik M w komórce pamięci wskazanej przez adres efektywny rozkazu, zaś akumulator A zapamiętaj w komórce wskazanej przez adres efektywny rozkazu zwiększony o 1. Oba rozkazy wykonują się tak samo w przypadku braku B-modyfikacji, zaś w wypadku jej występowania, do adresu rozkazu 26 PAD zostaje dodana podwojona zawartość B-rejestru, tak jak to opisano w pkt. 1.6.

Czas wykonania rozkazu 78 $\mu$ s

UMIEŚĆ W AKUMULATORZE

40(50) UMA lub UAK

Umieść w akumulatorze A zawartość komórki pamięci wskazanej przez adres efektywny rozkazu.

Czas wykonania rozkazu 44 $\mu$ s

PAMIĘTAJ AKUMULATOR

41(51) PAK

Zapamiętaj akumulator A w komórce pamięci wskazanej przez adres efektywny rozkazu.

Czas wykonania rozkazu 44 $\mu$ s

UMIEŚĆ W MNOŻNIKU

42(52) UMN

Umieść w mnożniku M zawartość komórki pamięci wskazanej przez adres efektywny rozkazu.

Czas wykonania rozkazu 44 $\mu$ s

PAMIĘTAJ I ZERUJ MNOŻNIK

43(53) PZM

Zapamiętaj mnożnik M w komórce pamięci wskazanej przez adres efektywny rozkazu, a następnie wyzeruj M.

Czas wykonania rozkazu 44 $\mu$ s

UMIEŚĆ W B-REJESTRZE

58(72) UMB

Umieść w B-rejestrze zawartość komórki pamięci wskazanej przez adres efektywny rozkazu.

Czas wykonania rozkazu 44 $\mu$ s

PAMIĘTAJ B-REJESTR

62(76) PAB

Zapamiętaj B-rejestr w komórce pamięci wskazanej przez adres efektywny rozkazu.

Czas wykonania rozkazu 44 $\mu$ s

4.1.1.3. Rozkazy działań arytmetycznych

Przy opisie rozkazów tej grupy przyjęto, że maszyna wykonuje operacje na liczbach całkowitych zapisanych w systemie znak – moduł (pkt. 1.3).

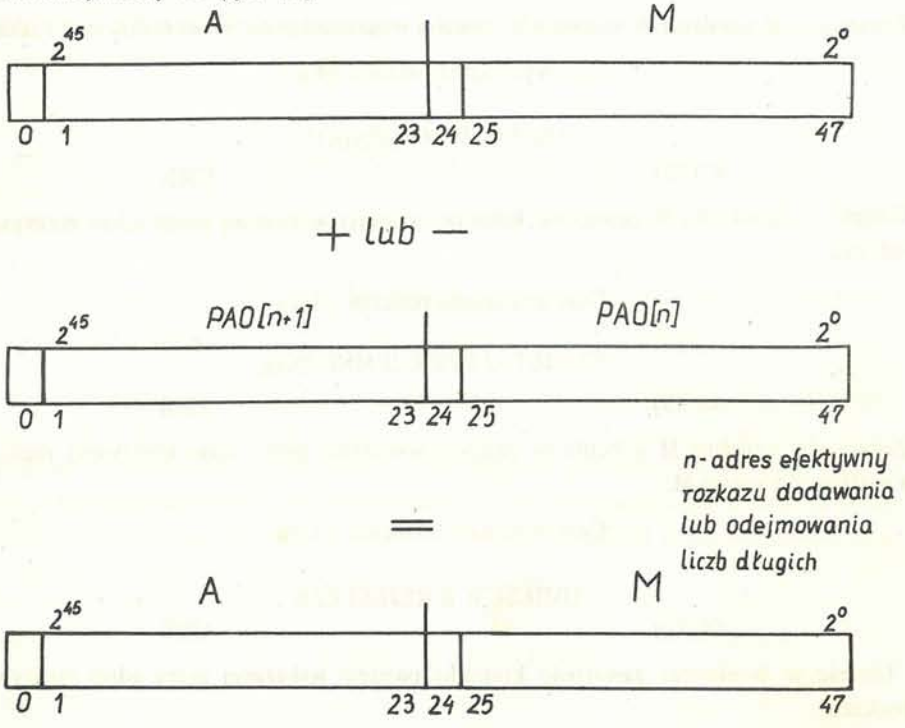
Po wykonaniu rozkazu z tej grupy zawartość licznika rozkazów LR zwiększa się o 1.

Po wykonaniu rozkazu z tej grupy maszyna przechodzi do następnego rozkazu.

DODAJ DO AM

23(27) DAM

Do zawartości akumulatora-mnożnika AM dodaj liczbę całkowitą długą umieszczoną w komórkach pamięci wskazanych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 (rys. 4.1).




Rys. 4.1. Operacje dodawania i odejmowania liczb długich

Jeżeli wartość bezwzględna wyniku jest większa od $2^{46}-1$ to wpisz jedynkę do wskaźnika nadmiaru WN, a następnie od wartości bezwzględnej wyniku odejmij $2^{46}$ i uzyskany rezultat wpisz do AM.
Jeżeli wartość wyniku równa się zero, to wyzeruj bity znaku A i M. Dodawane liczby mogą być niejednolite (tj. znaki A i M lub znaki liczb w PAO mogą być różne), natomiast wynik jest zawsze liczbą jednolitą.

Czas wykonania rozkazu 110 $\mu$ s

## ODEJMIJ OD AM

24(30) OAM

Od zawartości akumulatora-mnożnika AM odejmij liczbę całkowitą długą umieszczoną w komórkach pamięci wskazanych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 (rys. 4.1.).
Jeżeli wartość bezwzględna wyniku jest większa od $2^{46}-1$ to wpisz jedynkę do wskaźnika nadmiaru WN, a następnie od wartości bezwzględnej wyniku odejmij $2^{46}$ i uzyskany rezultat wpisz do AM. Jeżeli wartość wyniku równa się zero, to wyzeruj bity znaku A i M. Odejmowane liczby mogą być niejednolite (tj. znaki A i M lub znaki liczb w PAO mogą być różne), natomiast wynik jest zawsze liczb jednolitą.

Czas wykonania rozkazu 110 $\mu$ s

## DODAJ W STAŁYM PRZECINKU DO AKUMULATORA

32(40) DOS

Do zawartości akumulatora A dodaj liczbę umieszczoną w komórce pamięci wskazanej przez adres efektywny rozkazu.
Jeżeli wartość bezwzględna wyniku jest większa od $2^{23}-1$, to wpisz 1 do wskaźnika nadmiaru WN, a następnie od wartości bezwzględnej wyniku odejmij $2^{23}$ i uzyskany rezultat wpisz do A.
Jeżeli wartość wyniku równa się zero, to wyzeruj bit znaku A.

Czas wykonania rozkazu 44 $\mu$ s

## ODEJMIJ W STAŁYM PRZECINKU OD AKUMULATORA

33(41) ODS

Od zawartości akumulatora A odejmij liczbę umieszczoną w komórce pamięci wskazanej przez adres efektywny rozkazu.
Jeżeli wartość bezwzględna wyniku jest większa od $2^{23}-1$, to wpisz jedynkę do wskaźnika nadmiaru WN, a następnie od wartości bezwzględnej wyniku odejmij $2^{23}$ i uzyskany rezultat wpisz do A.
Jeżeli wartość wyniku równa się zero, to wyzeruj bit znaku A.

Czas wykonania rozkazu 44 $\mu$ s

## MNÓŻ W STAŁYM PRZECINKU PRZEZ MNOŻNIK

34(42) MNS

Zawartość mnożnika M pomnóż przez zawartość komórki pamięci wskazanej przez adres efektywny rozkazu i umieść wynik w akumulatorze-mnożniku AM. Po wykonaniu operacji bity znaku A i M są identyczne: równe zero, gdy znaki obu czynników były jednakowe, lub równe jeden, gdy znaki były różne.
Początkowa zawartość A nie ma żadnego wpływu na wynik operacji.

Czas wykonania rozkazu 300 $\mu$ s

## DZIEL W STAŁYM PRZECINKU AM

35(43) DZS

Jeżeli bezwzględna zawartość akumulatora A jest mniejsza od bezwzględnej zawartości komórki pamięci wskazanej przez adres efektywny rozkazu, to wykonaj następujące czynności:

a) bit znaku A przepisz do pozycji znaku M (bit znaku M nie ma żadnego wpływu na przebieg dzielenia)
b) zawartość AM podziel przez zawartość komórki pamięci wskazanej przez adres efektywny rozkazu
c) część całkowitą ilorazu wpisz do M, przy czym do pozycji znaku tego rejestru wpisz zero, gdy znaki dzielnej i dzielnika były jednakowe, lub jedynkę, gdy znaki były różne
d) resztę z dzielenia wpisz do A; ponieważ znak reszty jest zawsze równy znakowi dzielnej, pozostaw pozycję znaku A bez zmiany

Jeżeli bezwzględna zawartość A jest większa lub równa od bezwzględnej zawartości komórki pamięci wskazanej przez adres efektywny rozkazu, to zawartość AM pozostaw nie zmienioną i wpisz jedynkę do wskaźnika nadmiaru WN.

Czas wykonania rozkazu 330 $\mu$ s

## DODAJ DO B-REJESTRU

59(73) DOB

Do zawartości B-rejestru dodaj zawartość komórki pamięci wskazanej przez adres efektywny rozkazu.
Jeżeli wartość bezwzględna wyniku jest większa od $2^{23} - 1$, to wpisz jedynkę do wskaźnika nadmiaru WN, a następnie od wartości bezwzgldnej wyniku odejmij $2^{23}$ i uzyskany rezultat wpisz do B.
Jeżeli wartość wyniku równa się zero, to wyzeruj bit znaku B.

Czas wykonania rozkazu 44 $\mu$ s

## ODEJMIJ OD B-REJESTRU

60(74) ODB

Od zawartości B-rejestru odejmij zawartość komórki wskazanej przez adres efektywny rozkazu.
Jeżeli wartość bezwzględna wyniku jest większa od $2^{23} - 1$, to wpisz jedynkę do wskaźnika nadmiaru WN, a następnie od wartości bezwzględnej wyniku odejmij $2^{23}$ i uzyskany rezultat wpisz do B.
Jeżeli wartość wyniku równa się zero, to wyzeruj bit znaku B.

Czas wykonania rozkazu 44 $\mu$ s

### 4.1.1.4. Rozkazy działań logicznych

Po wykonaniu rozkazu z tej grupy zawartość licznika rozkazów zwiększa się o 1.

Po wykonaniu rozkazu z tej grupy maszyna przechodzi do następnego rozkazu.

## DODAJ LOGICZNIE DO AKUMULATORA

36(44) DOL

Wykonaj działanie „dodawania logicznego" nad 24 bitami zawartości akumulatora A i odpowiadającymi im 24 bitami komórki pamięci wskazanej przez adres efektywny rozkazu.
Wynik umieść w A.
Operacja „dodawania logicznego" zdefiniowana jest przez następującą tablicę:

| | | | | |
|---|---|---|---|---|
| bit A | 0 | 0 | 1 | 1 |
| bit w komórce pamięci | 0 | 1 | 0 | 1 |
| bit wyniku w A | 0 | 1 | 1 | 1 |

Czas wykonania rozkazu 44 $\mu$ s

Działanie rozkazu ilustruje przykład na rys. 4.2.

## ODEJMIJ SYMETRYCZNIE LOGICZNIE OD AKUMULATORA

37(45) OSL

Wykonaj działanie „sumy modulo dwa" nad 24 bitami akumulatora A i odpowiadającymi im 24 bitami zawartości komórki pamięci wskazanej przez adres efektywny rozkazu. Wynik umieść w A. Operacja „sumy modulo dwa" zdefiniowana jest przez następującą tablicę:

| | |
|---|---|
| bit w A | 0 0 1 1 |
| bit w komórce pamięci | 0 1 0 1 |
| bit wyniku w A | 0 1 1 0 |

Czas wykonania rozkazu 44 $\mu$ s

Działanie rozkazu ilustruje przykład na rys. 4.2.

MNÓŻ LOGICZNIE PRZEZ AKUMULATOR

38(46) MNL

Wykonaj działanie „mnożenia logicznego" nad 24 bitami akumulatora A i odpowiadającymi im 24 bitami zawartości komórki pamięci wskazanej przez adres efektywny rozkazu.
Wynik umieść w A.
Operacja „mnożenia logicznego" zdefiniowana jest przez następującą tablicę:

| | |
|---|---|
| bit w A | 0 0 1 1 |
| bit w komórce pamięci | 0 1 0 1 |
| bit wyniku w A | 0 0 0 1 |

Czas wykonania rozkazu 44 $\mu$ s

Działanie rozkazu ilustruje przykład na rys. 4.2.

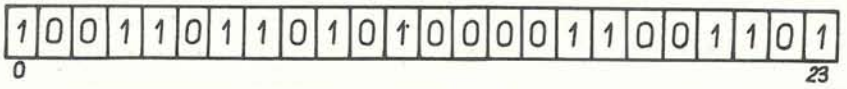
4.1.1.5. Rozkazy przesunięć

Po wykonaniu rozkazu z tej grupy zawartość licznika rozkazów LR zwiększa się o 1 (nie dotyczy rozkazów 55,7 i 55,0.

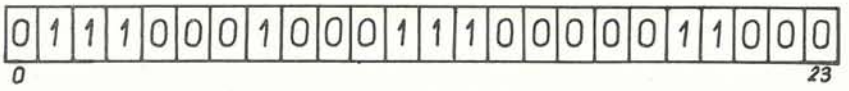
Po wykonaniu rozkazu z tej grupy maszyna przechodzi do następnego rozkazu (nie dotyczy rozkazów NOR i NOZ).

Niektóre rozkazy z tej grupy mają wydłużoną do 9 bitów część operacyjną, która podana jest w niniejszym opisie za pomocą dwóch liczb rozdzielonych przecinkiem. Pierwsza z nich określa zawartość bitów nr 3–8 rozkazu (a więc normalną część operacyjną), druga zaś określa zawartość bitów nr 9–11 rozkazu (wydłużenie części operacyjnej).
Czas wykonania rozkazów przesunięć arytmetycznych i cyklicznych 37 $\mu$s+n·5 $\mu$s, gdzie n jest ilością przesunięć.

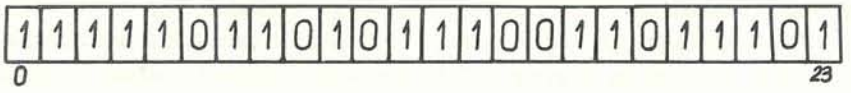
*Zawartosc A przed wykonaniem rozkazu*



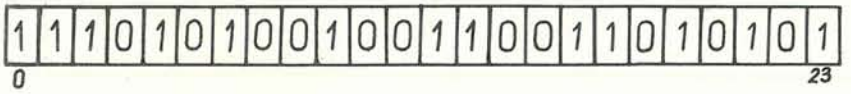

*Zawartość komórki pamięci przed wykonaniem rozkazu*



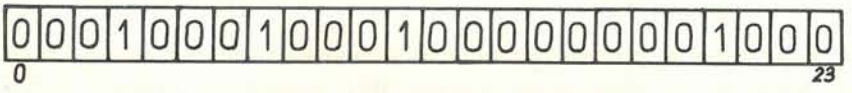

*Zawartość A po wykonaniu rozkazu dodawania logicznego 36 DOL*




*Zawartość A po wyk. rozkazu różnicy symetrycznej 37 OSL*




*Zawartość A po wyk. rozkazu mnożenia logicznego 38 MNL*



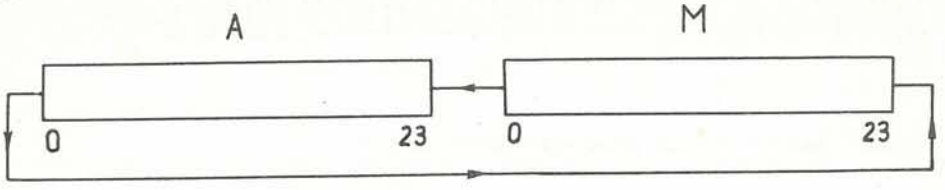

Rys. 4.2. Przykład ilustrujący działanie rozkazów wykonujących operacje logiczne

## PRZESUŃ W LEWO CYKLICZNIE DŁUGO

30(36) LCD

Zawartość akumulatora-mnożnika AM przesuń cyklicznie w lewo o ilość miejsc określoną przez 6 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr O

akumulatora A wchodzą na pozycję nr 23 mnożnika M. Bity wychodzące z pozycji nr O M wchodzą na pozycję nr 23 A (rys. 4.3). Działanie rozkazu ilustruje przykład na rys. 4.10.

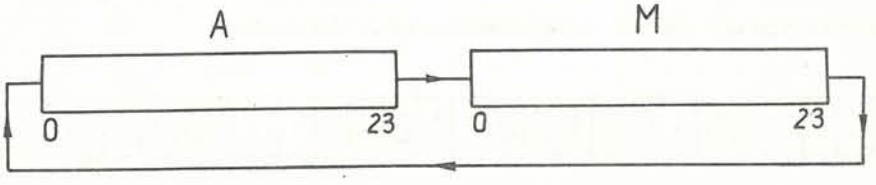



Rys. 4.3. Przesunięcie cykliczne AM w lewo

## PRZESUŃ W PRAWO CYKLICZNIE DŁUGO

31(37) PCD

Zawartość akumulatora-mnożnika AM przesuń cyklicznie w prawo o ilość miejsc określoną przez 6 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr 23 mnożnika M wchodzą na pozycję nr O akumulatora A. Bity wychodzące z pozycji nr 23 A wchodzą na pozycję nr 0 M (rys. 4.4). Działanie rozkazu ilustruje przykład na rys. 4.10.

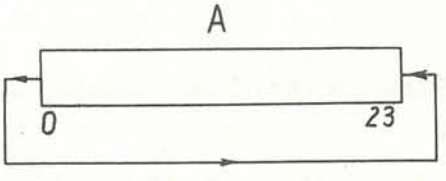



Rys. 4.4. Przesunięcie cykliczne AM w prawo

## PRZESUŃ W LEWO CYKLICZNIE AKUMULATOR

44(54) LCA

Zawartość akumulatora A przesuń cyklicznie w lewo o ilość miejsc określoną przez 6 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr O A wchodzą na pozycję nr 23 A (rys. 4.5). Działanie rozkazu ilustruje przykład na rys. 4.7.

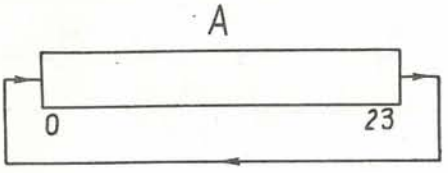



Rys. 4.5. Przesunięcie cykliczne A w lewo

## PRZESUŃ W PRAWO CYKLICZNIE AKUMULATOR

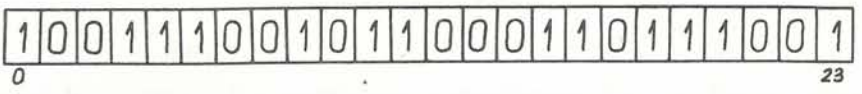
45(55) PCA

Zawartość akumulatora A przesuń cyklicznie w prawo o ilość miejsc określoną przez 6 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr 23 A wchodzą na pozycję nr O A (rys. 4.6).
Działanie rozkazu ilustruje przykład na rys. 4.7.

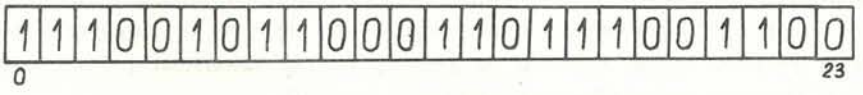



Rys. 4.6. Przesunięcie cykliczne A w prawo

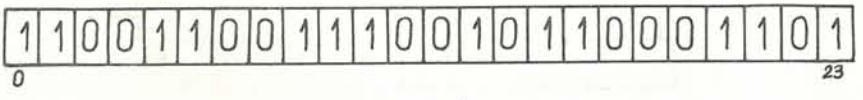
*Zawartość A przed wykonaniem rozkazu*




*Zawartość A po wykonaniu przesunięcia cyklicznego w lewo o 3*




*Zawartość A po wykonaniu przesunięcia cyklicznego w prawo o 5*



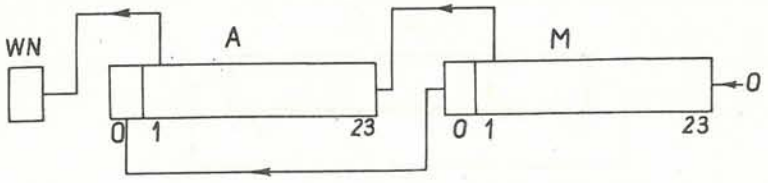

Rys. 4.7. Przykład działania rozkazów przesunięcia cyklicznego A

## PRZESUŃ W LEWO ARYTMETYCZNIE

46(56) LAR

Zawartość pozycji nr 1–23 akumulatora A i pozycji nr 1–23 mnożnika M przesuń w lewo arytmetycznie o ilość miejsc określoną przez 6 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr 1 M wchodzą na pozycję nr 23A. Za każdym

przesunięciem do pozycji nr 23 M wchodzi zero Do pozycji nr O A (pozycja znaku) wpisana zostaje zawartość pozycji nr O M. Jeżeli chociaż jeden bit równy 1 wychodzi z pozycji nr 1 A, to do wskaźnika nadmiaru WN wpisana zostaje jedynka.
Jeżeli 6 ostatnich bitów adresu efektywnego ma wartość zero, to jedynym efektem działania rozkazu będzie wpisanie do pozycji nr O A zawartości pozycji nr O M (rys. 4.8).
Działanie rozkazu ilustruje przykład na rys. 4.10.

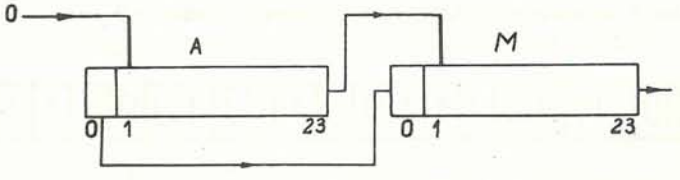



Rys. 4.8. Przesunięcie arytmetyczne AM w lewo

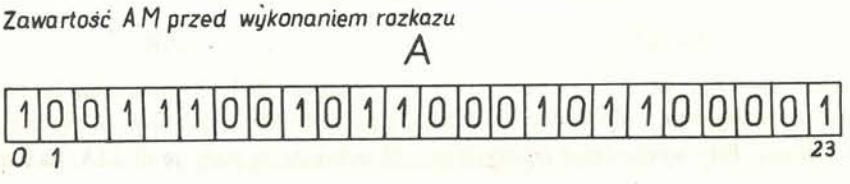
## PRZESUŃ W PRAWO ARYTMETYCZNIE

47(57) PAR

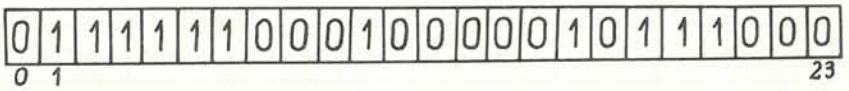
Zawartość pozycji nr 1–23 akumulatora A i pozycji nr 1–23 mnożnika M przesuń w prawo arytmetycznie o ilość miejsc określoną przez 6 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr 23 A wchodzą na pozycję nr 1 M. Bity wychodzące z pozycji nr 23 M są tracone. Za każdym przesunięciem na pozycję nr 1 A wchodzi zero. Na pozycję nr O M zostaje wpisana zawartość pozycji nr O A. Jeżeli wartość 6 ostatnich bitów adresu efektywnego równa się zero, to jedynym efektem działania rozkazu będzie wpisanie na pozycję nr O M zawartości pozycji nr O A (rys. 4.9). Działanie rozkazu ilustruje przykład na rys. 4.10.




Rys. 4.9. Przesunięcie arytmetyczne AM w prawo

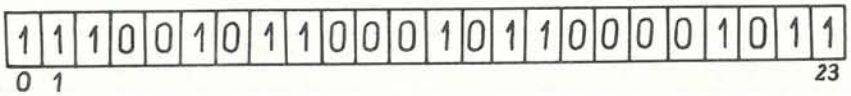


M

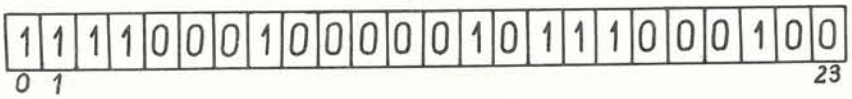



Zawartość AM po wyk. rozkazu przesunięcia cyklicznego AM w lewo o 3

A

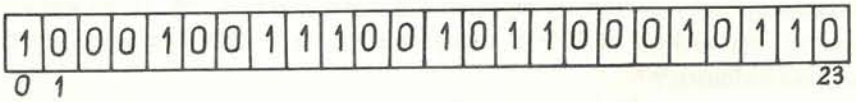



M

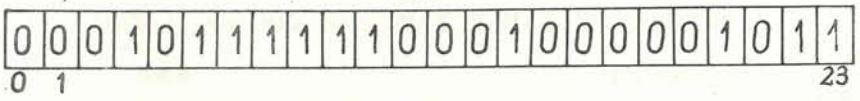



Zawartość AM po wyk. rozkazu przesunięcia cyklicznego AM w prawo o 4

A

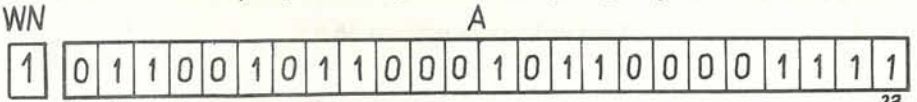



M

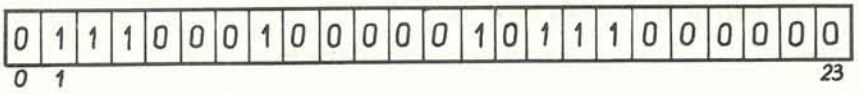



Zawartość AM i WN po wyk. rozkazu przesunięcia arytmetycznego AM w lewo o 3

WN




M




Rys. 4.10. Przykład działania rozkazów przesunięć długich cyklicznych i arytmetycznych

ZAOKRĄGLIJ

55,1(67,1) OKS

Wpisz jedynkę na pozycję nr 23 mnożnika M, pozostawiając bez zmiany pozycje nr 0–22 tego rejestru. Jeżeli na pozycji nr 1 M znajduje się jedynka, to do wartości bezwzględnej liczby całkowitej zapisanej w A dodaj 1. Jeśli wynik, co do wartości bezwzględnej, jest większy od $2^{23}-1$, to wyzeruj bity nr 1–23 A oraz wpisz jedynkę do wskaźnika nadmiaru WN.
Rozkaz ten nie zmienia bitów znaku A i M.

Czas wykonania rozkazu 36 $\mu$ s

PRZEPISZ WYKŁADNIK Z MNOŻNIKA DO B-REJESTRU

55,5(67,5) WMB

Przepisz pozycje nr 0 i 16–23 mnożnika M do odpowiadających im pozycji B-rejestru, a następnie wyzeruj bity nr 1–15 B. Do pozycji nr 0 M wpisz zawartość pozycji nr O A i wyzeruj pozycje nr 16–23 M.

Czas wykonania rozkazu 36 $\mu$ s

PRZEPISZ WYKŁADNIK Z B-REJESTRU DO MNOŻNIKA

55,3(67,3) WBM

Przepisz pozycje nr 0 i 16–23 B-rejestru do odpowiadających im pozycji mnożnika M. Pozostałe pozycje M pozostają bez zmiany.

Czas wykonania rozkazu 36 $\mu$ s

## NORMALIZUJ

55,7(67,7) NOR

Jeśli początkowa zawartość pozycji nr 1 akumulatora A jest równa 0, to zawartość akumulatora-mnożnika AM przesuwaj kolejno w lewo arytmetycznie o 1 (to znaczy wykonuj kolejno rozkaz LAR 1) nie zmieniając znaku A i za każdym przesunięciem odejmij 1 od B-rejestru. Przesuwanie zakończ, jeżeli do pozycji nr 1 A wpisana została 1, lub wykonane zostało 46 przesunięć. Jeżeli początkowa zawartość pozycji nr 1 A jest równa 1, to zawartość AM i B pozostaw nie zmienioną.

Opisany proces nazywa się normalizacją.

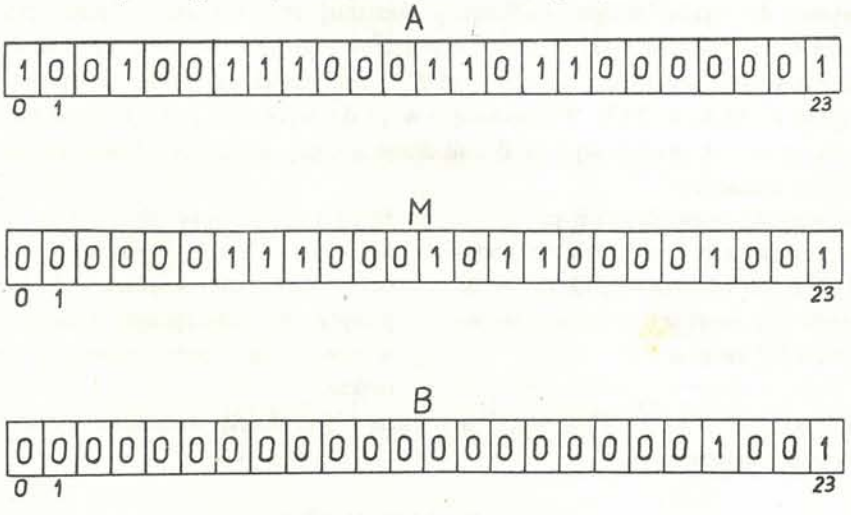
Jeżeli wartość bezwzględna B po wykonaniu normalizacji jest większa od 255, to licznik rozkazów LR zwiększ o 1, zaś w przeciwnym razie LR zwiększ o 2.

Jeżeli wartość bezwzględna B po wykonaniu normalizacji jest większa od 255, to przejdź do następnego rozkazu, zaś w przeciwnym razie przeskocz jeden rozkaz.

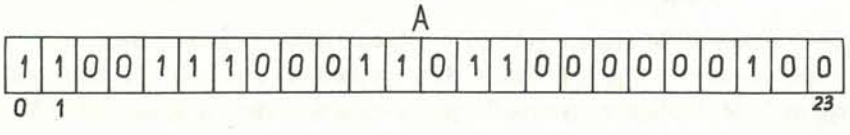
Czas wykonania rozkazu 53 $\mu$ s + n·5 $\mu$ s, gdzie n jest ilością przesunięć wykonanych podczas normalizacji.

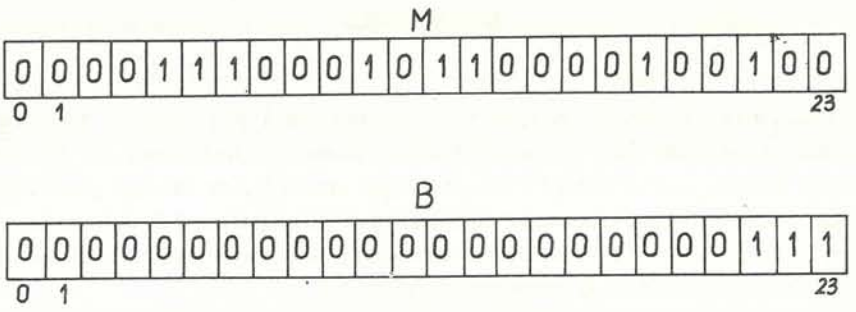
Działanie rozkazu ilustruje przykład na rys. 4.11.

Zawartość rejestrów przed wykonaniem rozkazu normalizacji 55,7 NOR




Zawartość rejestrów po wykonaniu rozkazu normalizacji 55,7 NOR






Rys. 4.11. Przykład działania rozkazu normalizacji 55,7 NOR

## NORMALIZUJ I ZAOKRĄGLIJ

55,0(67,0) NOZ

Jeżeli początkowa zawartość pozycji nr 1 akumulatora A równa jest 0, to zawartość akumulatora-mnożnika AM przesuwaj kolejno w lewo arytmetycznie o 1 (to znaczy wykonuj kolejno rozkaz LAR 1) nie zmieniając znaku A i za każdym przesunięciem odejmij 1 od B-rejestru. Przesuwanie zakończ, jeżeli na pozycję nr 1 A wpisana została 1 lub wykonane zostało 46 przesunięć. Jeżeli początkowa zawartość pozycji nr 1 A jest równa 1, to AM i B pozostaw bez zmiany. Opisany proces nazywa się normalizacją.

Następnie do liczby długiej całkowitej dodatniej reprezentowanej przez pozycje nr 1–23 A i pozycje nr 1–23 M dodaj liczbę $2^7 = 128$ (jedynka na pozycji nr 16 M). Jeżeli otrzymany wynik jest mniejszy od $2^{46}$, tzn. mieści się w AM, to wpisz go do pozycji nr 1–23 A i 1–23 M. W przeciwnym wypadku wyzeruj te pozycje, wpisz jedynkę na pozycję nr 1 A oraz dodaj 1 do B (tak dzieje się, gdy AM zawierał same jedynki na pozycjach znaczących).

Jeżeli wartość bezwzględna B po wykonaniu normalizacji i ewentualnym dodaniu 1 do B jest większa od 255, to licznik rozkazów LR zwiększ o 1, zaś w przeciwnym razie LR zwiększ o 2.

Jeżeli wartość bezwzględna B po wykonaniu normalizacji i ewentualnym dodaniu 1 do B jest większa od 255, to przejdź do następnego rozkazu, zaś w przeciwnym razie przeskocz jeden rozkaz.

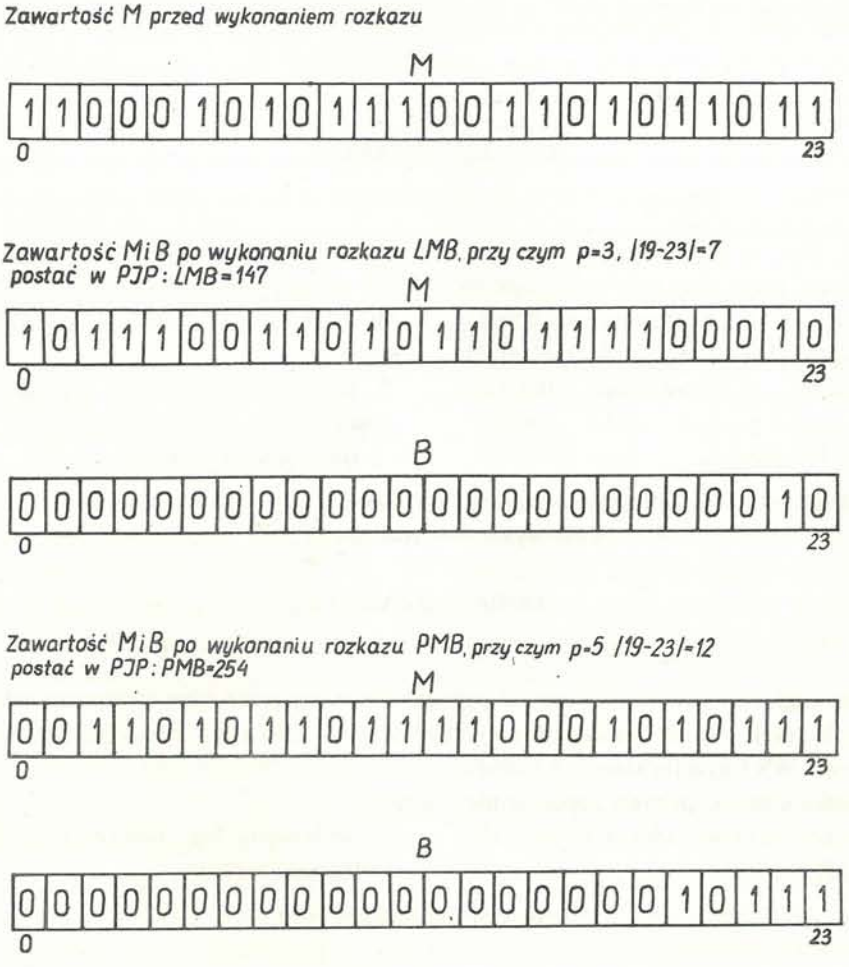
Czas wykonania rozkazu 65 $\mu$ s+n·5 $\mu$ s, gdzie n jest ilością przesunięć wykonanych podczas normalizacji.

## PRZESUŃ W LEWO MNOŻNIK ORAZ PRZEPISZ ZNAK DO B-REJESTRU

55,4(67,4) LMB

Zawartość mnożnika M przesuń cyklicznie w lewo o ilość miejsc określoną przez 5 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr O M wchodzą na pozycję nr 23 M. Następnie wyzeruj B-rejestr i wpisz do niego p ostatnich bitów M, gdzie

p określone jest przez zawartość bitów nr 15–18 adresu efektywnego rozkazu, przy czym, jeżeli p przekracza 8, to do B zostanie wpisane 8 ostatnich bitów M. Tak więc rozkaz umożliwia przepisanie z M do B od 0 do 8 wybranych bitów M. Uwaga: rozkaz zeruje B także w przypadku, gdy p = 0.
Działanie rozkazu ilustruje przykład na rys. 4.12.




Rys. 4.12. Przykład działania rozkazów 55,4 LMB i 55,2 PMB

## PRZESUŃ W PRAWO MNOŻNIK ORAZ PRZEPISZ ZNAK DO B-REJESTRU

55,2(67,2) PMB

Zawartość mnożnika M przesuń cyklicznie w prawo o ilość miejsc określoną przez 5 ostatnich bitów adresu efektywnego rozkazu. Bity wychodzące z pozycji nr 23 M

wchodzą na pozycję nr O M. Następnie wyzeruj B-rejestr i wpisz do niego p ostatnich bitów M, gdzie p określone jest przez zawartość bitów nr 15–18 adresu efektywnego rozkazu, przy czym jeżeli p przekracza 8, to do B wpisanych zostanie 8 ostatnich bitów M. Tak więc rozkaz umożliwia przepisanie z M do B od 0 do 8 wybranych bitów M.

Uwaga: rozkaz zeruje B także w przypadku, gdy p = 0.

Działanie rozkazu ilustruje przykład na rys. 4.12.

4.1.1.6. Pozostałe rozkazy podstawowe

ODEJMIJ OD PAMIĘCI

53(65) ODP

Jeżeli bity nr 1–23 komórki pamięci wskazanej przez adres efektywny rozkazu są równe zero, to do zawartości licznika rozkazów LR dodaj 1. W przeciwnym przypadku od bezwzględnej zawartości tej komórki pamięci odejmij 1, a następnie do LR dodaj 2.

Jeżeli bity nr 1–23 komórki pamięci wskazanej przez adres efektywny rozkazu są równe zero, to przejdź do następnego rozkazu.
W przeciwnym razie od bezwzględnej zawartości tej komórki pamięci odejmij 1 i przeskocz jeden rozkaz.

Bit znaku komórki pamięci pozostaje nie zmieniony.

Czas wykonania rozkazu 64 $\mu$ s

DODAJ DO PAMIĘCI

54(66) DOP

Do bezwzględnej zawartości komórki pamięci wskazanej przez adres efektywny rozkazu dodaj 1. Jeżeli wartość bezwzględna wyniku wynosi $2^{23}$, to wpisz jedynkę do wskaźnika nadmiaru WN i wyzeruj komórkę pamięci.

Bit znaku komórki pamięci pozostaje nie zmieniony.

Po wykonaniu tego rozkazu licznik rozkazów LR zwiększ o 1.

Po wykonaniu tego rozkazu przejdź do następnego rozkazu.

Czas wykonania rozkazu 64 $\mu$ s

UMIEŚĆ W B-REJESTRZE ADRES

57(71) UBA

Wyzeruj B-rejestr, a następnie wpisz do niego adres efektywny rozkazu.

Po wykonaniu tego rozkazu licznik rozkazów zwiększ o 1.

Po wykonaniu tego rozkazu przejdź do następnego rozkazu.

Czas wykonania rozkazu 30 $\mu$ s

### 4.1.2. Rozkazy definiowane przez system operacyjny

Poniżej opisano obszerną grupę rozkazów współpracy z urządzeniami zewnętrznymi. Wszystkie podane rozkazy są typu SLR z adresem mniejszym od 64. Należy podkreślić, że z technicznego punktu widzenia, są to tylko rozkazy skoku połączone z zapamiętaniem uzupełnionego licznika rozkazów, i zapaleniem wskaźnika WL (tak jak to opisano w pkt. 4.1.1.1). Jednakże, w komórkach pamięci, do których następować będą rozpatrywane skoki, znajdują się początkowe rozkazy bloków współpracujących z urządzeniami zewnętrznymi (opisane w pkt. 3.3). Innymi słowy opisane niżej rozkazy SLR są rozkazami, które powodują przejście do wykonania podprogramu współpracy z urządzeniem zewnętrznym. Podprogramy te stanowią integralną część systemu operacyjnego.

Niektóre rozkazy, spośród należących do tej grupy, wymagają podania pewnych parametrów np. ilości słów transmisji. Dopuszczalne wartości tych parametrów podane są przy opisie każdego rozkazu przy czym przekroczenie dopuszczalnego zakresu zmienności parametrów powoduje na ogół zakończenie wykonywania programu z podaniem przyczyny, oraz wartości błędnych parametrów. Ponadto drukowane są: zawartość licznika rozkazów LR, przy której nastąpiło zakończenie wykonywania programu oraz różnica zadeklarowanego i rzeczywistego czasu wykonywania programu w cyklach np. POZOSTAŁO 173 CYKLI. Czasami, po zakończeniu wykonywania programu system operacyjny drukuje kilka zawartości LR np.:

LR = 12 712 LUB 2941 LUB 2873.

Pierwsza liczba drukowana jest na podstawie zawartości komórki PAO [O+BD], czyli jest ślad ostatnio wykonanego przerwania. Następna liczba drukowana jest na podstawie zawartości komórki PAO [1+BD], a więc jest to ślad ostatnio wykonanego rozkazu SLR. Wreszcie trzecia liczba drukowana jest na podstawie zawartości komórki E2/13 wg symboliki tabulogramu systemu operacyjnego. W komórce tej przechowywany jest ślad rozkazów omawianej grupy, z wyjątkiem rozkazów SLR 3, 10, 30.

#### 4.1.2.1. Rozkazy czytania taśmy perforowanej

Po wykonaniu rozkazu z tej grupy maszyna przechodzi do następnego rozkazu (licznik rozkazów LR zwiększa się o 1).

*Uwaga 1.* Można przyjąć, że wykonanie rozkazu z tej grupy powoduje przesunięcie taśmy perforowanej o 1 rządek. Jednakże, w rzeczywistości, jak to opisano w pkt. 3.3.1, czytanie taśmy perforowanej odbywa się grupami po 16 rządków. Znajomość tego faktu może mieć pewne znaczenie podczas obserwacji przebiegu wykonywania programu, a także przy niektórych czynnościach operatorskich.

*Uwaga 2.* Jeżeli program użytkowy lub translator, po przeczytaniu wszystkich taśm perforowanych programu, żąda podłożenia pod czytnik dalszego ciągu taśmy (co sygnalizuje lampka LT), to wówczas operator wyciska klucz KT i podkłada pod czytnik taśmę z nowym programem, naciska przycisk ZO, w rezultacie czego wykonywanie programu zostaje zakończone, czemu towarzyszy wydruk:

LR = <licznik rozkazów> WYCZERPANO WEJŚCIE <nr czytnika>
POZOSTAŁO CYKLI <ilość cykli>

SLR 1

Wpisz bieżący rządek taśmy perforowanej 5, 6, 7 lub 8-kanałowej podłożonej pod czytnik 1 do B-rejestru, oraz do komórki PAO [27+BD]. Ponadto, jeżeli wczytany rządek taśmy jest znakiem cyfr (+27) lub liter (+31) w kodzie M2, to wpisz go również do komórki PAO [26+BD]. Rozkaz nie zmienia zawartości A, M.

SLR 5

Wpisz bieżący rządek taśmy perforowanej 5, 6, 7 lub 8-kanałowej podłożonej pod czytnik 2 do B-rejestru, oraz do komórki PAO [BD+61]. Ponadto, jeżeli wczytany rządek taśmy jest znakiem cyfr (+27) lub liter (+31) w kodzie M2, to wpisz go również do komórki PAO [60+BD]. Rozkaz nie zmienia zawartości A, M.

SLR 254.₹

Rozkaz ten wykonuje te same czynności co rozkaz SLR 1, gdy zawartość komórki PAO [254+BD] wynosi 1, oraz te same czynności co SLR 5, gdy zawartość komórki PAO [254+BD] wynosi 5.

4.1.2.2. Rozkazy drukowania

Po wykonaniu rozkazu z tej grupy maszyna przechodzi do następnego rozkazu (licznik rozkazów LR zwiększa się o 1).

SLR 2

Wyperforuj na perforatorze 1 znak określony przez 5 ostatnich bitów mnożnika M. Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 4

Dopisz do przygotowywanego wiersza wydruku na arkuszu drukarki wierszowej znak w kodzie M2 umieszczony na 5 ostatnich bitach mnożnika M. Jeśli na 5 ostatnich bitach M znajduje się znak cyfr (+27) lub liter (+31) kodu M2, to nie zostanie dopisany żaden znak, ale dalsze znaki będą drukowane, odpowiednio, jako należące do zbioru cyfr (obejmującego wszystkie znaki kodu M2 z wyjątkiem liter) lub liter (tabl. 1.1).
Znaki: pusty (+0) i powrotu karetki (+2) są pomijane. Znak nowej linii powoduje wydruk przygotowanego wiersza i przesuw papieru o 1 linię. Przekroczenie 120 znaków w jednym wierszu arkusza powoduje wydruk przygotowanego wiersza i automatyczne przejście do następnego wiersza.
Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 6

Wyperforuj na perforatorze 2 znak określony przez 5 ostatnich bitów mnożnika M.
Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 13

Wydrukuj na arkuszu drukarki wierszowej przygotowany wiersz a następnie przesuń papier o ilość linii określoną przez zawartość 6 ostatnich bitów mnożnika M.
Jeżeli na 6 ostatnich bitach M umieszczona jest liczba 0, to papier nie przesunie się – następny wiersz będzie drukowany na poprzednim.
Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 14

Dopisz do przygotowywanego wiersza wydruku na arkuszu drukarki wierszowej znak w kodzie drukarki DW umieszczony na 6 ostatnich bitach akumulatora A. Przekroczenie 120 znaków w jednym wierszu arkusza powoduje wydruk przygotowywanego wiersza i automatyczne przejście do następnego wiersza. Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 18

Czekaj na zakończenie poprzednio zainicjowanej operacji monitora, po czym wydrukuj na monitorze znak w kodzie M2 umieszczony na 5 ostatnich bitach mnożnika M.
Rozkaz nie zmienia zawartości A, M, B.

SLR 24

Dopisz do przygotowywanego wiersza wydruku na arkuszu drukarki wierszowej 4 znaki w kodzie drukarki umieszczone na bitach nr 0–5, 6–11, 12–17, 18–23 akumulatora A. Jeżeli pozycje w wierszu arkusza będziemy numerować od 0, to pierwszy znak (podczas wykonywania SLR 24) winien trafić na pozycję o numerze podzielonym przez 4. W przeciwnym wypadku nastąpi „cofnięcie się" do najbliższej pozycji podzielnej przez 4. W ten sposób do trzech znaków przygotowanych uprzednio przez SLR 4 lub SLR 14 może zostać zastąpionych przez znaki wypisywane przez SLR 24. Przekroczenie 120 znaków w jednym wierszu arkusza powoduje wydruk przygotowywanego wiersza i automatyczne przejście do następnego wiersza.
Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 255.☂

Rozkaz ten wykonuje te same czynności co
SLR 2, gdy zawartość komórki PAO [255+BD] wynosi 2
SLR 4, gdy zawartość komórki PAO [255+BD] wynosi 4
SLR 6, gdy zawartość komórki PAO [255+BD] wynosi 6
SLR 18, gdy zawartość komórki PAO [255+BD] wynosi 18.

Maksymalna wielkość wydruku w programie użytkowym jest deklarowana na początku tego programu. Jeżeli łączna ilość wyperforowanych rządków na każdym z perforatorów lub wydrukowanych linii na drukarce wierszowej przekroczy zadeklarowaną wartość maksymalną, to wykonywanie programu użytkowego zostaje zakończone, czemu towarzyszy wydruk:

LR = ⟨ licznik rozkazów ⟩
WYCZERPANO WYJŚCIE ⟨ typ urządzenia: PERF 1, PERF 2, DRUKARKA ⟩
POZOSTAŁO CYKLI ⟨ ilość cykli ⟩

4.1.2.3. Rozkazy współpracy z pamięcią bębnową

SLR 3

Czekaj na ewentualne zakończenie poprzednio zainicjowanej operacji bębnowej lub dotyczącej taśmy magnetycznej po czym zainicjuj operację bębnową i przeskocz dwa rozkazy (licznik rozkazów LR zwiększ o 3). Jeżeli przyjmiemy, że rozkaz SLR 3 znajduje się w komórce PAO [n], to parametry operacji bębnowej winny być rozmieszczone jak następuje:

adres bębnowy – w akumulatorze A
adres pamięci operacyjnej – w mnożniku M
operacja – w PAO [n+1]
(+0 – pisz na bęben, +1 – czytaj z bębna)
ilość słów w PAO [n+2]

W miejscu ilości słów można podać jej adres z 1 na bicie nr 1 słowa (P–modyfikacja). Rozkaz SLR 3 niszczy zawartość rejestrów A, M, B.

SLR 10

Przejdź do następnego rozkazu dopiero po zakończeniu operacji zainicjowanej przez ostatni rozkaz SLR 3 lub SLR 30. Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 30

Czekaj na ewentualne zakończenie poprzednio zainicjowanej operacji bębnowej lub dotyczącej taśmy magnetycznej, po czym zainicjuj operację bębnową i przeskocz dwa rozkazy (licznik rozkazów LR zwiększ o 3). Jeżeli przyjmiemy, że rozkaz SLR 30 znajduje się w komórce PAO [n], to parametry operacji bębnowej winny być rozmieszczone jak następuje:

| | |
|---|---|
| – adres bębnowy | – w akumulatorze A |
| – adres pamięci operacyjnej | – w B-rejestrze |
| – operacja (+0 – pisz na bęben<br>+1 – czytaj z bębna) | – w PAO [n+1] |
| – ilość słów | – w PAO [n+2] |

W miejscu ilości słów można podać jej adres z jedynką na bicie nr 1 słowa (P–modyfikacja). Rozkaz nie zmienia zawartości rejestrów A, M, B.

Parametry operacji SLR 3 lub SLR 30 muszą spełniać następujące warunki:
(uwaga: zestawy systemu operacyjnego opisano w pkt. 3.4)

dla operacji pisania

$$\left.\begin{matrix}16384\ \text{(I zestaw)}\\ 8192\ \text{(II zestaw)}\end{matrix}\right\} \leqslant \text{adres bębnowy} \leqslant 32768 - \text{ilość słów}$$

$$\left.\begin{matrix}1284\ \text{(I zestaw)}\\ 2436\ \text{(II zestaw)}\end{matrix}\right\} \leqslant \begin{matrix}\text{adres pamięci}\\ \text{operacyjnej}\end{matrix} \leqslant 12288 - \text{ilość słów}$$

ilość słów $> 0$

dla operacji czytania

$$0 \leqslant \text{adres bębnowy} \leqslant 32\,768 - \text{ilość słów}$$

$$\left.\begin{matrix}1306\ \text{(I zestaw)}\\ 2458\ \text{(II zestaw)}\end{matrix}\right\} \leqslant \begin{matrix}\text{adres pamięci}\\ \text{operacyjnej}\end{matrix} \leqslant 12\,288 - \text{ilość słów}$$

ilość słów $> 0$.

Jeżeli parametry są różne od dozwolonych, to wykonywanie programu zostaje zakończone, czemu towarzyszy wydruk:

| | |
|---|---|
| LR = ⟨ licznik rozkazów ⟩ | BŁĘDNE PARAMETRY SLR 3 LUB 30 |
| ADRES BEB | ⟨ adres bębnowy ⟩ |
| ADRES PW | ⟨ adres pamięci operacyjnej ⟩ |
| OPERACJA | ⟨ operacja ⟩ |
| ILOŚĆ | ⟨ ilość ⟩ |
| POZOSTAŁO CYKLI | ⟨ ilość cykli ⟩. |

Jeżeli w trakcie transmisji z pamięci bębnowej zostanie wykryty błąd, to maszyna zatrzymuje się ustawiając LR = 325 (I zestaw z Algolem) lub 333 (II zestaw bez Algolu). Pewne możliwości dalszego wykonywania programu opisano w pkt. 3.5.3.

4.1.2.4. Rozkazy współpracy z pamięcią taśmową

SLR 7

Czekaj na ewentualne zakończenie poprzednio zainicjowanej operacji bębnowej lub dotyczącej taśmy magnetycznej, po czym zainicjuj operację taśmy magnetycznej i przeskocz dwa rozkazy (licznik rozkazów LR zwiększ o 3). Jeżeli przyjmiemy, że rozkaz SLR 7 umieszczony jest w komórce PAO [n], to parametry operacji taśmy magnetycznej winny być rozmieszczone jak następuje:

| | |
|---|---|
| adres pamięci operacyjnej | – w B-rejestrze |
| operacja | – w PAO [n+1], na bitach nr 21–23 |

nr taśmy — w PAO [n+1], na bitach nr 18–20
ilość słów lub bloków — w PAO [n+2]

Operacje mogą być następujące:

0 – pisz blok na taśmę magnetyczną
1 – czytaj blok z taśmy magnetycznej
2 – kasuj odcinek taśmy magnetycznej określony przez ilość
3 – skocz do tyłu o liczbę bloków określoną przez ilość
4 – skocz do przodu o liczbę bloków określoną przez ilość
5 – odwiń taśmę magnetyczną do początku

Uwaga: podane kody operacji różnią się od kodów technicznych podanych w pkt. 2.3.2.
Operacja odwijania nie blokuje operacji na innych taśmach magnetycznych.
Rozkaz nie zmienia zawartości rejestrów A, M, B.
Parametry winny spełniać następujące warunki:

nr taśmy = 1, 2 lub 3
$20 \leqslant$ ilość $\leqslant 2048$.

Ponadto dla operacji pisania

$2436 \leqslant$ adres pamięci operacyjnej $\leqslant 8192$ – ilość

oraz dla operacji czytania

$2458 \leqslant$ adres pamięci operacyjnej $\leqslant 8192$ – ilość.

Jeżeli parametry są różne od dozwolonych, wykonywanie programu zostaje zakończone, czemu towarzyszy wydruk:

```
LR = ⟨ licznik rozkazów ⟩
BŁĘDNE PARAMETRY      SLR 7
ADRES  PW     ⟨ adres pamięci operacyjnej ⟩
OPERACJA      ⟨ nr taśmy X 8 + operacja ⟩
ILOŚĆ         ⟨ ilość ⟩
POZOSTAŁO CYKLI     ⟨ ilość cykli ⟩.
```

Taśma magnetyczna posiada na końcu wyróżniony tzw. obszar końcowy taśmy. W obszarze tym może być zapisany najwyżej 1 blok taśmowy. Żądanie zapisania lub odczytania drugiego bloku w obszarze końca taśmy powoduje zakończenie wykonywania programu z podaniem przyczyny:

```
KONIEC  TM
ADRES   PW    ⟨ adres pamięci operacyjnej ⟩
OPERACJA      ⟨ nr taśmy X 8 + operacja ⟩
ILOŚĆ         ⟨ilość ⟩
POZOSTAŁO CYKLI     ⟨ ilość cykli ⟩.
```

Takie samo zakończenie programu zachodzi w przypadkach, gdy operacja kasowania lub skocz do przodu kończy się w obszarze końca taśmy.

Gdy w czasie wykonywania operacji skocz do tyłu okaże się, że parametr < ilość > przekracza liczbę bloków zapisanych na taśmie (licząc od miejsca, w którym rozpoczęła się operacja wstecz), program zostaje zakończony z podaniem przyczyny:

| | | |
|---|---|---|
| POCZATEK | TM | |
| ADRES | PW | ⟨ adres pamięci operacyjnej ⟩ |
| OPERACJA | | ⟨ nr taśmy X 8 + operacja ⟩ |
| ILOŚĆ | | ⟨ ilość ⟩ |
| POZOSTAŁO CYKLI | | ⟨ ilość cykli ⟩. |

W czasie wykonywania operacji czytania układy kontrolne taśmy magnetycznej mogą wykryć błąd odczytu. Wówczas dokonuje się trzykrotnej próby odczytu i niezależnie od ich wyniku operację uważa się za zakończoną. Jednakże negatywny rezultat tych trzech prób może być wykryty rozkazem SLR 11, opisanym dalej.

Jeżeli w czasie wykonywania operacji zapisu na taśmę magnetyczną układy kontrolne wykryją błąd, to wówczas odcinek taśmy, na którym próba dokonania zapisu spowodowała powstanie błędu jest kasowany, po czym następuje zapis na następnym odcinku taśmy.

Jeżeli w trakcie wykonywania jakiejkolwiek operacji taśmy magnetycznej wystąpi uszkodzenie (system operacyjny może, na przykład, wnioskować o uszkodzeniu jednostki pamięci taśmowej, jeżeli operacja odwijania nie zakończy się w ciągu 300 sek.), to zapalany jest wówczas programowy wskaźnik uszkodzenia. Pojawienie się w dalszym ciągu programu jakiegokolwiek rozkazu dotyczącego taśmy magnetycznej powoduje zakończenie wykonywania programu z podaniem przyczyny:

| | |
|---|---|
| AWARIA TAŚMY | |
| ADRES PW | < adres pamięci operacyjnej > |
| OPERACJA | < nr taśmy X 8 + operacja > |
| ILOŚĆ | < ilość słów > |
| POZOSTAŁO CYKLI | < ilość cykli > |

Drukowane tu parametry są parametrami rozkazu SLR 7, który zainicjował transmisję, podczas której nastąpiło uszkodzenie jednostki pamięci taśmowej.

SLR 11

Czekaj aż zakończy się operacja taśmowa (z wyjątkiem odwijania) bezpośrednio poprzedzająca dany SLR 11, po czym:

- gdy operacją było czytanie, w czasie którego wystąpił błąd odczytu, to przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1)
- gdy operacją było czytanie w obszarze końca taśmy, w czasie którego wystąpił błąd odczytu, to przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 2)
- gdy operacja zakończyła się w obszarze końca taśmy (bez błędu), to przeskocz dwa rozkazy (licznik rozkazów LR zwiększ o 3)

– gdy nie zaszły żadne ww. okoliczności, to przeskocz 3 rozkazy (licznik rozkazów LR zwiększ o 4)

Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 12

Czekaj aż zakończy się odwijanie taśmy o numerze podanym w komórce pamięci znajdującej się za rozkazem SLR 12, po czym przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 1). Rozkaz nie zmienia zawartości rejestrów A, M, B.

4.1.2.5. Rozkazy współpracy z monitorem

SLR 8

Czekaj na zakończenie poprzednio zainicjowanej operacji monitora, po czym zainicjuj operację wypisywania na monitorze wskazanej ilości znaków w kodzie M2 określonych przez 5 ostatnich bitów kolejnych słów pamięci i przeskocz 2 rozkazy (licznik rozkazów LR zwiększ o 3). Jeżeli przyjmiemy, że rozkaz SLR 8 znajduje się w komórce PAO [n], to parametry operacji winny być rozmieszczone jak następuje:

adres pamięci operacyjnej – w PAO [n+1]
ilość znaków – w PAO [n+2]

Każdy z tych parametrów może być zastąpiony przez jego adres z jedynką na bicie nr 1 (P–modyfikacja).

Rozkaz nie zmienia zawartości rejestrów A, M, B.

SLR 9

Czekaj na zakończenie poprzednio zainicjowanej operacji monitora, po czym udziel zezwolenia na przesyłanie z monitora do określonego obszaru pamięci operacyjnej. Jeżeli monitor pracuje w reżimie wymuszonego czytania (klucz 23 na stoliku monitora wciśnięty) zainicjuj powyższą operację przesyłania. Następnie przeskocz dwa rozkazy (licznik rozkazów LR zwiększ o 3). Jeżeli monitor pracuje w reżimie normalnego czytania (klucz 23 na stoliku monitora wyciśnięty), operacja przesyłania zostanie zainicjowana po pobraniu (przez maszynę) z monitora pierwszego znaku. Jeżeli przyjmiemy, że rozkaz SLR 9 umieszczony jest w komórce PAO [n], to parametry operacji winny być rozmieszczone jak następuje:

adres pamięci operacyjnej – w PAO [n+1]
maksymalna ilość znaków – w PAO [n+2]

Każdy z tych parametrów może być zastąpiony przez jego adres z jedynką na bicie nr 1 (P-modyfikacja).

Rozkaz nie zmienia zawartości rejestrów A, M, B.

Operacja polega na pobieraniu z monitora kolejnych znaków i wpisywaniu ich do 5 najmłodszych pozycji (zerując pozostałe) kolejnych słów pamięci, poczynając od wskazanego adresu. Wpisywane są wszystkie znaki z tym, że:

a) jeżeli maszyna wczytała znak pusty (+0) lub znak powrotu karetki (+2), to następny znak będzie wpisany do tego samego słowa,
b) jeżeli maszyna wczytała znak nowej linii (+8), to następny znak będzie wpisany na początek obszaru pamięci,
c) jeżeli kolejny znak różny od nowej linii (+8) i ↑ (+26) jest wpisany do ostatniego słowa obszaru, to następny znak również będzie wpisany do tego samego słowa,
d) jeżeli znakiem wpisanym jest znak ↑ to operacja ulega zakończeniu.

Parametry operacji SLR 8 lub SLR 9 winny spełniać następujące warunki:

2458 ⩽ adres pamięci operacyjnej ⩽ 8192 – ilość znaków

ilość znaków ⩾ 1

Jeżeli parametry są różne od dozwolonych program zostaje zakończony z podaniem przyczyny:

LR = < licznik rozkazów >
BŁĘDNE PARAMETRY SLR 8 LUB 9
ADRES PW < adres pamięci operacyjnej >
ILOŚĆ < ilość znaków >
POZOSTAŁO CYKLI < ilość cykli >

*Uwaga:* Projektując parametry rozkazu SLR 9 należy pamiętać, że prócz 69 znaków tworzących wiersz, do obszaru pamięci wprowadzone zostaną także znaki cyfr i liter, których może być dość sporo. Należy także pamiętać, że niektórzy operatorzy naciskają niepotrzebnie kilkakrotnie znaki cyfr lub liter.

### SLR 18

Rozkaz ten opisano w pkt. 4.1.2.2.

### SLR 20

Czekaj na zakończenie operacji zainicjowanej przez ostatni rozkaz SLR 8, SLR 9 lub SLR 18, po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1) podając w B-rejestrze ilość znaków wypisanych na monitorze lub wczytanych z monitora (licząc od początku obszaru).
Rozkaz nie zmienia zawartości rejestrów A, M.

### SLR 21

Jeżeli trwa operacja zainicjowana przez ostatni SLR 8, SLR 9 lub SLR 18 przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 2), gdy zaś została zakończona przejdź do następnego rozkazu (LR zwiększ o 1). W obu przypadkach podaj w B-rejestrze ilość znaków wypisanych na monitorze lub wczytanych z monitora (licząc od początku obszaru).

Jako zakończenie operacji rozumie się również jej niezainicjowanie (np. podczas SLR 9, gdy monitor jest w stanie normalnego czytania).
Rozkaz nie zmienia zawartości rejestrów A, M.

4.1.2.6. Rozkaz czytania karty perforowanej

SLR 15

Wpisz bieżącą kartę do pamięci operacyjnej umieszczając kolejne kolumny w komórkach pamięci począwszy od 8100. Do pamięci wprowadzane są tylko kolumny zawierające znaki (nie puste). Za ostatnią kolumną z karty zostaje wpisana liczba + 4095. Po wykonaniu rozkazu przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). *Uwaga:* Jeżeli program użytkowy po przeczytaniu wszystkich kart perforowanych będących danymi tego programu żąda podłożenia dalszych kart (co sygnalizuje lampka 19 na pulpicie czytnika kart), to wówczas operator wyciska klucz 21 na tymże pulpicie, a następnie naciska przycisk ZO, w rezultacie wykonywanie programu zostaje zakończone, czemu towarzyszy wydruk:

LR = < licznik rozkazów > WYCZERPANO WEJŚCIE KART
POZOSTAŁO CYKLI < ilość cykli >

4.1.2.7. Rozkaz zakończenia programu

SLR 40

Rozkaz powoduje zakończenie wykonywania programu z podaniem przyczyny:

LR = < licznik rozkazów >
STOP
POZOSTAŁO CYKLI < ilość cykli >

4.1.3. Rozkazy definiowane przez podprogramy standardowe

Poniżej omówiono grupę rozkazów, definiowanych przez podprogramy standardowe. Użycie rozkazu z tej grupy w programie wymaga uprzedniego (podczas translacji) przepisania podprogramu z pamięci bębnowej do operacyjnej, co następuje w wyniku wykonania dyrektywy F (pkt. 5.3.2). Postać tej dyrektywy podana jest przy opisie każdego rozkazu; bliższe wyjaśnienia w zakresie zasad stosowania rozkazów tej grupy oraz dyrektyw podane są w pkt. 5.3.2 i 7.7.2.

Zarówno rozkazy czytania jak i drukowania mogą być wykonywane na rozmaitych urządzeniach zewnętrznych w zależności od zawartości komórek pamięci PAO [253+BD], PAO [254+BD] i PAO [255+BD]. I tak, gdy zawartość komórek PAO [253+BD] i PAO [254+BD] wynosi, odpowiednio 2458 i 1, to rozkazy czytania odnoszą się do czytnika I, zaś gdy zawartości wynoszą 2492 i 6 to podane rozkazy czytania odnoszą się do czytnika II.

Drukowanie wykonywane jest przez:
perforator I , gdy zawartość komórki PAO [255+BD] wynosi 2
perforator II, gdy zawartość komórki PAO [255+BD] wynosi 6
drukarkę wierszową, gdy zawartość komórki PAO [255+BD] wynosi 4
monitor, gdy zawartość komórki PAO [255+BD] wynosi 18.

Tablica 4.1

Typowe formaty wydruków

| Liczba | format wydruku | L1=1<br>L2=9 = 2452<br>L3=10 | L1=5<br>L2=3 =12140<br>L3=0 | L1=5<br>L2=3 =12150<br>L3=8 | L1=4<br>L2=0 =100003<br>L3=3 | L1=0<br>L2=3 =142<br>L3=2 |
|---|---|---|---|---|---|---|
| 19.4289 | | +1.942890000,+1 | +19.429 | +19428.900,−3 | +194,−1 | +.194,+12 |
| 67003.1673 | | +6.700316730,+4 | +67003.167 | +67003.167 | +6700,+1 | +.670,+5 |
| 42151200 | | +4.215120000,+7 | +42151.200,+3 | +42151.200,+3 | +4215,+4 | +.422,+8 |
| 0.0123456789 | | +1.234567890,−2 | +0.012 | +12345.679,−6 | +123,−4 | +.012 |
| 0.72,−29 ($=0.72\cdot10^{-29}$) | | +7.200000000,−30 | +0.001,−26 | +72000.000,−34 | +720,−32 | +.072,−28 |
| 412 | | +4.120000000,+2 | +412.000 | +41200.000,−2 | +412 | +.412,+3 |
| 0 | | 0 | 0 | 0 | 0 | 0 |

SLR 130.

Wydrukuj liczbę zmiennoprzecinkową umieszczoną w akumulatorze-mnożniku AM, po czym przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 2). Jeżeli przyjmiemy, że rozkaz znajduje się w komórce PAO [n], to format określający sposób wydruku musi być umieszczony w komórce PAO [n+1] w postaci trzech liczb L1, L2, L3 umieszczonych na pozycjach, odpowiednio, nr 9–13, 14–18, 19–23. Liczba L1 określa ilość cyfr przed kropką dziesiętną, liczba L2 ilość cyfr po kropce, zaś liczba L3 określa minimalną ilość cyfr znaczących w wydrukowanej liczbie. Przed liczbą drukowany jest znak, zaś po liczbie 4 spacje. Jeżeli zastosowany format wydruku uniemożliwia wydrukowanie liczby w formie poprawnej lub gdy ilość pozycji przeznaczona na cyfry po kropce jest zbyt mała, to zamiast 4 spacji na końcu liczby, drukowany jest, po przecinku, wykładnik potęgi liczby 10, przez którą to potęgę należy pomnożyć wydrukowaną liczbę. Typowe formaty wydruków podano w tablicy 4.1. Rozkaz niszczy zawartość rejestrów A, M, B.
W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F4,130–0

SLR 131.

Przeczytaj liczbę zmiennoprzecinkową dziesiętną i umieść ją w akumulatorze-mnożniku AM w postaci zmiennoprzecinkowej binarnej, po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1).
Wczytywana liczba ma postać $m \cdot 10^c$ gdzie m jest mantysą, c – cechą, zaś liczby m i c rozdzielone są przecinkiem. Mantysa m może zawierać co najwyżej 14 cyfr licząc (od lewej do prawej) od pierwszej cyfry różnej od zera (jeżeli zawiera więcej niż 14 cyfr, to każda następna cyfra wpisywana jest na pozycję 14-tej cyfry). Znak mantysy m można pominąć, gdy $m \geqslant 0$. Kropkę dziesiętną mantysy można pominąć, gdy część ułamkowa mantysy równa się 0. Cecha c jest zawsze liczbą całkowitą ze znakiem lub bez i musi być mniejsza od $2^{23}$ (w przeciwnym wypadku do B-rejestru zostaje wpisana 1 i wykonywanie rozkazu zostaje zakończone). Jeżeli cecha c ma wartość +1, to można ją pominąć. Wczytywanie liczby zostaje zakończone: z chwilą natrafienia na znak różny od cyfry, lub, w przypadku, gdy pominięto cechę, z chwilą natrafienia na znak różny od cyfry i przecinka i wreszcie przy wczytywaniu liczb całkowitych z chwilą natrafienia na znak różny od kropki, przecinka i cyfry. Wczytywana liczba musi być co do wartości bezwzględnej mniejsza od $2^{255} \cong 10^{77}$ w przeciwnym razie wynik uzyskany w AM jest błędny, zaś do B zostaje wpisana 1. Zakończenie wykonywania rozkazu z B = 0 oznacza poprawne wczytanie liczby.
Liczba binarna zmiennoprzecinkowa postaci $m_1 \cdot 2^{c_1}$ umieszczona w AM zawiera mantysę $m_1$ na bitach nr 1–23 A i 1–15 M, zaś cechę $c_1$ na bitach nr 16–23 M. Znak $m_1$ umieszczony jest na bicie nr OA, zaś znak $c_1$ na bicie nr OM. Przed zakończeniem wykonywania rozkazu następuje zaokrąglenie mantysy, polegające na dodaniu 1 do najmniej znaczącego bitu mantysy, gdy dwa poprzednie (obcięte) bity miały wartość 1.

Rozkaz SLR 131. służy w zasadzie do wczytywania, poprzez czytnik I lub II, liczb w kodzie M2 umieszczonych na taśmie perforowanej; można jednakże przystosować go

do czytania liczb z innych urządzeń. Czytanie liczb rozkazem SLR 131. poprzez czytnik I należy poprzedzić wpisaniem do komórki PAO [254+BD] liczby +1 i do komórki PAO [253+BD] liczby +2458 (w I zestawie z Algolem +1306), poprzez czytnik II wpisaniem odpowiednio liczb +5 i +2492.

*Uwaga:* przed rozpoczęciem wykonywania programu translator wpisuje się do komórki PAO[254+BD] liczbę +1 oraz do komórki PAO [253+BD] liczbę +2458 (w I zestawie +1306).

Przystosowanie rozkazu SLR 131. do czytania liczb z innych urządzeń jest możliwe, ale dość kłopotliwe. Podprogram definiujący rozkaz SLR 131. w celu wczytania znaku wykonuje rozkaz SLR, którego adres efektywny umieszczony jest w komórce PAO [254+BD], i po wykonaniu tego rozkazu pobiera znak z B lub z komórki o adresie określonym przez zwiększoną o 1 zawartość komórki PAO [253+BD]. Ostatnio przeczytany znak liter lub cyfr znajduje się w komórce, której adres zawiera komórka PAO [253+BD].

Na zakończenie podamy kilka przykładów liczb zapisanych w formie wymaganej przez rozkaz SLR 131.

| | | | |
|---|---|---|---|
| 439 | 0 | 0.0000001 | −0.127 |
| +4.39,+2 | +0 | 1,−7 | −127,−3 |
| 439000,−3 | −0 | +1,−7 | −12.7,−2 |
| 0.0439,+4 | | | |

W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywy:

```
F5,131−0
F6
```

SLR 134.

Wydrukuj liczbę całkowitą umieszczoną w akumulatorze A, po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1).

Drukowana liczba zajmuje na arkuszu 10 pozycji. Cyfry tworzące liczbę poprzedzone są co najmniej dwoma spacjami oraz, gdy liczba jest ujemna znakiem „−". Po wykonaniu rozkazu zawartość B-rejestru pozostaje nie zmieniona, natomiast zawartość A i M zostaje zniszczona.

W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F3,134−0

Przykłady wydruku:

```
  312
 −714
58112
    0
   −1
```

SLR 135.

Wydrukuj tekst umieszczony w kolejnych słowach pamięci po zęwszy od adresu umieszczonego w B-rejestrze, po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). W kolejnych słowach, znaki w kodzie M2 tworzące tekst, umieszczone są na bitach nr 1–5, 7–11, 13–17, 19–23. Tekst kończy słowo o zawartości +0. Rozkaz niszczy zawartość A, M, B.

W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F3,135–1

Przykład:

```
--------
--------
--------
UBA E12
SLR 135.
--------
--------
--------
E 12
= 37150311
= 14012016
= 05061436
= 30000000
+ 0
--------
--------
--------
```

Rezultatem działania podanego fragmentu programu w języku PJP będzie wydrukowanie tekstu: POLITECHNIKA

SLR 136.

Wydrukuj p spacji, po czym przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 2). Jeżeli przyjmiemy, że rozkaz SLR 136. znajduje się w komórce PAO [n], to ilość drukowanych spacji p umieszczona jest w komórce PAO [n+1]. Rozkaz pozostawia B-rejestr nie zmieniony, natomiast niszczy zawartość A, M. W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F3,136–2

SLR 137.

Wydrukuj znak powrotu karetki, a następnie p znaków nowej linii, po czym przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 2). Jeżeli przyjmiemy, że rozkaz SLR 137. znajduje się w komórce PAO [n], to ilość drukowanych znaków nowej linii p umieszczona jest w komórce PAO [n+1]. Rozkaz pozostawia B-rejestr nie zmieniony, natomiast niszczy zawartość A, M. W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F3,137–3

SLR 138.

Wydrukuj p znaków pustych (blanki), po czym przeskocz jeden rozkaz (licznik rozkazów LR zwiększ o 2). Jeżeli przyjmiemy, że rozkaz SLR 138. znajduje się w komórce PAO [n], to ilość drukowanych znaków pustych p umieszczona jest w komórce PAO [n+1]. Rozkaz pozostawia B-rejestr nie zmieniony, natomiast niszczy zawartość A, M. W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F3,138–4

SLR 160.

Zamień liczbę zmiennoprzecinkową umieszczoną w akumulatorze-mnożniku AM na liczbę stałoprzecinkową całkowitą i umieść ją w akumulatorze A. Część ułamkowa liczby umieszczonej w AM jest pomijana. Jeżeli, po zamianie liczba nie mieści się w akumulatorze A, to wpisz 1 do wskaźnika nadmiaru WN. Rozkaz nie zmienia zawartości B-rejestru.
W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F7,160–0

Czas wykonania rozkazu 0.5 ms.

SLR 161.

Zamień liczbę stałoprzecinkową całkowitą umieszczoną w akumulatorze A na liczbę zmiennoprzecinkową i umieść ją w akumulatorze-mnożniku AM. Rozkaz nie zmienia zawartości rejestru B. W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F7,161–1

Czas wykonania rozkazu 0.41 ms.

SLR 162.

Oblicz entier liczby zmiennoprzecinkowej umieszczonej w akumulatorze-mnożniku AM i umieść wynik, w postaci liczby zmiennoprzecinkowej w AM. Rozkaz nie zmienia zawartości rejestru B. W programie, w języku PJP, używającym ten rozkaz należy umieścić dyrektywę F7,162–2

Czas wykonania rozkazu 0.5 ms.

PO7 (czytaj: P zero siedem)

Wydrukuj zawartość rejestrów oraz ewentualnie zawartość wskazanego obszaru pamięci (nie więcej niż 63 komórki).
W celu uproszczenia dalszego opisu przyjmiemy, że rozkaz PO7 znajduje się w komórce PAO [n]. Żądania określające wielkość i sposób wydruku umieszczone są w części adresowej i części modyfikacyjnej rozkazu PO7. I tak, jeżeli, na bicie nr 1 rozkazu (P-modyfikacja) umieszczona będzie jedynka, to prócz zawartości rejestrów wydrukowany zostanie także obszar pamięci, którego adres początkowy znajduje się w komórce PAO [n+1]. Bit nr 2 rozkazu określa rodzaj wydruku:

ósemkowy (gdy bit = 1) lub dziesiętny (gdy bit = 0).

Część adresowa zawiera trzy liczby: L1 na bitach nr 9–14, L2 na bitach nr 15–20 i L3 na bitach nr 21–23. Liczba L1 ($1 \leqslant L1 \leqslant 63$, gdy bit nr 1 rozkazu = 1, oraz L1 = 0, gdy bit nr 1 rozkazu = 0) określa ilość słów drukowanego obszaru pamięci. Liczba L2 ($0 \leqslant L2 \leqslant 63$) określa częstość uruchamiania wydruków: po każdym wydruku L2 wykonania danego rozkazu PO7 nie będą powodowały drukowania (pierwszy wydruk następuje zawsze przy pierwszym wykonaniu rozkazu PO7).
Liczba L3 ($1 \leqslant L3 \leqslant 7$) określa łączną ilość wydruków wykonywanych przez dany rozkaz PO7. Każdy wydruk powoduje odjęcie 1 od liczby L3. Jeżeli w programie zachodzi potrzeba wykonania więcej niż 7 wydruków przez dany rozkaz PO7, to wówczas, należy po każdym wykonaniu danego rozkazu PO7, połączonym z wydrukiem, dodać 1 do liczby L3.
Forma wydruku jest następująca. Jako pierwszy, zawsze w systemie dziesiętnym, drukowany jest licznik rozkazów LR, przy którym został wykonany rozkaz PO7. Następnie drukowane są: B-rejestr, akumulator A, mnożnik M ósemkowo lub dziesiętnie. Zawartość wskaźnika nadmiaru WN, drukowana w formie liczby ósemkowej, kończy wiersz. W następnych wierszach drukowany jest, jeśli zażądano, wskazany obszar pamięci operacyjnej, przy czym pierwsza liczba określa adres pierwszego słowa.

Jeżeli rozkaz PO7 nie zawiera żądania drukowania obszaru pamięci, to po jego wykonaniu następuje przejście do następnego rozkazu (LR zwiększa się o 1), natomiast jeśli takie żądanie występuje, to po wykonaniu rozkazu następuje przeskoczenie jednego rozkazu (LR zwiększa się o 2).

Rozkaz nie zmienia zawartości rejestrów A, B, M; niszczy natomiast zawartość komórki PAO [1+BD].

W programie, w języku PJP, używającym ten rozkaz należy umieścić dyrektywę F2046,7–0

Przykłady:

| | |
|---|---|
| PO7 2 | wydruk zawartości rejestrów B, A, M w systemie dziesiętnym oraz wskaźnika WN; wydruk nastąpi jedynie przy pierwszym i drugim wykonaniu rozkazu PO7 |
| PO7 7+ | wydruk zawartości rejestrów B, A, M w systemie ósemkowym oraz wskaźnika WN; wydruk nastąpi jedynie przy pierwszych siedmiu wykonaniach rozkazu PO7 |
| PO7 = 34+ | wydruk zawartości rejestrów B, A, M w systemie ósemkowym oraz wskaźnika WN; wydruk nastąpi jedynie przy pierwszym, piątym, dziewiątym i trzynastym wykonaniu rozkazu PO7 |
| PO7 = 21772+↑<br>STS E12/19 | wydruk zawartości rejestrów B, A, M w systemie ósemkowym oraz wskaźnika WN i wydruk, również w systemie ósemkowym, zawartości 17 komórek pamięci operacyjnej począwszy od komórki o adresie określonym przez wartość etykiety E12/19; wydruk nastąpi jedynie przy pierwszym i sześćdziesiątym piątym wykonaniu rozkazu PO7 |

P 16

Do liczby zmiennoprzecinkowej umieszczonej w akumulatorze-mnożniku AM dodaj liczbę zmiennoprzecinkową umieszczoną w komórkach pamięci określonych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 (rys. 4.13), po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). B-modyfikacja adresu tego rozkazu powoduje dodanie do niego podwojonej zawartości B-rejestru. Rozkaz nie zmienia zawartości B. W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F2,16–0

Czas wykonania rozkazu 1.2 ms.

P 17

Od liczby zmiennoprzecinkowej umieszczonej w akumulatorze-mnożniku AM odejmij liczbę zmiennoprzecinkową umieszczoną w komórkach pamięci określonych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 (rys. 4.13), po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). B-modyfikacja adresu tego rozkazu powoduje dodanie do niego podwojonej zawartości B-rejestru. Rozkaz nie zmienia zawartości B.
W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F2,17–1

Czas wykonania rozkazu 1.2 ms.

P 18

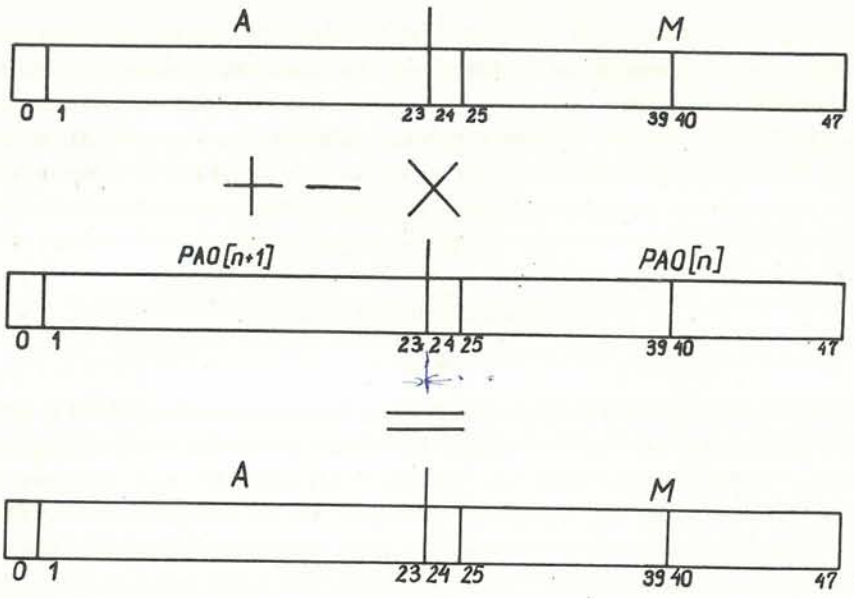
Pomnóż liczbę zmiennoprzecinkową umieszczoną w akumulatorze-mnożniku AM przez liczbę zmiennoprzecinkową umieszczoną w komórkach pamięci wskazanych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 i umieść wynik w AM (rys. 4.13), po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). B-modyfikacja adresu tego rozkazu powoduje dodanie do niego podwojonej zawartości B-rejestru. Rozkaz nie zmienia zawartości B.
W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F2,18–2

Czas wykonania rozkazu 1.9 ms.

P 19

Podziel liczbę zmiennoprzecinkową umieszczoną w akumulatorze-mnożniku AM przez liczbę zmiennoprzecinkową umieszczoną w komórkach pamięci wskazanych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 i umieść wynik w AM (rys. 4.13), po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). B-modyfikacja adresu tego rozkazu powoduje dodanie do niego podwojonej zawartości B-rejestru. Rozkaz nie zmienia zawartości B.
W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F2,19–3

Czas wykonania rozkazu 2.2 ms.




Rys. 4.13. Operacje arytmetyczne zmiennoprzecinkowe

P 20

Pomnóż liczbę stałoprzecinkową ułamkową długą umieszczoną w akumulatorze-mnożniku AM przez liczbę stałoprzecinkową ułamkową długą umieszczoną w komórkach pamięci wskazanych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 i umieść wynik w AM (rys. 4.14), po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). B-modyfikacja adresu tego rozkazu powoduje dodanie do niego podwojonej zawartości B-rejestru. Rozkaz nie zmienia zawartości B.
W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F2,20–4

Czas wykonania rozkazu 1.7 ms.

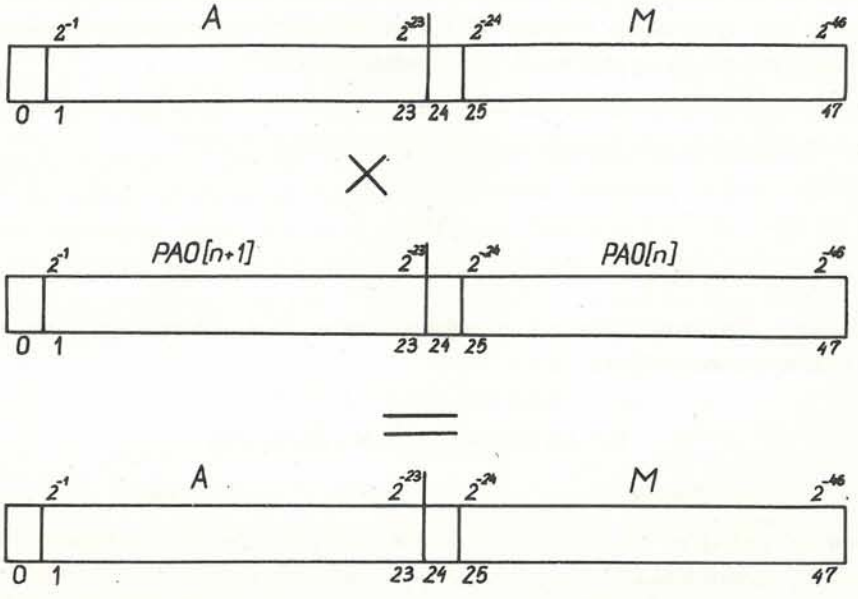
P 21

Podziel liczbę stałoprzecinkową ułamkową długą umieszczoną w akumulatorze-mnożniku AM przez liczbę stałoprzecinkową ułamkową długą umieszczoną w komórkach pamięci wskazanych przez: adres efektywny rozkazu i adres efektywny rozkazu zwiększony o 1 i umieść wynik w AM (rys. 4.14), po czym przejdź do następnego rozkazu (licznik rozkazów LR zwiększ o 1). Jeżeli wartość bezwzględna wyniku dzielenia jest nie

mniejsza od 1, to wpisz jedynkę do wskaźnika nadmiaru WN. B-modyfikacja adresu tego rozkazu powoduje dodanie do niego podwojonej zawartości B-rejestru. Rozkaz nie zmienia zawartości B.

W programie, w języku PJP, używającym ten rozkaz, należy umieścić dyrektywę F2,21–5

Czas wykonania rozkazu 1.9 ms.




Rys. 4.14. Mnożenie i dzielenie stałoprzecinkowe

## 4.2. ROZKAZY NIELEGALNE

W zbiorze rozkazów m.c. ZAM 41 istnieje grupa rozkazów, które mogą być wykonywane jedynie przy zapalonym wskaźniku legalności WL. Do grupy tej należą rozkazy: zatrzymania maszyny, ładowania rejestrów blokady dolnej BD i górnej BG oraz rozkazy współpracy z urządzeniami zewnętrznymi. Próba wykonania rozkazu z tej grupy przy zgaszonym WL sygnalizowana jest przerwaniem typu „nielegalny rozkaz" (pkt. 1.7 i 2.1).

Taka sama sygnalizacja pojawia się przy próbie wykonania rozkazu zmieniającego zawartość komórki pamięci o adresie leżącym poza granicami określonymi przez zawartość rejestrów BD i BG+127 przy zgaszonym wskaźniku legalności WL.

Omówione rozkazy nazywają się rozkazami nielegalnymi. W programie użytkowym rozkazy takie mogą występować, natomiast nie mogą być wykonywane. Próba wykonania takiego rozkazu powoduje przerwanie, a w ślad za tym wykonywanie programu zostaje zakończone, czemu towarzyszy wydruk (objaśniony w pkt. 4.1.2):

LR = < licznik rozkazów > NIELEGALNY ROZKAZ
POZOSTAŁO CYKLI < ilość cykli >
B = < zawartość B-rejestru >

Ponadto wystąpienie w programie użytkowym rozkazu sterującego (z wyjątkiem rozkazów SLR opisanych w pkt. 4.1.2), który wpisuje do licznika rozkazów LR liczbę leżącą poza przedziałem określonym przez zawartość rejestrów BD i BG+127, prowadzi na ogół do pojawienia się, po wykonaniu kilku rozkazów, rozkazu nielegalnego. Tak więc rozkazy tego typu można również zaliczyć do nielegalnych, jakkolwiek one same nie powodują pojawienia się przerwania typu „nielegalny rozkaz".

Obecnie podamy określenia rozkazów nielegalnych: zatrzymania maszyny, ustawiania zawartości rejestrów oraz współpracy z urządzeniami zewnętrznymi.

STOP

0(0) STS

Wpisz adres efektywny rozkazu do licznika rozkazów LR, a następnie zatrzymaj maszynę. Czas wykonania rozkazu 22 $\mu$ s.

USTAW DOLNĄ I GÓRNĄ BLOKADĘ

29(35) DGB

Zawartość pozycji nr 16–23 komórki pamięci wskazanej przez adres efektywny rozkazu wpisz do rejestru blokady dolnej BD, zaś zawartość pozycji nr 4–11 tej komórki wpisz do rejestru blokady górnej BG. Po wykonaniu tego rozkazu zawartość licznika rozkazów LR zwiększ o 1. Czas wykonania rozkazu 44 $\mu$ s.

Przejdziemy teraz do omówienia rozkazów współpracy z urządzeniami zewnętrznymi. Konkretnie reprezentacje tych rozkazów podano w pkt. 2.2 i 2.3 przy opisie poszczególnych urządzeń a także w dodatku 2. Rozkazy te (rys. 4.15) mają część operacyjną wydłużoną do 9 bitów, która podana jest w niniejszym opisie za pomocą dwóch liczb rozdzielonych przecinkiem. Pierwsza z nich określa zawartość bitów nr 3–8 rozkazu (a więc normalną część operacyjną), druga zaś określa zawartość bitów nr 9–11 rozkazu (wydłużenie części operacyjnej).
Znaczenie bitów części adresowej rozkazu jest następujące:
bit nr 12–14 określają numer rejestru lub wskaźnika, na którym będzie wykonany rozkaz
bity nr 15–18 określają numer kanału, do którego podłączono urządzenie
bity nr 19–23 określają ilość bitów przesyłanych do wskazanego rejestru urządzenia.

Wykonanie każdego z niżej omówionych rozkazów powoduje zwiększenie licznika rozkazów LR o 1.


Rys. 4.15. Struktura rozkazów współpracy z urządzeniami zewnętrznymi

## CZYTAJ DO MNOŻNIKA W LEWO

63,0(77,0) CML

Zawartość mnożnika M przesuń w lewo o wielkość określoną przez zawartość bitów nr 19–23 adresu efektywnego rozkazu. Jednocześnie przesuwaj w lewo zawartość rejestru wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu i za każdym przesunięciem bit wychodzący z najwyższej pozycji rozpatrywanego rejestru wprowadź do pozycji nr 23 M. Czas wykonania rozkazu 37 $\mu$s + n·5 $\mu$s, gdzie n jest ilością przesunięć.

## PISZ MNOŻNIK ZNAKAMI

63,1(77,1) PMZ

Kolejne znaki 4,6 lub 8-bitowe (w zależności od typu urządzenia, do którego przesyłane są znaki) zapisane w mnożniku M wpisuj jeden po drugim do rejestru wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu.
Czynność powyższa wykonywana jest aż do momentu wybrania wszystkich znaków z M. Bity nr 19–23 adresu efektywnego rozkazu nie odgrywają żadnej roli.
Rozkaz PMZ może być wykonywany jednocześnie z innymi rozkazami. Oznacza to, że po zainicjowaniu przesyłania znaków z mnożnika M maszyna może rozpocząć wykonywanie następnych rozkazów. Jeśli jednak wśród tych rozkazów pojawi się rozkaz dotyczący M lub powodujący przesunięcie cykliczne A, to jego wykonywanie zostanie wstrzymane aż do zakończenia wykonywania rozkazu PMZ.

## CZYTAJ DO MNOŻNIKA W PRAWO

63,2(77,2) CMP

Zawartość mnożnika M przesuń w prawo o wielkość określoną przez zawartość bitów nr 19–23 adresu efektywnego rozkazu. Jednocześnie przesuwaj w lewo zawartość rejestru wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu i za każdym przesunięciem bit wychodzący z najwyższej pozycji rozpatrywanego rejestru wprowadź do pozycji nr O M.
Czas wykonania rozkazu 37 $\mu$s +n·5 $\mu$s, gdzie n jest ilością przesunięć.

## PISZ WSKAŹNIK

63,3(77,3) PWT

Wpisz 1 do wskaźnika określonego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu. Bity nr 19–23 adresu efektywnego rozkazu nie odgrywają żadnej roli.
Czas wykonania rozkazu 35 $\mu$s.

## PISZ MNOŻNIK W LEWO

63,4(77,4) PML

Zawartość mnożnika M przesuń w lewo o wielkość określoną przez bity nr 19–23 adresu efektywnego rozkazu i za każdym przesunięciem wpisuj 0 do pozycji nr 23 M. Jednocześnie przesuwaj w prawo zawartość rejestru wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu i za każdym przesunięciem bit wychodzący z pozycji nr 0 M wpisz do najwyższej pozycji rozpatrywanego rejestru.
Czas wykonania rozkazu 37 $\mu$s $+n \cdot 5$ $\mu$s, gdzie n jest ilością przesunięć.

## CZYTAJ DO MNOŻNIKA ZNAKAMI

63,5(77,5) CMZ

Kolejne znaki 4,6 lub 8-bitowe (w zależności od typu urządzenia, z którego przesyłane są znaki) wpisywane do rejestru wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu czytaj do mnożnika M aż do całkowitego jego wypełnienia.
Bity nr 19–23 adresu efektywnego rozkazu nie odgrywają żadnej roli.
Rozkaz CMZ może być wykonywany jednocześnie z innymi rozkazami. Oznacza to, że po zainicjowaniu przesyłania znaków do mnożnika M, maszyna może rozpocząć wykonywanie następnych rozkazów. Jeśli jednak wśród tych rozkazów pojawi się rozkaz dotyczący M lub powodujący przesunięcie cykliczne A, to jego wykonywanie zostanie wstrzymane aż do zakończenia wykonywania rozkazu CMZ.

## PISZ MNOŻNIK W PRAWO

63,6(77,6) PMP

Zawartość mnożnika M przesuń w prawo o wielkość określoną przez bity nr 19–23 adresu efektywnego rozkazu i za każdym przesunięciem wpisz 0 do pozycji nr 0 M. Jednocześnie przesuwaj w prawo zawartość rejestru wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu i za każdym przesunięciem bit wychodzący z pozycji nr 23 M wpisz do najwyższej pozycji rozpatrywanego rejestru.
Czas wykonania rozkazu 37 $\mu$s $+n \cdot 5$ $\mu$s, gdzie n jest ilością przesunięć.

## CZYTAJ WSKAŹNIK

63,7(77,7) CWT

Zbadaj stan wskaźnika wskazanego przez bity nr 12–14 i 15–18 adresu efektywnego rozkazu. Jeżeli wskaźnik ten zawiera 0 (jest zgaszony), to do zawartości licznika rozkazów LR dodaj 2. Jeżeli wskaźnik ten zawiera 1 (jest zapalony), to wpisz 0 do wskaźnika i dodaj 1 do LR. Bity nr 19–23 adresu efektywnego rozkazu nie odgrywają żadnej roli.
Czas wykonania rozkazu 35 $\mu$s.
Uwaga: Rozkaz CWT 0 (czytaj i zeruj wskaźnik przyjęć części centralnej WP) powoduje zawsze zwiększenie licznika rozkazów LR o 2, niezależnie od stanu wskaźnika WP.

# 5. ELEMENTY PROGRAMU W JĘZYKU PJP

Program napisany w języku PJP składa się z *wierszy.*
Wierszem nazywać będziemy dowolny ciąg znaków alfabetu języka PJP, będącego podzbiorem kodu dalekopisowego M2, nie zawierający znaków powrotu karetki CR i nowej linii LF, zakończony ciągiem złożonym ze znaków CR i LF, zawierającym co najmniej jeden znak LF.
*Komentarzem* nazywać będziemy dowolny ciąg znaków z alfabetu języka PJP zaczynający siię od dwukropka i nie zawierający znaków powrotu karetki CR i nowej linii LF.
*Wierszem pustym* nazywać będziemy wiersz zawierający wyłącznie znaki spacji SP, cyfr FS, liter LS, powrotu karetki CR, nowej linii LF, puste BL lub komentarz.
Wiersz, który nie jest wierszem pustym oraz nie zaczyna się od przecinka lub apostrofu nazywać będziemy *wierszem znaczącym*, zaś wiersz zaczynający się od przecinka lub apostrofu – *wierszem warunkowym.*
Z kolei wiersze znaczące dzielą się na *wiersze informacyjne,* będące opisami słowa maszyny, oraz *dyrektywy i parametry,* niosące pewne informacje dla systemu operacyjnego i translatora.

## 5.1. ALFABET JĘZYKA PJP

W programie napisanym w języku PJP mogą występować następujące znaki kodu M2:

a) litery

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

b) cyfry

0 1 2 3 4 5 6 7 8 9

c) znaki pomocnicze ' , . : + – / ↑ =

d) znaki wydawnicze:
powrót karetki CR, linia LF, spacja SP, cyfry FS, litery LS, znaki puste BL

e) znaki w komentarzach:

? ( )

Wystąpienie na taśmie programu znaku różnego od wymienionych znaków kodu M2, (tj. znaku Ą, Ę, Ł, * ) jest błędem (błąd 1).

## 5.2. WIERSZE INFORMACYJNE

Wiersze informacyjne będące opisami słów maszyny, dzielą się na: rozkazy (pkt. 5.2.2), oraz liczby (dziesiętne) i szablony (liczby ósemkowe) (pkt. 5.2.1).

### 5.2.1. Zapis liczb

Opis słowa maszyny przy pomocy liczby dziesiętnej całkowitej utworzony jest przez ciąg co najwyżej 7 cyfr dziesiętnych, poprzedzonych zawsze znakiem + lub –. Wartość bezwzględna utworzonej w ten sposób liczby nie może przekraczać $2^{23}-1 = 8\,388\,607$, w przeciwnym razie translator sygnalizuje błąd (błąd 48).

Przykłady:

+256617

–38

–0

Wystąpienie znaku „+" oznacza, że bit nr 0 (bit znaku) tworzonego słowa jest zerem, wystąpienie zaś znaku „–" oznacza, że bit ten jest jedynką. Bity nr 1–23 tworzonego słowa określone są przez postać binarną modułu liczby dziesiętnej całkowitej.

Opis słowa maszyny przy pomocy liczby ósemkowej nazywa się *szablonem*. Szablon zaczyna się od znaku „=", po którym następuje ciąg, zawierający cyfry od 0 do 7, o długości od 1 do 8 cyfr. Kolejne cyfry od prawej do lewej określają odpowiednio trójki bitów nr 21–23, 18–20, 15–17, 12–14, 9–11, 6–8, 3–5, 0–2.

Jeżeli szablon zawiera mniej niż 8 cyfr, to brakujące pozycje uzupełnia się zerami z lewej strony.

Przykłady:

= 34570017

= 00077334 jest równoważny = 77334

Szablon zawierający więcej niż 8 cyfr jest niepoprawny (błąd 7).

### 5.2.2., Zapis rozkazów

Rozkaz w języku PJP ma następującą postać:

• < operacja > < adres > + ↵

Poszczególne elementy wiersza tworzącego rozkaz mogą być pominięte, z wyjątkiem części < operacja >.

Przykłady:

```
WMB
LCA 15
UMA 30.
SKO +5+↵
PMB = 137
PAK +0
UAM 24E60/15
```

PZM 3E1/O ↑
.ODS 2E3+ ↑
SMA 7/3
.STS 2B5

Znaczenie poszczególnych elementów tworzących rozkaz w języku PJP jest następujące:

a) wystąpienie znaku „·" przed literami określającymi operację oznacza, że bit nr 0 tworzonego słowa jest jedynką.
W przeciwnym przypadku bit ten jest zerem.

b) część < operacja > jest jednym z podanych w pkt. 4 skrótów trójznakowych i określa część operacyjną tworzonego rozkazu (bity nr 3–8 lub 3–11).

c) część < adres > określa część adresową tworzonego rozkazu, czyli zawartość bitów nr 9–23 lub 12–23. Adres rozkazu może być zapisany w czterech formach jako adres: rzeczywisty, przesunięty, względny lub symboliczny.

*Adres rzeczywisty* ma formę liczby całkowitej bez znaku, która nie może przekraczać 32 767.

*Adres przesunięty* tworzy liczba całkowita bez znaku i umieszczony za nią znak „·"(kropki). Część adresowa tworzonego rozkazu jest w tym wypadku sumą liczby całkowitej bez znaku i liczby 1280 (dla I zestawu z Algolem) lub 2432 (dla II zestawu bez Algolu). Adres taki nie ma nic wspólnego z D-modyfikacją, ponieważ dodanie liczby 1280 lub 2432 następuje już w procesie translacji.

*Adres względny* utworzony jest przez liczbę całkowitą bez znaku, poprzedzoną znakiem + lub –. W tym wypadku, poza pewnymi przypadkami szczególnymi, można przyjąć, że część adresowa rozkazu określona jest przez sumę adresu komórki pamięci, w której przewiduje się umieszczenie rozpatrywanego rozkazu i liczby całkowitej ze znakiem. Ściślej: w trakcie translacji translator tworzy dwa liczniki programowe q i b (pkt. 5.3), których wartości początkowe określone są przez dyrektywy V i Q, i których zawartość zwiększa się o 1 po translacji każdego wiersza informacyjnego. W tym ujęciu adres rozkazu powstaje w wyniku dodania do liczby całkowitej ze znakiem, aktualnej zawartości licznika q.

*Adres symboliczny* utworzony jest zawsze przy pomocy etykiety typu E lub B (pkt. 5.4), czyli symbolu, któremu przyporządkowuje się pewną wartość.

Adres symboliczny może posiadać następujące formy:

1. < liczba 1 > E < liczba 2 >/< liczba 3 >
   gdzie < liczba 1 > – liczba całkowita bez znaku lub szablon
   < liczba 2 > – liczba całkowita bez znaku
   lub < liczba 3 > o wartości 0–63

Część adresowa tworzonego rozkazu równa jest sumie < liczby 1 > i wartości (pkt. 5.4.1) etykiety E o numerze < liczba 2 >, zadeklarowanej w paragrafie o numerze < liczba 3 >. Gdy wartość < liczby 1 > wynosi 0, to < liczbę 1 > można pominąć. Jeżeli etykieta o numerze < liczba 2 > została zadeklarowana w tym samym paragrafie, co tworzony rozkaz, to można wówczas pominąć < liczbę 3 > wraz ze znakiem „/".

2. < liczba 1 > / < liczba 2 >

gdzie < liczba 1 > — liczba całkowita bez znaku
< liczba 2 > — liczba całkowita bez znaku o wartości 0 – 63

Część adresowa określona jest przez sumę < liczby 1 > i adresu pierwszego słowa należącego do paragrafu o numerze < liczba 2 > (ściślej: wartości dyrektywy Y paragrafu oznaczonego numerem < liczba 2 >).

3. < liczba 1 > B < liczba 2 >

gdzie < liczba 1> — liczba całkowita bez znaku lub szablon
< liczba 2> — liczba całkowita bez znaku o wartości 1 – 2047

Część adresowa określona jest przez sumę < liczby 1 > i wartości etykiety bębnowej B < liczba 2 >. Gdy wartość < liczby 1 > wynosi 0, to można ją pominąć.

Uwaga: należy pamiętać, że część adresowa zawiera 15 (lub 12, gdy rozkaz ma wydłużoną część operacyjną) bitów, toteż adres rozkazu po przetłumaczeniu programu (translacji) powinien być mniejszy od 32 768 (lub 4096, gdy część operacyjna jest wydłużona).

d) wystąpienie znaku „+" za adresem oznacza, że bit nr 2 tworzonego słowa (bit B-modyfikacji) jest jedynką, w przeciwnym wypadku bit ten jest zerem.

e) wystąpienie znaku „ ⍐ " za adresem oznacza, że bit nr 1 tworzonego słowa (bit P-modyfikacji) jest jedynką, w przeciwnym wypadku bit ten jest zerem.

## 5.3. DYREKTYWY I PARAMETRY JĘZYKA PJP

Prócz dotychczas omówionych wierszy informacyjnych opisujących tworzone słowa, program w języku PJP zawiera również dodatkowe informacje zwane tu dyrektywami i parametrami, określające m.in. położenie programu w pamięci operacyjnej i bębnowej, maksymalną objętość wyników, maksymalny czas wykonywania programu itp.

W celu bardziej precyzyjnego sformułowania znaczenia niektórych, dalej opisanych dyrektyw, wprowadźmy dwa umowne liczniki programowe q i b. Licznik q określa w trakcie translacji adres komórki pamięci, w której, w czasie wykonywania programu, znajdować się będzie tworzone słowo. Licznik b określa, w trakcie translacji, adres bębnowy miejsca w pamięci bębnowej, do którego zostanie wpisane tworzone słowo. Zawartości obu liczników ustawiane są przez odpowiednie dyrektywy, przy czym po każdym wierszu informacyjnym liczniki q i b zostają zwiększone o 1.

Uwaga: w dyrektywach, podobnie jak w adresach przesuniętych (pkt. 5.2.2), dopisanie znaku „." (kropki) po liczbie oznacza zwiększenie liczby o 1280 (w I zestawie z Algolem) lub o 2432 (w II zestawie bez Algolu).

### 5.3.1. Dyrektywa A.

Dyrektywa stosowana jest przy tworzeniu w programie obszarów przewidzianych do przechowywania zbiorów (tj. liczb, tekstów itp.).

Forma dyrektywy:

A < ilość opuszczonych słów >

Dyrektywa ta powoduje jednoczesne opuszczenie w pamięci operacyjnej i na bębnie ilości słów określonych przez liczbę umieszczoną za literą A. Ściślej: dyrektywa ta powoduje zwiększenie stanów liczników programowych „q" i „b" o liczbę umieszczoną za literą A.

Przykłady:

A21
A3
A42

### 5.3.2. Dyrektywa F.

Dyrektywa ta służy do dołączania podprogramów standardowych do programu właściwego. Dyrektywa ta może przyjmować następujące formy:

a) F < nr etykiety bębnowej, którą opatrzono pożądany podprogram >
lub

b) F < nr etykiety bębnowej,... >, < liczba 1 > − < liczba 2 >, < liczba 3 > − < liczba 4 >,.....

Dyrektywa ta powoduje wykonanie kolejno następujących czynności:

a) pobiera słowo z pamięci bębnowej, które opatruje etykietą bębnową o wskazanym numerze – słowo to, zgodnie z opisem struktury podprogramu (pkt. 6.2.1), określa ilość słów n podprogramu.
b) przepisuje n słów podprogramu w pamięci bębnowej od adresu równego zwiększonej o 1 wartości etykiety bębnowej opatrującej rozpatrywany podprogram do kolejnych miejsc pamięci bębnowej, których adres początkowy określa zawartość licznika programowego „b". Do chwili natrafienia, podczas przepisywania, na słowo złożone z samych jedynek ( = 77777777), do wartości bezwzględnej każdego słowa, które ma 1 na bicie nr 0, dodawana jest zawartość licznika programowego „q".
c) zawartość liczników programów „q" i „b" zostaje zwiększona o n, czyli o ilość słów podprogramu.
d) jeżeli dyrektywa F występuje w formie b, to do komórek

PAO [< liczba 1 >], PAO [< liczba 3>],

itd. zostaną wpisane rozkazy skoku bezwarunkowego SKO, z adresami równymi, odpowiednio, sumie zawartości licznika programowego q przed wykonaniem dyrektywy F i < liczby 2 >, < liczby 4> itd.

Przykład:

F2046,7–0

Dla ustalenia uwagi przyjmiemy, że zawartość licznika programowego q przed wykonaniem dyrektywy wynosiła 3000, zaś licznika programowego b 28 000. Etykieta B2046 ma wartość 2878, zaś zawartość słowa pamięci bębnowej o adresie 2878 wynosi 200. Wówczas wykonane zostaną następujące czynności:

a) z komórki pamięci bębnowej o adresie 2878 zostanie pobrana liczba określająca długość podprogramu – w tym przykładzie ma ona wartość 200.
b) w pamięci bębnowej nastąpi przepisanie 200 słów z obszaru o adresie 2879 do obszaru o adresie 28 000 (zawartość licznika programowego b przed wykonaniem dyrektywy F). Aż do natrafienia na słowo podprogramu złożone z samych jedynek ( = 77777777), do modułu każdego słowa, które ma jedynkę na bicie nr 0, dodawana będzie liczba 3000 (zawartość licznika programowego q przed wykonaniem dyrektywy F).
c) po wykonaniu dyrektywy zawartość licznika q wynosić będzie 3200, zaś licznika b 28 200.
d) do komórki PAO [BD+7] wpisany zostanie rozkaz skoku bezwarunkowego z adresem równym 3000 (zawartość licznika q przed wykonaniem dyrektywy zwiększona o liczbę 2).

W trakcie wykonywania programu ewentualny rozkaz PO7 spowoduje ustawienie licznika rozkazów LR na wartość BD+7. Tu zaś został wpisany rozkaz skoku bezwarunkowego z adresem 3000; w rezultacie licznik rozkazów LR zostanie ustawiony na 3000, czyli jako następny zostanie wykonany pierwszy rozkaz podprogramu.

### 5.3.3. Dyrektywa G

Dyrektywa ta anuluje działanie ostatniej dyrektywy H (opisanej poniżej). Dyrektywa ta stosowana jest w podprogramach standardowych. Forma dyrektywy jest następująca:

G

### 5.3.4. Dyrektywa H

Dyrektywa ta powoduje, że występujące za nią rozkazy z adresami *względnymi* lub *symbolicznymi* są zaopatrywane w jedynkę na bicie nr 0. Dyrektywa stosowana jest w podprogramach standardowych. Forma dyrektywy jest następująca:

H

### 5.3.5. Dyrektywa I

Dyrektywa ta umieszczona jest zawsze na początku programu, po nazwie translatora. Dyrektywa I wyznacza obszary pamięci bębnowej wykorzystywane w procesie translacji, a także określa inne, opisane dalej, parametry programu. Dyrektywa ta może przyjmować formy mniej lub więcej rozbudowane; w najprostszym przypadku ma ona postać „I", zaś w przypadku najbardziej złożonym postać tej dyrektywy jest następująca:

I0, < liczba 1 >
I1, < liczba 2 >
I2, < liczba 3 >
I3, < liczba 4 >
I4, < liczba 5 >
I5, < liczba 6 >
I6, < liczba 7 >
I7, < liczba 8 >
IA < liczba 9 > – < liczba 10 >, < liczba 11 > – < liczba 12 >,
< liczba 13 > – < liczba 14 >, < liczba 15 > – < liczba 16 >

Pierwsza część tej dyrektywy, do kodu IA, nadaje wartości parametrom I0, I1,..., I7 programu, odpowiednio, < liczba 1 >, < liczba 2 >..., < liczba 8 >. Pominięcie jednego lub kilku parametrów jest równoznaczne z nadaniem im wartości standardowych, podanych w tablicy 5.1.

Znaczenie parametrów I0, I1,..., I7 jest następujące:

*Parametry I0, I1* określają odpowiednio adres początku i końca obszaru bębnowego przeznaczonego na program wynikowy, przy czym miejsce pamięci o adresie I1 nie należy do tego obszaru. Wyjście tłumaczonego programu poza ten obszar traktowane jest jako błąd (błąd 5).

*Parametry I2, I3* określają odpowiednio adres początku i końca obszaru bębnowego przeznaczonego na listę rozkazów zawierających adresy symboliczne – etykiety E, przy czym miejsce pamięci o adresie I3 nie należy do tego obszaru. Pojemność tej listy nie powinna być mniejsza od maksymalnej ilości rozkazów zawierających adresy symboliczne (typu E) w poszczególnych sekcjach (pkt. 6.3.) programu. W przypadku, gdy pojemność tego obszaru okaże się za mała, translator PJP sygnalizuje błąd.

*Parametry I4, I5* określają odpowiednio adres początku i końca obszaru bębnowego przeznaczonego na listę rozkazów zawierających adresy symboliczne – etykiety B, przy czym miejsce pamięci o adresie I5 nie należy do tego obszaru. Pojemność tej listy nie powinna być mniejsza od ilości rozkazów zawierających etykiety bębnowe B w całym programie. W przypadku, gdy pojemność tego obszaru okaże się dla pewnego programu za mała, translator PJP sygnalizuje błąd.

*Parametr I6* określa maksymalny czas wykonywania programu w jednostkach zwanych cyklami (1 cykl = 0.35 sek.). Maksymalny czas wykonywania programu określa także parametr STS n znajdujący się w nagłówku programu. Na wykonanie programu przydziela się czas, który stanowi minimum z tych dwóch liczb.

*Parametr I7* określa maksymalną wielkość wydruków dla perforatorów i drukarki. Właściwe ograniczenie dla każdego z tych urządzeń stanowi liczba będąca minimum liczb: zadeklarowanej w nagłówku programu (dla danego urządzenia), określonej przez parametr I7.

Następna część dyrektywy I, umieszczona za kodem IA, zawiera co najwyżej 4 pary liczb bez znaków. Liczby te określają przedziały numerów wierszy warunkowych, które przy wczytywaniu programu traktowane będą jako wiersze znaczące. Jeżeli w programie nie ma wierszy warunkowych lub wszystkie istniejące należy ignorować, to dyrektywa przyjmuje postać „I" lub „IA".

Przykład:

```
IO,20480
I3,20480
IA 17–21, 75–78, 131–131.
```

W wyniku wykonania tej dyrektywy nastąpi zmiana standardowego podziału pamięci bębnowej (tablica 5.1 zestaw II). Obszar pamięci przeznaczony na program wynikowy zostanie zmniejszony do 12 288 słów, zaś obszar pamięci przeznaczony na listę rozkazów zawierających etykiety E zwiększy się o 4096 słów, w porównaniu do podziału standardowego. Ponadto wiersze warunkowe programu oznaczone numerami 17, 18, 19, 20, 21, 75, 76, 77, 78, 131 będą traktowane jako wiersze znaczące, a wiersze warunkowe oznaczone innymi numerami będą ignorowane.

Tablica 5.1.

Parametry standardowe translatora PJP

| | I Zestaw (z Algolem) | II zestaw (bez Algolu) |
|---|---|---|
| I0 | 20480 | 16384 |
| I1 | 32768 | 32768 |
| I2 | 16896 | 8704 |
| I3 | 20480 | 16384 |
| I4 | 16384 | 8192 |
| I5 | 16896 | 8704 |
| I6 | 2048 | 2048 |
| I7 | 2048 | 2048 |

### 5.3.6. Dyrektywa K

Dyrektywa ta wskazuje zakończenie programu. Forma dyrektywy:

K < liczba 1 >/< liczba 2 >

Dyrektywa ta powoduje zakończenie wprowadzania programu, następnie zakończenie jego translacji i w przypadku braku błędów formalnych wydrukowanie tekstu STA < liczba 1 >, a następnie przepisanie programu z pamięci bębnowej począwszy od adresu określonego przez wartość etykiety bębnowej B o numerze < liczba 2 > do pamięci

operacyjnej począwszy od komórki o adresie < liczba 1 > aż do komórki o adresie BG+127–16 = BG+111. Następnie rozpoczyna się wykonywanie programu począwszy od miejsca pamięci wskazanego przez < liczbę 1 >.

Przykłady:

K256./12
K3072/107
K1536/1

### 5.3.7. Dyrektywa Q

Dyrektywa ta nadaje wartość licznikowi programowemu „q" równą liczbie umieszczoną za literą Q. Dyrektywa ma formę:

Q < liczba >

Ponieważ musi być nadal zachowana, określona przez dyrektywę V, odpowiedniość między adresami w PAO i na bębnie, ulega również zmianie zawartość licznika programowego „b": zawartość jego zostaje zwiększona o różnicę wartości licznika „q" po i przed wykonaniem dyrektywy Q. Można to zapisać tak:

q' ← < liczba >
b' ← b+< liczba >–q

gdzie b' i q' oznaczają zawartości liczników po wykonaniu dyrektywy, zaś b i q przed.

Przykłady:

Q512.

Dyrektywa ta nadaje licznikowi „q" wartość 512+1280 = 1792 (I zestaw z Algolem) lub 512+2432 = 2944 (II zestaw bez Algolu).

Q3072

Dyrektywa nadaje licznikowi q wartość 3072.

### 5.3.8. Dyrektywa T

Dyrektywa ta wskazuje koniec sekcji programu. Pojawienie się jej w programie powoduje zakończenie adresowania rozkazów o adresach symbolicznych typu E (etykiety E), po czym następuje zlikwidowanie słownika etykiet E zakończonej sekcji. Dyrektywa ta ma formę:

T

### 5.3.9. Dyrektywa V

Dyrektywa określa współzależność pomiędzy zawartościami liczników programowych „q" i „b". Występuje w dwóch formach:

a) V < liczba 1 >, < liczba 2 >
   lub
b) VB, < liczba 3 >

Dyrektywa nadaje wartość licznikowi programowemu „b" w oparciu o aktualny stan liczników programowych „q" i „b" oraz liczby występujące w dyrektywie.
Dyrektywa w formie (a) poleca nadać licznikowi „b" taką wartość, ażeby miejsce pamięci bębnowej < liczba 1 >, licząc od początku obszaru przewidzianego na program wynikowy (obszar ten określają dyrektywy X, I), odpowiadało miejscu pamięci operacyjnej o adresie < liczba 2 >. Można to zapisać tak:

$$b' \leftarrow x+q+<\text{liczba } 1>-<\text{liczba } 2>$$

gdzie x – adres początku obszaru bębnowego przeznaczonego na program wynikowy

b' – zawartość licznika „b" po wykonaniu dyrektywy V
q – zawartość licznika „q"

Dyrektywa w formie (b) poleca tak ustawić zawartość licznika „b", aby bieżącej jego zawartości odpowiadał adres pamięci operacyjnej <liczba 3>. Można to zapisać tak:

$$b' \leftarrow b+q-<\text{liczba } 3>$$

gdzie: b – zawartość licznika „b" przed wykonaniem dyrektywy V.
Przykłady:

V512,3584
VB,102
VO,O
VB,1024.

### 5.3.10. Dyrektywa X

Dyrektywa ta wydziela na bębnie obszar przeznaczony na program wynikowy. Ma ona formę:

X < liczba >

Wielkość wydzielonego obszaru wynosi 512· < liczba > słów, przy czym obszar wydzielany jest wewnątrz obszaru wyznaczonego przez parametry IO, Il (pkt. 5.3.5), począwszy od adresu bębnowego wskazanego przez wyrażenie:

I1 – < liczba >· 512

Przykład:

X12

Dyrektywa powoduje wydzielenie w pamięci bębnowej obszaru 6144 słów przeznaczonych na program wynikowy.

### 5.3.11. Dyrektywa Y

Dyrektywa ta wskazuje początek nowego paragrafu.
Ma ona formę:

Y < liczba >

Dyrektywa nadaje numer < liczba > nowemu paragrafowi.
Parametr < liczba > może przyjmować wartości 0,1,...,63. Wartością dyrektywy Y jest zawartość licznika programowego q w chwili pojawienia się dyrektywy.

W zastosowaniach dyrektywa Y wykazuje podobieństwo do etykiet (pkt. 5.4). Toteż podobnie jak dla etykiet, wprowadza się określenie zasięgu ważności numeru paragrafu: zasięgiem ważności paragrafu jest sekcja, w której rozpatrywana dyrektywa została umieszczona.

### 5.3.12. Parametr określający czas wykonywania programu

Parametr określający maksymalny czas wykonywania programu ma postać:

STS < liczba >

gdzie < liczba > określa maksymalną ilość cykli programu (1 cykl = 0.35 sek.). Parametr w postaci:

STS < liczba > ↵

poleca jednocześnie translatorowi wydrukowanie słownika etykiet (pkt. 8.1) po zakończeniu translacji.

Jednakże, jak już wspomniano w pkt. 5.3.5., jako maksymalny czas wykonywania programu przyjmuje się czas określony przez minimum z ww. parametru i parametru I6. Parametr umieszcza się w programie pod dyrektywą I.
Przykład:

STS 1000 ↵

Parametr określa maksymalny czas wykonywania programu na 1000 cykli (≅ 350 sek.), a ponadto poleca translatorowi wydrukowanie słownika etykiet po zakończeniu translacji.

Jeżeli czas wykonywania programu przekroczy zadeklarowany, program zostaje zakończony, czemu towarzyszy wydruk (objaśniony w pkt. 4.1.2):
LR = < licznik rozkazów 1 > LUB < licznik rozkazów 2 >
LUB < licznik rozkazów 3 >
WYCZERPANO CZAS
B = < zawartość B w momencie przerwania wykonywania programu>

### 5.3.13. Parametry określające maksymalną wielkość wydruków

Maksymalną wielkość wydruku obu perforatorów oraz drukarki określają trzy parametry umieszczone bezpośrednio za parametrem określającym maksymalny czas wykonywania programu. Jednakże, jako rzeczywiste ograniczenie dla danego urządzenia, system operacyjny przyjmuje minimum z maksymalnej wielkości wydruku określonego dyrektywą I7 oraz odpowiedniego, niżej opisanego parametru.

Parametry te mają formę:

STS < liczba 1 >
STS < liczba 2 >
STS < liczba 3 >

< liczba 1 >, < liczba 2 > określa, odpowiednio, maksymalną ilość bloków po 16 rządków perforowanych przez perforator I i II, < liczba 3 > określa maksymalną ilość linii drukowanych przez drukarkę wierszową.

Przekroczenie któregoś z parametrów powoduje zakończenie wykonywania programu, czemu towarzyszy wydruk (objaśniony w pkt. 4.1.2):

LR = < licznik rozkazów >
WYCZERPANO WYJŚCIE < PERF 1, PERF 2, DRUKARKA >
POZOSTAŁO CYKLI < ilość cykli >

### 5.3.14. Wydruki po zakończeniu programu – POST MORTEM

Bezpośrednio przed zakończeniem części parametrowej nagłówka programu tj. przed „–0" można umieścić do 7 żądań wypisania zawartości pamięci operacyjnej i bębnowej, po zakończeniu wykonywania programu. Każde takie żądanie opisuje się przy pomocy dwóch wierszy. Pierwszy z tych wierszy określa adres początku obszaru oraz typ wydruku. Forma tego wiersza podana jest w tablicy 5.2. Drugi wiersz określa ilość drukowanych słów i ma postać:

STS < ilość drukowanych słów >

Sumaryczna ilość słów pamięci wypisywanych po zakończeniu programu nie może przekraczać 1024.

Tablica 5.2.

POST MORTEM

| | z pamięci operacyjnej | z pamięci bębnowej |
|---|---|---|
| oktalnie | PO4 < adres > | PO4 < adres > |
| dziesiętnie | PO5 < adres > | PO5 < adres > |

Budowa części < adres > jest typowa (pkt 5.2.2).

Uwaga: adresy symboliczne zawierające etykiety E odnoszą się do pierwszej sekcji programu.

Przykłady:

PO4 2912
STS 25 – wydruk 25 słów pamięci operacyjnej począwszy od komórki o adresie 2912; wydruk w systemie ósemkowym

PO5 E12/4
STS 18 – wydruk 18 słów pamięci operacyjnej począwszy od komórki o adresie określonym przez wartość etykiety E12/4; wydruk w systemie dziesiętnym

```
PO4 17200 ↑   – wydruk 120 słów pamięci bębnowej począwszy od
STS 120         komórki o adresie 17 200; wydruk w systemie ósemkowym
PO5 B17 ↑     – wydruk 21 słów pamięci bębnowej począwszy od komórki
STS 21          o adresie określonym przez wartość etykiety bęb-
                nowej B17; wydruk w systemie dziesiętnym.
```

## 5.4. ETYKIETY

Etykiety służą do nazywania (opatrywania nazwami) pewnych istotnych punktów programu. Ściślej: etykieta jest oznaczeniem wiersza informacyjnego występującego bezpośrednio za nią, w następnym wierszu.

Dopuszczalne jest poprzedzenie wiersza informacyjnego kilkoma etykietami.

W języku PJP istnieją dwa rodzaje etykiet: etykiety wewnętrzne (typu E) i etykiety bębnowe (typu B).

### 5.4.1. Etykiety wewnętrzne

Etykietę wewnętrzną zapisuje się w postaci litery E, za którą występuje jej numer przyjmujący wartości od 1 do 63.

*Wartość* etykiety wewnętrznej równa jest zawartości licznika programowego q w momencie jej wystąpienia.

Ponieważ zawartość licznika q można interpretować jako adres komórki pamięci operacyjnej, w której zostanie umieszczony aktualnie tłumaczony rozkaz (lub liczba), więc wartość etykiety można uważać za adres komórki pamięci, w której umieszczony zostanie rozkaz (lub liczba) opatrywany przez etykietę.

*Zasięgiem ważności* etykiety wewnętrznej jest paragraf, w którym została ona zadeklarowana. Oznacza to, że w paragrafie nie może występować kilka etykiet wewnętrznych o tych samych numerach. Jednakże wewnątrz sekcji (pkt 6.3) można odwoływać się do wszystkich etykiet, które zostały w niej umieszczone (w różnych paragrafach). Bliższe informacje na temat można znaleźć w pkt. 6.3.2.

Maksymalna ilość etykiet w paragrafie wynosi 63, zaś w całej sekcji (lub w programie utworzonym przez 1 sekcję) 512.

### 5.4.2. Etykiety bębnowe

Etykietę bębnową zapisuje się w postaci litery B, za którą występuje jej numer przyjmujący wartości od 1 do 2047. Etykiet bębnowych o numerach 2, 4, 5, 6, 7, 2046 nie wolno używać, ponieważ opatrują one standardowe podprogramy bębnowe, przechowywane stale w pamięci bębnowej.

*Wartość* etykiety bębnowej równa jest zawartości licznika programowego b w momencie jej wystąpienia.

*Zasięgiem ważności* etykiety bębnowej jest cały program. Oznacza to, że w programie nie może występować kilka etykiet bębnowych o tych samych numerach.

Maksymalna ilość etykiet bębnowych w programie wynosi 256.

# 6. STRUKTURA PROGRAMU W JĘZYKU PJP

W poprzednim, 5 rozdziale omówiliśmy szczegółowo elementy programu w języku PJP, co pozwoli obecnie podać opis struktury poszczególnych bloków tworzących program: nagłówka, podprogramów standardowych oraz programu właściwego. Bloki te przedstawione są na rys. 6.3.

## 6.1. NAGŁÓWEK PROGRAMU

Nagłówek zawiera informację o sposobie tłumaczenia programu oraz określa pewne parametry obowiązujące podczas wykonywania programu.

Struktura nagłówka, z wyjątkiem żądań POST MORTEM, które można pominąć, jest ustalona. Oznacza to, że poszczególne wiersze nagłówka muszą być w nim umieszczone w opisanej dalej kolejności.

a) PROG: < tytuł programu >
   <tytuł programu> musi być umieszczony w tym samym wierszu co słowo PROG, oraz nie może zawierać więcej niż 64 znaki kodu M2 (może zawierać również znaki kodu M2 nie należące do alfabetu języka PJP).
   Słowo PROG: wskazuje początek programu.

b) PJP:
   Wiersz ten określa nazwę translatora.
   Pominięcie tego wiersza lub umieszczenie go w formie różnej od podanej spowoduje wydrukowanie tekstu:

JAKI TRANSLATOR

c) dyrektywa I – dyrektywa ta została opisana w pkt. 5.3.5
d) parametr określający maksymalny czas wykonywania programu – forma tego parametru została opisana w pkt. 5.3.12
e) parametry określające maksymalną wielkość wydruku – formę tych parametrów opisano w pkt. 5.3.13
f) żądania wydruków określonych obszarów pamięci po zakończeniu wykonywania programu – POST MORTEM – forma tych żądań została opisana w pkt. 5.3.14
g) –0 (minus zero)
   Znaki te (–0) wskazują zakończenie listy parametrów i POST MORTEM
h) dyrektywa X – dyrektywa ta została opisana w pkt. 5.3.10.

Przykład:

```
PROG: OBLICZENIA STATYSTYCZNE
PJP:
I
STS 1500 ☂
STS 200
STS 0
STS 1200
–0
X8
```

W podanym przykładzie nagłówka programu dyrektywa I ma formę standardową, tj. poleca ominąć wszelkie wiersze warunkowe oraz parametrom I0,...,I7 nadać wartości standardowe. Jeżeli przyjmiemy, że program będzie realizowany przez II zestaw systemu operacyjnego (bez Algolu), to parametrom I0,...,I7 zostaną nadane następujące wartości (tabl. 5.1):

| | |
|---|---|
| I0 | 16384 |
| I1 | 32768 |
| I2 | 8704 |
| I3 | 16384 |
| I4 | 8192 |
| I5 | 8704 |
| I6 | 2048 |
| I7 | 2048. |

Parametr występujący w następnym wierszu określa maksymalny czas wykonywania programu na 1500 cykli. Jako wartość obowiązującą dla systemu operacyjnego, przyjmuje się, zgodnie z zasadami podanymi w pkt. 5.3.5, minimum z czasu określonego przez rozpatrywany parametr i parametr I6, czyli min (1500, 2048) = 1500.

Znak ☂ umieszczony za parametrem poleca translatorowi wydrukowanie słownika etykiet po zakończeniu translacji programu. Następne trzy parametry określają maksymalną wielkość wydruku dla obu perforatorów i drukarki. W rozpatrywanym przykładzie stanowią one właściwe ograniczenie, ponieważ I7 ma wartość standardową 2048, większą od wartości wszystkich trzech parametrów, i tym samym nie wpływa na wartość minimum. Jako maksymalną wielkość wydruku przez perforator I przyjęto 200 bloków X16 rządków = 3200 rządków. Perforator II nie jest używany w programie, toteż odpowiadający mu parametr ma wartość 0. Trzeci parametr określa maksymalną wielkość wydruku drukarki wierszowej na 1200 linii.

Liczba – O kończy część parametrową nagłówka, w której, jak widać, nie umieszczono żądań POST MORTEM.

Dyrektywa X8 wydziela w pamięci bębnowej obszar 8X512 = 4096 słów przeznaczonych na program wynikowy. Zatem, zgodnie z opisem dyrektywy X (pkt. 5.3.10), program wynikowy zostanie umieszczony w pamięci bębnowej począwszy od adresu 28672.

## 6.2. PODPROGRAMY STANDARDOWE

Omawiając rozkazy maszyny ZAM 41 omówiliśmy również w pkt. 4.1.3 grupę rozkazów definiowanych przez specjalne podprogramy. Podprogramy te przechowywane są w zablokowanym obszarze pamięci bębnowej (tj. obszarze, na którym niedozwolona jest operacja zapisu) wraz z systemem operacyjnym i translatorami. Dołączenie tych podprogramów do programu właściwego wymaga jedynie użycia w nim odpowiedniej dyrektywy F (pkt. 5.3.2).

Prócz opisanych, do każdego programu można dołączyć również inne podprogramy. Takie podprogramy, zbudowane wg dalej podanych zasad umieszcza się w programie między dyrektywą X a programem właściwym. W ten sposób można definiować nowe rozkazy z grupy PO4 – P22, SLR lub SKS. Podprogramy bębnowe warto stosować w tych przypadkach, gdy pewne złożone czynności powtarzać się będą w kilku rozmaitych programach, pisanych przez różne osoby w rozmaitych odstępach czasu.

### 6.2.1. Struktura podprogramów standardowych

Każdy podprogram standardowy rozpoczyna się od etykiety bębnowej. Bezpośrednio za nią umieszczona jest liczba, co najwyżej 4-cyfrowa, określająca ilość słów podprogramu.

Następne wiersze zawierają kolejno dyrektywy V, Q, H.
Dyrektywy te w każdym podprogramie mają identyczną formę:

```
VB, 0
Q 0
H
```

Dwie pierwsze dyrektywy pozostawiają nie zmieniony licznik programowy „b", zaś licznik „q" ustawiają na 0. Zatem podprogramy pisze się tak, jak gdyby miały być one umieszczone w pamięci operacyjnej począwszy od komórki o adresie 0.

Dyrektywa H, występująca w następnym wierszu, powoduje, że wszystkie rozkazy zawierające adresy względne lub symboliczne zaopatrzone zostaną w 1 na bicie nr 0. Jedynka ta, w trakcie dołączania podprogramu do programu właściwego, wskazuje rozkazy, których część adresowa musi być zwiększona o bieżącą zawartość licznika „q". Dalsza część podprogramu zawiera rozkazy podprogramu. Część rozkazową kończy słowo złożone z samych jedynek = 77777777. Dalej następują stałe liczbowe podprogramu, zmienne podprogramu, a na końcu dyrektywy G, T.
Dyrektywa G anuluje dyrektywę H, zaś dyrektywa T kończy sekcję.

### 6.2.2. Zasady konstrukcji podprogramów standardowych

Niniejszy podrozdział zawiera systematyczny opis metody konstrukcji podprogramów standardowych.
Na końcu podrozdziału umieszczono przykład.

6.2.2.1. Określenie zbioru czynności podprogramu

Na początku należy ustalić zbiór czynności wykonywanych przez projektowany podprogram. W szczególności należy ustalić, które z rejestrów A, M, B zmienią swoją zawartość, a jakie pozostaną nie zmienione po wykonaniu podprogramu. Podprogramy należy w miarę możliwości konstruować tak, aby zawartość rejestrów A, M, B nie ulegała zmianie, chyba, że w którymś z tych rejestrów umieszczane są wyniki podprogramu.

6.2.2.2. Określenie sposobu przenoszenia danych i wyników podprogramu oraz wybór rozkazu wywołującego podprogram

Dane i wyniki podprogramu przenoszone być mogą poprzez rejestry A, M, B, poprzez komórki pamięci znajdujące się za rozkazem skoku do podprogramu, poprzez komórki pamięci o ustalonych adresach, a także mogą być umieszczane w części adresowej rozkazu (dotyczy to wyłącznie rozkazów z grupy P04–P22).

Jako rozkazy skoku do podprogramu można stosować rozkazy SLR o adresach zawartych między (144+BD) i (250+BD), SKS oraz rozkazy z grupy P04–P22.

Zarówno wybór rozkazu, jak i sposobu przenoszenia danych i wyników dokonywany jest na ogół w oparciu o indywidualne przyzwyczajenia programisty.
Nie istnieją bowiem reguły pozwalające znaleźć najlepszy, spośród możliwych, rozkaz oraz najlepszy sposób przenoszenia danych i wyników podprogramu. Każdy zaproponowany sposób może być zwykle zastąpiony innymi, przy czym trudno jest określić, który z tych sposobów jest najlepszy. Toteż ograniczymy się tutaj tylko do podania kilku zaleceń, które w pewnym stopniu mogą ułatwić dokonanie wyboru.

a) Podprogramy wykonujące funkcje analogiczne do rozkazów podstawowych (opisanych w pkt. 4.1.1) powinny być wywoływane przez rozkazy mające formę analogiczną do tych rozkazów podstawowych. Będą to więc rozkazy z grupy P04–P22. I tak, dla przykładu można wprowadzić odpowiednik rozkazu DOP (dodaj 1 do komórki pamięci) dla liczb zmiennoprzecinkowych: P15. Rozkaz ten, z punktu widzenia programu właściwego będzie dodawał liczbę 1 do liczby zmiennoprzecinkowej umieszczanej w dwóch komórkach pamięci, o adresach wskazanych przez:
adres efektywny rozkazu i adres i adres efektywny rozkazu zwiększony o 1. (Rozpatrując bliżej działanie tego rozkazu łatwo stwierdzić, że rozkaz P15 spowoduje jedynie przejście do wykonania programu, który wykona opisane wyżej czynności).
Konstruując ten przykładowy podprogram należy też zwrócić uwagę na konieczność pozostawienia nie zmienionych zawartości rejestrów A, M, B po wykonaniu podprogramu.
Używanie rozkazów grupy P04 – P22 do innych rodzajów podprogramów jest dopuszczalne, choć nie zalecane, ze względu na stosunkowo małą ilość rozkazów tej grupy.

b) Podprogramy wykonujące typowe czynności czytania danych, drukowania wyników oraz dokonujące typowych obliczeń (na przykład obliczenia wartości funkcji $e^x$) powinny być wywoływane przez rozkazy grupy SLR 144. – SLR 250.

Dane i wyniki tych podprogramów mogą być przenoszone przez rejestry A, M, B, przez komórki pamięci znajdujące się za rozkazem SLR, a także przez komórki pamięci o ustalonych adresach.

Ten sposób wywoływania podprogramów warto stosować także w przypadku, gdy rozpatrywany podprogram stosowany jest w kilku programach operujących na tym samym zbiorze danych – wówczas stosowanie rozkazu SLR zamiast SKS zwiększa przejrzystość programów.

c) Inne podprogramy powinny być wywoływane rozkazem SKS. Dane i wyniki podprogramów wywoływanych tym rozkazem mogą być przenoszone, podobnie jak przy rozkazach SLR, poprzez rejestry A, M, B, poprzez komórki pamięci znajdujące się za rozkazem SKS, a także poprzez komórki o ustalonych adresach.

6.2.2.3. Określenie dopuszczalnego zakresu stosowania modyfikacji adresowych w rozkazach wywołujących podprogramy oraz innych adresach, będących danymi podprogramu

Należy określić, które z podanych adresów mogą być modyfikowane oraz ustalić typ B-modyfikacji (podwójna czy pojedyncza) stosowanej przy obliczaniu adresów efektywnych.

6.2.2.4. Określenie numeru etykiety bębnowej podprogramu

W celu określenia numeru etykiety bębnowej, która opatrywać będzie podprogram, należy przejrzeć etykiety bibliotecznych podprogramów standardowych oraz podprogramów zapisanych na stałe w pamięci bębnowej, a następnie wybrać numer etykiety (zawarty między 1 i 2047), który nie został użyty w żadnym z tych podprogramów.

6.2.2.5. Redakcja podprogramu

Po dokonaniu omówionych w poprzednich punktach ustaleń, można przystąpić do analizy metod realizacji postawionego zadania. Jej wynikiem będzie wykres operacyjny przedstawiający procesy zachodzące w podprogramie (symbolikę wykresów operacyjnych podano w dodatku 1).
Z kolei, na podstawie otrzymanego wykresu opracowuje się część rozkazową podprogramu i jednocześnie, związaną z nią, część zawierającą stałe i zmienne liczbowe.

Ze względu na bardzo szeroki zakres funkcji wykonywanych przez rozmaite podprogramy, trudno jest podać zasady pisania części rozkazowej podprogramu.
Tym niemniej w praktyce stosuje się kilka wygodnych metod realizacji niektórych funkcji występujących zwykle w podprogramach. Najczęściej stosowane metody opisano w pkt. 6.2.2.7 i 6.2.2.8.

6.2.2.6. Obliczanie ilości słów podprogramu

Jak już wspomnieliśmy, w następnym wierszu za etykietą bębnową podprogramu umieszcza się liczbę określającą ilość słów zajmowanych przez podprogram. Liczbę tę można wyznaczyć dwoma sposobami.

Pierwszy sposób, elementarny, polega po prostu na obliczaniu ilości rozkazów, oraz stałych i zmiennych liczbowych podprogramu. Należy pamiętać, że dyrektywa A pozostawia tyle wolnych komórek pamięci, ile wynosi liczba umieszczana za nią.

Drugi, nieco wygodniejszy sposób wykorzystuje do tego celu maszynę.
Mianowicie należy przyjąć długość podprogramu równą 1, a następnie przetłumaczyć podprogram (poprzedzony nagłówkiem i zakończony etykietą bębnową i dyrektywą K). Zmniejszona o 1 różnica wartości obu etykiet określa długość podprogramu.

6.2.2.7. Wykorzystanie P-modyfikacji w podprogramach standardowych

W celu objaśnienia korzyści wynikających ze stosowania P-modyfikacji w podprogramach rozpatrzmy, dla przykładu, następujące zadanie: ułożyć sekwencję rozkazów stanowiącą fragment podprogramu, która wprowadzi do akumulatora A zawartość komórki pamięci wskazanej przez adres efektywny ostatnio wykonanego rozkazu z grupy P04–P15.

Okazuje się, że poszukiwana sekwencja składa się z jednego rozkazu:

UMA 2. ☤

Oznacza to, że adres efektywny rozkazu UMA 2. ☤ równy jest adresowi efektywnemu ostatnio wykonanego rozkazu z grupy P04–P15. Poniższe wyjaśnienia dowiodą słuszności stwierdzenia.

Spróbujmy obliczyć adres efektywny rozkazu UMA 2. ☤
Zawartość części adresowej tego rozkazu wskazuje komórkę pamięci, w której został zapamiętany, w wyniku wykonania rozkazu z grupy P04–P15, uzupełniony licznik rozkazów ULR. Ponieważ rozkaz UMA 2. ☤ ma adres P-modyfikowany, więc liczba będąca adresem tego rozkazu (2+BD), wskazuje komórkę pamięci, na której 15 ostatnich bitach umieszczony jest adres pośredni rozkazu.

Ponieważ jednak w uzupełnionym liczniku rozkazów ULR na bicie nr 1, tj. bicie P-modyfikacji znajduje się zawsze 1, zatem liczba umieszczona na 15 ostatnich bitach słowa o adresie (2+BD) wskazuje adres komórki pamięci zawierającej nowy adres pośredni rozkazu (lub efektywny, jeżeli nie nastąpią dalsze modyfikacje). W rozpatrywanym przypadku komórka PAO [BD+2] zawiera uzupełniony licznik rozkazów, zatem 15 ostatnich bitów tej komórki zawiera stan licznika rozkazów przed wykonaniem rozkazu z grupy P04–P15, czyli adres komórki pamięci, w której znajduje się rozpatrywany rozkaz z grupy P04–P15. Oznacza to więc, że następny adres pośredni umieszczony jest na 15 ostatnich bitach słowa, zawierającego rozkaz z grupy P04–P15. Jeżeli umieszczony tam rozkaz z grupy P04–P15 nie posiada żadnych modyfikacji adresowych,

to ostatnio obliczony adres pośredni staje się adresem efektywnym, zaś w przeciwnym razie maszyna oblicza dalej adres efektywny w myśl ogólnie obowiązujących zasad.

Umieszczając w podprogramie rozkaz rozpatrywanego typu UMA 2. ↑ należy mieć na uwadze, że obliczanie adresu efektywnego tego rozkazu może wymagać również wykonania B-modyfikacji, jeżeli przy rozkazie z grupy P04–P15 tego rodzaju modyfikacja została umieszczona. Dlatego też przed wykonaniem rozkazu tego typu należy przywrócić, jeżeli była zmieniana, zawartość rejestru B w chwili wykonania rozkazu z grupy P04–P15.

Przedstawiona metoda może być stosowana również do wielu innych rozkazów. W szczególności, jak łatwo sprawdzić, rozkaz UAD 2. ↑ powoduje umieszczenie w AM zawartości komórek wskazanych przez adres efektywny rozkazu z grupy P04–P15, przy czym, jeżeli w rozkazie występuje B–modyfikacja, to zgodnie z określeniem rozkazu UAD, do adresu zostaje dodana podwojona zawartość B-rejestru.

Opisana metoda może być również, oczywiście, stosowana do rozkazów z grupy P16–P22. W tym wypadku należy jako adres rozkazu przyjąć 3. (3+BD), ponieważ rozkazy z tej grupy zapamiętują uzupełniony licznik rozkazów w komórce PAO [3+BD].

Pewna modyfikacja tej metody pozwala na stosowanie jej do rozkazów grupy SLR lub SKS. Jeżeli pewien adres, dla przykładu, został umieszczony w wierszu następnym za rozkazem SLR, to aby umieścić w mnożniku M zawartość komórki pamięci wskazanej przez ten adres należy wykonać poniższe dwa rozkazy:

```
DOP 1.
UMN 1. ↑
```

Zwiększenie o 1 uzupełnionego licznika rozkazów, który został zapamiętany w komórce (BD+1) w wyniku wykonania rozkazu SLR ma na celu przygotowanie właściwego adresu pośredniego dla rozkazu UMN. Wykonanie rozkazu UMN przebiega w sposób identyczny, jak poprzednio opisano.

Czasami zachodzi potrzeba obliczania adresu efektywnego rozkazu z grupy P04–P22 lub adresów towarzyszących rozkazowi SLR lub SKS.

W tym celu stosuje się metodę zbliżoną do opisanej, wykorzystującą rozkaz UBA, który umieszcza w B adres efektywny. Tak więc wykonanie rozkazu UBA 2. ↑ spowoduje umieszczenie w B adresu efektywnego ostatnio wykonanego rozkazu z grupy P04–P15. Obliczanie adresu przebiega dokładnie tak samo jak poprzednio opisano.

### 6.2.2.8. Uwagi dotyczące stosowania w podprogramach niektórych grup rozkazów

W podprogramach standardowych można stosować prócz rozkazów podstawowych, rozkazy definiowane przez system operacyjny, a także rozkazy definiowane przez podprogramy standardowe.

Rozkazy definiowane przez system operacyjny mogą być stosowane w podprogramach na ogólnie przyjętych zasadach. Należy jednak pamiętać, że rozkazy te (SLR) zapamiętują uzupełniony licznik rozkazów w komórce PAO [BD+1], tej samej, w której zapamiętują uzupełniony licznik rozkazów ULR rozkazy typu SLR wywołujące podprogramy standardowe.

Tak więc, jeżeli podprogram wywoływany jest rozkazem z grupy SLR, to ewentualny inny rozkaz SLR występujący w treści podprogramu musi zostać poprzedzony przechowaniem w komórce PAO [n] zawartości komórki PAO [BD+1]. Powrót do programu głównego nastąpi na podstawie zawartości komórki PAO [n].

Podprogramy standardowe realizujące czynności czytania i drukowania muszą być przystosowane do pracy z rozmaitymi urządzeniami zewnętrznymi. W tym celu w podprogramach standardowych zamiast rozkazów SLR 2, SLR 4, SLR 6 i SLR 18 stosuje się rozkaz SLR 255. ☂ . Komórka PAO [BD+255] zawiera początkowo liczbę +2 i jej zawartość może być zmieniana przez program główny na +4, +6, +18 lub ponownie na +2. Zatem, adresami efektywnymi rozkazu SLR 255. ☂ mogą być liczby 2, 4, 6, 18. Oczywiście, zastąpienie rozkazu SLR 4 przez SLR 255. ☂ może nastąpić tylko wówczas, gdy nie przewiduje się wydruku na drukarce wierszy dłuższych niż 69-znakowych.

Nieco bardziej skomplikowane są rozkazy czytania, bowiem rozkazy SLR 1 i SLR 5 prócz umieszczania przeczytanego znaku w B-rejestrze, umieszczają go jeszcze w komórce PAO[27+BD] (SLR 1) lub PAO [61+BD] (SLR 5), a także, gdy przeczytany znak jest znakiem liter LS, lub cyfr FS, w komórce PAO [26+BD] lub PAO [60+BD].

W związku z tym używane są dwie komórki sterujące: w komórce PAO [254+BD] umieszczona zostaje liczba +1 lub +5, zaś w komórce PAO [252+BD] adres pierwszej z dwu opisanych komórek (tj. 26+BD lub 60+BD). Z przedstawionego opisu wynika, że zamiast rozkazów SLR 1 i SLR 5 w podprogramach standardowych używać należy rozkazów SLR 254. ☂ , zaś zawartości komórek PAO [26+BD] i PAO [27+BD] lub PAO [60+BD] i PAO [61+BD] przesyłać należy rozkazem UAM 253. ☂ .

W ten sposób program, zmieniając zawartości komórek PAO [253+BD] PAO [254+BD], PAO [255+BD], może kierować operacje czytania i drukowania do rozmaitych urządzeń zewnętrznych.

Na zakończenie podamy kilka uwag dotyc ych wykorzystania w podprogramie standardowych rozkazów definiowanych przez inn odprogramy standardowe.

Stosując takie rozkazy należy pamiętać o możliwości zniszczenia zawartości komórek PAO [1+BD], PAO [2+BD], PAO [3+BD] przez rozkazy wywołujące podprogramy, należące do tej samej grupy, podobnie jak przy omówionej grupie SLR. Ponadto, używając rozkazów tego typu należy pamiętać o ich zdefiniowaniu. W tym celu w programie właściwym prócz dyrektywy F dołączającej rozpatrywany podprogram, należy umieścić także dyrektywy F dołączające podprogramy, które definiują rozkazy użyte w rozpatrywanym podprogramie.

### 6.2.2.9. Podprogramy standardowe definiujące kilka rozkazów

Dotychczas omawialiśmy podprogramy definiujące jeden rozkaz. Jednakże w praktyce występują, na ogół, podprogramy definiujące kilka rozkazów, zwłaszcza, gdy rozkazy wykonują zbliżone funkcje.

Podprogram taki składa się z kilku części, definiujących poszczególne rozkazy, przy czym na początku podprogramu znajdują się rozkazy skoku do tych części. Skoki te,

w połączeniu ze skokami wpisywanymi do komórek pamięci przez dyrektywę F spełniają rolę połączenia między rozkazem i definiującą go częścią podprogramu. Wyjaśnimy to na przykładzie.

Przedstawiony podprogram definiuje rozkazy SLR 197., SLR 198., SLR 199., które powodują obliczenie modułu liczby umieszczonej w akumulatorze A (SLR 197.) mnożniku M (SLR 198.), B-rejestrze (SLR 199.), a następnie umieszczają wynik w tym samym rejestrze. Dyrektywa F dołączająca ten podprogram do programu właściwego ma postać:

F 401, 197 – 0, 198 – 1, 199 – 2

W wyniku wykonania dyrektywy poza dołączeniem podprogramu do programu właściwego do komórek PAO [197+BD], PAO [198+BD], PAO [199+BD] zostaną wpisane rozkazy skoku;

do komórki PAO [197+BD] skok na pierwszy rozkaz podprogramu,
do komórki PAO [198+BD] skok na drugi rozkaz podprogramu,
do komórki PAO [199+BD] skok na trzeci rozkaz podprogramu.

Podprogram B 401 ma postać:

```
B401
+21
VB, 0
Q0
H
Y3
SKO E1: MODUL A
SKO E2: MODUL M
SKO E3: MODUL B
E1
MNL E4: MNOZENIE LOGICZNE
WRO 1.
E2
LCD 24: ZAMIANA AKUMULATORA Z MNOZNIKIEM
MNL E4: MNOZENIE LOGICZNE
LCD 24: PONOWNA ZAMIANA
WRO 1.
E3
PAK E5: PRZECHOWANIE
PAB E6
UMA E6
MNL E4: MNOZENIE LOGICZNE
PAK E6
UMB E6
UMA E5: ODTWORZENIE A
WRO 1.
```

```
=77777777
E4
=37777777
E5
A1
E6
A1
E7
G
T
```

Rozpatrzmy teraz czynności maszyny po wykonaniu rozkazu SLR 198. Zgodnie z przyjętym założeniem rozkaz ten ma obliczyć moduł liczby umieszczonej w M i wynik umieścić w M.

Po wykonaniu rozkazu SLR 198. w komórce PAO [1+BD] zostanie zapamiętany uzupełniony licznik rozkazów ULR, zaś jako następny zostanie wykonany rozkaz umieszczony w komórce PAO [198+BD].

W komórce tej, w wyniku działania dyrektywy F umieszczony został rozkaz skoku na drugi z kolei rozkaz podprogramu. Zatem w dalszej kolejności wykonany zostanie drugi rozkaz podprogramu a więc SKO E 2 (adres tego rozkazu został zwiększony o wartość licznika q – nie ma to, jak nietrudno się przekonać, znaczenia dla analizy działania podprogramu). Z kolei rozpoczyna się wykonywanie sekwencji rozkazów począwszy od rozkazu opatrzonego etykietą E 2. Rozkaz LCD 24 (do adresu tego rozkazu nie została dodana zawartość licznika q) powoduje przesunięcie cykliczne A i M o 24 pozycje w lewo – w rezultacie zawartość A przesunie się do M, zaś zawartość M do A. Następny rozkaz MNL E 4 powoduje pomnożenie logiczne A przez słowo zawierające jedynki na wszystkich bitach z wyjątkiem bitu znaku.

Następny rozkaz zamienia ponownie A i M. Rozakz WRO 1. powoduje powrót do programu właściwego na podstawie zapamiętanego licznika rozkazów w komórce PAO[1+BD].

### 6.2.2.10. Dokumentacja podprogramów standardowych

Każdy podprogram standardowy winien posiadać dokumentację zawierającą następujące elementy:

a) numer etykiety bębnowej,
b) ilość słów podprogramu,
c) listę i określenia rozkazów definiowanych przez podprogram, oraz, gdy podprogram realizuje czynności czytania i drukowania, opis możliwości realizacji tych czynności na rozmaitych urządzeniach zewnętrznych,
d) postać dyrektywy F dołączającej podprogram do programu właściwego,
e) listę rozkazów używanych w podprogramie, definiowanych przez inne podprogramy standardowe, oraz dyrektywy F powodujące dołączenie tych podprogramów do programu właściwego,

f) wykres operacyjny,
g) tabulogram podprogramu wraz ze słownikiem etykiet,
h) opis działania podprogramu,
i) uwagi dodatkowe dotyczące czasu wykonywania podprogramu, dokładności obliczeń, możliwości zmian sposobu wywoływania podprogramu itp.,
j) taśmę perforowaną podprogramu.

6.2.2.11. Przykład konstrukcji podprogramu

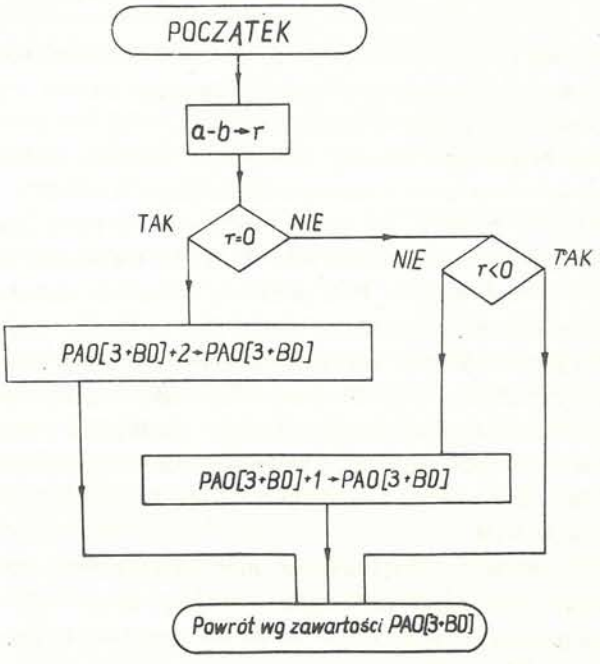
W niniejszym podrozdziale ułożymy podprogram porównywania liczb zmiennoprzecinkowych.
Ze względu na wygodę i przyzwyczajenia programistów podprogram zostanie skonstruowany tak aby rozkaz, który będzie wywoływał podprogram działał, z punktu widzenia programu właściwego, analogicznie do rozkazu POB (porównaj liczbę w PAO z zawartością B). Wynika stąd natychmiast wniosek, że rozkaz wywołujący podprogram powinien (jakkolwiek nie musi) być wybrany spośród rozkazów grupy PO4–P22.

Rozkazom P16–P21 zostały już przyporządkowane definicje (pkt. 4.1.3) toteż wyboru możemy dokonać wśród rozkazów PO4–P15, P22. Wybieramy rozkaz P22.
Opierając na opisie funkcji rozkazu POB podajemy określenie rozkazu P22: Porównaj liczby zmiennoprzecinkowe umieszczone w AM oraz w dwóch komórkach pamięci wskazanych przez: adres efektywny rozkazu oraz adres efektywny rozkazu zwiększony o 1. Jeżeli liczba umieszczona w AM jest mniejsza od liczby umieszczonej w komórkach pamięci, to przejdź do następnego rozkazu (dodaj 1 do licznika rozkazów LR), jeżeli liczba umieszczona w AM jest większa, to przeskocz jeden rozkaz (dodaj 2 do LR), zaś gdy liczby są równe, to przeskocz dwa rozkazy (dodaj 3 do LR). Rozkaz nie zmienia zawartości rejestrów A, M, B.
Przyjmiemy dalej, że rozkaz P22 może zawierać, modyfikacje adresowe typu P i B.
Etykiecie bębnowej, opatrującej podprogram, nadamy numer 402, dotychczas nie występujący w podprogramach standardowych. Zatem dyrektywa F, dołączająca podprogram do programu właściwego będzie miała postać:

F 402,22 – 0

Podamy tutaj dwa przykłady realizacji tego podprogramu. Pierwsza realizacja, znacznie prostsza, wykorzystywać będzie rozkaz odejmowania zmiennoprzecinkowego P17. Rozkaz ten, opisany w pkt. 4.1.3, definiowany jest przez inny podprogram. Tak więc stosując tę realizację podprogramu musimy dołączyć do programu właściwego także podprogram definiujący rozkaz P17. Na ogół, gdy program główny używa rozkazu porównania liczb zmiennoprzecinkowych, to używa także rozkazów wykonujących działania arytmetyczne w zmiennym przecinku, a więc między innymi P17.
Toteż dołączenie podprogramu definiującego rozkaz P17, nie wynika tylko z wymagań podprogramu porównywania liczb zmiennoprzecinkowych.

Istnieją jednak pewne programy np. porządkowania liczb, które operując na liczbach zmiennoprzecinkowych, nie wykonują żadnych działań arytmetycznych na nich.
W takich programach lepiej stosować, drugą realizację podprogramu, wprawdzie bardziej skomplikowaną, ale za to używającą wyłącznie rozkazów podstawowych.
Pierwsza realizacja programu określa, która z liczb jest większa, na podstawie znaku różnicy tych liczb.
Jeżeli różnicą dwóch liczb a–b jest dodatnia (a–b > 0), to a > b, zaś gdy różnica dwóch liczb jest ujemna (a–b < 0), to a < b. Przypadek, gdy liczby są równe badany jest wcześniej.




Rys. 6.1. Wykres operacyjny porównywania liczb zmiennoprzecinkowych (I wariant)

Na rys. 6.1 przedstawiono wykres operacyjny obrazujący czynności wykonywane w podprogramie.

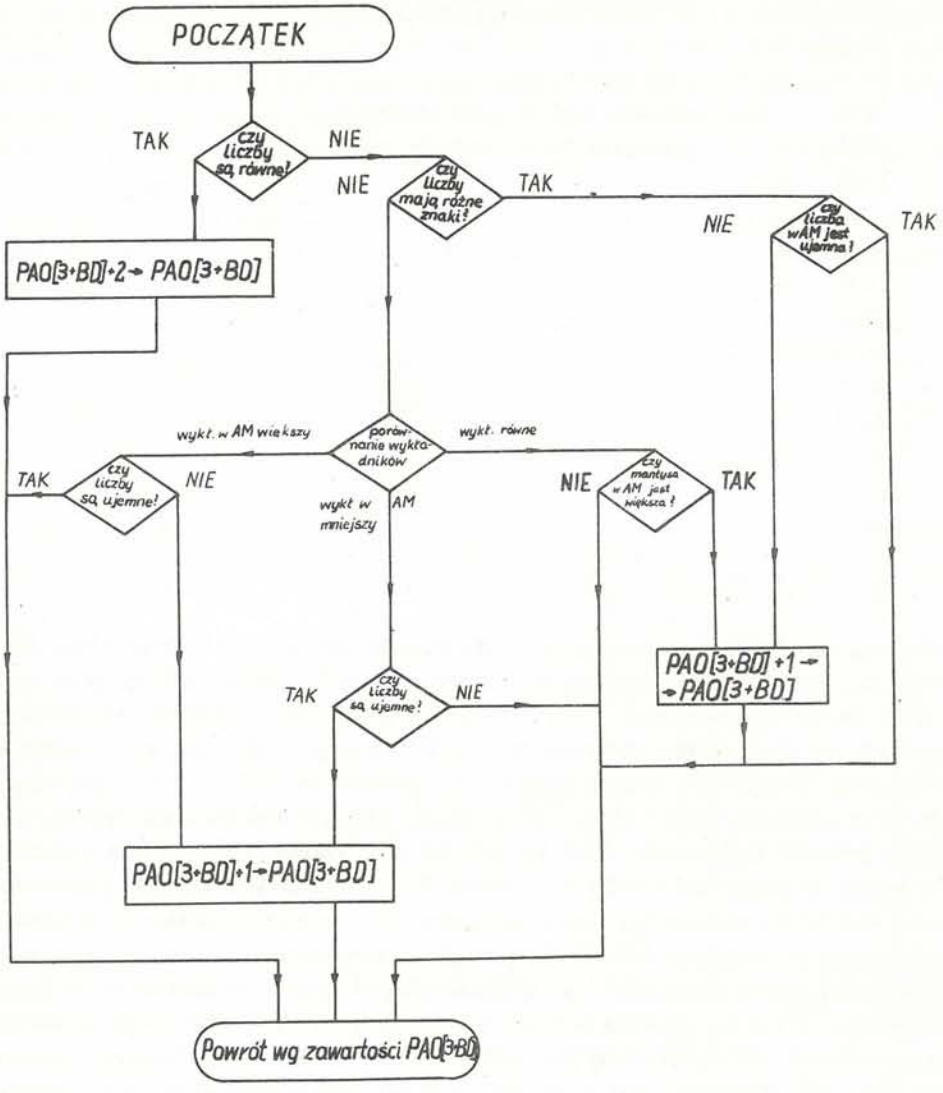
```
B402
+16
VB,0
Q0
H
Y5
PAM   E1      :PRZECHOWANIE AM
UMA   3.
```

```
PAK    E2      :PRZECHOWANIE SLADU ROZKAZU P22
UMA    1E1
P17    E2 ☂    :ODEJMOWANIE
SZA    +5      :SKOCZ, GDY WYNIK ROWNY 0
SMA    +2      :SKOCZ, GDY WYNIK UJEMNY
DOP    E2
UAM    E1
WRO    E2
DOP    E2
SKO    -4
=77777777
E1
A2
E2
A1
E3
G
T
```

Pierwszą czynnością wykonywaną przez podprogram, jest jak widać, zapamiętanie AM. Następnie, ze względu na przewidywane użycie rozkazu P17, który zapamiętuje uzupełniony licznik rozkazów ULR także w komórce PAO [3+BD], zawartość tej komórki przesyła się do komórki opatrzonej etykietą E2. Następnie odtwarza się zawartość A i wykonuje odejmowanie zmiennoprzecinkowe rozkazem P17 E 2 ☂. Łatwo sprawdzić, że adres efektywny tego rozkazu równy będzie adresowi efektywnemu rozkazu P22, który wywołał podprogram. Jeżeli po wykonaniu odejmowania zawartość A wynosi 0, to znaczy, że odejmowane liczby były równe. W tym wypadku do komórki przechowującej ślad (ULR) rozkazu P22 dodaje się liczbę +2, a następnie powraca do programu właściwego na podstawie zawartości tej komórki, przeskakując dwa rozkazy programu właściwego. Jeżeli akumulator A po odejmowaniu jest ujemny, to oznacza to, że liczba umieszczona w AM jest mniejsza od liczby umieszczonej w PAO – wówczas, po odtworzeniu zawartości AM, wykonywany jest rozkaz WRO, w wyniku którego maszyna powraca do programu właściwego, wykonując jako pierwszy, rozkaz znajdujący się w komórce pamięci za rozkazem P22. W wypadku, gdy A jest dodatni, do komórki opatrzonej etykietą E2 dodawana jest 1, w wyniku czego powracając do programu właściwego przeskakuje się jeden rozkaz.

Przejdziemy obecnie do przedstawienia drugiej wersji podprogramu. Wykorzystuje ona wyłącznie rozkazy podstawowe, co oczywiście spowodowało znaczne zwiększenie długości podprogramu.

Czynności wykonywane przez podprogram ilustruje wykres operacyjny przedstawiony na rys. 6.2.




Rys. 6.2. Wykres operacyjny porównywania liczb zmiennoprzecinkowych (II wariant)

```
B402
+48
VB,0
Q0
H
Y4
PAM     E1
```

```
UAD   3. ↑
OAM   E1
LAR   46
SNA   +5:  NADMIAR, GDY ODEJMOWANIE DALO WYNIK ROZNY
           :OD ZERA
UAM   E1:  LICZBY ROWNE
DOP   3.
DOP   3.
WRO   3.
UAD   3. ↑
OSL   1E1
SMA   E3:  MANTYSY MAJA ROZNE ZNAKI
PAB   E4
UAD   3. ↑
WMB:  WYKLADNIK LICZBY Z PAO
PAB   2E2
PAM   E2
UAM   E1
WMB:  WYKLADNIK LICZBY Z AM
PAB   2E1
PAM   E1
POB   2E2: POROWNANIE WYKLADNIKOW
SKO   E5:  WYKLADNIK LICZBY Z AM JEST MNIEJSZY
SKO   E6:  WYKLADNIK LICZBY Z AM JEST WIEKSZY
OAM   E2:  WYKLADNIKI ROWNE
SMA   +2:  MANTYSA LICZBY Z AM JEST MNIEJSZA
E7
DOP   3.
E8
UAM   E1
UMB   2E1
WBM
UMB   E4
WRO   3.
E5
SMA   E7:  SKOCZ, GDY MANTYSY UJEMNE
SKO   E8
E6
SMA   E8:  SKOCZ, GDY MANTYSY UJEMNE
SKO   E7
E3
UAM   E1
```

```
SMA   +2
DOP   3.
WRO   3.
=77777777
E1
A3
E2
A3
E4
A1
E9
G
T
```

Pierwszą czynnością wykonywaną w podprogramie jest zapamiętanie AM. Następnie, jak to ilustruje wykres operacyjny, bada się, czy obie liczby są identyczne.
Dokonuje się tego poprzez odejmowanie, po czym przesuwa się zawartość AM o 46 pozycji arytmetycznie w lewo. Jeżeli wynik odejmowania różnił się od zera tj. gdy odejmowano różne liczby, w wyniku przesunięcia zostanie wpisana 1 do wskaźnika nadmiaru WN. W przeciwnym razie do zawartości komórki PAO [BD+3], zostaje dodana 2, po czym powraca się do programu, przeskakując dwa rozkazy.

W przypadku, gdy liczby są różne następuje obliczenie sumy modulo dwa (rozkazem OSL) bitów znaków obu liczb. Pojawienie się, w wyniku wykonania rozkazu OSL, jedynki na pozycji nr 0 A oznacza, że znaki liczb są różne.
W tym wypadku dalsze postępowanie identyfikuje liczbę ujemną i ewentualnie dodaje 1 do komórki PAO [BD+3], po czym następuje powrót do programu.

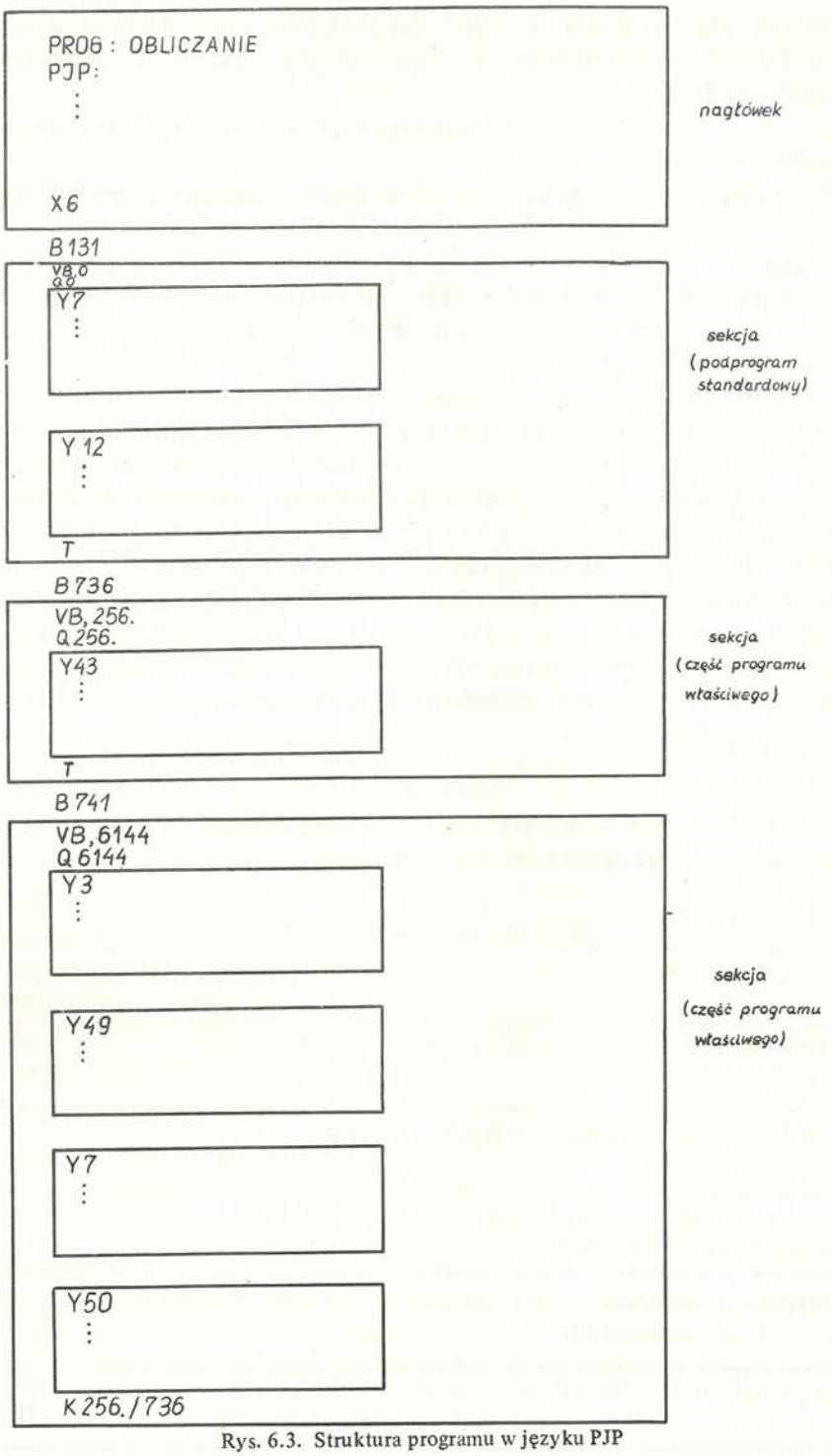
Jeżeli liczby mają identyczne znaki, to dalsze rozkazy przygotowują operację porównania wykładników.
Wynik porównania oraz znajomość znaku liczby pozwala łatwo określić, która z liczb jest większa.

W przypadku, gdy liczby mają identyczne znaki i identyczne wykładniki, należy określić, która z liczb ma większą mantysę. W tym celu odejmuje się od siebie obie mantysy i na podstawie znaku wyniku określa, która z nich jest większa. Następnie, po ewentualnym dodaniu 1 do komórki PAO [3+BD], następuje powrót do programu wg zawartości tej komórki.

## 6.3. PROGRAM WŁAŚCIWY

Program właściwy stanowi ostatnią i najważniejszą część programu w języku PJP.

Program właściwy umieszczony jest zawsze za podprogramami standardowymi, lub gdy one nie występują, za nagłówkiem programu. Koniec programu właściwego, a zarazem całego programu, wskazuje dyrektywa K.




Rys. 6.3. Struktura programu w języku PJP

Program właściwy składa się z sekcji (rys. 6.3), które z kolei dzielą się na paragrafy. *Sekcja* jest to fragment programu, zaczynający się od dyrektywy X lub T, zaś kończący się dyrektywą T lub K.

*Paragraf* jest to fragment programu zaczynający się od dyrektywy Y, zaś kończący się dyrektywą Y, T lub K.

W zależności od ilości sekcji w programie właściwym wyróżniamy programy właściwe jedno- i wielosekcyjne.

W praktyce spotykamy się najczęściej z programami właściwymi jednosekcyjnymi. Programy wielosekcyjne występują rzadko, ponieważ podział programu na sekcje stosuje się tylko wówczas, gdy ilość etykiet wewnętrznych w programie przekracza 512 lub gdy łączy się dwa różne programy.

Zasadniczą przyczyną, dla której unika się podziału programu właściwego na sekcje jest trudność w komunikacji między poszczególnymi sekcjami. Nie można się bowiem odwoływać w sekcji do etykiet wewnętrznych (pkt. 5.4.1) należących do innej sekcji. W związku z tym komunikacja między poszczególnymi sekcjami może odbywać się jedynie przy pomocy rozkazów zawierających adresy rzeczywiste lub symboliczne – etykiety bębnowe (które zachowują znaczenie w całym programie).

W momencie rozpoczęcia wykonywania programu właściwego w pamięci operacyjnej umieszczana jest zwykle tylko jedna sekcja. Toteż aby zainicjować, po wykonaniu rozkazów sekcji, działanie następnej, każda z sekcji, z wyjątkiem ostatniej, musi zawierać w sobie sekwencję rozkazów powodujących przepisanie następnej sekcji z pamięci bębnowej do pamięci operacyjnej.

### 6.3.1. Podział pamięci operacyjnej

Przewidując rozmieszczenie poszczególnych części programu w pamięci operacyjnej należy brać pod uwagę ograniczenia podane w tablicy 6.1.

Tablica 6.1

Niektóre parametry systemu m.c. ZAM 41

| | I zestaw (z Algolem) | II zestaw (bez Algolu) |
|---|---|---|
| Obszar pamięci, w którym może być umieszczona część rozkazowa programu właściwego | 1536<br>8191 | 2688<br>8191 |
| blokada dolna (zawartość rejestru BD w trakcie wykonywania programu) | 1280 | 2432 |
| obszar pamięci, w którym można realizować transmisję do/z taśmy magnetycznej | | 2688<br>8191 |
| blokada górna (zawartość rejestru BD + 127 – 16 w trakcie wykonywania programu) | 12271 | 12271 |
| obszar pamięci wykorzystywany przez czytnik kart | | 8100<br>8191 |

W dalszym ciągu rozpatrywać będziemy podział pamięci operacyjnej dla programów przygotowanych dla II zestawu. W zestawie I ograniczenia są nieco mniejsze, jednakże zestaw ten jest rzadko stosowany, ze względu na brak możliwości komunikacji z pamięcią taśmową i monitorem.

Łatwo zauważyć, że na część rozkazową programu przewidziano obszar zawierający 5504 komórki. Jeżeli uwzględnić, że obszar ten musi być jeszcze zmniejszony o obszary służące do transmisji do/z monitora, czytnika kart i taśmy magnetycznej, to okazuje się, że część rozkazowa programu może zajmować około 4 K (4096) komórek pamięci. Jak widać, nie jest to zbyt duży obszar, toteż właściwe rozplanowanie pamięci operacyjnej, zwłaszcza w dużych programach, jest istotnym problemem.

Trzeba jeszcze dodać, że nie omawiany dotychczas obszar pamięci począwszy od komórki o adresie 8192 do 12271 służyć może do przechowywania większych zbiorów danych, czy wyników pośrednich. Komunikacja z tym obszarem może się odbywać przy pomocy adresów rzeczywistych.

W świetle powyższych uwag jako najkorzystniejszy uważa się następujący podział pamięci operacyjnej:

0–2431 – obszar poniżej blokady dolnej (BD); tu znajduje się system operacyjny

2432–2687 – obszar wykorzystywany w sposób specjalny (skoki do podprogramów, modyfikatory)

2688–8191 – obszar przewidziany na część rozkazową (zwykle począwszy od 2688) oraz komórki służące do transmicji do/z monitora, pamięci taśmowej oraz czytnika kart

8192–12271 – obszar wykorzystywany do przechowywania większych zbiorów danych i wyników pośrednich.

### 6.3.2. Struktura wewnętrzna sekcji

Struktura sekcji jest identyczna w programach jedno- i wielosekcyjnych. Sekcja rozpoczyna się od dyrektyw V i Q, które wyznaczają położenie programu w pamięci operacyjnej i bębnowej.

W programach jednosekcyjnych, uwzględniających zalecenia zawarte w pkt. 6.3.1 dyrektywy te mają postać:

```
VB,256.          VO,256.
Q256.    lub     Q256.
```

Pierwszy wariant stosowany jest w przypadku, gdy program właściwy poprzedzają podprogramy.

Taka sama forma dyrektyw powinna występować w pierwszej sekcji programu wielosekcyjnego. Dalsze sekcje takiego programu zaczynać się będą również od dyrektyw V i Q, przy czym konkretna ich postać wynikać będzie z przyjętej organizacji przepisywania i inicjowania pracy poszczególnych sekcji.

Bezpośrednio pod dyrektywami V i Q umieszczona jest etykieta bębnowa, która wskazuje początek sekcji, a także jest wykorzystywana przez dyrektywę K.

Dalsze wiersze sekcji, tworzące jeden lub kilka paragrafów (najwyżej 64), zawierać będą część rozkazową, stałe i zmienne liczbowe oraz dyrektywy. Większe grupy zmiennych liczbowych należy umieszczać, jak zalecono w pkt. 6.3.1, w obszarze pamięci 8192–12271.

Sekcja kończy się dyrektywą T, lub gdy jest ostatnią sekcją programu, dyrektywą K.

Wewnątrz sekcji można stosować adresy symboliczne zawierające etykiety wewnętrzne deklarowane we wszystkich paragrafach tej sekcji oraz etykiety bębnowe całego programu.

### 6.3.3. Struktura wewnętrzna paragrafu

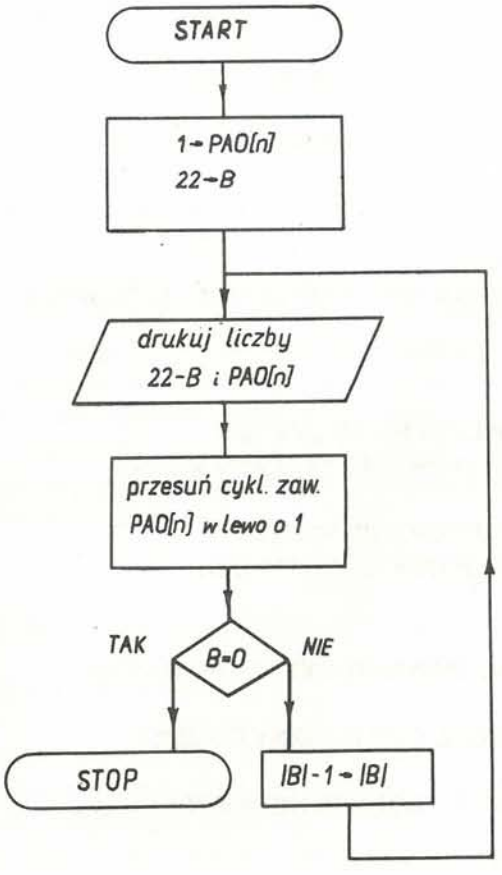
Paragrafem, w języku PJP, nazywamy fragment sekcji zaczynający się od dyrektywy Y i kończący się dyrektywą Y lub T lub K. Wewnątrz paragrafu mogą występować etykiety wewnętrzne opatrzone numerami od 1 do 63. Deklarowanie dwóch etykiet o identycznych numerach wewnątrz paragrafu jest zabronione. Etykieta o numerze O (EO) wprowadzana jest przez translator; opatruje ona pierwszy wiersz informacyjny paragrafu i może być używana w sekcji programu tak samo jak inne etykiety wewnętrzne.

Konstruując sekcję programu należy dążyć, by poszczególne paragrafy zawierały rozkazy wykonujące pewne zwarte fragmenty czynności sekcji. Zwiększa to w znacznym stopniu przejrzystość programu.

# 7. ZAGADNIENIA TECHNIKI PROGRAMOWANIA W JĘZYKU PJP

## 7.1. WPROWADZENIE

Aby wprowadzić czytelnika w konkretne problemy programowania rozpatrzmy prosty program obliczający i drukujący wartości wyrażenia $2^n$ dla n = 0,1,...,22.
W lewej kolumnie drukowane będą liczby n, zaś w prawej $2^n$. Wykres operacyjny obrazujący czynności wykonywane w programie przedstawiony jest na rys. 7.1.




Rys. 7.1. Wykres operacyjny tablicowania wyrażenia $2^n$

Jak widać, program nie jest skomplikowany. B-rejestr spełnia w nim rolę licznika; po każdym wydruku, z wyjątkiem ostatniego, jego zawartość zmniejsza się o 1. Tak więc instrukcja drukowania liczby (22–B) powodować będzie drukowanie kolejno liczb 0,1,...,22. Obliczenie liczby $2^n$ jest proste; opiera się na własności przesunięcia liczby dwójkowej w lewo: przesunięcie liczby o 1 pozycję w lewo jest równoważne pomnożeniu liczby przez 2. Korzystając z tego, łatwo wyznaczymy kolejno liczby $2^n$. Po osiągnięciu przez B zawartości równej 0 wykonywanie programu zostaje zakończone.

Program realizujący omówione czynności ma postać:

```
PROG:   OBLICZANIE POTEGI LICZBY 2
PJP:
I
STS     80 ☂
STS     60
STS     0
STS     0
–0
X1
VO,256.
Q256.
B404
Y63
UMA     E3
PAK     E4
UBA     22:     USTAWIENIE WARTOSCI POCZATKOWEJ B
E5
PAB     E1
UMA     E2
ODS     E1:     OBLICZENIE LICZBY 22-B
SLR     134.:   DRUKOWANIE WYKLADNIKA
UMA     E4
SLR     134.:   DRUKOWANIE POTEGI
SLR     137.:   DRUKOWANIE NOWEJ LINII
+1
UMA     E4
LCA     1:      PRZESUNIECIE CYKLICZNE W LEWO
PAK     E4
SOB     –11:    SKOCZ, GDY B ROZNY OD ZERA
SLR     40
F3,     134–0, 137–3:   DOLACZENIE PODPROGRAMU
E1
A1
E2
```

```
+22
E3
+1
E4
A1
K256./404
```

Zakładamy, że program będzie tłumaczony i wykonywany przy II zestawie systemu operacyjnego (bez Algolu).

Forma nagłówka programu jest typowa. Maksymalny czas wykonywania programu określono na 80 cykli, czyli około 30 s. Przyjęto, że perforator nie wydrukuje więcej niż 60 bloków po 16 rządków. Dyrektywa X1 wydziela na bębnie obszar 512 słów, począwszy od adresu 32 256 = I1–1·512, przeznaczonych na program wynikowy.

Dalsze dwie dyrektywy V i Q określają położenie programu na bębnie i w pamięci operacyjnej: program wynikowy zostanie umieszczony na bębnie począwszy od miejsca zerowego przydzielonego obszaru, a więc począwszy od adresu bębnowego 32 256. W pamięci operacyjnej program umieszczony będzie począwszy od komórki 2688 (= BD+256).

Program .właściwy składa się z jednej sekcji, która z kolei zawiera jeden paragraf, oznaczony numerem 63.

Działanie programu zostało już ogólnie omówione. Teraz podamy kilka uwag szczegółowych.

Dwa pierwsze rozkazy przenoszą liczbę +1 (= $2^0$) z komórki pamięci opatrzonej etykietą E 3 do komórki opatrzonej etykietą E 4. Rozkaz UBA 22 powoduje ustawienie wartości początkowej B, która wynosi 22. Dalsze rozkazy, za etykietą E 5, służą do obliczania liczby (22 – B): najpierw zapamiętuje się B, następnie w A umieszcza liczbę +22, i w końcu od liczby +22 umieszczonej w A odejmuje zapamiętaną zawartość B. Uzyskany w A wynik jest następnie drukowany przy pomocy rozkazu SLR 134. Następnie w A umieszcza się uprzednio przygotowaną zawartość komórki opatrzonej etykietą E 4, po czym drukuje się zawartość A rozkazem SLR 134. Rozkaz następny SLR 137. wraz z następującą po nim liczbą +1 powoduje wydruk nowej linii, przez co uzyskiwane wyniki mieć będą założoną formę tj. dwóch kolumn liczb. Trzy następne rozkazy powodują przesunięcie cykliczne o 1 w lewo liczby umieszczonej w komórce opatrzonej etykietą E 4. Rozkaz sterujący SOB powoduje, gdy zawartość B jest różna od zera, odjęcie 1 od zawartości B i skok do rozkazu odległego o 11 komórek w górę od rozkazu SOB.

Wówczas powtarzają się czynności już opisane:
obliczanie(22–B),drukowanie itd. Gdy zaś zawartość B wynosi 0, maszyna przechodzi do następnego rozkazu, którym jest SLR 40 – wykonywanie programu zostaje zakończone.

Umieszczana za omówionymi rozkazami dyrektywa F 3, 134–0, 137–3 powoduje dołączenie do programu podprogramu standardowego F 3, który przechowywany jest stale na bębnie. Ponadto do komórek PAO [134+BD] i PAO [137+BD] zostają wpisane rozkazy skoków, przez co rozkazy SLR 134. i SLR 137. zostają zdefiniowane.

Dalsze wiersze zawierają stałe i zmienne programu opatrzone etykietami.

Program kończy się dyrektywą K, która powoduje zakończenie tłumaczenia programu, przepisanie programu z pamięci bębnowej od adresu określonego przez wartość etykiety bębnowej B404 do pamięci operacyjnej, po czym rozpoczyna się wykonywanie programu począwszy od rozkazu umieszczonego w komórce PAO [256+BD].

## 7.2. PRZYGOTOWANIE PROBLEMU DO ROZWIĄZANIA NA MASZYNIE CYFROWEJ. WYKRESY OPERACYJNE

Rozwiązanie problemu przy pomocy maszyny cyfrowej musi być poprzedzone szeregiem czynności przygotowawczych, które poniżej omówimy.

Pierwszą z nich jest ścisłe sformułowanie problemu w sposób umożliwiający zastosowanie do jego rozwiązania maszyny cyfrowej. Czasami bowiem zleceniodawca, nie znając własności maszyn cyfrowych formułuje problem w sposób nieprecyzyjny, oraz stawia wymagania, których realizacja wymaga nakładów niewspółmiernie większych od osiągalnych rezultatów. Toteż omawiany etap powinien mieć charakter konsultacji między zleceniodawcą i programistą.

Z kolei, po ścisłym sformułowaniu rozwiązywanego problemu, programista przystępuje do wyboru metody rozwiązania i analizy numerycznej problemu. Wybór metody rozwiązania, spośród kilku możliwych, połączony z analizą numeryczną problemu, następuje w oparciu o wymagania dotyczące dokładności obliczeń, stopnia skomplikowania programu, czasu wykonywania programu, a czasami metoda rozwiązania jest narzucona przez zleceniodawcę. Analiza numeryczna ma na celu doprowadzenie metody rozwiązania do postaci umożliwiającej zastosowanie jej do ułożenia programu.

Następny, bardzo ważny etap stanowi przygotowanie wykresu operacyjnego, który jest graficznym przedstawieniem ciągu operacji wykonywanych podczas realizacji programu. Elementami wykresu operacyjnego są graficzne symbole operacji i decyzji podejmowanych przez program. Elementy połączone są ze sobą przy pomocy zorientowanych linii, których strzałki określają kolejność wykonywania operacji w programie. Należy podkreślić, że symbole wykresów operacyjnych mogą reprezentować zarówno czynności elementarne odpowiadające pojedynczym rozkazom, jak również czynności bardziej złożone jak np. drukowanie liczby.

Wykresy operacyjne umożliwiają programiście dobrą orientację w opracowywanym przez niego programie, zwłaszcza gdy program jest skomplikowany. Ponadto ułatwiają one późniejsze modyfikacje programu, szczególnie gdy są one wykonywane przez innego programistę. Z tego powodu wykres operacyjny powinien stanowić część dokumentacji każdego programu.

Symbole stosowane w wykresach operacyjnych podano w dodatku 1.

Dobrze opracowany wykres operacyjny umożliwia szybkie i bezbłędne ułożenie ciągu rozkazów programu. Pewne szczególne techniki programowania omówiono w dalszych punktach niniejszego rozdziału.

Z kolei napisane rozkazy przenosi się na taśmę perforowaną, która może być bezpośrednio odczytana przez czytnik maszyny.

Uruchamianie programu oraz dalszę czynności wykonywane przez maszynę po wprowadzeniu programu omówiono w rozdziale 8.

## 7.3. ADRESOWANIE ROZKAZÓW. ETYKIETY

W języku PJP istnieją trzy zasadnicze rodzaje adresów rozkazów: rzeczywiste (w tym przesunięte), względne i symboliczne (pkt. 5.2.2). W niniejszym podrozdziale podamy kilka wskazówek pozwalających wybrać najwygodniejszy, w danej sytuacji, rodzaj adresu.

Adresy rzeczywiste stosuje się zwykle przy adresowaniu rozkazów wykonujących operacje na komórkach pamięci powyżej 8192 oraz rozkazów przesunięć. Ponadto tego rodzaju adresy stosuje się także w rozkazach skoku do podprogramów (SLR). Stosowanie adresów rzeczywistych w innych przypadkach nie jest zalecane.

Znacznie szersze zastosowanie mają adresy względne. Występują najczęściej w rozkazach sterujących, jakkolwiek mogą się pojawiać także w innych rozkazach. Zaleca się, aby adresy względne o wartościach większych od 10 zastępować adresami symbolicznymi. W obszarze objętym przez adresy względne nie mogą znajdować się wiersze warunkowe.

Adresy symboliczne, tworzone przy pomocy etykiet, stosuje się przy rozkazach z rozmaitych grup.

W dużych programach mogą pojawić się dwa ograniczenia stosowania adresów symbolicznych. Pierwsze z nich, to ograniczona do 512 słów ilość etykiet w sekcji. Dlatego nie należy deklarować niepotrzebnie zbyt dużej ilości etykiet.

Na przykład rozkazy można zastąpić w taki sposób

```
UMA E2          UMA E2
UMB E7          UMB 1 E2
--------        --------
--------        --------
--------        --------
E2              E2
+27      ⟹     +27
E7              +41
+41             --------
--------        --------
--------        --------
--------        --------
```

przez co mogliśmy zrezygnować z deklaracji etykiety E 7. Tak samo należy wykorzystywać możliwości stosowania adresów względnych, wszędzie tam gdzie one istnieją.

Drugim ograniczeniem jest pojemność listy rozkazów do adresowania zawierających adresy symboliczne.

I także z tego powodu nie należy tworzyć adresów symbolicznych, tam gdzie mogą one być zastąpione przez adresy względne lub rzeczywiste.

Powyższe uwagi nie mają większego znaczenia w małych, kilkudziesięciorozkazowych programach.

## 7.4. ZASTOSOWANIE MODYFIKACJI ADRESOWYCH

Rozpatrzmy następujące zadanie: ułożyć fragment programu obliczający sumę zawartości 12 komórek pamięci, z których pierwsza ma adres równy wartości etykiety E27 (czyli E27 opatruje pierwsze słowo tego obszaru). Liczbę umieszczoną w tej komórce oznaczamy L1, liczbę w następnej komórce L2, itd., aż do L12.

Przedstawimy dwa rozwiązania: pierwsze z nich wykorzystywać będzie B-modyfikację, drugie zaś P-modyfikację.

| Rozwiązanie pierwsze | rozwiązanie drugie |
|---|---|
| -------- | -------- |
| -------- | -------- |
| -------- | -------- |
| UBA 10 | UMA E27 |
| UMA E27 | DOS E2 |
| DOS 1E27+ | ODP E1 |
| SOB –1 | SKO E3 |
| -------- | DOP E2 |
| -------- | SKO –4 |
| -------- | E1 |
| -------- | +10 |
| -------- | E2 |
| | STS 1E27 |
| | E3 |
| | -------- |
| | -------- |
| | -------- |

Rozwiązanie pierwsze, jak łatwo zauważyć, jest prostsze. Na początku do B ładuje się liczbę +10 (ilość liczb – 2). Następnie do A wprowadza się liczbę L1. Kolejny rozkaz powoduje dodanie do niej liczby umieszczonej w komórce pamięci o adresie równym wartości etykiety E27 zwiększonym o 1, a do tego jeszcze dodaną zawartością B-rejestru (+10). Zatem adres efektywny tego rozkazu wskazuje komórkę o adresie równym wartości etykiety E 27 zwiększonej o 11, a zatem liczbę L12.

Tak więc po wykonaniu tego rozkazu w A znajdzie się liczba L1+L12. Rozkaz SOB spowoduje odjęcie 1 od B, a następnie skok do poprzedniego rozkazu.
Wówczas wykona się ponownie rozkaz dodawania DOS, ale tym razem jego adres efektywny będzie o 1 mniejszy, toteż do otrzymanej poprzednio sumy L1+L12 zostanie dodana liczba L11, tak, że w A będzie liczba L1+L12+L11. Wówczas ponownie wykona się rozkaz SOB, następnie rozkaz dodawania, w wyniku działania którego w A będzie liczba L1+L12+L11+L10.

Powyższy proces sumowania będzie wykonywany aż do wyzerowania B-rejestru. Wówczas, gdy B = 0, do otrzymanej sumy zostanie dodana liczba L2, a następnie wykona się rozkaz SOB, którego wynikiem będzie tym razem jedynie przejście do następnego rozkazu.

Rozwiązanie drugie jest bardziej skomplikowane. Rozpoczyna się od załadowania liczby L1 do A. Następnie wykonywany jest rozkaz DOS. Jego adres efektywny znajduje się w komórce E 2: wynosi l E 27, czyli równy jest zwiększonej o 1 wartości etykiety E 27.
Zatem, do A zostanie dodana liczba L2. Z kolei rozkaz ODP powoduje odjęcie 1 od komórki opatrzonej etykietą E4 – komórka ta pełni rolę licznika.
Ponieważ zawartość tej komórki była różna od zera, więc po wykonaniu rozkazu ODP jeden rozkaz zostaje pominięty. Rozkaz DOP zwiększa adres umieszczony w komórce opatrzonej etykietą E 2 o 1, tak aby następny rozkaz DOS dodał do A następną komórkę sumowanego obszaru. Po wykonaniu rozkazu skoku SKO wykonywany jest rozkaz dodawania. W tym wypadku adres efektywny rozkazu będzie o 1 większy (wynik działania rozkazu DOP), toteż do A zostanie dodana liczba L3. Proces ten trwać będzie aż do momentu, gdy przed wykonaniem rozkazu ODP w komórce opatrzonej etykietą E 1 pojawi się zero.

Po analizie tego przykładu u czytelnika może pozostać wrażenie, że stosowanie P-modyfikacji prowadzi do niepotrzebnego skomplikowania programu.
Tak jednak nie jest.

Przedstawiony poniżej przykład wskazuje na konieczność racjonalnego wyboru typu modyfikacji adresowych, przy czym wyłączne stosowanie B-modyfikacji prowadzi, jak zobaczymy, do pewnego wydłużenia programu.

Zadanie polega na ułożeniu fragmentu programu, który podzieli zbiór 50 liczb dodatnich i ujemnych umieszczonych w pamięci operacyjnej począwszy od komórki o adresie 8192, na podzbiory liczb dodatnich i ujemnych. Ponieważ w szczególności wszystkie liczby mogą być dodatnie lub ujemne, przeto na każdy podzbiór przewidziano obszar 50 komórek pamięci.

Podobnie jak w poprzednim przykładzie, przytoczymy dwa warianty rozwiązań: rozwiązanie pierwsze wykorzystuje B-modyfikację, zaś rozwiązanie drugie B i P-modyfikacje.

Oba rozwiązania mają zbliżoną strukturę: najpierw, stosując B-modyfikację, umieszcza się w A liczbę ze zbioru liczb dodatnich i ujemnych. Następnie, po zbadaniu znaku liczby, przesyła się ją do jednego z dwóch obszarów. To przesłanie w pierwszym rozwią-

| Rozwiązanie pierwsze | rozwiązanie drugie |
|---|---|
| -------- | -------- |
| -------- | -------- |
| UBA 49 | UBA 49 |
| E1 | E1 |
| UMA 8192+ | UMA 8192+ |
| PAB E2 | SMA +4 |
| SMA +4 | PAK E3 |
| UMB E3 | DOP E3 |
| DOP E3 | SKO +3 |
| SKO +3 | PAK E4 |
| UMB E4 | DOP E4 |
| DOP E4 | SOB E1 |
| PAK 0+ | -------- |
| UMB E2 | -------- |
| SOB E1 | -------- |
| -------- | -------- |
| -------- | -------- |
| -------- | -------- |
| | -------- |
| E2 | -------- |
| A1 | -------- |
| E3 | E3 |
| STS E5 | STS E5 |
| E4 | E4 |
| STS E6 | STS E6 |
| E5 | E5 |
| A50 | A50 |
| E6 | E6 |
| A50 | A50 |
| -------- | -------- |
| -------- | -------- |

zaniu wykonywane jest przy zastosowaniu B-modyfikacji, w drugim zaś P-modyfikacji. Czynności te są powtarzane, aż do wyzerowania B-rejestru.

Na zakończenie omówimy jeszcze krótko zastosowanie rozkazów UAD i PAD. Jak wynika z opisu (pkt. 4.1.1.2), B-modyfikacja adresów tych rozkazów powoduje dodawanie do adresu podwojonej zawartości B.

W związku z tym rozkazy te nadają się bardzo dobrze do realizacji operacji na zbiorach liczb długich (48-bitowych). Zmiana bowiem, zawartości B-rejestru o 1, powoduje zmianę adresu efektywnego rozkazu o 2, a tym samym umożliwia przesłanie innej liczby. Stosując zamiast np. rozkazu UAD rozkaz UAM należałoby zmieniać zawartość B o 2, co byłoby kłopotliwe.

Powyższe wskazówki ilustruje przykład fragmentu programu, wpisujący 11 liczb +0 w postaci zmiennoprzecinkowej do pamięci począwszy od komórki 8192:

```
--------
--------
--------
UAM   E1
UBA   10
PAD   8192+
SOB   -1
--------
--------
--------
E1
-255
+0
--------
--------
--------
```

## 7.5. OPERANDY

W języku PJP operandy mogą być reprezentowane przez liczby całkowite dziesiętne lub ósemkowe oraz rozkazy. Liczby mogą reprezentować także liczby ułamkowe, zmiennoprzecinkowe itp. oraz teksty.

Wymienione operandy można z kolei podzielić na stałe, tj. te, które nie zmieniają się w trakcie wykonywania programu oraz zmienne, które z kolei mogą posiadać lub nie wartość początkową w chwili rozpoczęcia wykonywania programu.

Przy niezbyt dużej (np. 200) ilości operandów zmiennych o nieokreślonej wartości początkowej, można przewidzieć umieszczenie ich tuż obok części rozkazowej, rezerwując dla nich komórki pamięci przy pomocy dyrektywy A. Gdy zaś ilość tego rodzaju operandów jest duża i jednocześnie część rozkazowa programu jest obszerna, należy wówczas operandy te umieszczać w pamięci powyżej adresu 8192.
W tym wypadku nie rezerwuje się obszaru pamięci przy pomocy dyrektywy A, a rozkazy wykonujące działania na tych operandach mają adresy rzeczywiste.

Operandy zmienne o wartości określonej w chwili rozpoczęcia wykonywania programu mają taką samą postać jak operandy stałe. Posługując się tymi operandami należy pamiętać, że stosowanie ich w wielokrotnie wykonywanych fragmentach programu wymaga każdorazowego odtwarzania ich wartości początkowej.

Poniżej umieszczono przykładowy fragment programu zawierający operandy stałe oraz zmienne o określonej i nieokreślonej wartości początkowej.

```
--------
--------
--------
--------
--------
UMA    E27 ↑
DOP    E27
--------
--------
--------
E27
STS    E60/10: OPERAND ZMIENNY O USTALONEJ WARTOSCI
              :POCZATKOWEJ
E28
+2731  :  OPERAND STALY – NIE ZMIENIA WARTOSCI PODCZAS
       :  WYKONYWANIA PROGRAMU
Y10
SKO  E12
E60
A50  :  OBSZAR ZAREZERWOWANY NA OPERANDY ZMIENNE O NIE USTALONEJ
     :  WARTOSCI POCZATKOWEJ
UBA  127
--------
--------
--------
--------
```

## 7.6. INSTRUKCJE ZMIENNE

W praktyce programowania pojawia się czasami konieczność utworzenia rozkazów, których dokładna postać będzie mogła być określona dopiero w trakcie wykonywania programu. Innymi słowy program sam konstruuje rozkaz, a następnie go wykonuje.

Najczęściej występują dwa rodzaje takich rozkazów: rozkazy skoku („zwrotnice" o ustalonej części adresowej i kilku wariantach przepisywanych z poszczególnych fragmentów programu oraz rozkazy, których część adresowa musi być obliczona na podstawie wyników pośrednich programu. Rozpatrzymy na wstępie zastosowanie rozkazów jako przełączników (I rodzaj).

Weźmy pod uwagę następujący fragment programu, którego czynności polegają na drukowaniu na drukarce wierszowej i monitorze znaków wczytywanych przy pomocy czytnika I. Każdorazowo, pojawienie się znaku ↑ powoduje przełączenie z drukarki na monitor lub odwrotnie, przy czym znak ↑ nie jest przedrukowany.

Drukowanie rozpoczyna się na monitorze.

```
--------
--------
--------
--------
E4
SLR   1
SOB   +2
SKO   –2
UAM   26.
OAM   E1
LAR   46
SNA   E2
UMA   E2
UMN   E3
PAK   E3
PZM   E2
SKO   E4
E2
SKO   E5
E5
UMN   27.
SLR   18
SKO   E4
E6
UMN   27.
SLR   4
SKO   E4
E1
+27:  KOD ZNAKU CYFR FS
+26:  KOD ZNAKU ☂
E3
SKO   E6
--------
--------
--------
--------
```

Objaśnimy krótko działanie tego fragmentu programu. Po wczytaniu znaku rozkazem SLR 1 do B i komórki PAO [27+BD] (ewentualnie także do PAO [26+BD]) następuje badanie, czy wczytano znak pusty – jeśli tak, to wykonuje się ponownie rozkaz SLR 1. Jeśli wczytano znak różny od pustego, to rozkaz UAM 26. umieszcza się w M i A, odpowiednio zawartość komórek PAO [26+BD] i PAO [27+BD].

Z kolei, w celu zbadania, czy wczytany znak jest znakiem ↑, odejmuje się od akumulatora-mnożnika AM zawartości dwóch kolejnych komórek umieszczonych pod etykietą E1. Jeżeli wynikiem odejmowania jest liczba różna od zera, to rozkaz LAR 46 spowoduje zapalenie wskaźnika nadmiaru WN, co z kolei spowoduje skok do etykiety E2, dalej etykiety E5, po czym wydrukowanie przeczytanego znaku na monitorze.
Jeżeli WN = 0, to oznacza, że wczytano znak ↑, a to z kolei powoduje zamianę zawartości komórek opatrzonych etykietami E2 i E3, tak że następne znaki będą drukowane na drukarce wierszowej.

Kolejne znaki ↑ powodować będą za każdym razem przełączenie „zwrotnicy" umieszczonej w komórce opatrzonej etykietą E2.

Przejdziemy teraz do omówienia metody tworzenia rozkazów zmiennych drugiego rodzaju tj. takich, których część adresowa zostaje obliczana na podstawie wyników pośrednich programu. Rozpatrzmy następujący przykład:
wydrukować n znaków w kodzie M2 umieszczonych na 5 ostatnich bitach obszaru pamięci począwszy od komórki opatrzonej etykietą E49. Liczba n, określająca ilość znaków, umieszczona jest w komórce pamięci opatrzonej etykietą E62. Jako pierwszy musi być wydrukowany znak umieszczony w komórce opatrzonej etykietą E49, jako drugi znak z następnej komórki itd.

Podobnie jak w pkt. 7.4 podamy dwa rozwiązania: pierwsze wykorzystywać będzie B-modyfikację, drugie zaś P-modyfikację.

Rozwiązanie pierwsze

```
--------
--------
--------
UMA   E62
ODS   E10
DOS   E11
PAK   E12
UBA   1
ODB   E62
E12
A1
SLR   2
SOB   -2
--------
--------
--------
E10
+1
```

rozwiązanie drugie

```
--------
--------
--------
ODP   E62
SKO   +7
UMN   E48 ↑
SLR   2
DOP   E48
ODP   E62
SKO   +2
SKO   -5
--------
--------
--------
--------
--------
--------
E48
STS   E49
--------
```

```
E11                                  --------
                                     --------
UMN    E49+                          --------
--------                             --------
--------                             --------
--------                             --------
E62                                  E62
A1                                   A1
--------                             --------
--------                             --------
--------                             --------
E49                                  E49
--------                             --------
--------                             --------
--------                             --------
```

Rozpatrzmy na wstępie rozwiązanie pierwsze.

Aby spełnić żądanie określające kolejność drukowania znaków trzeba tak skonstruować pętlę, aby adres efektywny rozkazu UMN (umieścić w mnożniku) za każdym razem zwiększał się o 1.
Wiadomo przy tym, że rozkaz SOB sterujący pętlą będzie za każdym razem zmniejszał o 1 moduł zawartości B.
W ten sposób nasuwa się rozwiązanie polegające na umieszczeniu w B liczby ujemnej oraz odpowiednim zwiększeniu adresu rozkazu UMN.
Zatem w B musi być umieszczana liczba $-(n-1)$ i jednocześnie adresem rozkazu UMN musi być wartość etykiety E 49 zwiększona o $(n-1)$, czyli adresem efektywnym rozkazu UMN będzie wartość etykiety E 49 zwiększona o $(n-1)$ i zawartość B-rejestru. Na początku zawartość B wynosi $-(n-1)$, zatem adres efektywny rozkazu UMN równy jest wartości etykiety E49.
Po każdym rozkazie SOB zawartość modułu B będzie malała o 1, a tym samym adres efektywny rozkazu UMN wzrastać będzie o 1, a to zapewni pożądaną kolejność drukowania znaków.

Pojawia się tutaj jednakże pewna trudność: ponieważ adresem rozkazu UMN jest wartość etykiety E 49 zwiększona o $(n-1)$, przy czym liczba n jest nieznana przez rozpoczęciem wykonywania programu, więc nie można napisać adresu rozkazu UMN – trzeba go utworzyć w trakcie wykonywania programu.

Cztery pierwsze rozkazy rozwiązania pierwszego tworzą omawiany rozkaz UMN. Następnie utworzony rozkaz przesyłany jest do komórki pamięci opatrzonej etykietą E 12. Bezpośrednio za tą komórką znajduje się rozkaz drukowania znaków SLR 2. Ponadto, przed rozpoczęciem drukowania w B umieszcza się liczbę 1, następnie odejmuje n, i w rezultacie w B pojawia się liczba $1-n = -(n-1)$.

Rozwiązanie drugie, wykorzystujące P-modyfikacje, nie zawiera zmiennych instrukcji. Ponieważ pętle tego typu były już omawiane w pkt. 7.4 więc pominiemy tu analizę tego rozwiązania, przypuszczając, że czytelnik potrafi uczynić samodzielnie.

## 7.7. PODPROGRAMY

Dość często w programach zachodzi potrzeba wielokrotnego wykonywania identycznych czynności. W programie może być np. kilkanaście takich miejsc, w których trzeba znaleźć sumę liczb umieszczonych we wskazanym obszarze pamięci. Zwykle jest niemożliwe i niepotrzebne wypisywanie za każdym razem sekwencji rozkazów realizującej te czynności. Odpowiedni ciąg rozkazów wypisuje się jeden raz i gdy zajdzie potrzeba dokonuje się skoku do tego *podprogramu*, czyli jego wywołanie. Rozkazom skoku musi towarzyszyć, co najmniej, zapamiętanie aktualnej zawartości licznika rozkazów (śladu), aby po wykonaniu podprogramu można było powrócić do wykonywania, przerwanej przez rozkaz skoku do podprogramu, sekwencji rozkazów.

Prócz tego niektóre skomplikowane, ale typowe czynności mogą czasami być już opracowane w formie tzw. podprogramów standardowych umieszczonych w bibliotece podprogramów m.c.ZAM 41. Wówczas warto dołączyć taki podprogram także w przypadku, gdy będzie on wykonywany tylko raz.

Przystępując do opracowania podprogramu należy zastanowić się czy nadać mu formę podprogramu standardowego czy podprogramu wewnętrznego. Podprogramy standardowe, opisane w pkt. 6.2, mają strukturę nieco bardziej skomplikowaną, ale mogą być bez żadnych zmian dołączone do każdego programu. Można więc przyjąć jako zasadę, że podprogramowi nadaje się formę standardową, gdy jego cechy wskazują na możliwość zastosowania go w innych programach. Mogą to być dla przykładu podprogramy wydawnicze, podprogramy obliczające wartości funkcji, podprogramy konwersji znaków itp.

Wszelkim innym podprogramom nie opłaci się nadawać formy standardowej – przyjmują one wówczas formę podprogramów wewnętrznych.

### 7.7.1. Podprogramy wewnętrzne

Podprogramy wewnętrzne, ze względu na wygodę, wywoływane są zawsze rozkazem SKS, choć rozkazy z grupy SLR i P mogą być również stosowane.

Wywoływanie podprogramów wewnętrznych rozkazem SKS określa od razu ich ogólną strukturę. Na początku każdego podprogramu wewnętrznego znajdować się musi komórka pamięci służąca do przechowania śladu rozkazu SKS. Podprogram kończy się zwykle rozkazem WRO z adresem wskazującym na komórkę, w której zapamiętany jest ślad rozkazu SKS. Pozostałe rozkazy wynikają z funkcji realizowanych przez podprogram.

Podamy teraz przykład podprogramu wewnętrznego, który sumuje liczby całkowite umieszczone w komórkach pamięci począwszy od adresu umieszczonego w akumula-

torze A. Ilość sumowanych liczb umieszczona jest w B-rejestrze. Wynik sumowania zostaje umieszczony w A. Pierwsze słowo podprogramu opatrzono etykietą E 62.

```
--------
--------
--------
E62
A1
SOB    +2
SKO    E37:    GDY B = 0

PAK    E38:    ADRES PIERWSZEJ LICZBY
UMA    E39:    ZEROWANIE A
DOS    E38
DOP    E38:    ZWIEKSZENIE ADRESU
SOB    -2
E37
WRO    E62:    POWROT

E38
A1
E39
+0
--------
--------
--------
```

Powyższy podprogram może być wywołany przez program, którego fragment podany jest niżej.

```
--------
--------
--------
UMA    E20
UBA    7
SKS    E62:    WYWOŁANIE PODPROGRAMU
PAK    E23
--------
--------
--------
E20
STS    E21
E21
A50
```

```
E23
A1
--------
--------
--------
```

W wyniku wykonania powyższego fragmentu programu w komórce opatrzonej etykietą E23 zostanie umieszczona suma 7 liczb znajdujących się w komórkach pamięci, z których pierwsza opatrzona jest etykietą E 21.

Dane rozpatrywanego podprogramu tj. adres początku obszaru i ilość sumowanych liczb mogą być umieszczone w komórkach pamięci znajdujących się za rozkazem SKS. Jeżeli przyjmiemy, że w komórce znajdującej się za rozkazem SKS umieszczony jest adres obszaru a w następnej komórce ilość słów, to podprogram przyjmuje następującą postać:

```
--------
--------
--------
E62
A1
PAB     E40
DOP     E62
UBA     E62 ☂
PAB     E38
DOP     E62
UBA     E62 ☂
SOB     +2
SKO     E37:    SUMOWANIE O SLOW
UMA     E39
DOS     E38 ☂
DOP     E38
SOB     −2
UMB     E40
E37
WRO     E62
E38
A1
E39
+0
E40
A1
--------
--------
--------
```

Łatwo zauważyć, że powyższa wersja podprogramu nie zmienia zawartości rejestrów M, B.

Podprogram może być wywoływany przez następujący fragment programu:

```
--------
--------
--------
SKS     E62
STS     E21
+7
PAK     E23
--------
--------
--------
E21
A50
E23
A1
--------
--------
--------
```

Każdy podprogram może wywoływać wszelkie inne podprogramy, z wyjątkiem tych, które wywołały rozpatrywany, bezpośrednio lub poprzez inne podprogramy. Innymi słowy w podprogramach wewnętrznych o opisanej strukturze nie może występować rekursja – wymaga ona specjalnych metod programowania.

### 7.7.2. Wykorzystanie podprogramów standardowych

Podprogramy standardowe zostały szczegółowo opisane w pkt. 6.2. W niniejszym podrozdziale przedstawimy pewne aspekty ich wykorzystania.

Dołączenie podprogramu standardowego do programu właściwego następuje w wyniku wykonania dyrektywy F.

Wygodnie jest umieszczać dyrektywy F przed dyrektywą K.

Może się czasami zdarzyć, że dołączone podprogramy definiują różne rozkazy o tych samych kodach. W takich wypadkach należy w oparciu o dokumentację podprogramu, zmienić kod rozkazu oraz postać dyrektywy F.

Dla przykładu rozpatrzmy następującą sytuację. W programie właściwym używany jest rozkaz dodawania P16, opisany w pkt. 4.1.3. Zdefiniowanie tego rozkazu wymaga dołączenia podprogramu standardowego B2. Jednocześnie w programie używany jest rozkaz potęgowania, zdefiniowany przez podprogram B 405. Instrukcja tego podprogramu poleca, by wywoływać go również rozkazem P 16, co ze względu na wcześniejsze użycie tego rozkazu do wywoływania innego podprogramu, jest niedopuszczalne. Prócz tego wspomniana instrukcja zawiera również uwagę, że podprogramy mogą ewentualnie wywoływać również rozkazy P 17 – P 22, co z kolei pozwala na zmianę rozkazu wywołującego podprogram potęgowania – np. na P 22. Pociąga to za sobą konieczność zmiany dyrektywy F, która będzie miała teraz postać: F 405,22–0 zamiast F 405,16–0.

Większość opracowanych podprogramów standardowych, realizujących operacje czytania i drukowania, przystosowana jest do współpracy z rozmaitymi urządzeniami zewnętrznymi. Rolę przełączników pełnią komórki PAO [253+BD], PAO [254+BD], PAO [255+BD], do których, przed rozpoczęciem wykonywania programu, translator wpisuje, odpowiednio, liczby 1308 (I zestaw) lub 2458 (II zestaw), 1, 2, przystosowując w ten sposób podprogramy do pracy z czytnikiem I i perforatorem I. Zmieniając w odpowiedni, opisany w każdym podprogramie, sposób zawartości tych komórek, można przystosowywać podprogramy do pracy z rozmaitymi urządzeniami zewnętrznymi. Co więcej, w niektórych specjalnych zastosowaniach można, po odpowiednim przygotowaniu zawartości tych komórek symulować rozkaz czytania znaku umożliwiając w ten sposób czytanie liczby z monitora lub czytnika kart.

## 7.8. WYDRUKI ZAWARTOŚCI REJESTRÓW I KOMÓREK PAMIĘCI

W procesie opracowywania programu dość duże trudności nastręcza etap końcowy: uruchamianie. O ile błędy formalne programu (pkt. 8) dadzą się stosunkowo łatwo zidentyfikować i usunąć na podstawie listy błędów drukowanych przez translator, o tyle błędy merytoryczne są na ogół znacznie trudniejsze do wykrycia.

Z tego powodu w bardziej skomplikowanych programach, w trakcie ich uruchamiania, umieszcza się polecenia drukowania zawartości rejestrów lub komórek pamięci. Uzyskane wydruki ułatwiają przeprowadzenie lokalizacji błędu. Po zakończeniu uruchamiania programu polecenia te są usuwane. Aby czynność usuwania zbędnych poleceń uprościć, wprowadzono w języku PJP tzw. wiersze warunkowe (pkt. 5), które w zależności od postaci dyrektywy I mogą być traktowane jako wiersze puste lub jako wiersze znaczące. Jeśli rozpatrywane polecenia wprowadzimy do programu w formie wierszy warunkowych, to po jego uruchomieniu, usunięcie poleceń wymaga tylko zmiany dyrektywy I w nagłówku programu.

Rozpatrywane polecenia drukowania rejestrów i zawartości komórek pamięci mogą powodować wydruki w trakcie wykonywania lub po zakończeniu programu. W pierwszym wypadku polecenia te mają formę rozkazów wywołujących podprogramy wydruku (np. PO7, SLR 134. itp.), zaś w drugim wypadku mają formę parametrów umieszczonych w nagłówku programu, określających wydruki typu POST – MORTEM, opisane w pkt. 5.3.14.

Rozpatrzmy na wstępie drukowanie w trakcie wykonywania programu. W tym celu stosuje się najczęściej rozkaz PO7, jakkolwiek można stosować również wszelkie inne rozkazy wydruku jak np. SLR 130., SLR 134., SLR 135. itp.

Rozkazy warunkowe winny być umieszczone we wszystkich ważniejszych „węzłach" programu, tj. miejscach, które można traktować jako granice poszczególnych etapów wykonywania programu. Czasami rozkazy warunkowe umieszcza się także wewnątrz pętli programowych, co pozwala na sprawdzenie ich poprawności. Stosując rozkazy tego typu w formie wierszy warunkowych należy pamiętać, że nie mogą one być umieszczane w obszarze programu objętym adresami względnymi. Tak np. przedstawiony w lewej

kolumnie fragment programu jest poprawny dopóty, dopóki wiersze warunkowe są uwzględniane. Postać poprawna tego fragmentu, wykorzystując adresy symboliczne podana jest w prawej kolumnie.

```
--------                --------
--------                --------
--------                --------
SMA +4                  SMA E44
SZA +3                  SZA E44
'217 P07 7              '217 P07 7
SKO E19                 SKO E19
UMA E37                 E44
--------                UMA E37
--------                --------
--------                --------
--------                --------
```

Każdy wiersz warunkowy, a tym samym rozkaz warunkowy musi być opatrzony numerem (wiersz bez numeru ma numer 0), przy czym tym samym numerem można opatrzyć kilka wierszy warunkowych. Na ogół tymi samymi numerami opatrujemy grupy rozkazów np.:

```
--------
PAK E1
'47 P07 = 41007
'47 STS E30/15
'47 DOP -2
UMA E40/17
--------
--------
F2046,7-0
--------
--------
--------
```

Przedstawione rozkazy warunkowe drukować będą rejestry A, M, B i 33 komórki pamięci począwszy od adresu określonego przez wartość etykiety E30/15. Przedstawiony fragment jest także interesujący z innego powodu: powodować on będzie wydruk rejestrów i komórek pamięci przy każdym przejściu przez niego, nie zaś tylko 7 razy, jak by to było bez rozkazu DOP–2 (bowiem każde wykonanie rozkazu PO7 zmniejsza o 1 liczbę umieszczoną na trzech ostatnich bitach części adresowej). Należy zwrócić uwagę, że w programie wiersze warunkowe mogą być opatrywane rozmaitymi numerami np.

```
--------
--------
IA   17-17,31-47
--------
--------
'49  PO7 6 +
--------
--------
--------
'33  PO7 1 +
--------
--------
--------
--------
'17  PO7 1
--------
--------
--------
--------
```

Łatwo zauważyć, że przy podanej postaci dyrektywy I rozkaz warunkowy o numerze 49 będzie ignorowany.

Polecenia wydruków obszarów pamięci po zakończeniu wykonywania programu, POST MORTEM, umieszcza się w nagłówku programu, w sposób opisany w pkt. 5.3.14. Należy pamiętać, że łączna ilość drukowanych słów nie może przekroczyć 1024.

Oczywiście, to co tu powiedzieliśmy, nie wyczerpuje wszelkich możliwych zastosowań wierszy warunkowych – mogą one być dla przykładu stosowane w programach mających kilka wariantów itd.

## 7.9. REALIZACJA PROGRAMÓW WIELOSEKCYJNYCH

W dużych, wielosekcyjnych programach, pojawiają się znaczne trudności w komunikacji między poszczególnymi sekcjami. Z tego powodu programista powinien w ogóle unikać programów wielosekcyjnych, gdy zaś nie da się z nich zrezygnować, powinien umieszczać w sekcjach zamknięte fragmenty programu, mające niewielką ilość danych i wyników.
W ten sposób można będzie wyeliminować główne trudności.

W trakcie wykonywania programu wielosekcyjnego w pamięci operacyjnej znajduje się zwykle jedna sekcja. Po jej wykonaniu do pamięci wprowadza się następną (przepisuje z bębna) itd. aż do wykonania wszystkich sekcji.

Aby w maksymalnym stopniu uprościć te czynności przyjęto ustaloną organizację przepisywania sekcji z pamięci bębnowej do operacyjnej, wykorzystującą specjalny program sterujący wykonywaniem sekcji. Poniżej podano przykładowy fragment programu wielosekcyjnego, wykorzystujący wspomniany program sterujący.

```
--------
--------
--------
-0
X32:    KONIEC NAGLOWKA
B407
+500
VB,O
QO
--------
--------
--------
G
T: KONIEC PODPROGRAMOW STANDARDOWYCH
VB,780.: SEKCJA O
Q780.
B31
--------
--------
--------
SLR     40
T
VB,780.: SEKCJA 1
Q780.
B37
--------
--------
--------
UBA     2
PAR     23
SKO     260.: IDZ DO SEKCJI 2
T
VB,780.: SEKCJA 2
Q780.
B38
--------
--------
--------
```

```
UBA   O
UMA   +2
SKO   260.: IDZ DO SEKCJI O
+7
T
VB,256.:   SEKCJA STERUJACA
Q256.
B39
YO
UBA   1
UMA   E5
SKO   E4
Q260.
E4
PAK   E1
UMA   1E2+
ODS   E2+
PAK   +5:  ILOSC SLOW SEKCJI
UMN   E3
UMA   E2+
SLR   3: PRZEPISANIE SEKCJI Z BEBNA DO PAO
+1
A1
SLR   10
UMB   E1
SKO   780.+
E1
A1
E2
STS   B31
STS   B37
STS   B38
STS   B39
E3
STS   780.
E5
+0
F407,173–0,174–1
K256./39
```

Umieszczony w ostatniej sekcji, począwszy od komórki PAO [260+BD], program sterujący realizuje przepisywanie sekcji, której numer umieszczony jest w B-rejestrze, z pamięci bębnowej do operacyjnej. W akumulatorze A umieszczona jest liczba, która

wskazuje, do którego z kolei, licząc od początku, rozkazu należy skoczyć po przepisaniu sekcji.

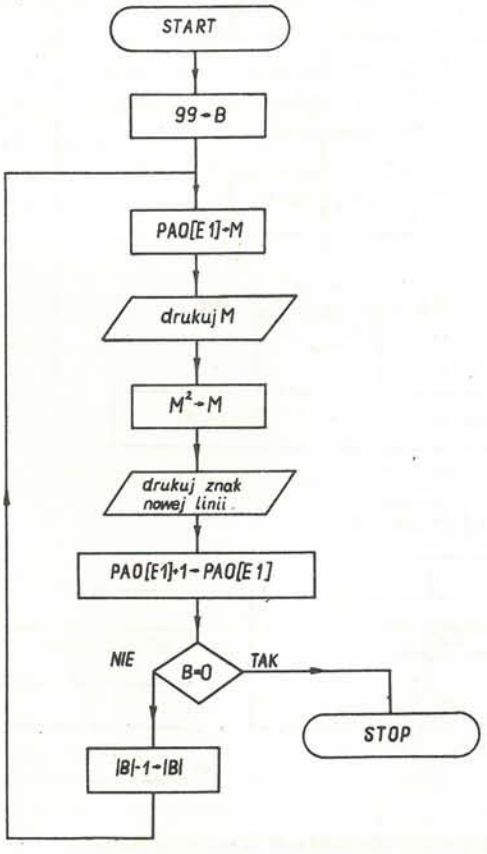
Ponieważ sekcja sterująca będzie wykonywana jako pierwsza, więc umieszczono w niej dodatkowo rozkazy ładowania B i A. W rezultacie jako druga zostanie wykonana sekcja nr 1, następnie sekcja nr 2, a w końcu sekcja nr 0.

Na końcu każdej sekcji znajduje się sekwencja rozkazów ładująca zawartości A i B, na podstawie zawartości których program sterujący umieści pożądaną sekcję.

Na końcu sekcji sterującej umieszcza się podprogramy standardowe, które znajdują się tam w czasie wykonywania całego programu. Z tego powodu wszystkie sekcje rozpoczynają się od adresu 280. zwiększonym o ilość słów podprogramów (w tym wypadku 500).

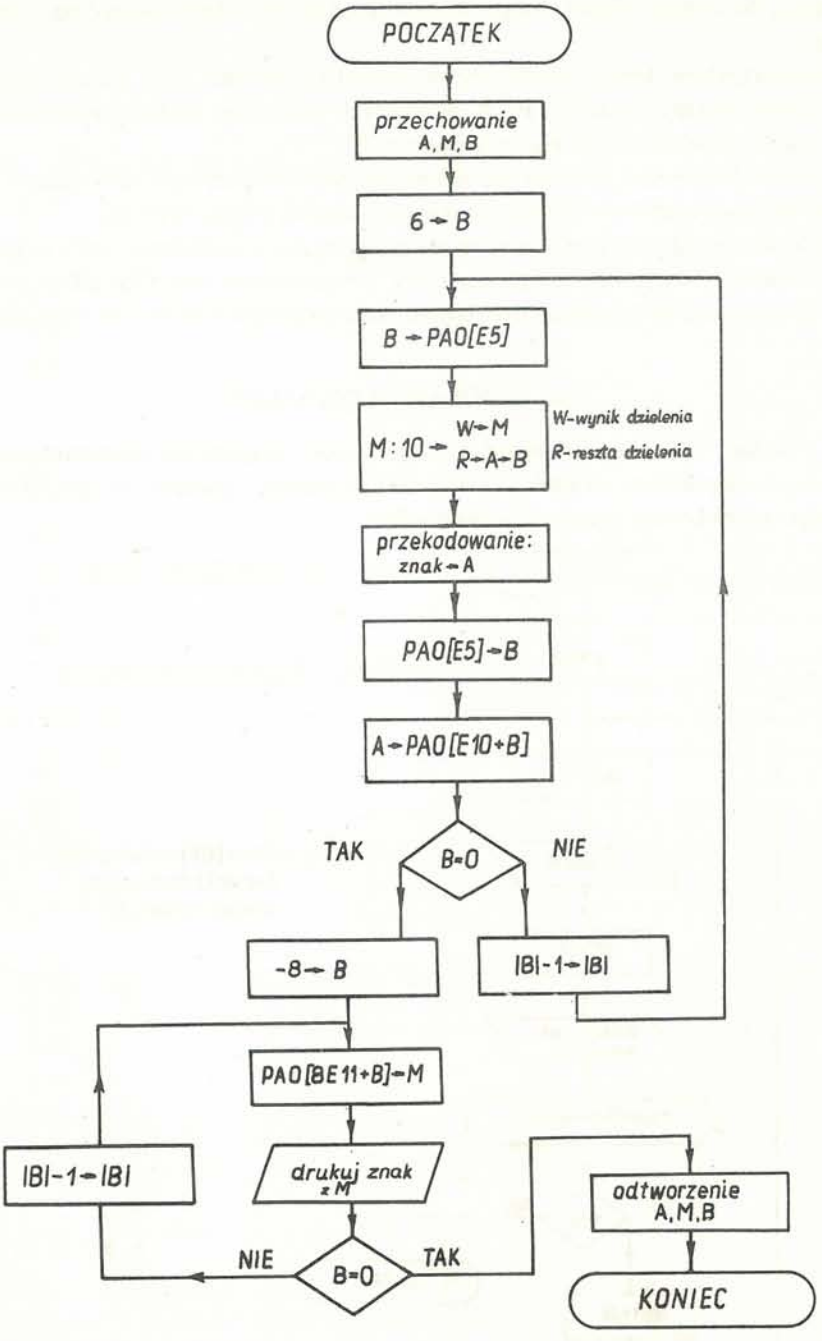
## 7.10. PROGRAM PRZYKŁADOWY

W rozdziale tym omówiliśmy już niektóre, typowe zagadnienia programowania. Aby omówione zagadnienia ukazać czytelnikowi wyraźniej, podamy tu przykład dość prostego, ale ciekawego programu w języku PJP.




PAO [E1] – zawartość komórki pamięci opatrzonej etykietą E1.

Rys. 7.2. Wykres operacyjny programu tablicowania funkcji $y = x^2$




Rys. 7.3. Wykres operacyjny podprogramu drukowania liczby

Zadaniem przedstawionego programu jest tablicowanie funkcji $y = x^2$ dla x zmieniającego się od 1 co 1 do 100. Wyniki drukowane będą w dwóch kolumnach: w lewej kolumnie wartości argumentu x, w prawej kolumnie wartości funkcji y. Ponadto założono, dla uproszczenia, że wszystkie drukowane liczby mają mieć 7 cyfr – stąd początkowe cyfry, dla podanego zakresu zmienności x, będą zerami.

Na rys. 7.2. przedstawiony jest wykres operacyjny rozpatrywanego programu. B-rejestr, jak łatwo zauważyć, spełnia jedynie rolę licznika. W każdym cyklu zawartość komórki pamięci opatrzonej etykietą E1 zwiększa się o 1, zaś zawartość B zmniejsza się o 1. Po wyzerowaniu się B-rejestru wykonywanie programu zostaje zakończone.

Podprogram podnoszący liczbę umieszczoną w mnożniku M, do kwadratu jest bardzo prosty, toteż omawiać go nie będziemy. Wyjaśnień wymaga natomiast przedstawiony na rys. 7.3 wykres operacyjny podprogramu drukowania liczby umieszczonej w M.

Podprogram składa się z dwóch części. W pierwszej poprzez 7 krotne dzielenie przez 10 uzyskuje się reszty będące kolejnymi, licząc od prawej cyframi drukowanej liczby. Każda cyfra zastępowana jest odpowiadającym jej kodem znakowym, który następnie umieszczany jest w przewidzianym obszarze pamięci. W drugiej części podprogramu otrzymane znaki są drukowane.

Przedstawimy dwa warianty programu, różniące się rodzajem zawartych w nim podprogramów. Pierwszy wariant programu, podany w lewej kolumnie, wykorzystuje podprogramy standardowe drukowania i podnoszenia liczby do kwadratu. W drugim, w prawej kolumnie, funkcje te wykonywane są przez podprogramy wewnętrzne.

```
PROG: PROGRAM PRZYKŁADOWY 3
PJP:
I
STS     1000 ⍑
STS     200
STS     0
STS     0
–0
X1
B410:  PODNOSZENIE DO KWADRATU
+7
VB,0
QO
H
Y3
PAM     E1
MNS     E1
UMA     1E1
WRO     1.
=77777777
E1
```

```
PROG: PROGRAM PRZYKŁADOWY 4
PJP:
I
STS     1000 ⍑
STS     200
STS     0
STS     0
–0
X1
VO,256.
Q256.
B1
Y1
UBA     99
UMN     E1
SKS     0/12: DRUKUJ M
SKS     0/3
SKS     0/12: DRUKUJ M
UMN     E2
SLR     2
UMN     1E2
```

```
A2
G
T
B411: DRUKOWANIE MNOZNIKA
+50
VB,0
QO
H
Y12
PAM    E1
PAB    E2
UMA    1.
PAK    E3
UBA    6
E4
PAB    E5
UMA    E6
DZS    E7
PAK    E8
UMB    E8
UMA    E9+
UMB    E5
PAK    E10+
SOB    E4
UMB    E12
UMN    8E11+
SLR    255.
SOB    -2
UAM    E1
UMB    E2
WRO    E3
=77777777
E1
A2
E2
A1
E3
A1
E5
A1
E6
+0
SLR    2
DOP    E1
SOB    1EO
SLR    40
E1
+1
E2
+2
+8
Y3: PODNOSZENIE DO KWADRATU
A1
PAM    E1
MNS    E1
UMA    1E1
WRO    EO
E1
A2
Y12: DRUKOWANIE MNOZNIKA
A1
PAM    E1
PAB    E2
UBA    6
E4
PAB    E5
UMA    E6
DZS    E7
PAK    E8
UMB    E8
UMA    E9+
UMB    E5
PAK    E10+
SOB    E4
UMB    E12
UMN    8E11+
SLR    255.
SOB    -2
UAM    E1
UMB    E2
WRO    EO
E1
A2
E2
```

```
E7
+10
E8
A1
E9
+13
+29
+25
+16
+10
+1
+21
+28
+12
+3
E11
+27
+4
E10
A7
E12
-8
G
T
VB,256.
Q256.
B1: PROGRAM WLASCIWY
Y1
UBA   99
UMN   E1
SLR   231.: DRUKUJ M
SLR   230.
SLR   231: DRUKUJ M
UMN   E2
SLR   2
UMN   1E2
SLR   2
DOP   E1
SOB   1 E0
SLR   40
E1
+1
A1
E5
A1
E6
+O
E7
+10
E8
A1
E9
+13
+29
+25
+16
+10
+1
+21
+28
+12
+3
E11
+27
+4
E10
A7
E12
-8
K256./1
```

```
E2
+2
+8
F410,230–0
F411,231–0
K256./1
```

Omówimy najpierw pierwszy wariant programu, umieszczony w lewej kolumnie. Postać nagłówka programu jest typowa. Umieszczony pod nim podprogram standardowy, oznaczony etykietą B 410 zawiera 7 słów. Podprogram definiuje rozkaz SLR 230.: podnoszenie do kwadratu liczby umieszczonej w M. Wykonanie rozkazu SLR 230. nie zmienia zawartości rejestrów A i B. Słowo złożone z samych jedynek (= 77777777) kończy część rozkazową podprogramu.

Następny podprogram standardowy, oznaczony etykietą B 411, definiuje rozkaz SLR 231.: wydruk liczby umieszczonej w M. Podprogram zaczyna się od rozkazów zapamiętujących zawartość rejestrów A, M, B oraz komórki PAO [1+BD].
Następnie do B-rejestru zostaje wpisana 6 po czym 7-krotnie wykonuje się dzielenie przez 10. Dzielenie poprzedzone jest wyzerowaniem A (ponieważ rozkaz DZS dzieli AM przez PAO [ ]). Reszta z dzielenia, umieszczona w A, przesyłana jest do B-rejestru, którego zawartość została uprzednio zapamiętana w komórce pamięci opatrzonej etykietą E 5. Teraz stosując B-modyfikację, można łatwo uzyskać kod odpowiadający uzyskanej cyfrze poprzez przesłanie do A słowa z obszaru pamięci opatrzonego etykietą E 9. (w obszarze tym umieszczone są kody kolejnych cyfr od 0 do 9).

Po odtworzeniu zawartości B-rejestru, akumulator A zapamiętywany jest w komórce, której adres równy jest sumie wartości etykiety E 10 i zawartości B-rejestru. Po 7-krotnym dzieleniu rozpoczyna się wykonywanie drugiej części podprogramu – wydruku cyfr. Przed cyframi drukowany jest znak cyfr kodu M2 – FS (+27) oraz spacja (+4).
Po odtworzeniu zawartości A, M, B następuje powrót do programu na podstawie przechowanego w komórce opatrzonej etykietą E 3 śladu rozkazu SLR 231.

Program właściwy, którego początek wskazuje etykieta B1, rozpoczyna się rozkazem UBA 99, w wyniku wykonania którego, do B-rejestru zostanie wpisana liczba 99. Następnie w mnożniku M umieszczona zostaje zawartość komórki pamięci opatrzonej etykietą E 1 (na początku = 1) i wydrukowana.
Z kolei liczba umieszczona w mnożniku zostaje podniesiona do kwadratu (rozkaz SLR 230.), po czym uzyskany wynik zostaje wydrukowany (SLR 231.). W celu uzyskania wyników pożądanej postaci dwóch kolumn, po wydruku obu liczb, drukowane są znaki powrotu karetki CR i nowej linii LF.
Po dodaniu 1 do zawartości komórki opatrzonej etykietą E 1 i odjęciu 1 od zawartości B-rejestru opisane czynności powtarzają się aż do wyzerowania B. Wówczas zostanie wykonany rozkaz SLR 40 : zakończenie programu.

Przedstawiony w prawej kolumnie drugi wariant programu różni się od omówionego jedynie rodzajem podprogramów oraz kilkoma innymi szczegółami. Toteż analizę tego wariantu pominiemy mając nadzieję, że czytelnik potrafi uczynić to samodzielnie.

# 8. URUCHAMIANIE PROGRAMÓW

## 8.1. TRANSLACJA PROGRAMU

W celu lepszego przybliżenia czytelnikowi zagadnień związanych z uruchomieniem programu w języku PJP prześledzimy teraz czynności maszyny podczas tłumaczenia programów podanych w pkt. 7.10. Wydruki podczas translacji, wykonywania, oraz po zakończeniu realizacji programu podane są poniżej.

```
PROGRAM PRZYKŁADOWY 3
ZAM 41 INSTYTUT INFORMATYKI PG
SLOWNIK E
    Y 3
EO   0
E1   5

SLOWNIK E
    Y 12
EO   0
E4   5
E1   22
E2   24
E3   25
E5   26
E6   27
E7   28
E8   29
E9   30
E11  40
E10  42
E12  49

SLOWNIK E
    Y 1
EO   2688
```

```
PROGRAM PRZYKŁADOWY 4
ZAM 41 INSTYTUT INFORMATYKI PG
SLOWNIK E
    Y 1
EO   2688
E1   2700
E2   2701
    Y 3
EO   2703
E1   2708
    Y 12
EO   2710
E4   2714
E1   2730
E2   2732
E5   2733
E6   2734
E7   2735
E8   2736
E9   2737
E11  2747
E10  2749
E12  2756

SLOWNIK B
B2046  2878
```

```
E1   2700
E2   2701

SLOWNIK B
B2046  2878
B2     3074
B3     6144
B4     3240
B5     5962
B6     3520
B7     6254
B410   32256
B411   32264
B1     32315
STA    2688

0000001   0000001
0000002   0000004
-------------------
-------------------
-------------------
0000099   0009801
0000100   0010000

LR=2699  STOP
POZOSTALO   327  CYKLI
POST MORTEM
KONIEC PM
```

```
B2     3074
B3     6144
B4     3240
B5     5962
B6     3520
B7     6254
B1     32256
STA    2688


0000001   0000001
0000002   0000004
0000003   0000009
-------------------
-------------------
-------------------
0000099   0009801
0000100   0010000

LR=2699  STOP
POZOSTALO   327  CYKLI
POST MORTEM

KONIEC PM
```

Po wyperforowaniu taśmy programu, wprowadza się ją do maszyny. Początek programu wskazuje tekst PROG:.

Tytuł programu, znajdujący się za słowem PROG:, jest przedrukowywany na arkuszu wynikowym, a w następnym wierszu drukowany jest tekst:

ZAM 41 INSTYTUT INFORMATYKI PG. Po wykonaniu tych czynności system operacyjny kontynuuje czytanie taśmy aż do znalezienia nazwy translatora programu.

Wówczas translator PJP przepisywany jest z pamięci bębnowej do operacyjnej, po czym przejmuje on funkcje czytania programu.

Kolejne rozkazy i liczby, tworzące program, wprowadzane są do pamięci bębnowej począwszy od adresu 32256 (1l–1x512). Liczby i rozkazy, z wyjątkiem rozkazów zawierających adresy symboliczne, umieszczane są na bębnie w formie zakończonej. Natomiast adresy rozkazów zawierające etykiety wewnętrzne nie mogą być zwykle obliczone przed wczytaniem całej sekcji programu (tj. po określeniu wartości wszystkich etykiet wewnętrznych), toteż rozkazy tego typu wprowadza się początkowo na bęben z wyzerowaną częścią adresową, a po wczytaniu całej sekcji uzupełnia brakujące adresy.

Po natrafieniu na dyrektywę T, kończącą sekcję, następuje zakończenie translacji sekcji, przy czym adresy rozkazów zawierających etykiety bębnowe są nadal nie ustalone. Ponadto, w rezultacie żądania umieszczonego w nagłówku programu, wydrukowany zostaje słownik etykiet wewnętrznych sekcji – SLOWNIK E, który podaje wartości wszystkich etykiet wewnętrznych występujących w sekcji. Słownik uporządkowany jest wg numerów paragrafów, natomiast etykiety należące do poszczególnych paragrafów drukowane są w kolejności występowania.

Kolejne sekcje programu tłumaczone są w ten sam sposób, oraz po zakończeniu każdej z nich drukowany jest słownik etykiet wewnętrznych: SLOWNIK E.

Tłumaczenie całego programu kończy się z chwilą natrafienia na dyrektywę K. Wówczas, podobnie jak po dyrektywie T, następuje adresowanie rozkazów zawierających etykiety wewnętrzny, po czym drukowany jest SLOWNIK E.

Następnie translator uzupełnia adresy rozkazów zawierających etykiety bębnowe B i drukuje słownik etykiet bębnowych B. Na początku słownika umieszczane są wartości etykiet bębnowych opatrujących podprogramy standardowe umieszczone stale w niedostępnym dla zapisu obszarze pamięci bębnowej. Dalej drukowane są wartości etykiet bębnowych programu.

Po zakończeniu translacji programu i wydrukowaniu słownika etykiet bębnowych zawartość pamięci bębnowej, począwszy od adresu wskazanego przez wartość etykiety bębnowej podanej w dyrektywie K, jest przepisywana do pamięci operacyjnej (adres określa również dyrektywa K). Następnie drukowany jest tekst STA i rozpoczyna się wykonywanie programu (pierwszy rozkaz znajduje się w komórce wskazanej przez dyrektywę K). Poniżej podano zawartości wybranych obszarów pamięci bębnowej i operacyjnej przed rozpoczęciem wykonywania programu.

Zawartości słów pamięci podano w formie rozkazów i liczb. Puste miejsca oznaczają zawartość nieokreśloną (wynik działania dyrektywy A). Zawartość słowa podana w formie rozkazu składa się z liczby, określającej zawartość bitów 3–8 rozkazu, następnie kodu literowego odpowiadającego tej liczbie oraz liczby określającej zawartość ostatnich 15 bitów rozkazu tj. jego adres. Znaki „.", „ ☂ ", „+" oznaczają, odpowiednio jedynkę na bicie nr 0,1,2 rozkazu.

Zawartość pamięci bębnowej po zakończeniu translacji programu.

| adres bębnowy | I wariant programu | | | II wariant programu | |
|---|---|---|---|---|---|
| 32256 | +7 | | | 57 uba | 99 |
| 32257 | .28 | pam | 5 | 42 umn | 2700 |
| 32258 | .34 | mns | 5 | 3 sks | 2710 |
| 32259 | .40 | uma | 6 | 3 sks | 2703 |
| 32260 | 52 | wro | 2433 | 3 sks | 2710 |
| 32261 | =77777777 | | | 42 umn | 2701 |

```
32262                                          2 slr     2
32263                                         42 umn  2702
32264   +50                                    2 slr     2
32265   .28  pam   22                        54 dop  2700
32266   .62  pab   24                        56 sob  2689
32267   40   uma 2433                         2 slr    40
32268   .41  pak   25                        +1
32269   57   uba    6                        +2
32270   .62  pab   26                        +8
32271   .40  uma   27
32272   .35  dzs   28                        28 pam  2708
32273   .41  pak   29                        34 mns  2708
32274   .58  umb   29                        40 uma  2709
32275   .40  uma   30+                       52 wro  2703
32276   .50  umb   26
32277   .41  pak   42+
32278   .56  sob    5
32279   .58  umb   49                        28 pam  2730
32280   .42  umn   48+                       62 pab  2732
32281     2  slr 2687 ☥                      57 uba     6
32282   .56  sob   15                        62 pab  2733
32283   .27  uam   22                        40 uma  2734
32284   .58  umb   24                        35 dzs  2735
32285   .52  wro   25                        41 pak  2736
32286   =77777777                           58 umb  2736
32287                                       40 uma  2737+
32288                                       58 umb  2733
32289                                       41 pak  2749+
32290                                       56 sob  2714
32291                                       58 umb  2756
32292   +0                                  42 umn  2755+
32293   +10                                  2 slr   255.☥
32294                                       56 sob  2724
32295   +13                                 27 uam  2730
32296   +29                                 58 umb  2732
32297   +25                                 52 wro  2710
32298   +16
32299   +10
32300   +1
32301   +21
32302   +28                                 +0
32303   +12                                 +10
```

```
32304          +3
32305          +27                          +13
32306          +4                           +29
32307                                       +25
32308                                       +16
32309                                       +10
32310                                       +1
32311                                       +21
32312                                       +28
32313                                       +12
32314          -8                           +3
32315          57   uba   99                +27
32316          42   umn   2700              +4
32317           2   slr   2663
32318           2   slr   2662
32319           2   slr   2663
32320          42   umn   2701
32321           2   slr      2
32322          42   umn   2702
32323           2   slr      2
32324          54   dop   2700              -8
32325          56   sob   2689
32326           2   slr     40
32327          +1
32328          +2
32329          +8
32330          .28  pam   2708
32331          .34  mns   2708
32332          .40  uma   2709
32333          52   wro   2433
32334          =77777777
32335
32336
32337          .28  pam   2732
32338          .62  pab   2734
32339          40   uma   2433
32340          .41  pak   2735
32341          57   uba      6
32342          .62  pab   2736
32343          .40  uma   2737
32344          .35  dzs   2738
32345          .41  pak   2739
```

```
32346          .58  umb  2739
32347          .40  uma  2740+
32348          .58  umb  2736
32349          .41  pak  2752+
32350          .56  sob  2715
32251          .58  umb  2759
32352          .42  umn  2758+
32353            2  slr  2687
32354          .56  sob  2725
32355          .27  uam  2732
32356          .58  umb  2734
32357          .52  wro  2735
32358          =77777777
32359
32360
32361
32362
32363
32364          +0
32365          +10
32366
32367          +13
32368          +29
32369          +25
32370          +16
32371          +10
32372          +1
32373          +21
32374          +28
32375          +12
32376          +3
32377          +27
32378          +4
42379
32380
23381
32382
32383
32384
32385
32386          -8
32387
32388
```

Zawartość pamięci operacyjnej przed rozpoczęciem wykonywania programu.

| adres PAO | I wariant programu | II wariant programu |
|---|---|---|
| 2662 | 1 sko 2703 | |
| 2663 | 1 sko 2710 | |
| 2685 | +2458 | +2458 |
| 2686 | +1 | +1 |
| 2687 | +2 | +2 |
| 2688 | 57 uba 99 | 57 uba 99 |
| 2689 | 42 umn 2700 | 42 umn 2700 |
| 2690 | 2 slr 2663 | 3 sks 2710 |
| 2691 | 2 slr 2662 | 3 sks 2703 |
| 2692 | 2 slr 2663 | 3 sks 2710 |
| 2693 | 42 umn 2701 | 42 umn 2701 |
| 2694 | 2 slr 2 | 2 slr 2 |
| 2695 | 42 umn 2702 | 42 umn 2702 |
| 2696 | 2 slr 2 | 2 slr 2 |
| 2697 | 54 dop 2700 | 54 dop 2700 |
| 2698 | 56 sob 2689 | 56 sob 2689 |
| 2699 | 2 slr 40 | 2 slr 40 |
| 2700 | +1 | +1 |
| 2701 | +2 | +2 |
| 2702 | +8 | +8 |
| 2703 | .28 pam 2708 | |
| 2704 | .34 mns 2708 | 28 pam 2708 |
| 2705 | .40 uma 2709 | 34 mns 2708 |
| 2706 | 52 wro 2433 | 40 uma 2709 |
| 2707 | =77777777 | 52 wro 2703 |
| 2708 | | |
| 2709 | | |
| 2710 | .28 pam 2732 | |
| 2711 | .62 pab 2734 | 28 pam 2730 |
| 2712 | 40 uma 2433 | 62 pab 2732 |
| 2713 | .41 pak 2735 | 57 uba 6 |
| 2714 | 57 uba 6 | 62 pab 2733 |
| 2715 | .62 pab 2736 | 40 uma 2734 |
| 2716 | .40 uma 2737 | 35 dzs 2735 |
| 2717 | .35 dzs 2738 | 41 pak 2736 |
| 2718 | .41 pak 2739 | 58 umb 2736 |
| 2719 | .58 umb 2739 | 40 uma 2737+ |
| 2720 | .40 uma 2740+ | 58 umb 2733 |

```
2721          .58  umb  2736          41   pak  2749+
2722          .41  pak  2752+         56   sob  2714
2723          .56  sob  2715          59   umb  2756
2724          .58  umb  2759          42   umn  2755+
2725          .42  umn  2758+          2   slr   255.☂
2726            2  slr  2687☂         56   sob  2724
2727          .56  sob  2725          27   uam  2730
2728          .27  uam  2732          58   umb  2732
2729          .58  umb  2734          52   wro  2710
2730          .52  wro  2735
2731          =77777777
2732
2733
2734                                   +0
2735                                   +10
2736
2737          +0                       +13
2738          +10                      +29
2739                                   +25
2740          +13                      +16
2741          +29                      +10
2742          +25                      +1
2743          +16                      +21
2744          +10                      +28
2745          +1                       +12
2746          +21                      +3
2747          +28                      +27
2748          +12                      +4
2749          +3
2750          +27
2751          +4
2752
2753
2754
2755
2756                                   −8
2757
2758
2759          −8
```

Jak pokazuje pobieżna analiza podanych zawartości pamięci, rozkazy i liczby tworzące program umieszczane w pamięci bębnowej w kolejności ich występowania. Warto zwrócić uwagę, że oba podprogramy B 410 i B 411 występują w pamięci bębnowej dwukrotnie: po raz drugi należą do programu właściwego, zaś adresy niektórych rozkazów zwiększone są o zawartość licznika programowego q.

Przejdziemy teraz do omówienia następnej fazy: wykonywania programu. Wynikiem jego działania jest wydruk tablicy funkcji $y = x^2$. Wykonanie rozkazu SLR 40 powoduje zakończenie realizacji programu, czemu towarzyszy wydruk zawartości licznika rozkazów LR, przy którym został wykonany rozkaz SLR 40, tekstu STOP, oraz różnicy zadeklarowanego i rzeczywistego czasu wykonywania programu w cyklach (1 cykl = 0,35 s.). Wiersze między tekstami POST MORTEM i KONIEC PM są puste, ponieważ w nagłówku nie umieszczono żądania wydruku zawartości pamięci po zakończeniu wykonywania programu.

Zawartości pamięci i wydruki programu podane w prawej kolumnie odnoszą się do drugiego wariantu programu opisanego w pkt. 7.9. Jego tłumaczenie i wykonywanie przebiega podobnie jak w omówionym, pierwszym wariancie programu.

## 8.2. URUCHAMIANIE PROGRAMÓW

Uruchomienie stanowi ostatni etap opracowywania programu w języku PJP. Polega ono na ewentualnym usunięciu błędów formalnych programu, tj. błędów polegających na niezgodności budowy programu z regułami języka PJP, a następnie na sprawdzeniu poprawności programu na podstawie uzyskanych wyników. Przyczyną błędnych wyników programu są błędy merytoryczne, wynikające z błędnego zaprogramowania zadania. Ich identyfikacja, w przeciwieństwie do błędów formalnych, jest na ogół znacznie trudniejsza.

### 8.2.1. Usuwanie błędów formalnych

Błędy formalne drukowane są w trakcie wczytywania programu, przy czym pewne grupy błędów, wypisywane są po wczytaniu całego programu. Odnalezienie błędu formalnego w programie ułatwia słownik etykiet wewnętrznych i bębnowych programu drukowany bezpośrednio pod listą błędów (żądania drukowania słownika etykiet programu opisano w pkt. 5.3.12). Każdy błąd formalny programu specyfikowany jest przy pomocy jednego wiersza podającego rodzaj błędu i miejsce jego wystąpienia. Rodzaj błędu podany jest przy pomocy numeru, którego znaczenie podaje słownik błędów (tabl. 8.1).

Miejsce wystąpienia błędu określane jest przez podanie zawartości licznika programowego q, odpowiadającej położeniu wiersza informacyjnego zawierającego błąd.

Forma wydruku błędu dla nr nr błędów różnych od 8, 9 i 15 jest następująca:

BLAD ⟨numer błędu⟩ ⟨wartość licznika q⟩ ⟨numer paragrafu⟩ ⟨pozycja wewnątrz paragrafu⟩

Dla błędów o nr 8, 9, 15 liczba określająca wartość licznika q podana jest jako ostatnia, zaś dwie poprzednie liczby nie mają znaczenia. W tym przypadku wydruk ma postać:

BLAD ⟨numer błędu⟩ ⟨liczba⟩ ⟨liczba⟩ ⟨wartość licznika q⟩

Na podstawie otrzymanego wydruku zawierającego wykaz błędów i słownik etykiet dokonuje się identyfikacji błędów formalnych, a następnie spostrzeżone błędy usuwa się.

Identyfikacja błędu jest najłatwiejsza, gdy wiersz informacyjny, zawierający błąd, opatrzony jest etykietą. Jednakże często tak nie jest, toteż aby odnaleźć błędny wiersz, trzeba najpierw odszukać w słowniku etykiet etykietę o wartości najbliższej do podanej w wydruku błędu, zawartości licznika programowego q. Etykieta opatruje wiersz informacyjny odległy o n wierszy informacyjnych od wiersza zawierającego błąd, gdzie n jest różnicą wartości etykiety i licznika q, w wykazie błędów.

Przedstawiony proces usuwania błędów formalnych zilustrujemy przykładem. Przypuśćmy, że do maszyny został wprowadzony program, którego zadaniem jest drukowanie 6 kolejnych liczb naturalnych począwszy od 200. Postać programu jest następująca:

```
PROG:  PROGRAM PRZYKŁADOWY 5
PJP:
I
STS    100
STS    100
STS    0
STS    0
–0
X 1
VO,256.
Q 256.
B 12
Y 18
UBA    5
UMA    E 16
SLR    134.
DOP    E 15
SOB    1/18
SLR    40
E 16
+200
F3,134–0
K256./12
```

Tablica 8.1

Słownik błędów

| Numer błędu | treść błędu |
|---|---|
| 0 | błąd maszyny |
| 1 | niedozwolony znak |
| 2 | znak znaczący między znakami powrotu karetki CR i nowej linii LF |
| 3 | nieprawidłowa część operacyjna |
| 4 | znaki powrotu karetki CR lub nowej linii LF w części operacyjnej rozkazu |
| 5 | wykroczenie poza dozwolony obszar pamięci operacyjnej lub bębnowej |
| 6 | błąd syntaktyczny w zakończeniu poprzedniego wiersza |
| 7 | więcej niż 8 cyfr w szablonie |
| 8 | niezadeklarowana etykieta wewnętrzna |
| 9 | niezadeklarowany paragraf |
| 10 | przepełnienie listy etykiet wewnętrznych |
| 11 | błąd syntaktyczny |
| 12 | numer paragrafu lub etykiety większy od 63 |
| 13 | przepełnienie listy rozkazów zawierających etykiety bębnowe w części adresowej; zakończenie translacji |
| 14 | brak dyrektywy I na początku programu; zakończenie translacji |
| 15 | niezadeklarowana etykieta bębnowa |
| 16 | etykieta zadeklarowana podwójnie |
| 17 | niezadeklarowana etykieta bębnowa w dyrektywie K |
| 18 | przepełnienie listy rozkazów zawierających etykiety wewnętrzne w części adresowej; zakończenie translacji |
| 19 | przepełnienie listy etykiet bębnowych |
| 20 | brak dyrektywy K |
| 26 | błędna dyrektywa X; zakończenie translacji |
| 31 | brak przecinka w dyrektywie V; zakończenie translacji |
| 32 | 32-gi błąd; zakończenie translacji |
| 44 | niezadeklarowany podprogram przy dyrektywie F |
| 48 | za duża liczba |
| 49 | etykieta bębnowa zadeklarowana podwójna |
| 50 | niepoprawne parametry dyrektywy K |
| 51 | paragraf zadeklarowany podwójnie |

Pobieżna analiza wskazuje, że rozkaz DOP E 15 zawiera błędny numer etykiety: powinno być E 16. Ale na razie tego nie zauważyliśmy i wprowadzamy program do maszyny, która dostarcza następującego wydruku:

```
PROGRAM PRZYKLADOWY 5
ZAM 41 INSTYTUT INFORMATYKI PG
BLAD 8 3587 7     2691
SLOWNIK E
   Y 18
```

EO 2688
E 16 2694
SLOWNIK B
B2046 2878
B2 3074
B3 6144
B4 3240
B5 5962
B6 3520
B7 6254
B12 32256

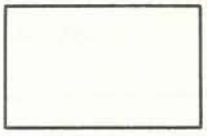
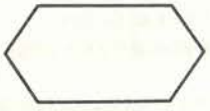
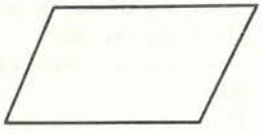
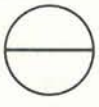
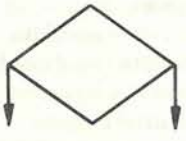
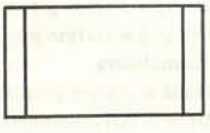
Na podstawie otrzymanego wydruku przystępujemy do identyfikacji błędu. Odnajdujemy w słowniku błędów (tablica 8.1) znaczenie błędu nr 8. Okazuje się, że błąd polega na wystąpieniu niezadeklarowanej etykiety, a ponadto stwierdzamy, że wiersz informacyjny będący źródłem błędu pojawił się przy zawartości licznika programowego q równej 2691. Teraz, w celu znalezienia błędnego wiersza, w słowniku etykiet wewnętrznych poszukujemy etykiet o wartości najbardziej zbliżonej do 2691. Okazuje się, że etykiety EO i E16 z paragrafu 18 są równoodległe od błędnego wiersza.
Dalsze poszukiwania wykonujemy w oparciu o wartość E16 wynoszącą 2694. Wiersz poprzedzający (SLR 40) tę etykietę został wprowadzony przy q = 2693 itd., skąd wynika, że przy q = 2691 wprowadzony został wiersz DOP E15.
Oczywiście! Zamiast etykiety E16 wpisano omyłkowo E15. Poprawiamy błąd na taśmie perforowanej i ponownie wpuszczamy program, który tym razem jest poprawny i drukuje poprawne wyniki.

### 8.2.2. Identyfikacja błędów merytorycznych

Jak już wspomnieliśmy identyfikacja błędów merytorycznych jest na ogół trudniejsza niż identyfikacja błędów formalnych. Nie podamy tutaj, ze względu na wielką różnorodność spotykanych programów, konkretnych recept identyfikacji tego rodzaju błędów, natomiast spróbujemy dać kilka wskazówek, które rozpatrywany proces identyfikacji, niewątpliwie w pewnym stopniu ułatwią.

1. W dużych programach trzeba koniecznie stosować rozkazy warunkowe wydruku zawartości rejestrów i pamięci. Szczegóły związane z ich wykorzystaniem omówiono w pkt. 7.7.
2. Czasami można wykonać program przy uproszczonej postaci danych np. samych zerach. Na ogół wyniki uzyskane w ten sposób łatwiej sprawdzić i ewentualnie zidentyfikować błąd w programie.
3. Praktyka uruchamiania programów wskazuje, że przyczyną błędnego działania programu jest prawie zawsze zawarty w nim błąd, a nie błędne działanie maszyny, jak to usiłują dowodzić niektórzy początkujący programiści.

**Dodatek 1.** Symbole stosowane w wykresach operacyjnych

| SYMBOL | NAZWA |
|---|---|
|  | PROCES |
|  | PRZYGOTOWANIE DANYCH |
|  | WPROWADZENIE WYPROWADZENIE |
|  | ŁĄCZNIK |
|  | DECYZJA |
|  | PROCES OKREŚLONY Z GÓRY |
| | |

## Dodatek 2

Rozkazy języka PJP

| typ rozkazu | rozkaz: kod literowy | rozkaz: kod dziesięt. | rozkaz: kod ósemkowy | nazwa rozkazu |
|---|---|---|---|---|
| sterujące | SKO | 1 | 1 | Skocz |
| | SLR | 2 | 2 | Skocz pamiętając licznik rozkazów |
| | SKS | 3 | 3 | Skocz ze śladem |
| | P04–P15 | 4–15 | 4–17 | Wykonaj rozkaz programowany |
| | P16–P22 | 16–22 | 20–26 | |
| | POL | 39 | 47 | Porównaj logicznie z akumulatorem |
| | SZA | 48 | 60 | Skocz przy zerze akumulatora |
| | SMA | 49 | 61 | Skocz przy minusie akumulatora |
| | SNA | 50 | 62 | Skocz przy nadmiarze |
| | SUB | 51 | 63 | Skocz i umieść w B-rejestrze |
| | WRO | 52 | 64 | Wróć |
| | SOB | 56 | 70 | Skocz po odjęciu 1 do B-rejestru |
| | SRB | | | |
| | POB | 61 | 75 | Porównaj z B-rejestrem |
| przesyłania | UAD | 25 | 31 | Umieść w AM długo |
| | PAD | 26 | 32 | Pamiętaj AM długo |
| | UAM | 27 | 33 | Umieść w AM |
| | PAM | 28 | 34 | Pamiętaj AM |
| | UMA | 40 | 50 | Umieść w akumulatorze |
| | UAK | | | |
| | PAK | 41 | 51 | Pamiętaj akumulator |
| | UMN | 42 | 52 | Umieść w mnożniku |
| | PZM | 43 | 53 | Pamiętaj i zeruj mnożnik |
| | UMB | 58 | 72 | Umieść w B-rejestrze |
| | PAB | 62 | 76 | Pamiętaj B-rejestr |
| działania arytmetyczne | DAM | 23 | 27 | Dodaj do AM |
| | OAM | 24 | 30 | Odejmij od AM |
| | DOS | 32 | 40 | Dodaj w stałym przecinku do akumulatora |
| | ODS | 33 | 41 | Odejmij w stałym przecinku od akumulatora |
| | MNS | 34 | 42 | Mnóż w stałym przecinku przez mnożnik |
| | DZS | 35 | 43 | Dziel w stałym przecinku AM |
| | DOB | 59 | 73 | Dodaj do B-rejestru |
| | ODB | 60 | 74 | Odejmij od B-rejestru |
| działania logiczne | DOL | 36 | 44 | Dodaj logicznie do akumulatora |
| | OSL | 37 | 45 | Odejmij symetrycznie logicznie od akumulatora |
| | MNL | 38 | 46 | Mnóż logicznie przez akumulator |

| typ rozkazu | rozkaz: kod literowy | rozkaz: kod dziesięt. | rozkaz: kod ósemkowy | nazwa rozkazu |
|---|---|---|---|---|
| przesunięcia | LCD | 30 | 36 | Przesuń w lewo cyklicznie długo |
| | PCD | 31 | 37 | Przesuń w prawo cyklicznie długo |
| | LCA | 44 | 54 | Przesuń w lewo cyklicznie akumulator |
| | PCA | 45 | 55 | Przesuń w prawo cyklicznie akumulator |
| | LAR | 46 | 56 | Przesuń w lewo arytmetycznie |
| | PAR | 47 | 57 | Przesuń w prawo arytmetycznie |
| | OKS | 55,1 | 67,1 | Zaokrąglij |
| | WMB | 55,5 | 67,5 | Przepisz wykładnik z mnożnika do B-rejestru |
| | WBM | 55,3 | 67,3 | Przepisz wykładnik z B-rejestru do mnożnika |
| | NOR | 55,7 | 67,7 | Normalizuj |
| | NOZ | 55,0 | 67,0 | Normalizuj i zaokrąglij |
| | LMB | 55,4 | 67,4 | Przesuń w lewo mnożnik oraz przepisz znak do B-rejetru |
| | PMB | 55,2 | 67,2 | Przesuń w prawo mnożnik oraz przepisz znak do B-rejestru |
| pozostałe | ODP | 53 | 65 | Odejmij od pamięci |
| | DOP | 54 | 66 | Dodaj do pamięci |
| | UBA | 57 | 71 | Umieść w B-rejestrze adres |
| nielegalne | STS | 0 | 0 | Stop |
| | DGB | 29 | 35 | Ustaw dolną i górną blokadę |
| | CML | 63,0 | 77,0 | Czytaj do mnożnika w lewo |
| | PMZ | 63,1 | 77,1 | Pisz mnożnik znakami |
| | CMP | 63,2 | 77,2 | Czytaj do mnożnika w prawo |
| | PWT | 63,3 | 77,3 | Pisz wskaźnik |
| | PML | 63,4 | 77,4 | Pisz mnożnik w lewo |
| | CMZ | 63,5 | 77,5 | Czytaj do mnożnika znakami |
| | PMP | 63,6 | 77,6 | Pisz mnożnik w prawo |
| | CWT | 63,7 | 77,7 | Czytaj wskaźnik |

Rozkazy współpracy z urządzeniami zewnętrznymi
oraz rozkaz zakończenia programu

| typ rozkazu | rozkaz | funkcje rozkazu |
|---|---|---|
| czytanie taśmy perforowanej | SLR 1 | czytaj rządek taśmy na czytniku I |
| | SLR 5 | czytaj rządek taśmy na czytniku II |
| | SLR 254.☂ | czytaj rządek taśmy |
| drukowanie | SLR 2 | drukuj znak na perforatorze I |
| | SLR 4 | drukuj znak na drukarce wierszowej |
| | SLR 6 | drukuj znak na perforatorze II |
| | SLR 13 | przesuń papier na drukarce wierszowej |
| | SLR 14 | drukuj znak na drukarce wierszowej |
| | SLR 18 | drukuj znak na monitorze |
| | SLR 24 | drukuj 4 znaki na drukarce wierszowej |
| | SLR 255.☂ | drukuj znak |
| operacja monitora | SLR 8 | drukuj znaki na monitorze |
| | SLR 9 | czytaj znaki z monitora |
| | SLR 18 | drukuj znak na monitorze |
| | SLR 20 | czekaj na zakończenie operacji monitora |
| | SLR 21 | sprawdź, czy operacja monitora jeszcze trwa |
| operacja bębnowa | SLR 3 | zainicjuj operację bębnową |
| | SLR 10 | czekaj na zakończenie operacji bębnowej |
| | SLR 30 | zainicjuj operację bębnową |
| operacja taśmowa | SLR 7 | zainicjuj operację taśmową |
| | SLR 11 | czekaj na zakończenie operacji taśmowej |
| | SLR 12 | czekaj na zakończenie odwijania |
| czytanie karty | SLR 15 | czytaj kartę perforowaną |
| zakończenie programu | SLR 40 | zakończenie programu |

Rozkazy definiowane przez podprogramy standardowe

| typ rozkazu | rozkaz | funkcje rozkazu |
|---|---|---|
| czytanie liczby zmiennoprz. | SLR 131. | czytaj liczbę zmiennoprzecinkową |
| drukowanie liczb | SLR 130. | drukuj liczbę zmiennoprzecinkową |
| | SLR 134. | drukuj liczbę stałoprzecinkową |
| drukowanie znaków | SLR 135. | drukuj tekst |
| | SLR 136. | drukuj spacje |
| | SLR 137. | drukuj linie |
| | SLR 138. | drukuj znaki puste |
| konwersja | SLR 160. | zamień liczbę zmiennoprzecinkową na liczbę stałoprzecinkową |
| | SLR 161. | zamień liczbę stałoprzecinkową na liczbę zmiennoprzecinkową |
| | SLR 162. | oblicz entier |
| operacje arytmetyczne | P 16 | dodaj w zmiennym przecinku |
| | P 17 | odejmij w zmiennym przecinku |
| | P 18 | pomnóż w zmiennym przecinku |
| | P 19 | podziel w zmiennym przecinku |
| | P 20 | pomnóż ułamki |
| | P 21 | podziel ułamki |
| wydruk rejestrów i pamięci | P07 | wydrukuj zawartość rejestrów i komórek pamięci operacyjnej |

## Dodatek 3

Zestawienie rejestrów i wskaźników zapisywanych i odczytywanych rozkazami z grupy 63

| kanał | nazwa rejestru lub wskaźnika | nr rej. (poz. nr 12–14 rozkazu) | nr kanału (poz. nr 15–18 rozkazu) | rozkazy współpracujące |
|---|---|---|---|---|
| wewnątrz. | Wskaźnik przyjęć | 0 | 0 | CWT, PWT |
| | Rejestr zegara (12-bitowy) | 1 | 0 | CML, CMP |
| pamięć bębnowa | Wskaźnik błędu | 0 | 1 | CWT |
| | Wskaźnik czytania | 1 | 1 | CWT, PWT |
| | Wskaźnik pisania | 2 | 1 | PWT |
| | Rejestr adresu (18-bitowy) | 0 | 1 | PMP |
| | Bufor znaku | 1 | 1 | PMZ, CMZ |
| | Rejestr błędu (3-bitowy) | 2 | 1 | CML, CMP |
| stolik operatora I | Wskaźnik zgł. operatora | 0 | 2 | CWT |
| | Wskaźnik błędu | 1 | 2 | CWT |
| | Wskaźnik czytania | 2 | 2 | PWT |
| | Wskaźnik „czytaj oktalnie" | 3 | 2 | PWT |
| | Rejestr kluczy (4-bitowy) | 0 | 2 | CMP, CML |
| | Rejestr lampek (4-bitowy) | 1 | 2 | PMP, PML |
| | Rejestr czytnika (8-bitowy) | 2 | 2 | CMP, CML |
| | Rejestr perforatora (8-bitowy) | 3 | 2 | PMP, PML |
| drukarka wierszowa | Wskaźnik drukowania | 0 | 3 | PWT |
| | Wskaźnik zgł. operatora | 1 | 3 | CWT |
| | Wskaźnik błędu | 2 | 3 | CWT |
| | Bufor znaku | 0 | 3 | PMZ |
| | Rejestr przesuwu (6-bitowy) | 1 | 3 | PMP, PML |
| | Rejestr lampek (4-bitowy) | 2 | 3 | PMP, PML |
| | Rejestr kluczy (4-bitowy) | 3 | 3 | CMP, CML |
| stolik operatora II | Wskaźnik zgł. operatora | 0 | 4 | CWT |
| | Wskaźnik błędu | 1 | 4 | CWT |
| | Wskaźnik czytania | 2 | 4 | PWT |
| | Rejestr kluczy (4-bitowy) | 0 | 4 | CMP, CML |
| | Rejestr lampek (4-bitowy) | 1 | 4 | PML, PMP |
| | Rejestr czytnika (8-bitowy) | 2 | 4 | CMP, CML |
| | Rejestr perforatora (8-bitowy) | 3 | 4 | PMP, PML |

| kanał | nazwa rejestru lub wskaźnika | nr rej. (poz. nr 12–14 rozkazu) | nr kanału (poz. nr 15–18 rozkazu) | rozkazy współpracujące |
|---|---|---|---|---|
| monitor | Wskaźnik drukowania | 0 | 5 | PWT, CWT |
| | Wskaźnik zgł. operatora | 1 | 5 | CWT |
| | Rejestr znaku (5-bitowy) | 0 | 5 | PML, PMP |
| | Rejestr kluczy (4-bitowy) | 1 | 5 | CML, CMP |
| | Rejestr lampek (4-bitowy) | 2 | 5 | PMP, PML |
| kanał pamięci taśmowej | Wskaźnik błędu | 0 | 6 | CWT |
| | Rejestr wskaźników (12-bitowy) | 0 | 6 | CMP, CML |
| | Rejestr operacji (6-bitowy) | 1 | 6 | PMP, PML |
| | Rejestr adresu (15-bitowy) | 2 | 6 | PMP, PML |
| | Rejestr ilości słów (15-bitowy) | 3 | 6 | PMP, PML |
| czytnik kart | Wskaźnik „czytaj kartę" | 0 | 8 | PWT |
| | Wskaźnik błędu | 1 | 8 | CWT |
| | Wskaźnik zgł. operatora | 2 | 8 | CWT |
| | Rejestr czytnika (12-bitowy) | 0 | 8 | CMP, CML |
| | Rejestr kluczy (4-bitowy) | 1 | 8 | CMP, CML |
| | Rejestr lampek (4-bitowy) | 2 | 8 | PMP, PML |

## LITERATURA

1. Praca zbiorowa pod red. W.M. Turskiego. Wybrane zagadnienia systemów operacyjnych, PWN 1971.
2. *D.D. Mc Cracken:* Programowanie maszyn cyfrowych, PWN 1962.
3. *J. Bańkowski, K. Fiałkowski:* Programowanie w języku maszyny, PW 1971.
4. Dokumentacja techniczno-ruchowa maszyny cyfrowej ZAM 41.
5. Praca zbiorowa pod red. J. Swianiewicza: Podstawowy Język Programowania m.c. ZAM 41.

Cena zł 25.—