C. Demonstracja pracy maszyny liczącej.
D. Interpretacja kodów zewnętrznych. Kodowanie automatyczne.
F. Logika matematyczna.
G. Porządkowanie. Wybieranie. Zliczanie.
I. Specjalne działania arytmetyczne (zmiennoprzecinkowe, ze zwiększoną dokładnością).
J. Obliczenie wartości funkcji. Układanie tablic. Sumowanie szeregów.
K. Interpolacja. Ekstrapolacja. Aproksymacja.
M. Teoria liczb.
N. Algebra liniowa.
O. Równania algebraiczne nieliniowe, równania przestępne i układy.
Q. Przybliżone różniczkowanie. Przybliżone całkowanie.
R. Równania różnicowe.
S. Równania różniczkowe zwyczajne.
T. Równania różniczkowe cząstkowe.
U. Równania całkowe.
V. Inne zagadnienia analizy matematycznej.
X. Statystyka matematyczna.
W tablicy 4-12 przedstawiona jest karta klasyfikacyjna podprogramu.

5. ORGANIZACJA PRACY NA MASZYNIE TYPOWEJ

[5.1. Programy wprowadzające, 5.2. Programy wyprowadzające, 5.3. Uruchomienie maszyny, 5.4. Szukanie błędów w programach]

5.1. PROGRAMY WPROWADZAJĄCE

5.1.0. Programy i materiał liczbowy wprowadzamy do maszyny za pomocą specjalnych programów wprowadzających, które z symboli wydziurkowanych na taśmie perforowanej i odczytanych przez maszynę (tabl. 2-5) formują słowa (17 albo 34 bitowe) i umieszczają je pod odpowiednimi adresami w pamięci.

Mała maszyna cyfrowa nakłada dość trudne w praktyce do realizacji warunki na programy wprowadzające. Szczególnie ważne jest, aby program przeznaczony do wprowadzania rozkazów i pseudorozkazów (punkt 3.0) zapisanych w kodzie wewnętrznym w adresach względnych (punkt 3.1), dalej zwanym podstawowym programem wprowadzającym, działał stosunkowo szybko i był możliwie krótki.

Obecnie omówimy trzy programy wprowadzające, a mianowicie:

1) podstawowy program wprowadzający, przeznaczony do wprowadzania słów krótkich (rozkazów lub pseudorozkazów) i umieszczania tych słów krótkich w odpowiednich miejscach w pamięci;

2) stałoprzecinkowy program wprowadzająco-przeliczający dla dziesięciocyfrowych liczb dziesiętnych (ostatnia cyfra parzysta) co do modułu mniejszych od jedności; program ten przelicza liczby z systemu dziesiętnego na binarny i umieszcza wyniki w odpowiednich miejscach pamięci;

3) zmiennoprzecinkowy program wprowadzająco-przeliczający dla ośmiocyfrowych zmiennoprzecinkowych liczb dziesiętnych; program ten przelicza liczby ze zmiennoprzecinkowego systemu dziesiętnego na zmiennoprzecinkowy system binarny i umieszcza wynik w odpowiednich miejscach pamięci.

Podstawowy program wprowadzający znajduje się na stałe w maszynie (jest on nagrany na ścieżce nr zero), pozostałe programy wprowadzające w zależności od potrzeb wprowadzamy do maszyny (oczywiście za pomocą podstawowego programu wprowadzającego).

Przy korzystaniu z każdego z wyżej wymienionych programów wprowadzających nastawiamy warunek wejścia na pierwszej pozycji wejścia, za pomocą przełącznika Y0 (Załącznik 2).

5.1.1. Podstawowy program wprowadzający. Jak już wspominaliśmy, program ten służy do wprowadzania słów krótkich (rozkazów lub pseudorozkazów). Program ten rozróżnia słowa dwóch długości: słowa, które nie podlegają przeadresowaniu przy wprowadzaniu, kodowane w sześciu rządkach taśmy perforowanej i słowa, które podlegają przeadresowaniu w trakcie wprowadzania, kodowane w siedmiu rządkach taśmy perforowanej. Podstawowy program wprowadzający dopuszcza korzystanie w jednej wprowadzonej sekwencji rozkazów i pseudorozkazów z co najwyżej dwunastu modyfikatorów (stałych przeadresowania), oznaczonych odpowiednio literami X, N, H, L, M, S, G, B, D, F, J, K. Jako zasadę przy posługiwaniu się programem przyjęto używać w przypadku wprowadzania podprogramów, zawierających stałą przeadresowania, modyfikatora K.

Podstawowy program wprowadzający korzysta z 14 miejsc roboczych na ścieżce nr 1.

Do sterowania podstawowym programem wprowadzającym służą specjalne rozkazy, tzw. rozkazy taśmowe, wprowadzone do maszyny przez ten program, umieszczone pod adresem 0101, a następnie wykonane przez podstawowy program wprowadzający. W tablicy 5-1 są podane wszystkie rozkazy taśmowe, jakich używamy przy wprowadzaniu programu do maszyny.

Perforowanie taśmy. Przywykliśmy do pisania liczb począwszy od cyfry najbardziej znaczącej, a skończywszy na cyfrze najmniej znaczącej. Okazało się, że znaczne uproszczenia podstawowego programu wprowadzającego można uzyskać wprowadzając liczby w kierunku przeciwnym do tego, w jakim przywykliśmy pisać liczby. Tę pozorną trudność możemy rozwiązać w sposób bardzo prosty, a mianowicie dziurkujemy kolejne cyfry liczby na taśmie w kolejności od najbardziej znaczącej do najmniej znaczącej, wprowadzamy zaś taśmę w kierunku przeciwnym do dziurkowania. Musimy przy tym pamiętać, że zaczynamy wówczas wprowadzanie od ostatniego roz-

Tablica 5-1

Rozkazy taśmowe

Lp.	Kod	Czynności wykonywane
1	M1 *n*	przeczytanie jednego słowa, a następnie przekazanie sterowania rozkazowi, którego adres jest zapisany pod adresem *n*
2	M2 *n*	przeczytanie jednego słowa, a następnie przekazanie sterowania rozkazowi zapisanemu pod adresem *n*
3	M5 *n*	przeczytanie jednego słowa, a następnie „stop" z pobraniem do rejestru dyspozytora rozkazu zapisanego pod adresem *n*
4	S4 *n*	przesłanie następnie odczytanego słowa krótkiego przez program wprowadzający pod adres *n*
5	G4 *n*	przeczytanie jednego słowa, a następnie przewinięcie taśmy perforowanej o jeden rządek

Tablica 5-2

Przykład rozkazów taśmowych końca czytania

Kod rozkazu	Kolejność symboli dziurkowanych	Kod zewnętrzny symboli dziurkowanych
	0	10 000
	0	10 000
	0	10 000
00 0000	0	10 000
	0	10 000
	0	10 000
	M	01 000
	2	10 010
M2 1777	1	10 001
	7	10 111
	7	10 111
	7	10 111

kazu programu, a kończymy wprowadzanie wprowadzając pierwszy rozkaz programu. Dziurkując taśmę korzystamy z programów zapisanych na formularzach EM (rys. 3-1), dziurkujemy rozkaz po rozkazie w kolejności wzrastających adresów.

Przystępując do dziurkowania, dziurkujemy najpierw rozkazy dotyczące czynności maszyny, po wprowadzeniu programu. W tablicy 5-2 podany jest przykład rozkazów przekazujących sterowanie po zakończeniu czytania rozkazowi stojącemu pod adresem 1777'. Następnie dziurkujemy kolejne rozkazy (albo pseudorozkazy) w kolejności ustalonej na arkuszu programowym EM. Jeśli w części adresowej występuje stała przeadresowania, np. K, to dziurkujemy ją również za pomocą odpowiedniego klawisza dziurkarki. Pomiędzy kolejnymi ósemkami słów krótkich dziurkujemy odstępy wejścia, które ułatwią nam kontrolę taśmy i ewentualne poprawki na taśmie. Na końcu dziurkujemy grupę rozkazów taśmowych początku czytania i wartości stałych przeadresowania dla wydziurkowanego programu. W tablicy 5-3 podaliśmy przykład rozkazów

taśmowych dotyczących początku czytania, przy założeniu, że ostatni rozkaz podprogramu umieszczamy pod adresem 0533′, a stała przeadresowania K (modyfikator) ma wartość 0237′.

Korekcja taśmy. Jeśli jakiś rozkaz w podprogramie został źle wydziurkowany, ale przy tym żadna cyfra nie została opuszczona ani też nie zostało wydziurkowane więcej cyfr, to między końcem czytania a pierwszym rozkazem podprogramu umieszczamy grupę rozkazów: S4-adres błędnie wydziurkowanego rozkazu, a następnie właściwą postać tego rozkazu. Aby uniknąć zbędnego klejenia taśmy, należy zostawić duży odstęp między rozkazami końca czytania a pierwszym rozkazem programu i w tym miejscu wydziurkujemy dodatkową grupę rozkazów korekcji taśmy.

Tablica 5-3

Przykład rozkazów taśmowych początku czytania

Kod rozkazu	Kolejność symboli dziurkowanych	Kod zewnętrzny symboli dziurkowanych
	S	01 001
	4	10 100
S4 0533	0	10 000
	5	10 101
	3	10 011
	3	10 011
	0	10 000
	0	10 000
00 0237	0	10 000
(modyfikator	2	10 010
K)	3	10 011
	7	10 111
	S	01 001
	4	10 100
S4 0116	0	10 000
	1	10 001
	1	10 001
	6	10 110

Ponieważ program wprowadzamy w kierunku przeciwnym do kierunku dziurkowania, więc kolejność czytania grup rozkazów początku czytania i końca czytania jest taka jak w tablicach 5-4 i 5-5.

Jak już wspominaliśmy, podstawowy program wprowadzający dysponuje 12 modyfikatorami, których wartości zapisujemy odpowiednio w komórkach o adresach: X = = (0102), N = (0104), H = (0105), L = (0106), M = (0107), S = (0110), G = (0111), B = (0112), D = (0113), F = (1014), J = (0115) i K = (0116). Pozostałe dwie komórki robocze 0100 i 0101 będziemy oznaczali odpowiednio literami r i t. Komórkę 0100 będziemy nazywali relatywizatorem programu wprowadzającego, a komórkę

Tablica 5-4

Przykład początku czytania

Taśma	Kod zewnętrzny	Symbole na klawiaturze dziurkarki	
● • ● ●	10 110	6	
● • ●	10 001	1	
● • ●	10 001	1	S4 0116
● •	10 000	0	
● • ●	10 100	4	
• ● • ●	01 001	S	
● • ● ● ●	10 111	7	
● • ● ●	10 011	3	
● • ●	10 010	2	00 0237
● •	10 000	0	modyfikator K
● •	10 000	0	
● •	10 000	0	
● • ● ●	10 011	3	
● • ● ●	10 011	3	
● • ● ●	10 101	5	S4 0533
● •	10 000	0	
● • ●	10 100	4	
• ● • ●	01 001	S	

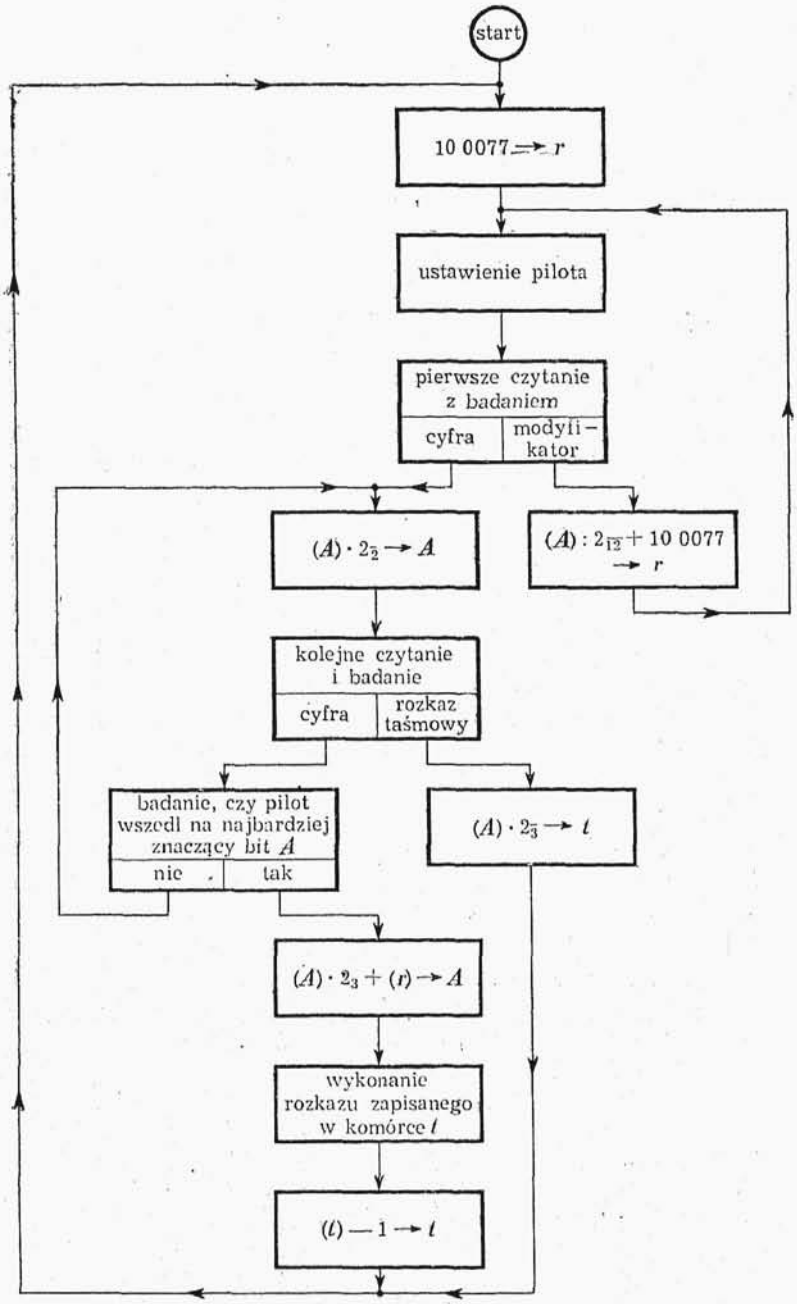
Tablica 5-5

Przykład końca czytania

Taśma	Kod zewnętrzny	Symbole na klawiaturze dziurkarki	
● • ● ● ●	10 111	7	
● • ● ● ●	10 111	7	
● • ● ● ●	10 111	7	M2 1777
● • ●	10 001	1	
● • ●	10 010	2	
• ● •	01 000	M	
● •	10 000	0	
● •	10 000	0	
● •	10 000	0	00 0000
● •	10 000	0	
● •	10 000	0	
● •	10 000	0	

0101, w której umieszczony jest rozkaz taśmowy, będziemy nazywali transferatorem programu wprowadzającego.

Na rysunku 5-1 podany jest schemat blokowy podstawowego programu wprowadzającego, w tabl. 5-6 zaś podane są kolejne rozkazy tego programu. Podstawowy program wprowadzający jest programem trójwariantowym, pierwszy wariant — czytanie rozkazu (pseudorozkazu) niezmodyfikowanego, drugi wariant — czytanie rozkazu (pseudorozkazu) modyfikowanego, trzeci wariant — czytania rozkazu taśmowego.



Rys. 5-1. Schemat blokowy podstawowego programu wprowadzającego

Tablica 5-6

Podstawowy program wprowadzający

Kolejny adres	Kolejny rozkaz		Objaśnienia
0000	12	0027	przesłanie rozkazu 10 0077 do relatywizatora
1	14	0100	
2	12	0076(1)	przesłanie i ustawienie w akumulatorze pilota
3	22	0040	
4	24	2424	czytanie jednego rządka taśmy
5	03	0022	skok w przypadku odczytania modyfikatora
6	23	0002	przesunięcie zawartości akumulatora o dwie pozycje w lewo
7	24	0000	kolejne czytanie jednego rządku taśmy
0010	03	0025	skok w przypadku odczytania rozkazu taśmowego
1	23	0000	sprawdzenie położenia pilota, skok, gdy pilot nie znajduje
2	06	0006	się na bicie α_0
3	23	0003	ustawienie w akumulatorze odczytanego słowa krótkiego
4	00	0100	wykonanie rozkazu zapisanego w relatywizatorze
5	00	0101	wykonanie rozkazu zapisanego w transformatorze
6	12	0101	
7	11	0076(1)	przeadresowanie zawartości transformatora
0020	14	0101	
1	02	0000	skok do początku programu w przypadku słowa krótkiego
2	22	0014	modyfikowanego, sformowanie właściwej postaci modyfika-
3	10	0027	tora
4	02	0001	skok do rozkazu 0001
5	23	0003	ustawienie w akumulatorze odczytanego rozkazu taśmowego
6	02	0020	skok do rozkazu 0020
7	10	0077	parametr

(1) Pod adresem 0076 zapisany jest parametr 2^{-16}, w postaci pseudorozkazu 00 0001.

Tablica 5-7

Wariant I: Kolejne zawartości licznika rozkazów przesyłane do rejestru dyspozytora przy czytaniu rozkazu (pseudorozkazu) nie modyfikowanego

0000	0006	0012	0011
0001	0007		0012
0002	0010	0006	
0003	0011	0007	0013
0004	0012	0010	0014
0005		0011	0015
0006		0012	0016
0007	0006		0017
0010	0007	0006	0020
0011	0010	0007	0021
0012	0011	0010	

Dla łatwiejszego zrozumienia działania podstawowego programu wprowadzającego podaliśmy w tabl. 5-7, 5-8 i 5-9 kolejne zawartości licznika rozkazów przesyłane do rejestru dyspozytora, dla wariantów pierwszego, drugiego i trzeciego.

Tablica 5-8

Wariant II: Kolejne zawartości licznika rozkazów przesyłane do rejestru dyspozytora przy czytaniu rozkazu (pseudorozkazu) modyfikowanego

0000	0002	0006	0012	0010
0001	0003	0007		0011
0002	0004	0010	0006	0012
0003	0005	0011	0007	0013
0004	0006	0012	0010	0014
0005	0007		0011	0015
0022	0010	0006	0012	0016
0023	0011	0007		0017
0024	0012	0010	0006	0020
0001		0011	0007	0021

Tablica 5-9

Wariant III: Kolejne zawartości licznika rozkazów przesyłane do rejestru dyspozytora przy czytaniu rozkazu taśmowego

0000	0012	0010	0006
0001		0011	0007
0002	0006	0012	0010
0003	0007		0025
0004	0010	0006	0026
0005	0011	0007	0020
0006	0012	0010	0021
0007		0011	
0010	0006	0012	
0011	0007		

5.1.2. Stałoprzecinkowy program wprowadzająco-przeliczający. W odróżnieniu od podstawowego programu wprowadzającego, program wprowadzająco-przeliczający wprowadza do maszyny pojedyncze liczby, dlatego też wprowadzanie liczby za pomocą tego podprogramu jest wolniejsze od wprowadzania liczb zakodowanych w postaci par pseudorozkazów za pomocą podstawowego programu wprowadzającego. Dla zapisania jednej liczby dziesiętnej dziesięciocyfrowej (ostatnia cyfra parzysta), co do modułu mniejszej od jedności, potrzebujemy aż 16 rządków taśmy perforowanej z podziałem na następujące grupy:

1 rządek + 4 rządki + 1 rządek + 10 rządków.

Pierwszy rządek przeznaczony jest na zapis symbolu końca przeliczenia, w tym celu perforujemy na tej pozycji przed pierwszą liczbą cyfrę 0, a przed następnymi liczbami

literę D. Dalsze cztery rządki przeznaczone są na adres (zapisany w kodzie ósemkowym), pod który ma być przesłana odczytana liczba. Następnie jeden rządek przeznaczony jest na znak liczby przeliczonej. Pozostałe 10 rządków służy do zapisania kolejnych dziesięciu cyfr liczby przeliczonej. Liczby dziesiątkowe przeznaczone do perforowania zapisujemy na specjalnych arkuszach programowych D.

Oznaczmy kolejne cyfry rozwinięcia l w postaci dziesiętnej przez d_1; możemy liczbę l zapisać w postaci:

$$l = \sum_{i=1}^{10} d_i 10^{-i}. \tag{5-1}$$

Dla przeliczenia wyrażeniu (5-1) nadamy postać Hornera:

$$(\ldots(d_{10} \cdot 10^{-1} + d_9) \cdot 10^{-1} + d_8)10^{-1} + \ldots + d_1)10^{-1}. \tag{5-2}$$

Stałoprzecinkowy program wprowadzająco-przeliczający, podobnie jak podstawowy program wprowadzający, wprowadza liczby w kierunku od najmniej znaczącej cyfry do najbardziej znaczącej cyfry (w kierunku przeciwnym do dziurkowania). Liczba 10^{-1} jest liczbą dwójkowo niewymierną i dlatego bezpośrednie korzystanie z formuły (5-2) prowadziłoby do dużych błędów. Dla uniknięcia tej trudności, zamiast mnożyć kolejne cyfry rozwinięcia d_i przez 10^{-1}, będziemy je dzielić przez 10, dzielenie wykonujemy z zaokrągleniem według reszty.

Poświęćmy trochę miejsca dla omówienia metody zaokrąglenia przy dzieleniu. Jeśli dowolną liczbę dodatnią w rozwinięciu dziesiętnym k-cyfrową mamy podzielić przez jakąś inną liczbę dodatnią w rozwinięciu dziesiętnym i chcemy w wyniku otrzymać najlepsze k-cyfrowe przybliżenie wyniku, to za najmniej znaczącą cyfrę dzielnej zapisujemy połowę dzielnika i w ten sposób otrzymaną liczbę dzielimy przez nasz dzielnik. Omówimy jeszcze powyższą metodę na przykładach.

Przykład 5-1. Niech $k = 2$, dzielna będzie równa 2,2, zaś dzielnik 3. Wykonujemy dzielenie

```
0,733...
2,2        :3
21
 10
  9
 10
  :
```

Najlepszym dwucyfrowym przybliżeniem wyniku jest liczba 0,73. Jeśli teraz zastosujemy metodę dzielenia z zaokrągleniem, to zamiast dzielić liczbę 2,2 będziemy dzielić liczbę 2,215 przerywając dzielenie po otrzymaniu drugiej cyfry wyniku:

```
0,73
2,215    :3
21
 11
  9
```

Przykład 5-2. Niech $k = 2$, dzielna będzie równa 1,7, dzielnik zaś będzie równy 2,4. Wykonujemy dzielenie:

```
0,708...
1,7     :2,4
168
 200
 192
   8
   :
```

Najlepszym dwucyfrowym wynikiem będzie liczba 0,71. Jeśli teraz zastosujemy metodę dzielenia z zaokrągleniem, to zamiast dzielić liczbę 1,7 będziemy dzielić liczbę 1,712 przerywając dzielenie po otrzymaniu drugiej cyfry wyniku:

```
0,71
1,712   :2,4
168
 32
 24
```

Jak widać z powyższych przykładów, metoda dzielenia z zaokrągleniem pozwala na podanie k-cyfrowego przybliżenia wyniku z zaokrągleniem bez obliczenia $k + 1$ cyfry wyniku. Uzasadnienie takiego postępowania jest niesłychanie proste; przeprowadzimy je dla przypadku dzielenia dwu liczb dodatnich mniejszych od jedności l_1, l_2. Niech obie te liczby mają k-cyfrowe rozwinięcia przy podstawie g, wówczas możemy je przedstawić w postaci.

$$l_1 = \sum_{i=1}^{k} a_i g^{-i}, \qquad l_2 = \sum_{i=1}^{k} b_i g^{-i}. \tag{5-3a}$$

Jeśli chcemy otrzymać k-cyfrowe przybliżenie wyniku dzielenia liczb l_1 i l_2, musimy do $k + 1$ cyfry wyniku dodać $\frac{1}{2}g$

$$\frac{\sum_{i=1}^{k} a_i g^{-i}}{\sum_{i=1}^{k} b_i g^{-i}} + \frac{g}{2} \cdot g^{-k-1}. \tag{5-3b}$$

Poniższe wyrażenie równoznaczne jest z wyrażeniem (5-3b)

$$\frac{\sum_{i=1}^{k} a_i g^{-i}}{\sum_{i=1}^{k} b_i g^{-i}} + \frac{1}{2} g^{-k}. \tag{5-3c}$$

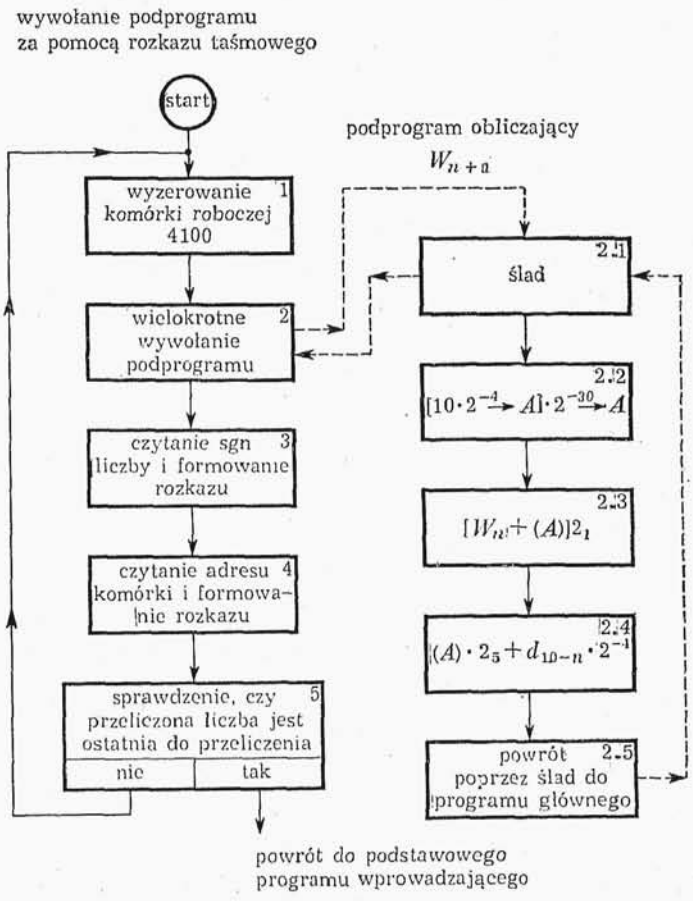
Po sprowadzeniu wyrażenia (5-3c) do wspólnego mianownika otrzymamy zgodnie

Tablica 5-10

Stałoprzecinkowy program wprowadzająco-przeliczający

Operator lub predykat	Kolejny adres	Kolejny rozkaz		Objaśnienia
1	$k+0000$	12	0077	położenie $W_0 = 0$
	1	14	4100	
2	2	04	$k+0036$	pierwsze wywołanie podprogramu oblicz. W_{n+1}
	3	04	$k+0036$	drugie wywołanie, jak wyżej
	4	04	$k+0036$	trzecie wywołanie, jak wyżej
	5	04	$k+0036$	czwarte wywołanie, jak wyżej
	6	04	$k+0036$	piąte wywołanie, jak wyżej
	7	04	$k+0036$	szóste wywołanie, jak wyżej
	$k+0010$	04	$k+0036$	siódme wywołanie, jak wyżej
	1	04	$k+0036$	ósme wywołanie, jak wyżej
	2	04	$k+0036$	dziewiąte wywołanie, jak wyżej
	3	04	$k+0036$	dziesiąte wywołanie, jak wyżej
3	4	12	$k+0047$	sformowanie znaku liczby przeliczonej
	5	24	0000	poprzez sformowanie rozkazu przesyła-
	6	14	$k+0031$	nia
4	7	24	0000	
	$k+0020$	23	0002	
	1	24	0000	
	2	23	0002	czytanie adresu, pod który mamy przesłać
	3	24	0000	wynik
	4	23	0002	
	5	24	0000	
	6	22	0003	
	7	11	0007	sformowanie rozkazu przesyłającego wy-
	$k+0030$	14	$k+0032$	nik przeliczenia pod zadany adres
	1	77	4100	miejsce na rozkaz przesyłania formujący znak liczby
	2	14	7777	miejsce na rozkaz przesyłania wyniku pod zadany adres
5	3	24	0000	sprawdzenie, czy przeliczona liczba jest ostatnią z liczb przeznaczonych do prze-
	4	03	$k+0000$	liczenia, a jeśli tak, to powrót do podsta-
	5	02	0000	wowego programu wprowadzającego
2.1	6	77	7777	miejsce na łącznik
2.2	7	12	0065	
	$k+0040$	22	0036	formowanie wyrażenia
2.3	1	10	4100	$W_n \cdot 2^{-4} + d_{10-n} \cdot 2^{-4} + 10 \cdot 2^{-38}$
	2	23	0001	
2.4	3	24	0000	
2.5	4	17	0065	obliczenie W_{n+1}
	5	14	4100	$W_{n+1} \rightarrow W_n$
2.6	6	01	$k+0036$	powrót do programu głównego parametr
	7	20	4000	

Podprogram realizujący algorytm $W_{n+1} = (W_n \cdot 2^{-4} + d_{10-n} \cdot 2^{-4} + 10 \cdot 2^{-38}) : (2^{-4} \cdot 10)$



Rys. 5-2. Schemat blokowy stałoprzecinkowego programu wprowadzająco-przeliczającego

z poprzednio podaną metodą dzielenia z zaokrągleniem:

$$\frac{\sum_{i=1}^{k} a_i g^{-i} + \frac{1}{2}\sum_{i=1}^{k} b_i g^{-k-i}}{\sum_{i=1}^{k} b_i g^{-i}}. \tag{5-3d}$$

Wzór (5-2) możemy przedstawić jako następującą formułę rekurencyjną:

$$W_0 = 0, \quad W_{n+1} = (W_n^* + d_{10-n}) : 10, \tag{5-4a}$$

gdzie $n = 1,2,\ldots,9$.

Ponieważ w maszynie nie dysponujemy stałoprzecinkową liczbą 10, tylko liczbą $2^{-4} \cdot 10$, musimy jeszcze pomnożyć licznik i mianownik wzoru (5-4a) przez 2^{-4} oraz powiększyć licznik o połowę mianownika pomnożonego przez 2^{-33} (zaokrąglenie dzielenia), ostatecznie otrzymamy wzór w postaci:

$$W_0 = 0, \quad W_{n+1} = (W_n \cdot 2^{-4} + d_{10-n} \cdot 2^{-4} + 10 \cdot 2^{-38}) : 2^{-4} \cdot 10, \tag{5-4b}$$

gdzie $n = 1,2,\ldots,9$.

Musimy jeszcze oszacować błąd maksymalny (definicja błędu maksymalnego podana jest przez J. Łukaszewicza i M. Warmusa — Bibliografia [12]), jaki popełniamy przeliczając liczby stałoprzecinkowe dziesiętne na liczby stałoprzecinkowe binarne. Na powstanie błędu przy obliczeniu W_{n+1} składają się dwa czynniki: błąd, z jakim obliczyliśmy W_n, oraz błąd zaokrąglenia przy obliczeniu W_{n+1}. Odejmując stronami od wyrażenia obarczonego błędami wyrażenia dokładne otrzymamy następujący związek między błędem maksymalnym w $(n+1)$-kroku Δ_{n+1} a błędem maksymalnym w n-kroku Δ_n i błędem zaokrąglenia $\pm 2^{-34}$, przy założeniu, że $\Delta_0 = 0$:

$$\Delta_{n+1} = \frac{1}{10}\Delta_n + 2^{-34}. \tag{5-5a}$$

Korzystając z wzoru (5-5a) możemy z łatwością oszacować błąd maksymalny Δ wyniku przeliczenia przy użyciu wzoru (5-4b):

$$\Delta = \Delta_{10} < \frac{10}{9} \cdot 2^{-34}. \tag{5-5b}$$

Jak widać z powyższego oszacowania, błąd przeliczenia jest niewielki i przeliczenie liczb dziesięciocyfrowych z ostatnią cyfrą parzystą (tzw. dziewięć i pół cyfr znaczących) w systemie dziesiętnym na system binarny jest uzasadnione, ponieważ nie prowadzi do większych strat dokładności.

Schemat blokowy stałoprzecinkowego programu przeliczająco-wprowadzającego jest przedstawiony na rys. 5-2. W tablicy 5-10 podane są kolejne rozkazy programu.

5.1.3. Zmiennoprzecinkowy program wprowadzająco-przeliczający. Podobnie jak stałoprzecinkowy program wprowadzająco-przeliczający, program ten działa wolno, ponadto wymaga on programu działań zmiennego przecinka, dokładniej podprogramu sformowania liczby zmiennoprzecinkowej.

Zmiennoprzecinkowy program wprowadzająco-przeliczający służy do wprowadzania do maszyny liczby postaci:

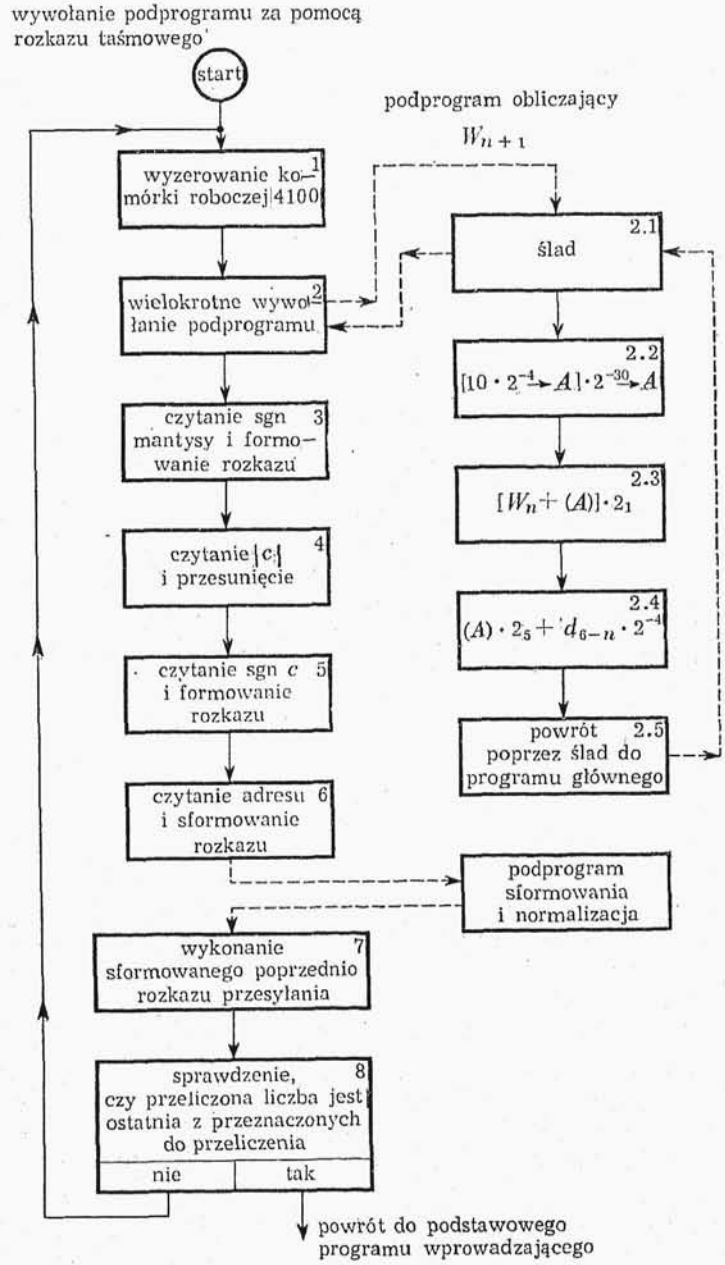
$$10^c \cdot m, \tag{5-6}$$

gdzie $-9 \leqslant c \leqslant 9$, mantysa zaś jest liczbą dziesiętną ośmiocyfrową co do modułu mniejszą od jedności. Dla zapisania jednej liczby postaci (5-6) potrzebujemy 16 rządków na taśmie perforowanej z podziałem na następujące grupy:

1 rządek + 4 rządki + 1 rządek + 1 rządek + 1 rządek + 8 rządków.

Podobnie jak w stałoprzecinkowym programie wprowadzająco-przeliczającym pierwszy rządek przeznaczony jest na zapis symbolu końca przeliczania, dalsze cztery rządki przeznaczone są na adres (zapisany w kodzie ósemkowym), pod który ma być przesłana odczytana liczba. Następnie jeden rządek przeznaczony jest na znak cechy i jeden na moduł cechy, dalej jeden znak przeznaczony jest na znak mantysy i pozostałe osiem rządków na moduł mantysy.

Algorytm przeliczenia mantysy, z którego korzysta zmiennoprzecinkowy program wprowadzający, jest analogiczny od algorytmu przedstawionego w punkcie 5.1.2. Cecha dziesiętna liczby postaci (5-6) jest przeliczana metodą słownika; szczegółową budowę takiego słownika omówimy w punkcie 6.1. Schemat blokowy programu jest



Rys. 5-3. Schemat blokowy zmiennoprzecinkowego programu wprowadzająco-przeliczającego

przedstawiony na rys. 5-3. Zmiennoprzecinkowy program wprowadzająco-przeliczający jest wywoływany rozkazami taśmowymi, a po zakończeniu przeliczenia sekwencji liczb postaci (5-6), przekazuje sterowanie podstawowemu programowi wprowadzającemu.

5.1.4. Program ustawiający standartowe podprogramy. Duże ułatwienie przy składaniu i przeadresowywaniu programów daje umieszczenie podprogramów funkcji elementarnych w stałych miejscach pamięci. Wadą tego rodzaju rozwiązania jest nieekonomiczne wykorzystanie pamięci, wynikające z tego, że nie zawsze wszystkie podprogramy funkcji elementarnych są wykorzystane przy obliczeniach. Pewnym kompromisem jest wywołanie podprogramów przez tzw. słownik, w którym adresy wywoływania poszczególnych podprogramów są ustalone raz na zawsze. Zakładając, że przy obliczeniach dysponujemy co najwyżej 26 podprogramami, możemy przyjąć, że słownik mieści się na ścieżce nr 37. Na każdy podprogram w słowniku wykorzystujemy dwa słowa krótkie.

Dla podprogramu nr 0 (np. $\sqrt{x}$) wykorzystujemy adresy 3714, 3715, dla podprogramu nr 1 (np $\sqrt[3]{x}$) wykorzystujemy adresy 3716, 3717 itd. Dla podprogramu nr 25 wykorzystujemy adresy 3776 i 3777. Pod pierwszym z adresów przeznaczonych w słowniku na wywołanie podprogramu zostawimy miejsce na ślad, pod drugim rozkaz 02 do pierwszego rozkazu podprogramu. Powrót po wykonaniu podprogramu odbywa się przez ślad zostawiony przez program główny w słowniku.

Przy założeniu, że programy składają się z parzystej ilości słów krótkich oraz że pierwszy rozkaz podprogramu ma adres parzysty, można za pomocą prostego programu ustawić podprogramy w pamięci jeden za drugim, tak że wszystkie miejsca pamięci są wykorzystane. Dla wywołania programu „ustawiającego" umieszczamy na końcu podprogramu (złożonego wyłącznie z rozkazów i pseudorozkazów) grupę rozkazów taśmowych:

1) 00 Numer podprogramu (od 0000' do 0030'),
2) M2 0127,
3) 00 Ilość słów krótkich (od 0000' do 3777'),
4) M2 0117.

Tablica 5-11

Program ustawiający podprogramy

	00	0000						
	M5	0117						
0117	14	0103	0130	14	0132	0140	02	0000
0120	12	0144	1	12	0103	1	11	0144
1	11	0103	2	14	7777	2	03	0000
2	10	0147	3	12	0144	3	05	0144
3	14	0116	4	14	0101	4	14	3713
4	10	0140	5	12	0103	5	11	7777
5	14	0103	6	10	0145	6	14	3777
6	02	0000	7	14	0144	7	24	0001
7	10	0146					S4	0147

Podprogramy zamknięte wprowadzamy kolejno jeden za drugim (kolejność dowolna).

Kolejne rozkazy programu ustawiającego, łącznie z rozkazami taśmowymi podane są w tabl. 5-11.

Ponadto program ustawiający umożliwia programiście autokontrolę, czy przypadkiem rozkazy programu głównego nie są wprowadzane na miejsca, na które poprzednio wprowadziliśmy podprogramy. W tym celu na końcu programu głównego należy umieścić grupę rozkazów taśmowych:

1) S4 g,
2) 14 $g-1$,
3) M2 0141,

gdzie g oznacza najwyższy adres zajęty przez program główny. W przypadku pomyłki w podziale pamięci program główny nie zostanie wprowadzony do maszyny.

5.2. PROGRAMY WYPROWADZAJĄCE

Wyprowadzanie wyników obliczeń oraz programów zapisanych w pamięci dokonujemy za pomocą programów wyprowadzających. Przy normalnej eksploatacji maszyny musimy dysponować co najmniej czterema programami wyprowadzającymi:

1) programem do wyprowadzania zawartości kolejnych miejsc pamięci w postaci rozkazów i pseudorozkazów,
2) programem wyprowadzająco-przeliczającym dla liczb całkowitych,
3) stałoprzecinkowym programem wyprowadzająco-przeliczającym,
4) zmiennoprzecinkowym programem wyprowadzająco-przeliczającym.

W odróżnieniu od programów wprowadzających, programy wyprowadzające mają prostą strukturę. Ograniczymy się do omówienia tylko dwóch programów wyprowadzających, a mianowicie stałoprzecinkowego programu wyprowadzająco-przeliczającego (bardzo uproszczonego) i programu dla wyprowadzania zawartości kolejnych miejsc pamięci w postaci rozkazów i pseudorozkazów.

Stałoprzecinkowy program wyprowadzająco-przeliczający łącznie z podstawowym programem wprowadzającym i parametrami jest nagrany na stałe na ścieżce nr zero. Program ten służy do drukowania liczb stałoprzecinkowych co do modułu mniejszych od jedności w układzie dziesiętnym, ponadto służy do drukowania odstępów potrójnych i napisów trzyznakowych.

Stałoprzecinkowy program wyprowadzająco-przeliczający zastosowany do liczby x ($|x|<1$), drukuje symbol $+0$, lub -0, w zależności od znaku liczby x oraz 12 cyfr dziesiętnych liczby x. Cyfry te uzyskuje się przez wielokrotne mnożenie x przez $2^{-4} \cdot 10$, drukowanie cyfry, która znalazła się na miejscu charakterystycznym akumulatora oraz arytmetyczne przesuwanie wyniku o 4 miejsca w lewo (czyli mnożenie wyniku przez 2^4). Po wydrukowaniu liczby program drukuje potrójny odstęp. Odstęp taki można również drukować bez drukowania jakiejkolwiek liczby.

Uwaga: Napis -0, tworzymy przez dodanie do napisu $+0$, liczby ósemkowej