55. Dalsze przykłady programów

551. Mnożenie macierzy. Jednym z często spotykanych fragmentów obliczeń jest mnożenie macierzy. Niechaj mamy macierz A o p wierszach i q kolumnach (tabl. 5-10) oraz macierz B o q wierszach i r kolumnach (tabl. 5-11). Jak wiadomo, pod iloczynem macierzy A i B rozumiemy macierz C o p wierszach i r kolumnach (tabl. 5-12), której każdy wyraz c_{ik} jest określony wzorem

$$c_{ik} = \sum_{j=1}^{q} a_{ij}\, b_{jk} .$$

Tablica 5-10

Oznaczenia elementów a_{ij} macierzy A

i \ j	1	2	...	q
1	a_{11}	a_{12}	...	a_{1q}
2	a_{21}	a_{22}	...	a_{2q}
...	...	...	...	...
p	a_{p1}	a_{p2}	...	a_{pq}

Tablica 5-11

Oznaczenia elementów b_{jk} macierzy B

j \ k	1	2	...	r
1	b_{11}	b_{12}	...	b_{1r}
2	b_{21}	b_{22}	...	b_{2r}
...	...	...	...	...
q	b_{q1}	b_{q2}	...	b_{qr}

Tablica 5-12

Oznaczenia elementów c_{ik} macierzy C

i \ k	1	2	...	r
1	c_{11}	c_{12}	...	c_{1r}
2	c_{21}	c_{22}	...	c_{2r}
...	...	...	...	...
p	c_{p1}	c_{p2}	...	c_{pr}

Dla ułożenia programu obliczenia macierzy C założymy, że żadna z trzech macierzy nie zawiera więcej niż 100 elementów. Dla elementów macierzy A przeznaczymy miejsca w pamięci od 101 do 200, dla elementów macierzy B - miejsca od 201 do 300, wreszcie dla elementów macierzy C - miejsca od 301 do 400.

Pierwszy element pierwszego wiersza macierzy A, tzn. a_{11}, umieścimy w miejscu pamięciowym 101, następny element tego samego wiersza - 102 itd., aż do wyczerpania elementów pierwszego wiersza. Bezpośrednio po tym lokujemy elementy drugiego wiersza i w podobny sposób elementy dalszych wierszy. W konsekwencji element a_{ij} znajdzie się w miejscu pamięciowym o numerze $100+q(i-1)+j$.

Elementy macierzy B ulokujemy kolumnami, mianowicie pierwszy element pierwszej kolumny, tzn. b_{11}, umieścimy w miescu pamięciowym 201, następny element tej samej kolumny w 202 itd., aż do wyczerpania kolumny. Bezpośrednio po tym lokujemy elementy drugiej kolumny i następnych. W konsekwencji element b_{jk} znajdzie się w miejscu pamięciowym $200+(k-1)q+j$.

Obliczane elementy macierzy C będziemy lokowali w miejscach pamięciowych poczynając od numeru 301 w sposób analogiczny do lokowania elementów macierzy A tak, że element c_{ik} znajdzie się w miejscu pamięciowym o numerze $300+(i-1)r+k$ (tabl. 5-13).

Tablica 5-13

Lokata w pamięci elementów macierzy A, B i C

Macierz A		Macierz B		Macierz C	
N \ Element		N \ Element		N \ Element	
101	a_{11}	201	b_{11}	301	c_{11}
102	a_{12}	202	b_{21}	302	c_{12}
...	...	...	...	...	...
100+q	a_{1q}	200+q	b_{q1}	300+r	c_{1r}
100+q+1	a_{21}	200+q+1	b_{12}	300+r+1	c_{21}
...	...	...	...	...	...
100+2q	a_{2q}	200+2q	b_{q2}	300+2r	c_{2r}
...	...	...	...	...	...
100+(i-1)q+j	a_{ij}	200+(k-1)q+j	b_{jk}	300+(i-1)r+k	c_{ik}

Obliczenie przeprowadzimy metodą sum cząstkowych, której program zawiera tabl. 53-1. Ponieważ składniki nie są tu znane, dodawanie musi być poprzedzane obliczeniem iloczynów $a_{ij}\,b_{jk}$. Kolejne sumy będziemy oznaczali symbolem S_{ijk}.

$$S_{ijk} = \sum_{h=1}^{j} a_{ih}\,b_{hk} .$$

Z tego wynika, że

$$c_{ik} = S_{iqk} .$$

Program składa się z dziewięciu elementów struktury. Element 01 zawiera rozkazy przygotowawcze ogólne, element 02 zawiera rozkazy przygotowawcze dotyczące obliczania wyrazów leżących w określonym wierszu macierzy C, element 03 zawiera rozkazy przygotowawcze dotyczące obliczania poszczególnych wyrazów macierzy C, element 04 wreszcie zawiera rozkazy obliczania iloczynów $a_{ij}\,b_{jk}$ oraz sum S_{ijk}. Za każdym przebiegiem elementu 04 liczba składników S_{ijk} wzrasta o 1 i z chwilą, gdy osiąga ona wartość q, element 05/04 kieruje przebieg do elementu rozgałęźnego 07/03 poprzez element 06, którego zadaniem jest lokowanie w pamięci elementów c_{ik}.

Tablica 5-14

Program obliczania iloczynu dwóch macierzy

ES	N	Rozkaz	Wynik operacji
01	001	C 000	Ustawienie sumatora na zero
	002	B 051	Ustawienie sumatora na -p
02	003	A 051	Obliczenie i-1
	004	A 050	Obliczenie i
	005	C 061	Przeniesienie i do pamięci
	006	B 053	Ustawienie sumatora na -r
03	007	A 053	Obliczenie k-1
	008	A 050	Obliczenie k
	009	C 063	Przeniesienie k do pamięci
	010	C 060	Położenie $S_{ijk} = 0$
	011	B 052	Ustawienie sumatora na -q
04	012	A 052	Obliczenie j-1
	013	A 050	Obliczenie j
	014	C 062	Przeniesienie j do pamięci
	015	A 022	Przeniesienie rozkazu D 100+ +(i-1)q+j-1 do sumatora
	016	A 050	Obliczenie rozkazu D 100+ +(i-1)q+j
	017	C 022	Przeniesienie rozkazu D 100+ +(i-1)q+j do pamięci
	018	A 023	Przeniesienie rozkazu E 200+ +(k-1)q+j-1 do sumatora
	019	A 050	Obliczenie rozkazu E 200+ +(k-1)q+j
	020	C 023	Przeniesienie rozkazu E 200+ +(k-1)q+j
	021	A 060	Przeniesienie $S_{i,j-1,k}$ do sumatora
	022	D 100 +(i-1)q+j	Przeniesienie a_{ij} do rejestru mnożenia

Tablica 5-14 (c.d)

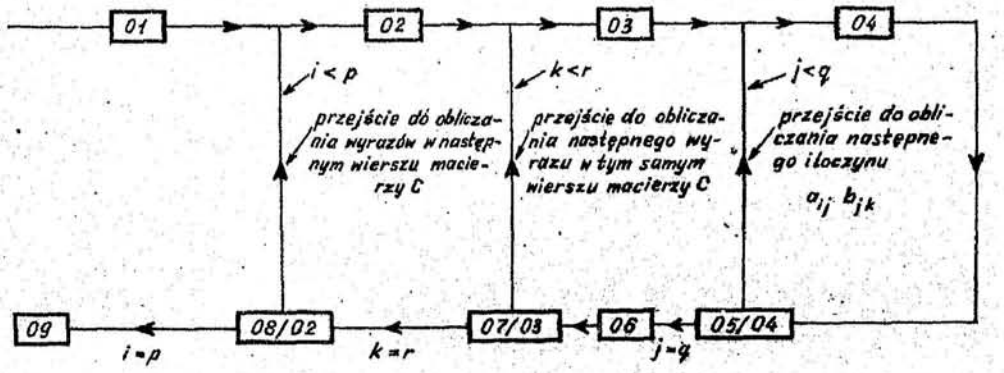
RS	N	Rozkaz	Wynik operacji
	023	E 200 +(k-1)q+j	Obliczenie S_{ijk}
	024	C 060	Przeniesienie S_{ijk} do pamięci
05/04	025	A 062	Przeniesienie j do sumatora
	026	B 052	Obliczenie j-q
	027	X 012	Powrót do rozkazu 012, jeśli j < q
06	028	C 000	Ustawienie sumatora na zero
	029	A 033	Przeniesienie rozkazu C 300+ +(i-1)r+k-1 do sumatora
	030	A 050	Obliczenie rozkazu C 300+ +(i-1)r+k
	031	C 033	Przeniesienie rozkazu C 300+ +(i-1)r+k do pamięci
	032	A 300	Przeniesienie c_{ik} do sumatora
	033	C 300 +(i-1)r+k	Przeniesienie c_{ik} do pamięci
07/03	034	A 063	Przeniesienie k do sumatora
	035	B 053	Obliczenie k-r
	036	X 007	Powrót do rozkazu 007, jeśli k<r
08/02	037	C 000	Ustawienie sumatora na zero
	038	A 061	Przeniesienie i do sumatora
	039	B 051	Obliczenie i-p
	040	X 003	Powrót do rozkazu 003, jeśli i<p
09	041	Z	Zatrzymanie maszyny

Tablica 5-14 c.d)

Lokata danych i wyników obliczeń z wyjątkiem elementów macierzy

N	Wielkości stałe	N	Wielkości zmienne
050	1	060	S_{ijk}
051	p	061	i
052	q	062	j
053	r	063	k

Element 07/03 kontroluje, czy ostatnio obliczony wyraz c_{ik} jest ostatnim w wierszu macierzy C, czy nie. Jeśli obliczony wyraz nie jest ostatni, to przebieg kieruje się do elementu 03, który powiększa wskaźnik k o jedność. Jeśli obliczony wyraz jest ostatni w wierszu, to przebieg kieruje się do elementu rozgałęźnego 08/02, który kontroluje, czy ukończony wiersz macierzy C jest ostatni. Jeśli wiersz nie jest ostatni, to przebieg wraca do elementu 02, który powiększa o jedność wartość wskaźnika i.



Rys. 5-8. Struktura programu obliczania iloczynu dwóch macierzy

Program tego obliczenia podany jest w tabl. 5-14, struktura programu - na rys. 5-8.

552. Wyróżnik rozkazu warunkowego przy użyciu metod kolejnych przybliżeń. W rozpatrzonych dotychczas programach wyróżnikiem rozkazu warunkowego

była zawsze różnica między rzeczywiście wykonaną liczbą przebiegów poprzez pewien element struktury a z góry znaną liczbą potrzebnych przebiegów. W wielu jednakże obliczeniach używamy takich metod, przy użyciu których liczba potrzebnych przebiegów nie jest znana podczas programowania i musi być wyznaczona przez maszynę. Do metod takich należy przede wszystkim obliczanie wartości funkcji danej przez rozwinięcie na szereg oraz obliczanie wartości funkcji określonej wzorem rekurencyjnym. Obliczana wartość funkcji może być liczbą niewymierną lub też liczbą wymierną, do przedstawienia której trzeba użyć większej liczby miejsc znakowych, aniżeli jest ich w maszynie. W obu przypadkach musimy zadowolić się wartością przybliżoną. W programie obliczenia musi wówczas figurować pewna stała charakteryzująca stopień żądanego przybliżenia. Stałą tę będziemy oznaczali literą ε, kolejne zaś przybliżenia obliczanej wartości przez a_1, $a_2, \ldots, a_n, a_{n+1}$ itd. Wówczas warunek osiągnięcia żądanego przybliżenia wyrazi się w postaci nierówności

$$|a_n - a_{n+1}| < \varepsilon ,$$

z czego wynika, że wyróżnikiem rozkazu warunkowego będzie

$$\varepsilon - |a_n - a_{n+1}|$$

Jeśli wiadome jest, że ciąg kolejnych przybliżeń jest monotoniczny, określanie wartości bezwzględnej można opuścić.

553. Obliczanie odwrotności. W niektórych maszynach nie ma operacji dzielenia, wobec czego zamiast dzielenia wykonuje się mnożenie przez odwrotność. Obliczanie odwrotności odbywa się często na podstawie wzoru iteracyjnego. Mianowicie, jeśli

$$y_{n+1} = y_n (2 - x y_n), \qquad (5.1)$$

to $y_n \to 1/x$, gdy $n \to \infty$. Wartość y_1, czyli pierwszego przybliżenia odwrotności musi być przy tej metodzie dana z góry. Można to osiągnąć np. przez umieszczenie przybliżonej tablicy odwrotności w pamięci, przy tym może to być pamięć o stosunkowo dużym czasie oczekiwania. Dokładność tego pierwszego przybliżenia może być

niewielka. W zakresie dodatnich wartości x musi być tylko spełniona nierówność

$$0 < y_1 < 2/x. \tag{5.2}$$

Ciąg kolejnych przybliżeń poczynając najdalej od drugiego wyrazu jest monotoniczny i rosnący. Dowiedziemy najpierw, że, jeśli którykolwiek wyraz ciągu jest mniejszy od odwrotności, to wszystkie następne wyrazy są od niego większe, tzn., jeśli

$$xy_n < 1, \tag{5.3}$$

to

$$y_{n+1} > y_n . \tag{5.4}$$

Rzeczywiście, wprowadzając dla ułatwienia oznaczenie

$$z = 1-xy_n$$

będziemy mogli napisać

$$y_n = \frac{1-z}{x} \tag{5.5}$$

oraz

$$y_{n+1} = \frac{1-z^2}{x}. \tag{5.6}$$

Jeśli zachodzi (5.3), to $z < 1$ i prawdziwość (5.4) wynika wprost z porównania prawych stron (5.5) i (5.6). Zatem ciąg wyrazów y_n jest monotoniczny i rosnący, jeśli $xy_1 < 1$.

Załóżmy obecnie, że $xy_1 > 1$. Jeśli zachowany będzie poza tym warunek (5.2), to i w tym przypadku $z < 1$ i z (5.6) wynika, że $xy_2 < 1$, a tym samym dla każdego n równego co najmniej dwóm zachodzi (5.4). Z tego wynika, że wyróżnikiem rozkazu warunkowego jest wielkość

$$\varepsilon - (y_{n+1} - y_n)$$

Program obliczania odwrotności na podstawie wzoru (5.1) podaje tabl. 5-15. Struktura tego programu jest taka sama, jak programu dodawania metodą sum cząstkowych, podanego w tablicy 5-5.

Tablica 5-15

Program obliczania wartości funkcji $y = 1/x$ na podstawie wzoru iteracyjnego $y_{n+1} = y_n (2-xy_n)$

ES	N	Rozkaz	Wynik operacji
01	001	C 000	Ustawienie sumatora na zero
	002	A 025	Przeniesienie y_n do sumatora
	003	C 024	Przeniesienie y_n do pamięci
	004	D 023	Przeniesienie x do rejestru mnożenia
	005	E 024	Obliczenie xy_n
	006	C 000	Przeniesienie xy_n do pamięci
	007	A 021	Przeniesienie 2 do sumatora
	008	B 000	Utworzenie różnicy $2-xy_n$
	009	C 000	Przeniesienie $2-xy_n$ do pamięci
	010	D 000	Przeniesienie $2-xy_n$ do rejestru mnożenia
	011	E 024	Obliczenie y_{n+1}
	012	C 025	Przeniesienie y_{n+1} do pamięci
02/01	013	A 022	Przeniesienie ε do sumatora
	014	A 024	Obliczenie $\varepsilon + y_n$
	015	B 025	Obliczenie $\varepsilon - (y_{n+1} - y)$
	016	X 001	Powrót do rozkazu 001, jeśli $y_{n+1} - y_n > \varepsilon$
03	017	Z	Zatrzymanie maszyny

Lokata danych i wyników obliczeń

N	Wielkości stałe	N	Wielkości zmienne
021	2	024	y_n
022	ε	025	y_{n+1} , na początku y_1
023	x		

554. Wyciąganie pierwiastka kwadratowego. Niewiele tylko maszyn zawiera operację wyciągania pierwiastka kwadratowego, wobec czego operacja ta musi być wykonywana na podstawie odpowiednio skonstruowanego programu. Również w tym przypadku dogodne są wzory iteracyjne. Jeśli mianowicie $q > 1$, zaś

$$p_{j+1} = \frac{1}{2}(p_j + q/p_j) \tag{5.7}$$

lub

$$p_{j+1} = \frac{1}{2}(3p_j - p_j^3/q), \tag{5.8}$$

to

$$p_{j+1} \longrightarrow \sqrt{q}, \text{ gdy } j \longrightarrow \infty .$$

Ciągi określone przez każdy z tych wzorów są b. szybko zbieżne. Jeśli $q > 0$, to ciąg wyrazów p_j określonych wzorem (5.7) jest monotoniczny i malejący poczynając najdalej od drugiego wyrazu. Aby dowieść tego zauważmy, że dla dowolnych wartości p_j i q mamy zawsze

$$(p_j^2 - q)^2 > 0. \tag{5.9}$$

Zastosujmy obecnie następujące przekształcenia

$$(p_j^2-q)^2 = p_j^4+q^2 - 2p_j^2 q = 4p_j^2\left[\left(\frac{p_j^2+q}{2p_j}\right)^2 - q\right] =$$
$$= 4p_j^2\left\{\left[\frac{1}{2}(p_j+q/p_j)\right]^2 - q\right\}.$$

Na zasadzie (5.7) wielkość w nawiasie kwadratowym jest to p_{j+1}. W połączeniu z (5.9) z tego wynika,

$$p_{j+1}^2 - q > 0. \tag{5.10}$$

Nierówność ta jest ważna dla każdej wartości j, zatem ważna jest już poczynając od drugiego wyrazu. Pierwszy wyraz możemy obrać dowolnie.

Rozpatrzmy obecnie interesującą nas różnicę $p_{j+1} - p_j$. Mamy ogólnie

$$p_{j+1} - p_j = \frac{1}{2}(p_j + q/p_j) - p_j = \frac{1}{2}(q/p_j - p_j) = \frac{q-p_j^2}{2p_j}.$$

Ponieważ w myśl (5.9) licznik ostatniego ułamka jest ujemny, dla każdego j równego co najmniej dwóm spełniona jest nierówność

$$p_{j+1} - p_j < 0,$$

czyli ciąg jest monotoniczny i malejący poczynając od drugiego wyrazu. Z tego wynika, że wyróżnikiem rozkazu warunkowego jest w danym przypadku wielkość

$$\varepsilon - (p_j - p_{j+1}).$$

Program obliczenia pierwiastka kwadratowego podług wzoru (5.7) jest podany w tabl. 5-16. Struktura tego programu jest identyczna ze strukturą programu obliczania odwrotności. Na p_1 można przyjąć dowolną liczbę dodatnią. Zwłaszcza wygodne jest położenie $p_1 = 1$ lub $p_1 = q$.

T a b l i c a 5-16

Program obliczania wartości funkcji $p = \sqrt{q}$ na podstawie wzoru iteracyjnego $p_{n+1} = \frac{1}{2}(p_n + q/p_n)$

ES	N	Rozkaz	Wynik operacji
.01	001	C 000	Ustawienie sumatora na zero
	002	A 020	Przeniesienie p_n do sumatora
	003	C 019	Przeniesienie p_n do pamięci
	004	A 016	Przeniesienie q do sumatora
	005	D 019	Przeniesienie p_n do rejestru mnożenia
	006	E 019	Obliczenie $p_n^2 + q$
	007	G 019	Obliczenie $(p_n^2 + q)/p_n$
	008	E 017	Obliczenie p_{n+1}
	009	C 020	Przeniesienie p_{n+1} do pamięci
02/01	010	A 018	Przeniesienie ε do sumatora
	011	A 020	Obliczenie $\varepsilon + p_{n+1}$
	012	B 019	Obliczenie $\varepsilon - (p_n - p_{n+1})$
	013	X 001	Powrót do rozkazu 001, jeśli $p_n - p_{n+1} > \varepsilon$
03	014	Z	Zatrzymanie maszyny

Tablica 5-16 (c.d.)

Lokata danych i wyników obliczeń

N	Wielkości stałe	N	Wielkości zmienne
016	q	019	p_n
017	1/2	020	p_{n+1} , na początku q
018	ε		

555. Obliczanie logarytmów naturalnych. Niechaj mamy ciąg wielkości a_k i ciąg wielkości b_k określonych wzorami

$$a_{k+1} = \frac{a_k + b_k}{2}, \tag{5-11}$$

$$b_{k+1} = \sqrt{a_{k+1}\, b_k}. \tag{5-12}$$

Wyrazy każdego z tych ciągów dążą do tej samej granicy y, tzn. $y = \lim a_{k+1} = \lim b_{k+1}$, gdy $k \to \infty$. Wartość tej granicy zależy od wartości a_0 i b_0. Jeśli

$$a_0 = 1 + x^2 \tag{5.13}$$

$$b_0 = 2x, \tag{5.14}$$

to

$$y = \frac{x^2-1}{\ln x}. \tag{5.15}$$

Wymienione wzory można wykorzystać do obliczania wartości logarytmów naturalnych. Przypuśćmy, że zadaniem naszym jest obliczenie tablicy tych logarytmów w zakresie od 1 do M, co h.

Wykorzystanie wzorów podanych na początku tego rozdziału będzie polegało na tym, że wzór (5.15) przepiszemy w postaci

$$\ln x = \frac{x^2-1}{y} \tag{5.16}$$

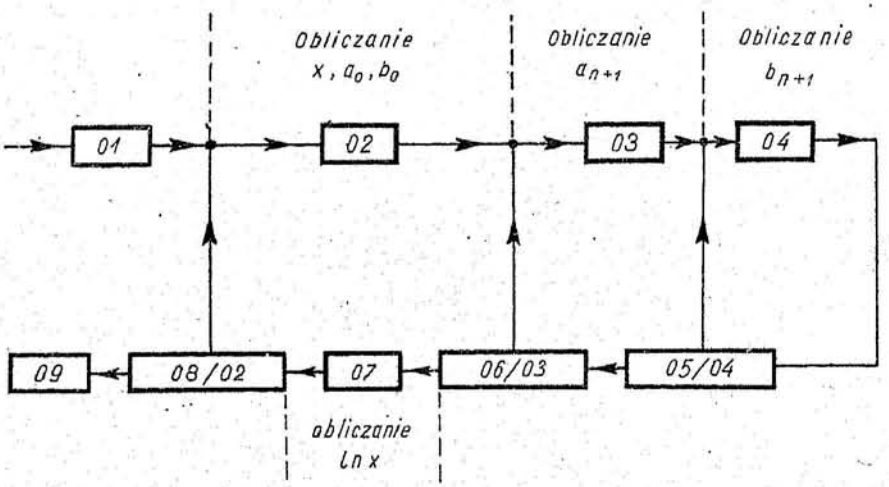
i będziemy najpierw obliczali przybliżoną wartość y na podstawie wzorów iteracyjnych (5.11) i (5.12), a następnie ln x z (5.16).

Wykres struktury programu obliczania logarytmów naturalnych podany jest na rys. 5-9. Ułożenie programu na podstawie wykresu struktury jest już rzeczą stosunkowo łatwą. Zbadania wymaga tylko wyróżnik rozkazu warunkowego zawartego w elemencie 06/03. Jako miarę dokładności obliczenia y przyjmiemy wartość bezwzględną różnicy $a_{k+1} - b_{k+1}$. Można dowieść, że różnica ta jest zawsze dodatnia. Stwierdzimy najpierw, że jeśli

$$a_k > b_k,$$

to

$$a_{k+1} > b_{k+1}.$$



Rys. 5-9. Struktura programu obliczania logarytmów naturalnych

Ponieważ wszystkie a i wszystkie b są dodatnie, wystarczy dowieść, że, jeśli

$$a_k^2 > b_k^2,$$

to

$$a_{k+1}^2 > b_{k+1}^2.$$

Rozpatrzmy wobec tego różnicę $a_{k+1}^2 - b_{k+1}^2$. Na podstawie wzorów (5-11) i (5-12) możemy napisać

$$a_{k+1}^2 - b_{k+1}^2 = \left(\frac{a_k+b_k}{2}\right)^2 - b_k \frac{a_k+b_k}{2} = \frac{a_k^2-b_k^2}{4},$$

z czego natychmiast jest widoczne, że, jeśli $a_k > b_k$, to $a_{k+1} > b_{k+1}$. Pozostaje nam jeszcze sprawdzić znak różnicy a_1-b_1. Ze wzorów (5.13) i (5.14) wynika

$$a_1-b_1 = 1 + x^2-2x = 1-x^2,$$

zatem $a_1-b_1 > 0$, dla każdego x. Wobec tego wyróżnikiem rozkazu warunkowego zawartego w elemencie struktury 06/03 jest wyrażenie

$$\varepsilon - (a_{k+1} - b_{k+1}).$$

Program obliczenia zawiera tabl. 5-17.

Tablica 5-17

Program obliczania logarytmów naturalnych w granicach od 1 do M, co h, na podstawie wzorów:
$\ln x = (x^2-1)/y$; $y = \lim a_{k+1} = \lim b_{k+1}$;
$a_{k+1} = (a_k+b_k)/2$; $b_{k+1} = \sqrt{a_{k+1} \cdot b_k}$;
$a_0 = 1 + x^2$; $b_0 = 2x$.

ES	N	Rozkaz	Wynik operacji
01	001	C 000	Ustawienie sumatora na zero
	002	A 041	Przeniesienie 1 do sumatora
	003	C 050	Położenie $x_0 = 1$
02	004	A 050	Przeniesienie x-h do sumatora
	005	A 043	Obliczenie x
	006	C 050	Przeniesienie x do pamięci
	007	A 041	Przeniesienie 1 do sumatora
	008	D 050	Przeniesienie x do rejestru mnożenia
	009	E 050	Obliczenie a_0
	010	C 051	Przeniesienie a_0 do pamięci
	011	D 050	Przeniesienie x do rejestru mnożenia
	012	E 042	Obliczenie b_0
	013	C 053	Przeniesienie b_0 do pamięci

Tablica 5-17 (c.d.)

ES	N	Rozkaz	Wynik operacji
03	014	A 051	Przeniesienie a_k do sumatora
	015	A 053	Obliczenie a_{k+1}
	016	C 052	Przeniesienie a_{k+1} do pamięci
	017	D 052	Przeniesienie a_{k+1} do rejestru mnożenia
	018	E 053	Obliczenie $b_k \cdot a_{k+1}$
04	019	C 000	Ustawienie sumatora na zero
	020	A 057	Przeniesienie p_{j+1} do sumatora
	021	C 056	Przeniesienie p_j do pamięci
	022	A 055	Przeniesienie $a_{k+1}\ b_k$ do sumatora
	023	D 056	Przeniesienie p_j do rejestru mnożenia
	024	E 057	Obliczenie $p_j^2 + a_{k+1}\ b_k$
	025	G 057	Obliczenie $(p_j^2 + a_{k+1}\ b_k)/p_j$
	026	E 040	Obliczenie p_{j+1}
	027	C 057	Przeniesienie p_{j+1} do pamięci
05/04	028	A 044	Przeniesienie ε do sumatora
	029	A 057	Obliczenie $\varepsilon + p_j$
	030	B 056	Obliczenie $\varepsilon - (p_{j+1} - p_j)$
	031	X 019	Powrót do rozkazu 019, jeśli $p_{j+1} - p_j > \varepsilon$
06/03	032	C 000	Ustawienie sumatora na zero
	033	A 044	Przeniesienie ε do sumatora
	034	A 054	Obliczenie $\varepsilon + b_{k+1}$
	035	B 052	Obliczenie $\varepsilon - (a_{k+1} - b_{k+1})$
	036	X 014	Powrót do rozkazu 014, jeśli $a_{k+1} - b_{k+1} > \varepsilon$
07	037	C 000	Ustawienie sumatora na zero
	038	B 041	Ustawienie sumatora na -1
	039	D 050	Przeniesienie x do rejestru mnożenia

Tablica 5-17 (c.d.)

ES	N	Rozkaz	Wynik operacji
	040	E 050	Obliczenie x^2-1
	041	G 052	Obliczenie ln x
	042	C 058	Przeniesienie ln x do pamięci
	043	W 058	Wydrukowanie wyniku
08/02	044	A 050	Przeniesienie x do sumatora
	045	B 045	Obliczenie różnicy x-M
	046	X 004	Powrót do rozkazu 004, jeśli x<M
09	047	Z	Zatrzymanie maszyny

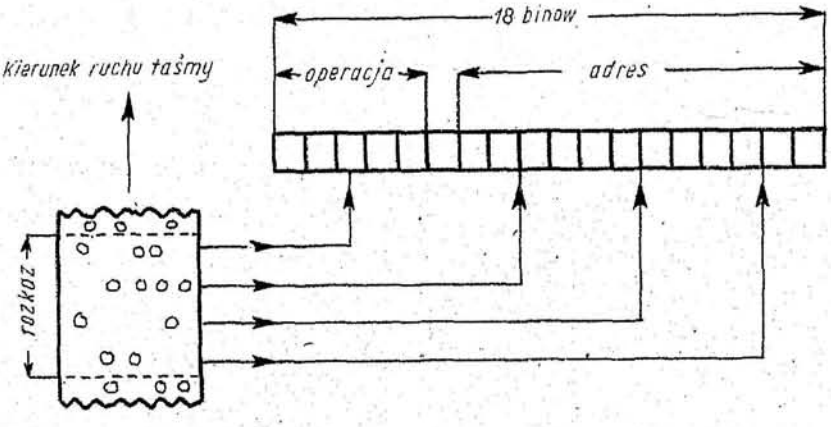
Lokata danych i wyników obliczeń

N	Wielkości stałe	N	Wielkości zmienne
040	1/2	050	x
041	1	051	a_k
042	2	052	a_{k+1}
043	h	053	b_k
044	ε	054	b_{k+1}
045	M	055	a_{k+1} b_k
		056	p_j
		057	p_{j+1}
		058	ln x

56. Program wstępny

Jedną z faz przygotowania maszyny do rozwiązania problemu obliczeniowego jest ręczne umieszczenie programu obliczeń w rejestrze wejściowym. Program ten należy przenieść do pamięci, ponieważ tylko z tego miejsca może on kierować bezpośrednio pracą maszyny. Akcją przenoszenia kieruje z kolei inny program, który będziemy nazywali wstępnym. Program wstępny jest w danej maszynie stały, tzn. jest on niezależny od programu obliczeń.

Czynności, które należy wykonać dla przeniesienia programu obliczeń z rejestru wejścia do rejestru pamięci, wynikają nie tylko z różnych zasad fizycznych rejestracji, lecz również z odmiennej struktury informacji a częściowo nawet z odmiennej ich zawartości. W rejestrze wejścia, stosującym przeważnie pięciorzędową taśmą papierową, każdy znak ma tylko 5 binów, kompletny zaś rozkaz składa się z kilkunastu do kilkudziesięciu binów. Dla sformułowania na taśmie jednego rozkazu trzeba użyć kilku znaków. Z informacji zawartych w tych znakach maszyna musi zbudować rozkaz. Do kierowania tą budową służy właśnie program wstępny. Przykładowy schemat przenoszenia znaków z taśmy do pamięci podany jest na rys. 5-10. W przykładzie tym założono dla uprosz-



Rys. 5-10. Przykład schematu przenoszenia rozkazów z taśmy dalekopisowej do pamięci

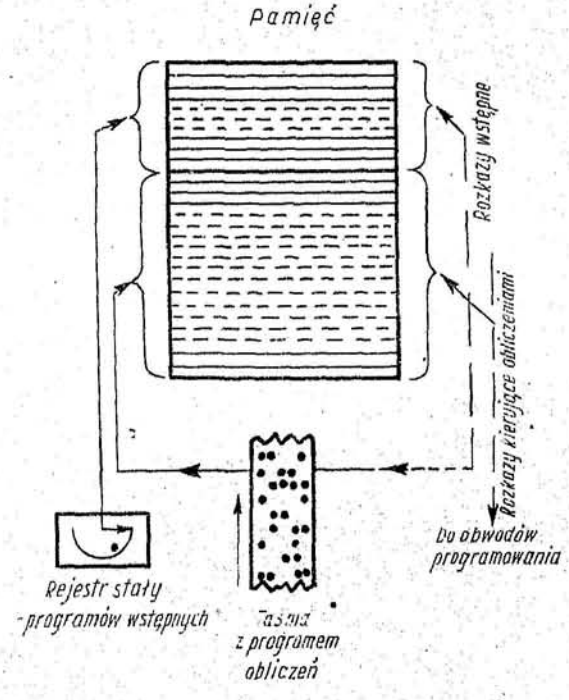
czenia, że miejsc liczbowych w rejestrze pamięci jest 1000 i że każdy znak dziesiętny adresu jest zakodowany w postaci czterech binów. Takie założenie nie jest, oczywiście, jedynym możliwym. Jeśli zapis w pamięci jest czysto dwójkowy, to poszczególne cyfry dziesiętne nie mają w ogóle indywidualnych odpowiedników i schemat z rys. 5-10 nie może być stosowany.

Nie jest rzeczą konieczną, aby każdy rozkaz na taśmie składał się z tej samej liczby znaków. Wzgląd zachowania możliwej prostoty procesu ręcznej perforacji taśmy prowadzi raczej do zmiennej liczby znaków na rozkaz. Powoduje to konieczność oddzielania w jakiejś umownej postaci grup znaków przedstawiających poszczególne rozkazy. Jako kryterium oddzielania może np. służyć znak sygnalizujący koniec

rozkazu. Znak ten sam przez się nie jest częścią rozkazu, a zatem nie powinien on być przekazywany do pamięci.

Jeśli odczytywanie rozkazów odbywa się tak, jak to założyliśmy poprzednio, w kolejności numeracji miejsc pamięciowych, to jest rzeczą nieodzowną umieszczanie programu wstępnego na początkowych miejscach pamięci. Program wstępny musi zawierać co najmniej jeden subprogram. Jest nim mianowicie wielokrotnie przebiegany ciąg rozkazów kierujących przeniesieniem do pamięci jednego rozkazu programu obliczeń.

Skoro program obliczeń jest umieszczany w pamięci pod kierunkiem programu wstępnego, program zaś wstępny, kierując tą pracą, sam się już znajduje w pamięci, to musi on być tam umieszczany jakąś inną drogą, nie poprzez urządzenia wejścia. Możliwe



Rys. 5-11. Schemat ułożenia programów w pamięci oraz obwodów kontrolnych

rozwiązanie polega na tym, że program wstępny jest umieszczany w pewnym rejestrze pomocniczym, w którym

jest zarejestrowany trwale przy użyciu takiego samego kodowania, jakie jest stosowane w rejestrach pamięci. Takim rejestrem pomocniczym może np. być zespół wybieraków obrotowych. W tym przypadku program wstępny realizuje się przez dołączenie styków pola wybieraków do potencjałów odpowiadających bądź "0", bądź "1".

Przy użyciu wybieraków obrotowych układ wygląda tak, jak na rys. 5-11. Po naciśnięciu przycisku startowego rozpoczynają ruch wybieraki przekazując program wstępny do pamięci. Po zakończeniu tego procesu następuje uruchomienie taśmy na wejściu i przeniesienie programu obliczeń do pamięci.

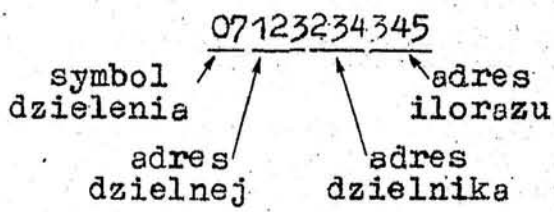
57. Rozkazy wieloadresowe

Rozkazom prostym każdej operacji towarzyszy jeden tylko adres, można jednakże tworzyć rozkazy zawierające dwa, trzy lub cztery adresy. Brzmienie rozkazu dwuadresowego dla operacji dodawania jest następujące: "Weź liczbę z miejsca pamięciowego a i dodaj do niej liczbę pobraną z miejsca pamięciowego b, pozostawiając przy tym sumę w miejscu pamięciowym b". Jasne jest, że rozkaz dwuadresowy tego typu jest równoważny trzem rozkazom jednoadresowym, a mianowicie: Aa, Ab i Cb.

Rozkazy dwuadresowe powstały w wyniku osobliwości konstrukcyjnych paru pierwszych wielkich maszyn cyfrowych i objęły swym zakresem tylko dodawanie. Mnożenie i dzielenie wykonuje się przy użyciu rozkazów jednoadresowych.

W rozkazach trójadresowych obok operacji podaje się adresy obu czynników działania oraz adres wyniku. W ten sposób trójadresowy rozkaz dodawania ma w zasadzie budowę następującą: "Weź liczbę z miejsca pamięciowego a, dodaj do niej liczbę pobraną z miejsca pamięciowego b i umieść sumę w miejscu pamięciowym c".

Rozkazy trójadresowe obejmują wszystkie działania arytmetyczne. W maszynie o układzie dziesiętnym rozkaz podzielenia liczby zawartej w miejscu pamięciowym 123 przez liczbę zawartą w miejscu pamięciowym 234 i umieszczenie ilorazu w miejscu pamięciowym 345 ma po zakodowaniu postać następującą



Każdy rozkaz trójadresowy można zastąpić przez trzy rozkazy jednoadresowe, dodawanie przez Aa, Ab i Cc, odejmowanie przez Aa, Bb i Cc, mnożenie przez Da, Eb i Cc, dzielenie przez Aa, Gb i Hc.

Są w użyciu również rozkazy trójadresowe o bardziej skomplikowanej budowie, zawierające, obok adresów czynników działania, również pewne dodatkowe informacje, którymi mogą być zmiana znaku liczby lub pobranie jej wartości bezwzględnej. Na przykład rozkaz mnożenia może mieć w takim systemie brzmienie następujące: "Pobierz wartość bezwzględną liczby zawartej w miejscu pamięciowym a, pomnóż ją przez wziętą z odwrotnym znakiem liczbę pobraną z miejsca pamięciowego b i umieść iloczyn w miejscu pamięciowym c". Dla zastąpienia takiego rozkazu trójadresowego trzeba byłoby w ogólnym przypadku użyć więcej niż trzech rozkazów jednoadresowych.

Rozkazy czteroadresowe różnią się od trójadresowych tylko tym, że zawierają jeszcze adres następnego rozkazu. W ten sposób zbędna staje się zasada wykonywania rozkazów w kolejności umieszczenia ich w pamięci. Korzyść eksploatacyjna rozkazów czteroadresowych w stosunku do trójadresowych polega na elastyczności wykorzystywania miejsc pamięciowych.

Zagadnienie celowości stosowania rozkazów wieloadresowych jest nieco podobne do dyskutowanego poprzednio zagadnienia powiększania liczby rozkazów w maszynie. Również w tym przypadku chęć ułatwienia programowania powoduje komplikacje konstrukcyjne i wzrost kosztów maszyny. Są zwolennicy obu systemów, ale nie wydaje się, aby zostało dowiedzione, że któryś z systemów wieloadresowych po uwzględnieniu wszystkich wad i zalet byłby korzystniejszy od systemu jednoadresowego.

Należy pamiętać poza tym o tendencji powiększania pojemności pamięci, co powoduje z kolei wzrost numerów adresów. Łatwo jest sprawdzić, że łączna liczba binów potrzebnych na oznaczenie operacji i trzech adresów przekraczałaby w wielu przypadkach pojemność miejsca liczbowego w pamięci.

58. Mikroprogramowanie

Wypełnienie rozkazu przez maszynę wymaga wykonania odpowiednio dobranego ciągu operacji elementarnych, czyli mikrooperacji. Pod mikrooperacją będziemy rozumieli taki ciąg stanów współpracujących ze sobą zespołów maszyny, w którym zmiany zachodzą bądź podczas jednego okresu pracy generatora sterującego, bądź podczas wielu kolejnych okresów, w ciągu których potencjały sterujące przebiegami pozostają niezmienne. Typowymi mikrooperacjami są np. przeniesienia zapisów z sumatora do rejestru przesunięć, z pamięci do sumatora lub z pamięci do rejestru mnożenia.

Ciąg mikrooperacji potrzebnych do wykonania danego rozkazu jest określony przez pewien program mikrooperacji, który będziemy nazywali mikroprogramem. Mikroprogram jest sekwencją mikrorozkazów, w każdym z których jest część określająca mikrooperację oraz adres; ten ostatni wyznacza część maszyny, która daną mikrooperację kontroluje.

Liczba mikrorozkazów potrzebnych do ułożenia programów wszystkich rozkazów jest rzędu parudziesięciu lub kilkudziesięciu; dokładna wartość zależy od szczegółów konstrukcji maszyny. Liczba natomiast mikrooperacji efektywnie wykonywanych w ciągu jednego rozkazu waha się od kilkudziesięciu do kilkuset. Dopiero, być może, z perspektywy mikroprogramów staje się widoczne, jak, w gruncie rzeczy, skomplikowaną dla maszyny czynnością jest wykonanie operacji arytmetycznej.

W skład mikrorozkazów potrzebnych do ułożenia programów wszystkich rozkazów musi wchodzić co najmniej jeden rozkaz warunkowy, którego skutek może zależeć od znaku liczby zarejestrowanej w określonym rejestrze i w określonej chwili lub też od wartości określonego binu.

Jak widać z powyższego, między mikroprogramem a programowaniem zachodzą daleko idące podobieństwa. Jeśli chodzi o różnice, to do najbardziej zasadniczych należy odmienna lokalizacja mikrorozkazów. Rozkazy są rejestrowane w pamięci, mikrorozkazy natomiast mogą być trwale rejestrowane w schemacie organu sterującego maszyny.