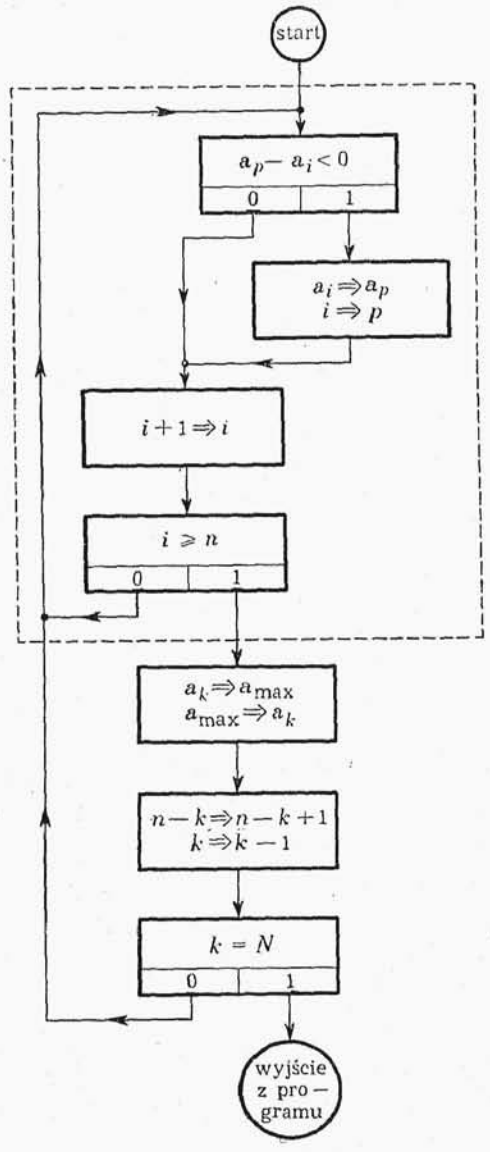


Rys. 4-4. Schemat blokowy drugiej metody porządkowania liczb

Linią przerywaną ujęto szukanie maksymalnej liczby w ciągu $k \div n$, gdzie $k = 0, 1, \ldots, n-1$ oraz $i = k, k+1 \ldots, n$, przy czym $k \leqslant p < n$.

uwagę) itd. aż zostanie nam tylko jedna liczba, wówczas kończymy nasz proces. Na rysunku 4-4 podany jest schemat blokowy programu realizującego powyższy algorytm postępowania.

4.2. PROGRAMY OBLICZENIOWE O STAŁYM PRZECINKU

4.2.0. Omawiana przez nas UPMC, jak wiemy, wykonuje operacje arytmetyczne na liczbach z przedziału $<-1, 1-2^{-33}>$, jednak w większości problemów fizycznych i technicznych argumenty wartości funkcji przebiegają znacznie szersze przedziały, jeśli więc chcemy korzystać bezpośrednio z działań arytmetycznych maszyny, musimy wprowadzić nowe zmienne przebiegające przedział $<-1, 1-2^{-33}>$. Oczywiście, nie wystarczy tylko sprowadzanie danych początkowych do wyżej wymienionego przedziału, musimy ponadto zapewnić sobie mieszczenie się w tym przedziale wszystkich wyników pośrednich. Taką metodę prowadzenia obliczeń będziemy nazywali metodą stałego przecinka.

W czasie analizy problemu, w przypadku programowania ze stałym przecinkiem, wyróżniamy następujące etapy:

1) analiza równań i formuł zadania, w celu określenia granic przedziałów, zmienności argumentów i wartości funkcji,

2) wybór mnożników dla każdej z wielkości zmiennych, zapewniających mieszczenie się w przedziale $<-1, 1-2^{-33}>$ iloczynu mnożnika i danej wielkości zmiennej,

3) wprowadzenie nowych zmiennych, przebiegających już przedział $<-1, 1-2^{-33}>$.

Należy w tym miejscu bardzo mocno podkreślić trudności związane z określeniem granic przedziałów zmienności argumentów i wartości funkcji, nie istnieje bowiem żadna uniwersalna metoda dokładnego określenia przedziałów zmienności. W praktyce ogra-

niczymy się albo do bardzo przybliżonej analizy zmienności przedziałów, albo określimy przedziały z fizycznego sensu problemu.

Sprowadzenie sumy do przedziału $<-1, 1-2^{-33}>$, gdy każdy ze składników sumy należy również do przedziału $<-1, 1-2^{-33}>$, można wykonać kilkoma metodami. Podamy obecnie jedną z takich metod podaną przez J. Kotlarskiego, dość przydatną (jak to dalej pokażemy na przykładzie) dla całkowania numerycznego. Metoda ta zapewnia maksymalną dokładność przy obliczeniach sum metodą stałego przecinka.

4.2.1. Niech będzie dany układ N liczb $a_1, a_2, a_3, \ldots, a_N$, przy czym $-1 \leqslant a_n < 1$ dla $n = 1,2,\ldots,N$. Zbudujemy ciąg średnich arytmetycznych

$$S_n = \frac{a_1 + a_2 + \ldots + a_n}{n} \quad \text{dla} \quad n = 1,2,\ldots,N;$$

wyrazy tego ciągu możemy obliczyć korzystając z wzoru rekurencyjnego

$$S_1 = a_1,$$

$$S_{n+1} = S_n + \frac{a_{n+1} - S_n}{n+1} \quad \text{dla} \quad n = 1,2,\ldots, N-1.$$

Program realizujący powyższy wzór rekurencyjny jest prosty, składa się on z 14 rozkazów. Związek między S_N a $\sum_{i=1}^{N} a_i$ jest następujący:

$$\sum_{i=1}^{N} a_i = NS_N.$$

Jak widać, $-1 \leqslant S_n < 1$ dla $n = 1,2,\ldots, N$.

W omawianym programie będziemy korzystali z następujących komórek pamięci:

$$\begin{aligned} \text{adres roboczy} &= b + 4000, \\ < S_n > &= b + 4004, \\ < \tfrac{1}{2} a_n > &= b + 4002, \\ < n \cdot 2^{-16} > &= b + 0006. \end{aligned}$$

Pod adresem $b + 0006$ przed rozpoczęciem obliczeń znajduje się 2^{-16}.

Program będzie wymagał jednej stałej równej 2^{-16}.

Zakładając, że a_n znajduje się w akumulatorze, otrzymamy następujący początek programu:

$k + 0000$		miejsce na ślad,	
	1	21	0001,
	2	14	$b + 4002$,
$k + 0003$		12	$b + 0006$,
	4	10	$< 2^{-16} >$,
	5	14	$b + 0006$.

Jak łatwo widzieć, zawsze prawdziwe jest oszacowanie (przy założeniu, że $-1 \leqslant a < 1$ dla $i = 1,2,3,\ldots, N$):

$$-1 \leqslant \frac{a_{n+1} - S_n}{n+1} < 1 \qquad \text{dla } n = 1,2,\ldots, N-1;$$

$$\frac{a_{n+1} - S_n}{n+1} = \frac{2^{-16}(a_{n+1} - S_n)}{2^{-16}(n+1)} = \frac{2^{-15}\left(\frac{a_{n+1}}{2} - \frac{S_n}{2}\right)}{2^{-16}(n+1)}.$$

Wykonane przekształcenie zapewnia nam zawieranie się wszystkich wyników w przedziale $< -1, +1)$

$k+0006$	13	$b+4004$	obliczenie
7	21	0001	$2^{-15}\left(\frac{a_{n+1}}{2} - \frac{S_n}{2}\right)$,
$k+0010$	10	$b+4002$	
1	21	0017	
2	17	b+0006	$\frac{a_{n+1} - S_n}{n+1}$,
3	10	$b+4004$	S_n,
4	14	$b+4004$	$S_{n+1} \Longrightarrow S_n$,
5	01	$k+0000$	powrót do programu głównego.

Przykład 4-1. Obliczyć całkę:

$$\mathcal{J} = \int_0^1 \text{arc sin} \sqrt{\frac{x}{1+x}}\, dx$$

z dokładnością $\varepsilon = 0{,}00001$, stosując przybliżoną metodę numerycznego całkowania, mzt metodę prostokątów. Analiza przedziałów zmienności kolejnych funkcji występujących pod całką $\mathcal{J}$ prowadzi do następujących oszacowań:

$$0 \leqslant \frac{x}{1+x} \leqslant \frac{1}{2},$$

$$0 \leqslant \sqrt{\frac{x}{1+x}} \leqslant \frac{\sqrt{2}}{2},$$

$$0 \leqslant \text{arc sin} \sqrt{\frac{x}{1+x}} \leqslant \frac{\pi}{4}.$$

Jak widać z powyższych oszacowań, jedynym potrzebnym przekształceniem jest podzielenie w ułamku $\frac{x}{1+x}$ licznika i mianownika przez 4. Wprowadzając nową zmienną $t = \frac{1}{4}x$, otrzymamy

$$\mathcal{J} = 4 \int_0^{\frac{1}{4}} \text{arc sin} \sqrt{\frac{t}{\frac{1}{4}+t}}\, dt.$$

Jeśli oznaczymy funkcję podcałkową przez $f(t)$, to formuła prostokątów dla całki

$$\int_a^b f(t)\, dt,$$

ma postać

$$\int_a^b f(t)\mathrm{d}t \approx h\,[f(t_0) + f(t_1) + \ldots + f(t_{n-1})];$$

gdzie $t_0 = a$, $t_n = b$, $h = \dfrac{b-a}{n}$ oraz $t_j = t_0 + hj$ dla $j = 1,2,\ldots,n-1$.

W naszym przypadku przyjmiemy:

$$h = 0{,}00025, \quad t_0 = 0, \quad t_n = \tfrac{1}{4}, \quad t_j = j \cdot 0{,}00025.$$

Do obliczeń będziemy używali trzech podprogramów:

1) podprogramu do obliczania pierwiastka kwadratowego,
2) podprogramu do obliczania arc sin y,
3) podprogramu sumowania metodą średnich arytmetycznych.

Stosując metodę sumowania średnich arytmetycznych zauważmy, że przybliżona wartość całki $\mathcal{J}$ jest równa:

$$\mathcal{J} = 4hnS_n$$

gdzie S_n jest średnią arytmetyczną liczb $f(t_0),\ldots,f(t_{n-1})$.

Ponieważ $n = 1000$, zaś $h = 0{,}00025$, mamy więc

$$\mathcal{J} = S_n.$$

Z powyższego wzoru widać, że nie musimy obliczać wartości sumy, wystarczy obliczyć jej średnią arytmetyczną.

Podprogram dla obliczania pierwiastka kwadratowego składa się z 51 słów krótkich, wywołanie podprogramu wygląda następująco: liczbę, którą chcemy pierwiastkować, umieszczamy w akumulatorze, po czym wykonujemy operację 04 k + 0000 (gdzie k jest stałą przeadresowania podprogramu), po wykonaniu podprogramu wynik zostaje w akumulatorze. Podprogram korzysta z ośmiu miejsc roboczych krótkich o adresach 0100 ÷ 0107.

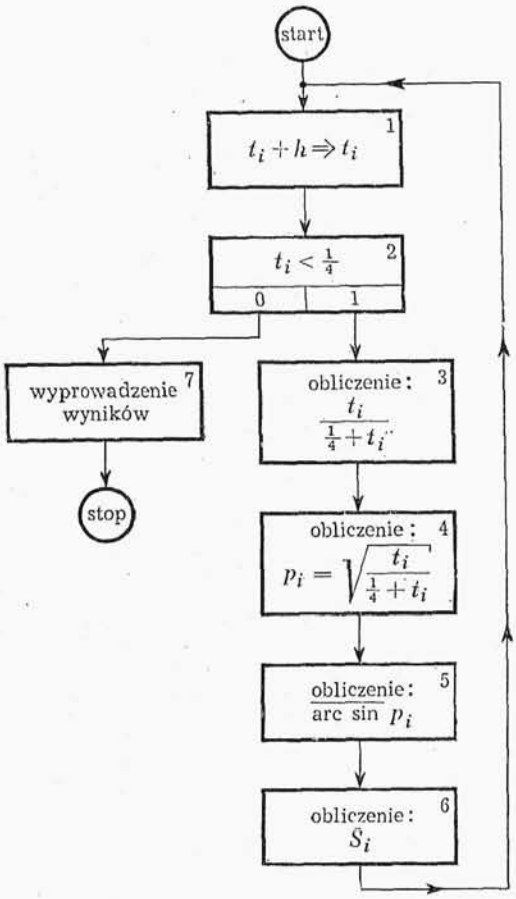
Podprogram dla obliczania arc sin y, składa się z 48 słów krótkich, wywołanie podprogramu wygląda następująco: argument umieszczamy w akumulatorze, po czym wykonujemy operację 04 k' + 0000 (gdzie k' jest stałą przeadresowania podprogramu), po wykonaniu podprogramu wynik zostaje w akumulatorze. Podprogram korzysta z czterech miejsc roboczych krótkich o adresach 0100 ÷ 0103.

Podprogram dla sumowania metodą średnich arytmetycznych składa się z 14 słów krótkich; wywołanie podprogramu wygląda następująco: kolejny wyraz przeznaczony do sumowania umieszczamy w akumulatorze, po czym wykonujemy operację 04 k'' +0000 (gdzie k'' jest stałą przeadresowania programu), po wykonaniu podprogramu S_n zostaje pod adresem 4114, $n \cdot 2^{-16}$ zaś zostaje pod adresem 0116. Podprogram korzysta z siedmiu miejsc roboczych krótkich o adresach 0110 ÷ 0116.

Schemat blokowy płaski naszego programu przedstawiony jest na rys. 4-5.

Ze względu na brak miejsca nie będziemy rysowali schematu blokowego liniowego, natomiast bezpośrednio na podstawie schematu płaskiego zakodujemy program główny, przyjmując, że operatory ostatecznego sformułowania wyniku i wyprowadzenia wy-

niku z maszyny umieścimy za pętlą cyklu, podprogramy zaś w kolejności wyżej wymienionej umieścimy za programem głównym.



Rys. 4-5. Rysunek do przykładu 4-1

Przyjmując, że program główny ma miejsca robocze podane w tabl. 4-8, kolejne operatory i predykaty programu głównego, po zakodowaniu, przedstawiamy w tabl: 4-9.

Musimy pamiętać, że miejsca robocze $b + 4002$, 4114 i 0116 przed wykonaniem programu należy wyzerować, możemy dokonać tego za pomocą operatora przygotowania, który umieszczamy przed programem głównym.

Tablica 4-8

Adres	$b+4000$	$b+4002$	$b+4004$	$b+0006$	$b+0007$
Zawartość adresu	h	t_i	$t_i+\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$

Tablica 4-9

Program opracowany przy założeniu, że w chwili początkowej pod adresem $b + 4002$ znajduje się zero

Operator lub predykat	Kolejny adres	Kolejny rozkaz		Objaśnienia
1	$g+0000$	12	$b+4002$	przy wprowadzeniu programu do maszyny musimy wyzerować adres $b+4002$
	1	10	$b+4000$	
	2	14	$b+4002$	$t_i - \frac{1}{4} < 0$
2	3	11	$b+0006$	
	4	06	$g+0015$	
3	5	10	$b+0007$	$\left(t_i - \frac{1}{4}\right) + \frac{1}{2} = t_i + \frac{1}{4}$
	6	14	$b+4004$	$t_i + \frac{1}{4} \rightarrow b+4003$
	7	12	$b+4002$	$t_i \rightarrow A$
	$g+0010$	17	$b+4004$	$\frac{t_i}{\frac{1}{4} + t_i}$
4	1	04	$k+0000$	wywołanie podprogramu pierwiastka
5	2	04	$k'+0000$	wywołanie podprogramu arc sin y,
6	3	04	$k''+0000$	wywołanie podprogramu obliczenia S_n
	4	02	$g+0000$	
7	5	04	<podprogram wyprowadzający>	
8	6	05	0000	stop

Podprogramu przeliczenia i drukowania wyników nie będziemy tu omawiać, zajmiemy się nim szczegółowo w rozdz. 5.

Rozmieszczając miejsca robocze programu głównego w pamięci maszyny musimy pamiętać, że $b > 116'$.

4.3. PROGRAMY OBLICZENIOWE O ZMIENNYM PRZECINKU

4.3.0. Przez postać normalną (dwójkowo-dwójkową) liczby binarnej N będziemy rozumieli postać[1]:

$$N = 2^c \cdot m,$$

[1] Postacią normalną liczby binarnej nazywana jest również postać tzw. dwójkowo-dziesiętna, w której liczba N jest postaci

$$N = 10^x \cdot a,$$

gdzie x jest liczbą całkowitą, $-10 < a < 10$ oraz $1 \leqslant |a| < 10$ (Bibliografia [19]).

gdzie c całkowite, oraz

$$-1 < m \leqslant -\tfrac{1}{2} \text{ albo } \tfrac{1}{2} \leqslant m < 1, \tag{4-1}$$

przy czym liczby c i m spełniające podane wyżej warunki są nazywane odpowiednio cechą i mantysą liczby N.

Oczywiście, realizując taką postać w maszynie ustalimy przedział zmienności c; najczęściej stosowanym przedziałem jest przedział $<-32, +31>$. W niektórych maszynach bywa używany przedział większy, np. $<-64, +63>$.

Operacje arytmetyczne na liczbach postaci normalnej będziemy nazywali operacjami w zmiennym przecinku.

4.3.1. Postać liczb. W omawianym niżej podprogramie liczby $N \neq 0$ przedstawione są w postaci normalnej. Jeśli $N = 0$, to przyjmujemy, że $c = m = 0$.

Zapis liczby zmiennoprzecinkowej N, składający się z cechy c i mantysy m, stanowi w pamięci maszyny 34-bitową (długą) liczbę postaci:

$$2^{-5}c^* + 2^{-6}m^*, \tag{4-2}$$

gdzie

$$c^* = \begin{cases} c, & \text{jeśli } c \geqslant 0, \\ 64 + c, & \text{jeśli } c < 0, \end{cases} \qquad m^* = \begin{cases} m, & \text{jeśli } m \geqslant 0, \\ 2 + m, & \text{jeśli } m < 0. \end{cases}$$

Inaczej mówiąc, na bardziej znaczących 6 bitach słowa długiego zapamiętana jest cecha, a na pozostałych 28 bitach mantysa — obie w arytmetyce uzupełnieniowej. Oprócz zera można w ten sposób przedstawić (z około ośmioma znaczącymi cyframi dziesiętnymi) liczby spełniające nierówności (4-1) i nierówność

$$-32 \leqslant c \leqslant 31, \tag{4-3}$$

tj. liczby o wartości bezwzględnej z przedziału

$$<2^{-32}, 2^{31}> \approx <1{,}16 \cdot 10^{-10}, 2{,}14 \cdot 10^{9}>.$$

4.3.2. Program działań w zmiennym przecinku będzie się składał z następujących części:

1. Podprogramu rozformowania, który rozdzieli cechy i mantysy obu argumentów działania.

2. Podprogramów wykonujących poszczególne działania arytmetyczne i podających wynik w postaci pary pseudocechy γ i pseudomantysy μ, według następujących wzorów:

dla dodawania

$$\begin{array}{llll} \text{jeśli } c_1 \geqslant c_2, & \gamma = c_1 + 1, & \mu = \tfrac{1}{2}m_1 + 2^{-c_1+c_2-1}m_2, & \\ \text{jeśli } c_2 < c_1 & \gamma = c_2 + 1, & \mu = 2^{c_1-c_2-1}m_1 + \tfrac{1}{2}m_2; & \end{array} \tag{4-4}$$

dla odejmowania

$$\begin{array}{llll} \text{jeśli } c_1 \geqslant c_2, & \gamma = c_1 + 1, & \mu = \tfrac{1}{2}m_1 - 2^{-c_1+c_2-1}m_2, & \\ \text{jeśli } c_2 < c_1, & \gamma = c_2 + 1, & \mu = 2^{c_1-c_2-1}m_1 - \tfrac{1}{2}m_2; & \end{array} \tag{4-5}$$

dla mnożenia

$$\gamma = c_1 + c_2, \quad \mu = m_1 m_2; \tag{4-6}$$

dla dzielenia

$$\gamma = c_1 - c_2 + 1, \quad \mu = \frac{m_1}{2m_2}. \tag{4-7}$$

3. Podprogramu normalizującego pseudomantysę wyniku; obliczenie cechy wyniku następuje poprzez zsumowanie pseudocechy z liczbą całkowitą v, tak dobraną, że liczba $2^v\mu$ jest mantysą w sensie nierówności (4-1).

4. Podprogramu sformowania wyniku, czyli zakodowania go jako liczby postaci (4-2), jeśli cecha wyniku spełnia nierówność (4-3). W przypadku gdy cecha wyniku jest większa od 31, podprogram sygnalizuje to przez wywołanie podprogramu dzwonka. W przypadku gdy cecha wyniku jest mniejsza od—32, wynik kładziemy równy zeru.

4.3.3. Sposób posługiwania się programem. Podprogram działania arytmetycznego jest wywoływany za pomocą rozkazu skoku ze śladem. W chwili wykonania tego rozkazu argumenty działania powinny znajdować się: jeden w akumulatorze, drugi w komórce 4100. W związku z tym rozróżniamy dwa rodzaje działań: działania symetryczne, tj. dodawanie, mnożenie, i działania niesymetryczne, tj. odejmowanie i dzielenie „lewe" (odjemna lub dzielna w akumulatorze, odjemnik lub dzielnik w komórce 4100) i „prawe" (odjemna lub dzielna w komórce 4100, odjemnik lub dzielnik w akumulatorze). Przy działaniach symetrycznych, tj. dodawaniu i mnożeniu, jest obojętne, który składnik lub czynnik znajduje się w akumulatorze, a który w komórce 4100.

Poszczególne działania wywołuje się za pomocą następujących rozkazów:

dodawanie za pomocą rozkazu	04 k + 0000,
„lewe" odejmowanie za pomocą rozkazu	04 k + 0066,
„prawe" odejmowanie za pomocą rozkazu	04 k + 0050,
mnożenie za pomocą rozkazu	04 k + 0077,
„lewe" dzielenie za pomocą rozkazu	04 k + 0136,
„prawe" dzielenie za pomocą rozkazu	04 k + 0116.

(podprogram zmiennoprzecinkowych działań arytmetycznych mieści się w komórkach $k + 0000 \div k + 0223$).

Wynik dowolnego działania arytmetycznego znajduje się po jego wykonaniu na miejscu obu argumentów, tj. w akumulatorze i w komórce 4100.

Dla wykonania zmiennoprzecinkowego działania arytmetycznego, przy wykonywaniu ciągu działań arytmetycznych należy

1) przesłać argumenty do „pozycji początkowych",
2) wywołać podprogram tego działania,
3) przesłać wynik działania do odpowiedniej komórki.

W wielu przypadkach można (m.in. dzięki wprowadzeniu dwóch rodzajów odejmowań i dzieleń) skrócić lub w ogóle opuścić rozkazy 1) i 3).

Dla zilustrowania tej uwagi rozpatrzmy następujący przykład. Należy obliczyć wyrażenie

$$\left(\frac{a-b}{c}+\frac{d}{e-fg}\right)^2,$$

w którym liczby a, b, c, d, f, e, g są dane w zmiennoprzecinkowej postaci. Można to zrobić za pomocą następującego ciągu rozkazów:

12 <a>
14 4100 } przesłanie $a \rightarrow 4100$, $b \rightarrow A$ (A akumulator),
12 <b>

04 k+0050 „prawe" odejmowanie (odjemnik w akumulatorze) $a-b$,
12 <c> przesłanie $c \rightarrow A$,
04 k+0116 „prawe" dzielenie (dzielnik w akumulatorze) $\frac{a-b}{c}$,
14 R przesłanie $\frac{a-b}{c}$ do komórki roboczej R,

12 <f>
14 4100 } przesłanie $f \rightarrow 4100$, $g \rightarrow A$,
12 <g>

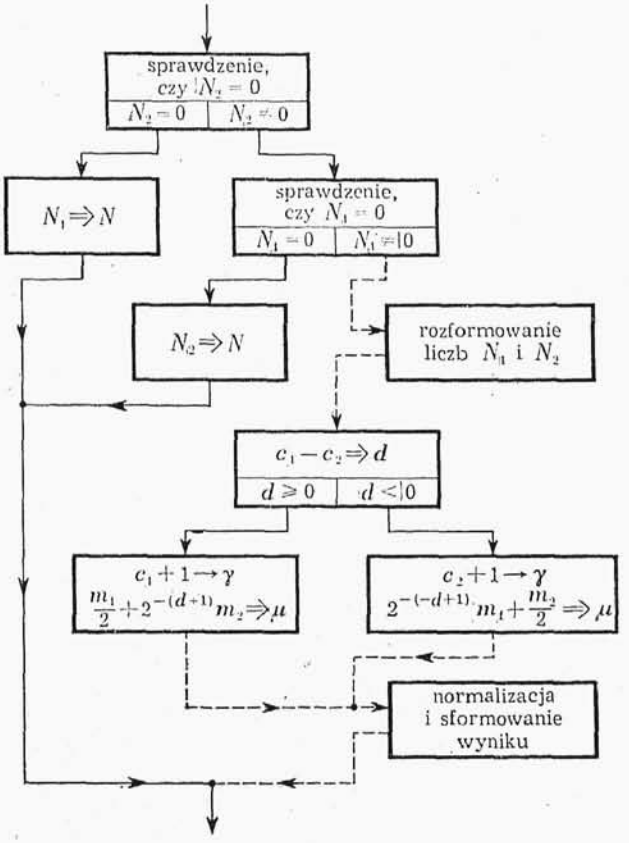
04 k+0077 mnożenie gf,
12 <e> przesłanie $e \rightarrow A$,
04 k+0066 „lewe" odejmowanie (odjemna w akumulatorze) $e-fg$,
12 <d> przesłanie $d \rightarrow A$,
04 k+0136 „lewe" dzielenie (dzielna w akumulatorze) $\frac{d}{e-fg}$,
12 R przesłanie $(R) = \frac{a-b}{c} \rightarrow A$,
04 k+0000 dodawanie $\frac{a-b}{c}+\frac{d}{e-fg}$,
04 k+0077 mnożenie $\left(\frac{a-b}{c}+\frac{d}{e-fg}\right)\left(\frac{a-b}{c}+\frac{d}{e-fg}\right)$.

4.3.4. Uwagi. Program działań arytmetycznych na liczbach postaci zmiennoprzecinkowej wykorzystuje komórki robocze 4100, 4102, 4104 oraz parametry

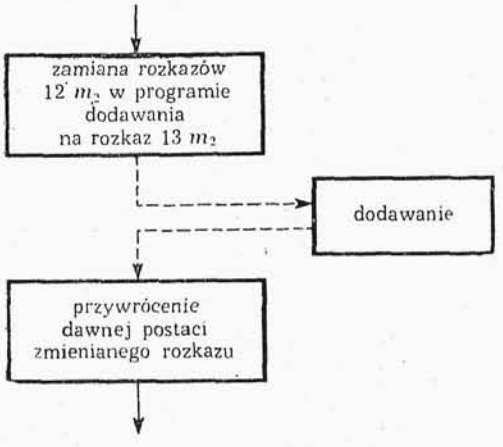
01 0000 = (0066),
00 0001 = (0076),
00 0000 = (0077),

z pamięci stałej. Część tego programu można wykorzystać dla przekształcenia liczb danych w postaci stałoprzecinkowej do postaci zmiennoprzecinkowej. Przed wywołaniem tego podprogramu za pomocą rozkazu 04 k+0162, należy wyzerować komórkę 0104 i umieścić przekształcaną liczbę w akumulatorze. Wynik przekształcenia znajduje się także w komórce 4100.

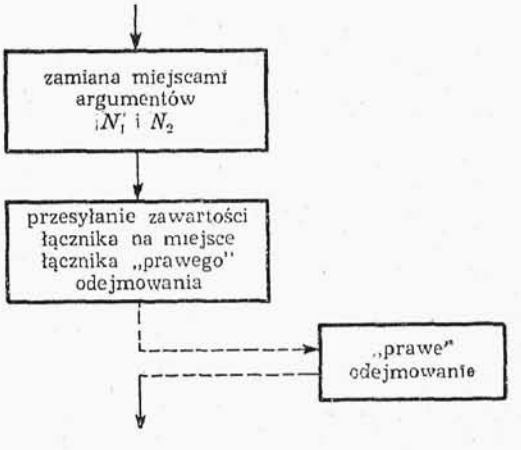
W tablicy 4-10 podajemy średnią liczbę rozkazów składającą się na wykonanie działań arytmetycznych dla poszczególnych postaci liczb. Dane przedstawione w tabl. 4-10



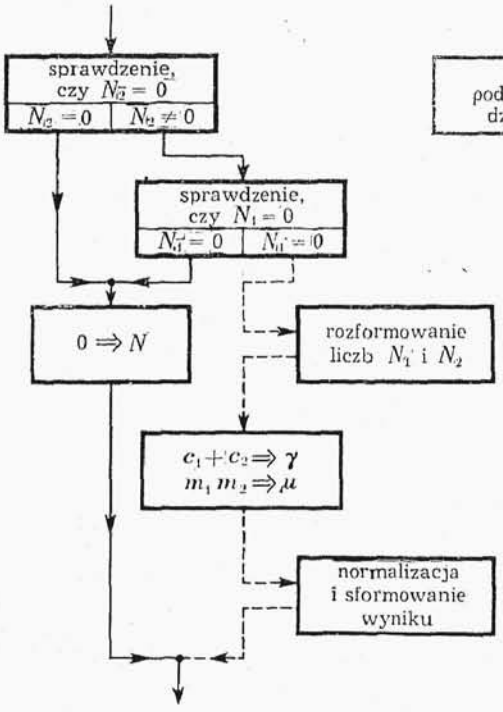
Rys. 4-6. Schemat blokowy sumowania w zmiennym przecinku



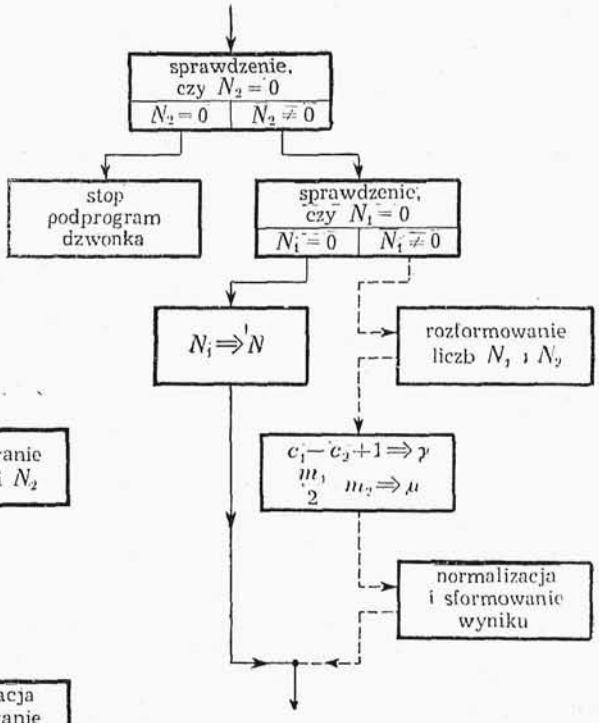
Rys. 4-7. Schemat blokowy podprogramu „prawego" odejmowania zmiennoprzecinkowego



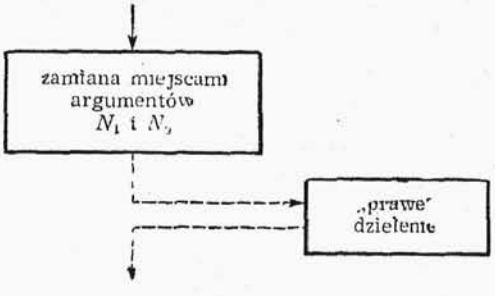
Rys. 4-8. Schemat blokowy podprogramu „lewego" odejmowania zmiennoprzecinkowego



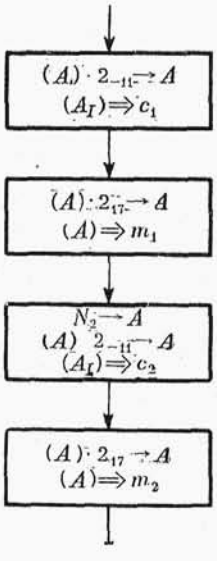
Rys. 4-9. Schemat blokowy podprogramu mnożenia liczb zmiennoprzecinkowych



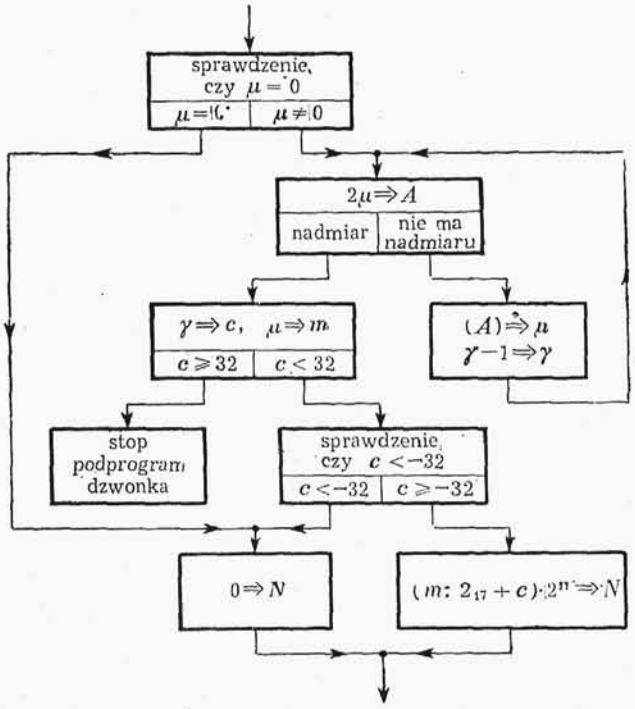
Rys. 4-10. Schemat blokowy podprogramu „prawego" dzielenia liczb zmiennoprzecinkowych



Rys. 4-11. Schemat blokowy podprogramu „lewego" dzielenia liczb zmiennoprzecinkowych



Rys. 4-12. Schemat blokowy podprogramu rozformowania argumentów liczb zmiennoprzecinkowych



Rys. 4-13. Schemat blokowy podprogramu normalizacji i sformowanie wyniku działań zmiennoprzecinkowych

Tablica 4-10

Działanie	Przypadek	Liczba rozkazów
dodawanie	$N_1, N_2 \neq 0$ $N_1 = 0$ lub $N_2 = 0$	$46+7v$ 7
„lewe" odejmowanie	$N_1, N_2 \neq 0$ $N_1 = 0$ lub $N_2 = 0$	$67+7v$ 28
„prawe" odejmowanie	$N_1, N_2 \neq 0$ $N_1 = 0$ lub $N_2 = 0$	$60+7v$ 21
mnożenie	$N_1, N_2 \neq 0$ $N_1 = 0$ lub $N_2 = 0$	40 6
„lewe" dzielenie	$N_1, N_2 \neq 0$ $N_2 = 0$	51 15
„prawe" dzielenie	$N_1, N_2 \neq 0$ $N_1 = 0$	43 7
Przejście od liczb stałoprzecinkowych do zmiennoprzecinkowych		$14+7v$

pozwalają na oszacowanie czasów potrzebnych na wykonanie działań zmiennoprzecinkowych.

Liczba całkowita nieujemna v wyznaczająca czas przejścia od pseudomantysy (ewentualnie od liczby stałoprzecinkowej do jej mantysy) jest z definicji równa liczbie zer w pseudomantysie między przecinkiem a jej pierwszą cyfrą znaczącą (w układzie dwójkowym).

Na rysunkach 4-6÷4-13 przedstawione są schematy blokowe poszczególnych części jednoadresowego programu zmiennego przecinka, liniami przerywanymi oznaczono wywoływanie podprogramów i powrót po wykonaniu podprogramu. W załączniku 4 (umieszczonym na końcu książki) podany jest omawiany wyżej program działań zmiennego przecinka.

4.3.5. Przykład programu korzystającego z jednoadresowych podprogramów zmiennego przecinka[1]. Niech będzie dany układ n równań algebraicznych liniowych:

$$X = AX + F, \tag{4-8}$$

Elementy macierzy A i współrzędne wektora F są liczbami zmiennoprzecinkowymi z przedziału przyjętego przez jednoadresowy program zmiennego przecinka (punkt 4.3.1), gdzie $i, j, = 1,2, \ldots, n$.

Przypuśćmy, że rozwiązanie układu (4-8) chcemy znaleźć za pomocą metody iteracyjnej Seidla określonej za pomocą wzorów

$$x_i^{(k)} = \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + \sum_{j=1}^{n} a_{ij} x_j^{(k-1)} + f_i, \tag{4-9}$$

gdzie $i = 1,2, \ldots, n$, $k = 1,2, \ldots, l$.

Zakładamy oczywiście, że metoda jest zbieżna. Przybliżenie początkowe $X^{(0)}$ określamy dowolnie; przyjmując na przykład, że równa się ono wektorowi wyrazów wolnych F.

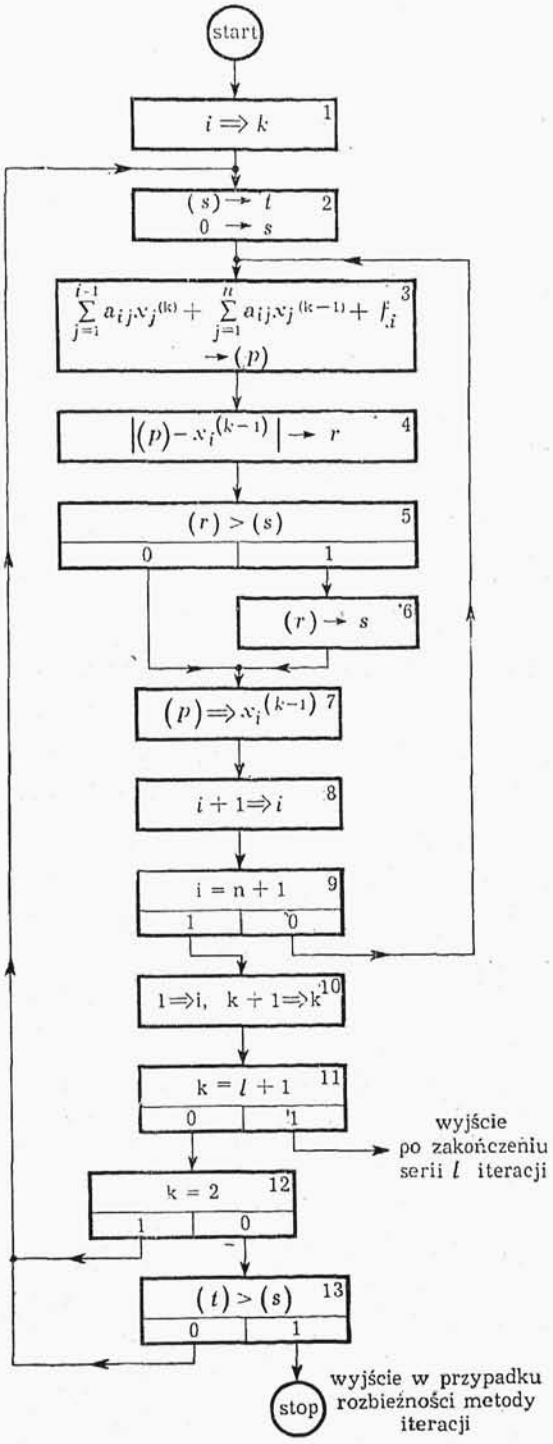
Za kryterium rozbieżności przyjmiemy następujący związek:

$$\max_{1 \leqslant i \leqslant n} \left| x_i^{(k)} - x_i^{(k-1)} \right| \geqslant \max_{1 \leqslant i \leqslant n} \left| x_i^{(k-1)} - x_i^{(k-2)} \right|. \tag{4-10}$$

Program realizujący metodę iteracyjną Seidla korzysta z n^2+2n komórek pamięci (długich), w których kolejno znajdują się wielkości $f_1, f_2, \ldots, f_n, x_1, x_2, \ldots, x_n, a_{11}, a_{12}, \ldots, a_{1n}, \ldots, a_{n1}, a_{n2}, \ldots, a_{nn}$. Ponadto program korzysta z czterech komórek długich oznaczonych literami p, r, c, t oraz z jednej komórki krótkiej, w której umieszczamy bieżącą wartość k (liczbę wykonanych kroków iteracji). Przed wywołaniem podprogramu, program główny umieszcza parametry $00\ l+1$ (l—ilość kroków iteracji, które należy wykonać) oraz 12 a_{11} odpowiednio pod adresami 0106, 0107, w akumulatorze zaś parametr $2n \cdot 2^{-17}$ (n-ilość równań). Przyjęto, że komórki robocze mają następujące adresy:

$$p = 4112,$$
$$r = 4114,$$
$$s = 4116,$$
$$t = 4120,$$
$$k = 0111.$$

[1] Przykład opracowany przez Z. Jankowską.



Rys. 4-14. Rysunek do przykładu

Tablica 4-11

Operator lub predykat	Kolejny adres	Kolejny rozkaz		Objaśnienia
1	$k+0000$	77	7777	miejsce na ślad
	1	14	0110	$2n \longrightarrow 0110$
	2	12	0076	$l \longrightarrow k$
	3	14	0111	
2	4	12	4116	$(s) \longrightarrow t$
	5	14	4120	
	6	12	0077	$0 \longrightarrow s$
	7	14	4116	
	$k+0010$	12	0107	formowanie rozkazów w pętli sumowania
	1	14	$k+0024$	
	2	11	0110	
	3	14	$k+0026$	
	4	14	$k+0044$	
	5	10	0031	formowanie rozkazów w pętli równania
	6	14	$k+0057$	
	7	11	0031	formowanie rozkazów w pętli równania
	$k+0020$	11	0110	
	1	14	$k+0022$	
3	2	12	7777	przesłanie x_i do p
	3	14	4112	
	4	12	7777	
	5	14	4100	
	6	12	7777	
	7	04	$k_{019}+77$	wywołanie mnożenia zmiennoprzecinkowego
	$k+0030$	12	4112	
	1	04	k_{019}	wywołanie dodawania zmiennoprzecinkowego
	2	14	4112	
3	3	12	$k+0024$	k_{019} — stała przeadresowania podprogramu zmiennoprzecinkowego
	4	10	0067	
	5	14	$k+0024$	
	6	11	0110	
	7	14	$k+0026$	
	$k+0040$	11	0107	
	1	03	$k+0024$	
4	2	12	4112	
	3	14	4100	przygotowanie odejmowania zmiennoprzecinkowego
	4	12	7777	
	5	04	$k_{019}+66$	wywołanie „lewego“ odejmowania
	6	04	$k_{019}+226$	wywołanie operacji wartości bezwzględnych zmiennoprzecinkowych
	7	14	4114	
5	$k+0050$	12	4116	
	1	04	$k_{019}+50$	wywołania „prawego“ odejmowania zmiennoprzecinkowego
	2	23	0000	
	3	03	$k+0056$	

Tablica 4-11 (d. c.)

Operator lub predykat	Kolejny adres	Kolejny rozkaz		Objaśnienia
6	$k+0054$	12	4114	przesłanie zawartości r do s
	5	14	4116	
7	6	12	4112	przesłanie zawartości p na miejsce x_i
	7	14	7777	
8	$k+0060$	12	$k+0026$	przeadresowanie pętli
	1	11	0110	
	2	14	$k+0026$	
	3	12	$k+0057$	
	4	10	0067	
	5	14	$k+0057$	
	6	12	$k+0022$	
	7	10	0067	
	$k+0070$	14	$k+0022$	
	1	11	0110	
	2	14	$k+0044$	
9	$k+0073$	11	0107	wyjście z pętli równania lub powtórzenie jej
	4	03	$k+0022$	
10	5	12	0111	powiększenie zawartości licznika iteracji
	6	10	0076	
	7	14	0111	
11	$k+0100$	11	0106	sprawdzenie krotności iteracji
	1	03	$k+0103$	
	2	01	$k+0000$	
12	3	10	0106	sprawdzenie, czy zachodzi warunek $k = 2$
	4	11	0067	
	5	03	$k+0004$	
13	$k+0106$	12	4120	sprawdzenie zbieżności iteracji, wywołanie „lewego" odejmowania zmiennoprzecinkowego
	7	14	4100	
	$k+0110$	12	4116	
	1	04	$k_{019}+66$	
	2	23	0006	
	3	03	$k+0004$	

Schemat blokowy programu jest podany na rys 4-14. Program ten jest ekonomiczny tylko w przypadku, gdy znaczna większość współczynników w macierzy $||a_{ij}||$ jest różna od zera.

W przypadku gdy wśród współczynników macierzy $||a_{ij}||$ jest dużo zer, stosujemy program bardziej złożony, wybierający niezerowe współczynniki macierzy $||a_{ij}||$ za pomocą skal logicznych.

Szczegółowo omówimy kolejne rozkazy programu w tabl. 4-11.