$$I = I_4 = 1.010 = -\frac{3}{4},$$

$$R = 2^{-4}\ (1.100 + 0.100) = 0.0000.$$

Przykład 2-5.

$$A = -\frac{9}{64} = 1.110111, \quad M = \frac{3}{8} = 0.011.$$

Sprawdzenie nierówności (2-1):

$$1 \neq 0, \quad A_1 = 0{,}001111,$$

jak widać z tabl. 2-4, nierówność jest spełniona, wobec czego możemy przystąpić do dzielenia.

pierwszy krok:	$0 = 0$,	$A_2 = 0.00011$,	$I_1 = 1.00000$,
drugi krok:	$0 = 0$,	$A_3 = 1.1101$,	$I_2 = 1.10000$,
trzeci krok:	$1 \neq 0$,	$A_4 = 0{,}000$,	$I_3 = 1.10000$,
czwarty krok:	$0 = 0$,	$A_5 = 1.101$,	$I_4 = 1.10100$,
piąty krok:	$1 \neq 0$,	$A_6 = 1.101$,	$I_5 = 1.10100$,
szósty krok:	$1 \neq 0$,	$A_7 = 1.101$,	$I_6 = 1.10100$,

$$I = I_6 = 1.10100 = -\frac{3}{8},$$

$$R = 2^{-6}\ (1.101 + 0.011) = 0.00000.$$

2.2. KRÓTKI OPIS MASZYNY TYPOWEJ

2.2.0. Obecnie omówimy zasady organizacji małej UPMC. Maszyna taka mimo niskich parametrów i prostoty organizacji, wystarczy do wyłożenia zasadniczych elementów programowania w całej rozciągłości. Przedstawiona niżej typowa mała UPMC będzie maszyną szeregową stałoprzecinkową, pracującą w binarnej arytmetyce uzupełnieniowej. Maszyna wykonuje operacje na liczbach dwójkowo wymiernych z przedziału $<-1,\ 1-2^{-33}>$.

W maszynie wyróżniamy dwa rodzaje słów:

1) słowa krótkie o długości 17 pozycji binarnych — w skrócie 17 B[1],

2) słowa długie o długości 34 pozycji binarnych — w skrócie 34 B.

Rozkazy (instrukcje) dla maszyny są kodowane binarnie i przedstawione za pomocą słów krótkich.

Maszyna może wykonywać wszystkie operacje arytmetyczne, logiczne zarówno na słowach krótkich, jak i na słowach długich. Zasadniczo słowa 17 bitowe służą do przedstawiania rozkazów, a słowa 34 bitowe — liczb. Gdy używamy słowa 17 bitowe jako

[1] Literę B będziemy czytali bit, 17B czytamy 17 bitów.

liczby, wtedy maszyna automatycznie dopełnia tę liczbę w rejestrach arytmometru do 34 bitów zerami (od 17 do 34 bitu), tzn. na siedemnastu miejscach najmniej znaczących liczby

$$\alpha_0 \cdot \alpha_1 \ldots \alpha_7 \ldots \alpha_{16}\, \alpha_{17}\, \alpha_{18} \ldots \alpha_{33}$$

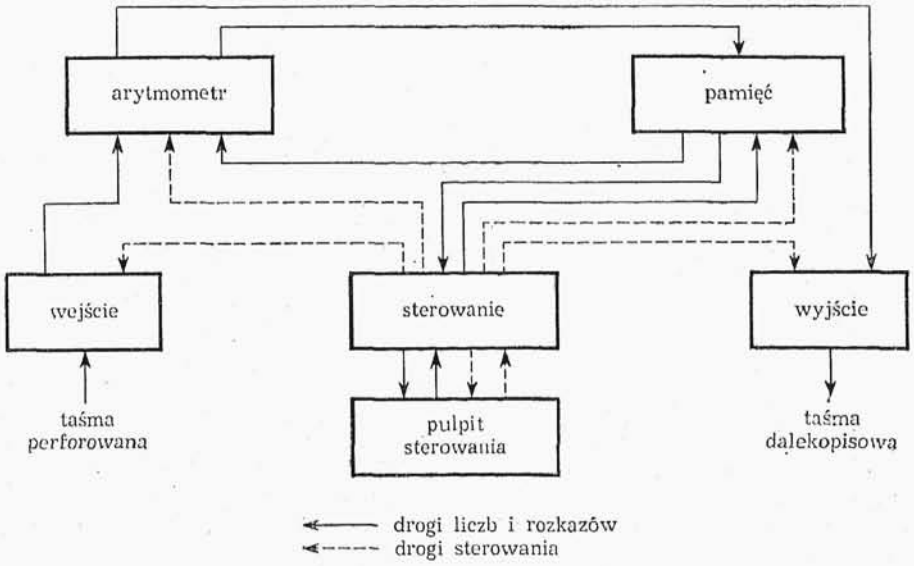
cyfry od α_{17} do α_{33} są zerami.

Prędkość maszyny wynosi około 100 operacji/s.

Maszyna typowa składa się z siedmiu zasadniczych części:

1) wejścia,
2) wyjścia,
3) pamięci,
4) sterowania,
5) arytmometru,
6) pulpitu sterowania,
7) zasilacza.

Schemat ideowy powiązania między sobą pierwszych sześciu podzespołów pokazano na rys. 2-1.



Rys. 2-1. Schemat przykładowej UPMC

W dalszym ciągu zajmiemy się krótkim omówieniem pierwszych sześciu części maszyny. Zasilaczem nie będziemy się zajmowali ze względu na charakter naszych rozważań.

2.2.1. Wejście. Wejście do maszyny stanowi fotoczytnik standartowej taśmy dalekopisowej czytający skokowo, o prędkości czytania ~50 rządków na sekundę. Czytnik jest uruchomiony specjalnym rozkazem maszyny, a odczytany wynik zostaje umieszczony w tzw. miejscu charakterystycznym akumulatora (punkt 2.2.5); poza tym czytnik

samoczynnie przesuwa taśmę, o ile na niej wydziurkowany jest symbol odstępu wejścia i zatrzymuje się na pierwszym rządku, na którym wydziurkowany jest symbol różny od odstępu wejścia. Czytnik czytający taśmę perforowaną neguje pierwszą pozycję rządka taśmy, ponadto każda z pozycji odczytanych przez wejście może być blokowana

Tablica 2-5

Kod dziurkarki i wejścia

Lp.	Symbole na klawiaturze dziurkarki	Kod na taśmie perforowanej	Kod odczytywany przez fotoczytnik
0	„odstęp"	0 0 0 0 0	czytnik nie czyta
1	X	0 0 0 1 1	1 0 0 1 1
2	N	0 0 1 0 1	1 0 1 0 1
3	H	0 0 1 1 0	1 0 1 1 0
4	L	0 0 1 1 1	1 0 1 1 1
5	M	0 1 0 0 0	1 1 0 0 0
6	S	0 1 0 0 1	1 1 0 0 1
7	G	0 1 0 1 0	1 1 0 1 0
8	B	0 1 0 1 1	1 1 0 1 1
9	D	0 1 1 0 0	1 1 1 0 0
10	F	0 1 1 0 1	1 1 1 0 1
11	J	0 1 1 1 0	1 1 1 1 0
12	K	0 1 1 1 1	1 1 1 1 1
13	0 (zero)	1 0 0 0 0	0 0 0 0 0
14	1	1 0 0 0 1	0 0 0 0 1
15	2	1 0 0 1 0	0 0 0 1 0
16	3	1 0 0 1 1	0 0 0 1 1
17	4	1 0 1 0 0	0 0 1 0 0
18	5	1 0 1 0 1	0 0 1 0 1
19	6	1 0 1 1 0	0 0 1 1 0
20	7	1 0 1 1 1	0 0 1 1 1
21	8	1 1 0 0 0	0 1 0 0 0
22	9	1 1 0 0 1	0 1 0 0 1
23	+	1 1 0 1 0	0 1 0 1 0
24	—	1 1 0 1 1	0 1 0 1 1
25	:	1 1 1 0 0	0 1 1 0 0
26	=	1 1 1 0 1	0 1 1 0 1

(tzn. niezależnie od wyniku czytania wejście stawia na zablokowaną pozycję zero) za pomocą specjalnego układu przełączników (punkt 2.2.6). Taśmę dziurkujemy na dziurkarce dalekopisowej o odpowiednio dobranym kodzie. Kod dziurkarki jest przedstawiony w tabl. 2-5.

2.2.2. Wyjście. Wyjściem z maszyny jest odbiornik dalekopisowy (arkuszowy) odpowiednio przystosowany, prędkość drukowania do 10 znaków na sekundę. Wyjście jest uruchomione za pomocą specjalnego rozkazu maszyny, na który zostaje wydruko-

wana zawartość tzw. miejsca charakterystycznego akumulatora (punkt 2.2.5), po zanegowaniu pierwszej pozycji. Kod wyjścia jest podany w tabl. 2-6.

Tablica 2-6

Kod wyjścia

Lp.	Kod wewnętrzny maszyny	Kod zewnętrzny dalekopisu	Symbole drukowane
0	1 0 0 0 0	0 0 0 0 0	niewykorzystane
1	1 0 0 0 1	0 0 0 0 1	powrót karetki
2	1 0 0 1 0	0 0 0 1 0	nowy wiersz
3	1 0 0 1 1	0 0 0 1 1	X /
4	1 0 1 0 0	0 0 1 0 0	odstęp
5	1 0 1 0 1	0 0 1 0 1	N ,
6	1 0 1 1 0	0 0 1 1 0	H niewykorzystane
7	1 0 1 1 1	0 0 1 1 1	L)
8	1 1 0 0 0	0 1 0 0 0	M .
9	1 1 0 0 1	0 1 0 0 1	S ' (apostrof)
10	1 1 0 1 0	0 1 0 1 0	G niewykorzystane
11	1 1 0 1 1	0 1 0 1 1	B ?
12	1 1 1 0 0	0 1 1 0 0	D niewykorzystane
13	1 1 1 0 1	0 1 1 0 1	F niewykorzystane
14	1 1 1 1 0	0 1 1 1 0	J dzwonek
15	1 1 1 1 1	0 1 1 1 1	K (
16	0 0 0 0 0	1 0 0 0 0	P 0
17	0 0 0 0 1	1 0 0 0 1	Q 1
18	0 0 0 1 0	1 0 0 1 0	W 2
19	0 0 0 1 1	1 0 0 1 1	E 3
20	0 0 1 0 0	1 0 1 0 0	R 4
21	0 0 1 0 1	1 0 1 0 1	T 5
22	0 0 1 1 0	1 0 1 1 0	Y 6
23	0 0 1 1 1	1 0 1 1 1	U 7
24	0 1 0 0 0	1 1 0 0 0	I 8
25	0 1 0 0 1	1 1 0 0 1	O 9
26	0 1 0 1 0	1 1 0 1 0	Z +
27	0 1 0 1 1	1 1 0 1 1	A —
28	0 1 1 0 0	1 1 1 0 0	C :
29	0 1 1 0 1	1 1 1 0 1	V =
30	0 1 1 1 0	1 1 1 1 0	przestawia na drukowanie cyfr i znaków
31	0 1 1 1 1	1 1 1 1 1	przestawia na drukowanie liter

2.2.3. Pamięć. Pamięcią wewnętrzną w maszynie typowej jest bęben magnetyczny, wirujący z prędkością 100 obrotów na sekundę. Na bębnie znajdują się 32 ścieżki robocze, z których każda podzielona jest na 32 komórki po 34B (słowa długie), które z kolei dzielą się na dwie komórki po 17B (słowa krótkie); traktując sprawę formalnie, na bębnie mamy nie 32 ścieżki, lecz aż 128 ścieżek; wynika to z faktu, że na każdej

ścieżce słowa długie nagrywamy w czterech fazach(1), wobec czego ze względu na czas oczekiwania (punkt 1.1.6) musimy rozważyć 128 ścieżek zamiast 32. Ponadto na bębnie znajdują się jeszcze trzy ścieżki zegarowe, którymi tu nie będziemy się zajmowali. Jak widać z powyższego, pamięć jest podzielona na 1024 komórki po 34B (lub, jak kto woli, na 2048 komórek po 17B). Każdej komórce przyporządkowany pewien numer (który pozwala ją odnaleźć w pamięci) nazywamy dalej adresem komórki pamięci lub krócej adresem.

Obecnie omówimy system numeracji komórek pamięci. Każdej komórce 17 B przyporządkowano kolejną liczbę naturalną w systemie ósemkowym(2) począwszy od 0000′ do 3777′(3). Komórki 34 B, jak już wspominaliśmy, składają się z dwóch komórek 17B; komórki 17B wchodzące w skład 34B mają adresy: pierwszy parzysty, drugi nieparzysty. Komórce 34 B przyporządkowujemy adres parzystej komórki 17B, wchodzącej w skład komórki 34B powiększony o liczbę ósemkową 4000′. Tak więc adresy komórek 34B przebiegają liczby ósemkowe od 4000′ do 7776′ co 0002′.

Tak na przykład komórka długa o adresie 5760 składa się z dwóch komórek krótkich o adresach 1760 i 1761. Podobnie komórka długa o adresie 7326 składa się z dwóch komórek krótkich o adresach 3326 i 3327.

Należy w tym miejscu wspomnieć o tym, że ze względu na szeregową budowę maszyny bardziej znacząca część słowa długiego znajduje się w komórce 17B o adresie nieparzystym, mniej znacząca zaś część słowa długiego znajduje się w komórce 17B o adresie parzystym.

2.2.4. Sterowanie. Sterowanie składa się z elementów, jak rejestry, pętle operacji itd. Zasadniczymi elementami, które musimy omówić, aby zrozumieć kod rozkazowy maszyny, są: rejester L zwany inaczej *licznikiem rozkazów* (punkt 1.1.2.1), oraz rejestr D zwany inaczej *rejestrem dyspozytora*. Zanim zajmiemy się omówieniem pracy tych rejestrów poświęcimy trochę miejsca na zapoznanie się z budową rozkazów.

Rozkazy w naszej maszynie są kodowane w słowach krótkich (17B), pięć najbardziej znaczących pozycji binarnych przeznaczono na kod operacji (rodzaj operacji), 12 pozostałych pozycji zostało wykorzystane w następujący sposób: najbardziej znacząca pozycja binarna została wykorzystana jako znak adresu długiego (w systemie ósemko-

(1) Inaczej mówiąc, za ostatnim bitem pierwszego słowa długiego nagrywamy kolejno: ostatni bit ósmego słowa długiego, ostatni bit szesnastego słowa długiego, ostatni bit dwudziestego czwartego słowa długiego, następnie nagrywamy przedostatni bit pierwszego słowa długiego itd.

(2) Wprowadzenie liczb ósemkowych ma na celu skrócenie zapisu, zamiast pisać trzy cyfry binarne — piszemy jedną cyfrę ósemkową. Przejście od systemu binarnego do ósemkowego jest natychmiastowe, wartość cyfry ósemkowej e wyznaczamy z wzoru

$$e = 4\alpha_i + 2\alpha_{i-1} + \alpha_{i-2}$$

gdzie α_i, α_{i-1}, α_{i-2} są to kolejne trzy cyfry binarne, które zastępujemy jedną cyfrą ósemkową.

(3) W dalszym ciągu dla rozróżniania liczb binarnych ósemkowych i dziesiętnych przyjmujemy następującą konwencję: część ułamkową od całkowitej oddzielamy w liczbach binarnych kropką, w liczbach ósemkowych — apostrofem, a w liczbach dziesiętnych — przecinkiem. Dla liczb całkowitych w przypadkach wątpliwych na końcu liczby binarnej stawiamy kropkę, na końcu zaś liczby ósemkowej — apostrof.

wym odpowiada mu liczba 4000') pozostałych 11 przeznaczono na zapis adresów słów krótkich (za pomocą 11 pozycji binarnych można zapisać w systemie ósemkowym liczby od 0000' do 3777'). Widzimy, że rozkaz ma następującą budowę:

$$5\,\text{B} + 1\,\text{B} + 11\,\text{B} = 17\,\text{B}.$$

Zajmiemy się obecnie opisem rejestru dyspozytora — jest to rejestr na 17 B, poszczególne pozycje oddziałują na poszczególne zespoły sterowania zgodnie z podziałem funkcyjnym rozkazu, a więc pierwszych pięć pozycji rejestru uruchamia poprzez tzw. matrycę operacji pętlę poszczególnych operacji, szósta pozycja służy do określania, czy wybieramy z pamięci adres słowa krótkiego, czy też długiego, pozostałych jedenaście służy do określania adresu w pamięci albo parametrów operacji (szczegóły w punkcie 2.3).

Omówimy bliżej licznik rozkazów, jest to rejestr na 11 B; licznik rozkazów służy do zapamiętania adresów, pod którymi znajdują się kolejno wykonywane rozkazy. Rejestr L jest powiązany z półsumatorem, który służy do powiększania zawartości rejestru L o jedynkę, ponadto istnieją drogi przesyłania zawartości rejestru L do części adresowej rejestru D i odwrotnie do przesyłania części adresowej rejestru D do rejestru L oraz drogi przesyłania zawartości rejestru L do pamięci i odwrotnie.

Postaramy się obecnie wyjaśnić zasady pracy maszyny. Maszyna pracuje dwutaktowo.

Pierwszy takt pracy. Zawartość licznika rozkazów zostaje przesłana do części adresowej rejestru dyspozytora, a następnie zostaje powiększona o jedynkę. Rozkaz o części operacyjnej równej zeru i o adresie słowa krótkiego (jest to tzw. rozkaz pobrania z pamięci, patrz punkt 2.3.4) powoduje pobranie słowa zapisanego pod wskazanym adresem i umieszczenie go w rejestrze dyspozytora.

Drugi takt pracy. Zostaje wykonany rozkaz znajdujący się w rejestrze dyspozytora.

Po drugim takcie powtarza się pierwszy itd.

2.2.5. Arytmometr. W skład arytmometru wchodzi kilka rejestrów i układów. Ze względu na opis, potrzebny nam dla zrozumienia działania poszczególnych rozkazów, wymienimy następujące elementy arytmometru:

1) rejestr podstawowy zwany akumulatorem (w skrócie oznaczany A) o długości 68 B, pozycje akumulatora będziemy numerowali zgodnie z numeracją pozycji liczb binarnych w arytmetyce uzupełnieniowej (punkt 2.1.1);

2) rejestr mnożnej (w skrócie oznaczany M) o długości 34 B;

3) rejestr sygnału W (w skrócie oznaczany W) o długości 1 B;

4) rejestr sygnału N (w skrócie oznaczany N) o długości 1 B;

5) układ realizujący uzupełnienie liczby (w skrócie zwany uzupełnieniem);

6) układ realizujący sumę dwóch liczb (w skrócie zwany sumatorem).

Obecnie pokrótce omówimy rejestry A i M.

Akumulator. Słowa długie z pamięci możemy przesyłać do 34 bardziej znaczących pozycji akumulatora i, odwrotnie, zawartość bardziej znaczących 34 pozycji możemy przesyłać do pamięci (oczywiście pod adres słowa długiego), podobnie słowa krótkie z pamięci maszyny możemy przesyłać na 17 najbardziej znaczących pozycji akumulatora i, odwrotnie, zawartość 17 najbardziej znaczących pozycji akumulatora możemy prze-

syłać do pamięci (oczywiście pod adres słowa krótkiego). Możemy również przesuwać arytmetycznie zawartość akumulatora w lewo albo w prawo o ilość pozycji mniejszą lub równą 63, podobnie możemy przesuwać zawartość akumulatora cyklicznie[1]) w lewo albo w prawo o ilość pozycji mniejszą lub równą 63. Ponadto możemy przesyłać zawartość akumulatora przez sumator i wynik umieszczać z powrotem w akumulatorze (oczywiście w sumatorze dokonujemy sumowania zawartości akumulatora z zawartością dowolnego adresu pamięci n), możliwe jest również przesyłanie z pamięci przez układ uzupełniania do akumulatora. W przypadku przekroczenia zakresu bit przekroczenia zakresu nie ginie, lecz zostaje umieszczony na specjalnej dodatkowej pozycji akumulatora; po przesunięciu zawartości akumulatora w kierunku pozycji najmniej znaczących, zawartość pozycji przekroczenia zakresu zostaje umieszczona na pozycji zerowej akumulatora, ponadto bit ten bierze udział w operacji dodawania i odejmowania, w pozostałych operacjach bit ten nie bierze udziału. Pięć najbardziej znaczących pozycji akumulatora będziemy nazywali miejscem charakterystycznym akumulatora, na to miejsce wprowadzimy rządek taśmy perforowanej odczytany przez wejście oraz zawartość tych pięciu pozycji przesyłamy na wyjście.

Rejestr mnożnej. Możliwe jest tylko przesyłanie z pamięci do rejestru mnożnej; zawartość komórek krótkich jest przesyłana z pamięci na 17 bardziej znaczących pozycji rejestru M.

2.2.6. Pulpit sterowania. Służy do manipulacji ręcznych oraz obserwacji pracy maszyny, przyczyn zatrzymania itp. Na pulpicie sterowania znajdują się neonówki i przełączniki. W Załączniku 2 zajmiemy się omówieniem roli poszczególnych neonówek i przełączników (Załącznik 2 umieszczony jest na końcu książki).

2.3. KOD ROZKAZOWY

2.3.0. Dla potrzeb dalszych rozważań wprowadzimy kilka definicji:

Definicja 1. Będziemy mówili, że n jest adresem krótkim, zamiast mówić, że n jest adresem słowa krótkiego.

Definicja 2. Będziemy mówili, że n jest adresem długim, zamiast mówić, że n jest adresem słowa długiego.

Definicja 3. Będziemy mówili krótko — zawartość adresu n, zamiast mówić słowo zapisane pod adresem n.

Definicja 4. Będziemy mówili krótko — przesłanie słowa (z pamięci do któregoś z rejestrów albo odwrotnie), zamiast mówić przesłanie słowa (z pamięci do któregoś z rejestrów albo odwrotnie) i zapisanie go (w którymś z rejestrów albo w którejś z komórek pamięci) na miejscu poprzedniej zawartości, przy czym zawartość miejsca, z którego przesyłamy (komórki pamięci albo któregoś z rejestrów), nie ulega zmianie.

Definicja 5. Będziemy mówili krótko — sekwencja rozkazów, zamiast mówić skończony ciąg rozkazów umieszczony pod kolejnymi adresami w pamięci, takich,

---

[1]) Przez przesuwanie cykliczne rozumiemy takie przesuwanie, przy którym zakładamy, że najbardziej znaczący bit α_0 i najmniej znaczący bit α_{67} są sąsiednimi bitami.

że maszyna musi je wykonywać jeden po drugim w kolejności wzrastających adresów (pod którymi zapisane są rozkazy) począwszy od pierwszego wyrazu ciągu, niezależnie od zawartości komórek pamięci i rejestrów wykorzystanych w tym ciągu obliczeń.

Definicja 6. Zawartość adresu n będziemy oznaczali krótko (n), podobnie zawartość akumulatora A będziemy oznaczali (A), zawartość rejestru M będziemy oznaczali (M), zawartość rejestru L będziemy oznaczali (L) i zawartość rejestru D będziemy oznaczali (D).

2.3.1. Operacje przesyłania. Operacje przesyłania służą do przesyłania liczb (rozkazów) z pamięci do rejestrów A i M oraz zawartości rejestru A do pamięci. Obecnie omówimy kolejno te operacje i podamy proste przykłady ich zastosowania.

Plus przesyłanie: 1) gdy n jest adresem długim, prześlij zawartość adresu n do 34 bardziej znaczących pozycji akumulatora, 2) gdy n jest adresem krótkim, prześlij zawartość adresu n do 17 najbardziej znaczących pozycji akumulatora. Jeśli liczba przesłana do akumulatora jest różna od zera, generuj sygnał $W = 1$, jeśli zaś jest zerem, generuj sygnał $W=0$. Generuj sygnał $N=0$.

Kod operacji: 12 n.

Zapis symboliczny: $(n) \rightarrow A$.

Minus przesyłanie: 1) gdy n jest adresem długim, prześlij uzupełnienie zawartości adresu n do 34 bardziej znaczących pozycji akumulatora, 2) gdy n jest adresem krótkim, prześlij uzupełnienie zawartości adresu n do 17 najbardziej znaczących pozycji akumulatora. Jeśli liczba przesyłana do akumulatora jest różna od zera, generuj sygnał $W = 1$, jeśli zaś jest zerem, generuj sygnał $W = 0$. Jeśli nastąpiło przekroczenie zakresu, generuj sygnał $N = 1$.

Kod operacji: 13 n.

Zapis symboliczny: $-(n) \rightarrow A$.

Przesyłanie do pamięci: 1) gdy n jest adresem długim, prześlij zawartość 34 bardziej znaczących pozycji akumulatora do komórki pamięci o adresie n, 2) gdy n jest adresem krótkim, prześlij zawartość 17 najbardziej znaczących pozycji akumulatora do komórki pamięci o adresie n. Jeśli zawartość akumulatora jest ujemna, generuj sygnał $W = 1$, jeśli zawartość akumulatora jest nieujemna, generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Uwaga: W wyniku operacji przesyłanie do pamięci zawartości akumulatora nie ulega zmianie.

Kod operacji: 14 n.

Zapis symboliczny: $(A) \rightarrow n$.

Przesyłanie do M: 1) gdy n jest adresem długim, prześlij zawartość adresu n do rejestru mnożnej M; 2) gdy n jest adresem krótkim, prześlij zawartość adresu n do 17 bardziej znaczących pozycji rejestru mnożnej M. Jeśli liczba przesłana do rejestru mnożnej M jest ujemna, generuj sygnał $W = 1$, jeśli jest nieujemna, generuj sygnał $W = 0$. Generuj sygnał $N=0$.

Kod operacji: 27 n.

Zapis symboliczny: $(n) \rightarrow M$.

Rozpatrzmy obecnie przykłady zastosowania powyższych operacji.

Przypuśćmy, że chcemy przesłać zawartość komórki pamięci o adresie n do komórki pamięci o adresie m. Wykonujemy to w sposób następujący:

1) $(n) \rightarrow A$,
2) $(A) \rightarrow m$.

Załóżmy, że w komórce pamięci o adresie n mamy jakąś liczbę l (co do modułu mniejszą od 1), przypuśćmy, że dla pewnych celów potrzebna nam jest nie liczba l, ale liczba $-l$, która musi być umieszczona w komórce pamięci o adresie m. Wykonujemy wtedy operacje:

1) $-(n) \rightarrow A$,
2) $(A) \rightarrow m$.

2.3.2. Operacje arytmetyczne (dokończenie). *Dodawanie*: 1) gdy n jest adresem długim, do 34 bardziej znaczących pozycji akumulatora dodaj zawartość adresu n; 2) gdy n jest adresem krótkim, do 17 najbardziej znaczących pozycji akumulatora dodaj zawartość adresu n. Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś jest liczbą nieujemną, generuj sygnał $W = 0$. Jeśli zaś nastąpiło przekroczenie zakresu, generuj sygnał $N = 1$.

Kod operacji: 10 n.

Zapis symboliczny: $(A) + (n) \rightarrow A$.

Odejmowanie: 1) gdy n jest adresem długim od 34 bardziej znaczących pozycji akumulatora odejmij zawartość adresu n; 2) gdy n jest adresem krótkim, od 17 najbardziej znaczących pozycji akumulatora odejmij zawartość adresu n. Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś liczbą nieujemną, generuj sygnał $W = 0$. Jeśli nastąpiło przekroczenie zakresu, generuj sygnał $N = 1$.

Kod operacji: 11 n.

Zapis symboliczny: $(A) - (n) \rightarrow A$.

Mnożenie: pomnóż zawartość adresu n przez zawartość rejestru mnożnej M; wynik umieść w akumulatorze. Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś jest liczbą nieujemną, generuj sygnał $W = 0$. Jeśli nastąpiło przekroczenie zakresu, generuj sygnał $N = 1$[1]).

Uwaga: wykonanie operacji mnożenia nie zmienia zawartości rejestru M.

Kod operacji: 16 n.

Zapis symboliczny: $(M) \cdot (n) \rightarrow A$.

Zaokrąglenie: wykonaj na zawartości akumulatora operacje zaokrąglenia, według algorytmu podanego w punkcie 2.1.6. Jeśli zawartość akumulatora jest ujemna, generuj sygnał $W = 1$, jeśli zawartość akumulatora jest nieujemna, generuj sygnał $W = 0$. Jeśli zaś w wyniku zaokrąglenia nastąpiło przekroczenie zakresu, generuj sygnał $N = 1$.

Kod operacji: 15 n.

Zapis symboliczny: $z(A) \rightarrow A$.

---

[1]) Przy mnożeniu przekroczenie zakresu może nastąpić tylko w przypadku pomnożenia -1 przez -1, wówczas wynik mnożenia równa się $+1$ (punkt 2.1.5).

Dzielenie: Zawartość akumulatora podziel przez zawartość adresu n, umieszczając iloraz na 34 bardziej znaczących pozycjach akumulatora, resztę zaś umieszczając na 34 mniej znaczących pozycjach akumulatora (iloraz i reszta są traktowane jako dwa niezależne słowa długie). Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś wynik jest liczbą nieujemną, generuj sygnał $W = 0$. Jeśli $|(A)| \geqslant |(n)|$, generuj sygnał $N = 1$, przy czym dzielenie nie zostaje wykonane.

Uwaga: W przypadku gdy maszyna nie wykonuje dzielenia, przechodzi ona do następnego taktu pracy.

Kod operacji: 17 n.

Zapis symboliczny: $(A) : (n) \rightarrow A$.

Tworzenie wartości bezwzględnej: zastąp zawartość akumulatora bezwzględną wartością zawartości akumulatora. Jeśli liczba zawarta w akumulatorze jest różna od zera, generuj sygnał $W = 1$, jeśli zaś jest równa zeru, generuj sygnał $W = 0$. Jeśli nastąpiło przekroczenie zakresu(1) generuj sygnał $N = 1$.

Kod operacji: 26 n.

Zapis symboliczny: $|(A)| \rightarrow A$.

Rozpatrzmy obecnie przykłady zastosowania powyższych operacji. We wszystkich rozpatrywanych przykładach zakładamy, że w wyniku wykonywanej operacji nie nastąpiło przekroczenie zakresu. Przypuśćmy, że chcemy dodać zawartość adresu n_1 do zawartości adresu n_2 i wynik umieścić pod adresem n_3 — wykonujemy to w następujący sposób:

1) $(n_1) \rightarrow A$,
2) $(A) + (n_2) \rightarrow A$,
3) $(A) \rightarrow n_3$.

Przypuśćmy z kolei, że chcemy pomnożyć zawartość adresu n_1 przez zawartość adresu n_2 wynik mnożenia zaokrąglić i przesłać pod adres n_3; wykonujemy to w następujący sposób:

1) $(n_1) \rightarrow M$,
2) $(M) \cdot (n_2) \rightarrow A$,
3) $z(A) \rightarrow A$,
4) $(A) \rightarrow n_3$.

Dzielenie zawartości adresu n_1 przez zawartość adresu n_2 i umieszczenie wyniku pod adresem n_3 wykonujemy w następujący sposób:

1) $(n_1) \rightarrow A$,
2) $(A) : (n_2) \rightarrow A$,
3) $(A) \rightarrow n_3$.

Zastanówmy się jeszcze, jak obliczyć różnice wartości bezwzględnych zawartości adresu n_1 i n_2, a wynik umieścić pod adresem n_3. Dla wykonania powyższego zadania potrzebna nam jest jeszcze jedna pomocnicza komórka pamięci (tzw. komórka robocza) o adresie m. Nasze zadanie rozwiązujemy wówczas następująco:

---

(1) Przekroczenie zakresu przy obliczaniu wartości bezwzględnej może nastąpić tylko w przypadku, gdy $(A) = -1$.

1) $(n_2) \rightarrow A$,
2) $|(A)| \rightarrow A$,
3) $(A) \rightarrow m$,
4) $(n_1) \rightarrow A$,
5) $|(A)| \rightarrow A$,
6) $(A) - (m) \rightarrow A$,
7) $(A) \rightarrow n_3$.

Do grupy operacji arytmetycznych zaliczamy jeszcze dwie operacje.

Mnożenie przez 2^n: zawartość akumulatora pomnóż przez 2^n. Liczba n nie może być większa od 63 (adres Mod 64). Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś wynik jest liczbą nieujemną, generuj sygnał $W = 0$. Jeśli nastąpiło przekroczenie zakresu, generuj sygnał $N = 1$.

Kod operacji: 20 n.

Zapis symboliczny: $(A) \cdot 2^n \rightarrow A$.

Dzielenie przez 2^n: zawartość akumulatora pomnóż przez 2^{-n}. Liczba n nie może być większa od 63 (adres Mod 64). Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś wynik jest liczbą nieujemną, generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Kod operacji: 21 n.

Zapis symboliczny: $(A) : 2^n \rightarrow A$.

Zastanówmy się obecnie nad przykładem zastosowania operacji mnożenia przez 2^{-n}. Przypuśćmy, że chcemy pomnożyć zawartość adresu n_1 przez $\frac{1}{64}$ i wynik przesłać pod adres n_2; możemy zrealizować to dwoma sposobami:

a) korzystając z operacji mnożenia, pomnożyć zawartość adresu n_1 przez $\frac{1}{64}$ i wynik przesłać pod adres n_2,

b) skorzystać z operacji dzielenia przez 2^n, a mianowicie

1) $(n_1) \rightarrow A$,
2) $(A) : 2^6 \rightarrow A$,
3) $(A) \rightarrow n_2$;

przy korzystaniu z operacji mnożenia musimy pamiętać jeszcze liczbę $\frac{1}{64}$, a wykonanie naszego zadania dłużej by trwało, ponieważ we wszystkich trzech operacjach występują czasy oczekiwania. Natomiast korzystając z drugiego sposobu mamy czas oczekiwania tylko w dwóch operacjach, mianowicie w pierwszej i trzeciej.

2.3.3. Operacje logiczne. Przez operacje logiczne będziemy rozumieli operacje wykonywane na zawartościach akumulatora, przy czym każda pozycja akumulatora jest traktowana jako oddzielna zmienna zero-jedynkowa.

Koniunkcja: weź koniunkcję (1) zawartości akumulatora i zawartości adresu n.

(1) Przez koniunkcję dwu liczb binarnych α_i, β_i rozumiemy operację określoną następującymi wzorami:

$$\alpha_i \cap \beta_i = \begin{cases} 0, & \text{jeśli } \alpha_i = 0, \\ \beta_i, & \text{jeśli } \alpha_i = 1. \end{cases}$$

Uwaga: Jeśli adres n jest adresem krótkim, to maszyna traktuje to jakby na 17 mniej znaczących pozycjach zawartości adresu długiego były same zera.

Jeśli wynik jest liczbą różną od zera, generuj sygnał $W = 1$, jeśli zaś wynik jest zerem, generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Kod operacji: 30 n.

Zapis symboliczny: $(A) \cap (n) \rightarrow A$.

Przesuwanie cykliczne (patrz odnośnik do punktu 2.2.5) *w prawo*: zawartość akumulatora przesuń o n miejsc binarnych cyklicznie w prawo. Liczba n nie może być większa od 63 (adres Mod 64). Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś wynik jest liczbą nieujemną, generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Kod operacji: 22 n.

Zapis symboliczny: $(A) : 2_n \rightarrow A$.

Przesuwanie cykliczne w lewo: zawartość akumulatora przesuń o n miejsc binarnych cyklicznie w lewo. Liczba n nie może być większa od 63 (adres Mod 64). Jeśli wynik jest liczbą ujemną, generuj sygnał $W = 1$, jeśli zaś wynik jest liczbą nieujemną, generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Kod operacji: 23 n

Zapis symboliczny: $(A) \cdot 2_n \rightarrow A$.

Rozpatrzmy przykład zastosowania powyższych operacji: przypuśćmy, że dla pewnego celu musimy w liczbie 17B, znajdującej się pod adresem krótkim n_1, położyć na pozycje binarne 7, 8, 9, 10 zera i wynik umieścić pod adresem krótkim n_2. Jak to wykonać? Zadanie to rozwiązujemy stosując operację koniunkcji; w tym celu załóżmy, że pod adresem krótkim m znajduje się liczba binarna, która ma na pozycjach 0, 1, 2, 3, 4, 5, 6 jedynki, na pozycjach 7, 8, 9, 10 zera i na pozycjach 12, 13, 14, 15, 16 jedynki.

1) $(n_1) \rightarrow A$,
2) $(A) \cap (m) \rightarrow A$,
3) $(A) \rightarrow n_2$.

Jak wiemy, w wyniku podzielenia dwóch liczb (oczywiście, o ile dzielenie to jest wykonalne) otrzymujemy wynik na 34 bardziej znaczących pozycjach akumulatora, reszta z dzielenia zaś zostaje umieszczona na 34 mniej znaczących pozycjach akumulatora. Przypuśćmy, że wynik dzielenia chcemy przesłać pod adres n_1, resztę zaś pod adres n_2 (oba adresy są oczywiście adresami długimi), możemy to wykonać na dwóch drogach:

a. Korzystając z operacji przesuwania cyklicznego w prawo

1) $(A) \rightarrow n_1$,
2) $(A) : 2_{34} \rightarrow A$,
3) $(A) \rightarrow n_2$.

Operację koniunkcji maszyna wykonuje na zawartości akumulatora:

$$\alpha_0 \cdot \alpha_1 \alpha_2 \ldots \alpha_{33}$$

i zawartości adresu n

$$\beta_0 , \beta_1 \beta_2 \ldots \beta_{33}$$

realizując koniunkcję kolejnych par α_i, β_i dla $i = 0, 1, 2, \ldots, 33$.

b. Korzystając z operacji przesuwania cyklicznego w lewo

1) $(A) \rightarrow n_1$,

2) $(A) \cdot 2_{34} \rightarrow A$,

3) $(A) \rightarrow n_2$.

2.3.4. Operacje organizacyjne. *Czytanie:* Odczytaj kolejny rządek taśmy perforowanej z uwzględnieniem położenia przełączników „blokada wejścia" (patrz Załącznik 2, pulpit sterowania) i wynik zsumuj logicznie z zawartości miejsca charakterystycznego akumulatora, przesuwając jednocześnie cyklicznie poprzednią zawartość akumulatora o pięć pozycji binarnych w prawo. Jeśli na którymś z uwarunkowanych miejsc (za pomocą przełączników „warunków wejścia Y", patrz Załącznik 2) pojawi się jedynka, to generuj sygnał $W = 1$, jeśli zaś na żadnym z uwarunkowanych miejsc nie pojawi się „jedynka" albo jeśli żadna pozycja nie jest uwarunkowana, generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Kod operacji: 24 n.

Zapis symboliczny: czytaj.

Drukowanie: wydrukuj znak odpowiadający zawartości miejsca charakterystycznego akumulatora, z uwzględnieniem położenia przełączników „blokada wyjścia" (patrz Załącznik 2, pulpit sterowania). Jeśli na którymś z uwarunkowanych miejsc za pomocą przełączników „warunek wyjścia Z" (patrz Załącznik 2) pojawi się jedynka, to generuj sygnał $W = 1$, jeśli zaś na żadnym z uwarunkowanych miejsc nie pojawi się jedynka albo jeśli żadna pozycja nie jest uwarunkowana, to generuj sygnał $W = 0$. Generuj sygnał $N = 0$.

Kod operacji: 25 n.

Zapis symboliczny: drukuj.

Przykładów na zastosowanie tych dwóch rozkazów nie będziemy omawiali, omówimy je dopiero przy programach wprowadzających i wyprowadzających w rozdz. 5.

Skok: do licznika rozkazów L prześlij część adresową wykonywanego rozkazu (tzn. następny rozkaz pobierz z komórki o adresie n). Nie zmieniaj zawartości rejestrów N i W.

Kod operacji: 02 n.

Zapis symboliczny: $n \rightarrow L$.

Powyższy rozkaz ma następujące zastosowanie: przypuśćmy, że po wykonaniu pewnej sekwencji rozkazów znajdujących się pod adresami $k, \ldots, k + p$, chcemy przejść do sekwencji, której pierwszy rozkaz znajduje się pod adresami n, wówczas pod adresem $k + p + 1$ umieszczamy rozkaz skokowy,

$$n \rightarrow L,$$

po którego wykonaniu maszyna zacznie wykonywać kolejne rozkazy sekwencji zaczynającej się pod adresem n.

Skok z podstawieniem: do licznika rozkazów L prześlij część adresową słowa odczytanego pod adresem n. Nie zmieniaj zawartości rejestrów N i W.

Kod operacji: 01 n.

Zapis symboliczny: $(n) \rightarrow L$.

Rozkaz skoku z podstawieniem ma podobne zastosowanie jak rozkaz skoku, z tą różnicą, że w powiązaniu z dalej omówionym rozkazem skoku ze śladem daje bardzo wygodną organizację wywoływania podprogramów.

Skok ze śladem: zawartość licznika rozkazów L prześlij pod adres n, do licznika rozkazów L prześlij zaś $n + 1$. Nie zmieniaj zawartości rejestrów N i W.

Kod operacji: 04 n.

Zapis symboliczny: $(L) \rightarrow n$,
$n + 1 \rightarrow L$.

Rozkaz ten jest ciekawym rozwiązaniem organizacyjnym, daje on duże ułatwienie przy organizacji programu głównego (rozdz. 4).

Rozpatrzmy przykład zastosowania rozkazów skok ze śladem i skok z podstawieniem. Przypuśćmy, że mamy sekwencję rozkazów umieszczonych pod adresami od $n + 1$ do $n + p$, którą w trakcie wykonywania programu wykonujemy wielokrotnie. W celu wywołania w odpowiednim miejscu programu głównego tej sekwencji, umieszczamy w programie głównym rozkaz skok ze śladem (o części adresowej równej n). Wykonanie tego rozkazu powoduje zapisanie pod adresem n zawartości licznika rozkazów, po czym maszyna zaczyna wykonywać kolejne rozkazy sekwencji począwszy od rozkazu zapisanego pod adresem $n + 1$. Jeżeli pod adresem $n + p + 1$ umieścimy rozkaz skok z podstawieniem (o części adresowej równej n), to po wykonaniu ostatniego rozkazu sekwencji (o adresie $n + p + 1$) maszyna pobierze rozkaz zapisany pod adresem $n + p + 1$, który spowoduje powrót do programu głównego, według stanu licznika rozkazów zapisanego pod adresem n.

Pierwszy skok warunkowy: jeżeli zawartość rejestru $W = 1$, to część adresową wykonywanego rozkazu prześlij do licznika rozkazów L; jeżeli zaś zawartość rejestru $W = 0$, nie zmieniaj zawartości licznika rozkazów L.

Uwaga: Rozkaz pierwszego skoku warunkowego zeruje rejestr W i nie zmienia zawartości rejestru N.

Kod operacji: 03 n.

Zapis symboliczny: $n \rightarrow L$,
$W = 1$.

Zajmiemy się omówieniem przykładów zastosowania pierwszego rozkazu skoku warunkowego. Przypuśćmy, że mamy odjąć od zawartości adresu n_1 zawartość adresu n_2 i jeśli wynik odejmowania jest ujemny (czyli $(n_1) < (n_2)$), przejść do sekwencji rozpoczynającej się od rozkazu zapisanego pod adresem n. W tym zaś przypadku, gdy wynik odejmowania jest nieujemny (czyli $(n_1) \geqslant (n_2)$). Mamy wykonać kolejny rozkaz:

1) $(n_1) \rightarrow A$,
2) $(A) - (n_2) \rightarrow A$,
3) $n \rightarrow L$,
$W = 1$.

Zupełnie analogicznie działa rozkaz drugiego skoku warunkowego.

Drugi skok warunkowy: jeśli zawartość rejestru $W = 0$, to część adresową wykonywanego rozkazu prześlij do licznika rozkazów L; jeśli zaś zawartość rejestru $W = 1$, nie zmieniaj zawartości licznika rozkażów L.

Uwaga: Rozkaz drugiego skoku warunkowego zeruje rejestr W, nie zmienia zawartości rejestru N.

Kod operacji: 06 n.

Zapis symboliczny: $n \rightarrow L$,
$W = 0$.

Skok ze śladem przy nadmiarze: jeśli zawartość rejestru $N = 1$, to zawartość licznika rozkazów (L) prześlij pod adres n, zaś $n + 1$ prześlij do licznika rozkazów L; jeśli natomiast zawartość rejestru $N = 0$, nie przesyłaj zawartości licznika rozkazów L pod adresem n oraz nie zmieniaj zawartości licznika rozkazów.

Uwaga: Rozkaz skoku ze śladem przy nadmiarze zeruje rejestr N i nie zmienia zawartości rejestru W.

Kod operacji: 37 n.

Zapis symboliczny: $(L) \rightarrow n$,
$n + 1 \rightarrow L$,
$N = 1$.

W przypadku gdy przełącznik W10 (patrz pulpit sterowania, Załącznik 2) znajduje się w dolnym położeniu, dla uniknięcia przekroczenia zakresu korzystamy z rozkazu skoku ze śladem przy nadmiarze; rozkaz ten działa podobnie do rozkazu pierwszego skoku warunkowego z tym, że w przypadku $N = 1$ nie pobiera rozkazu stojącego pod adresem n, lecz przesyła — podobnie jak rozkaz skoku ze śladem — zawartość licznika rozkazów pod adres n, po czym pobiera rozkaz stojący pod adresem $n + 1$.

Pobranie rozkazu: pobierz rozkaz zawarty w komórce o adresie krótkim n, wykonaj go i powróć do poprzedniej sekwencji rozkazów, o ile rozkaz wykonywany nie był rozkazem skokowym. Nie zmieniaj zawartości rejestrów N i W.

Kod operacji: 00 n.

Zapis symboliczny: $(n) \rightarrow D$.

Rozkaz ten jest opisem operacji wykonywanym przez maszynę w pierwszym takcie (punkt 2.2.4), jest to pobranie rozkazu do wykonania. W pewnych przypadkach wygodnie jest go stosować w programach.

Stop z pobraniem: prześlij rozkaz według wskazań części adresowej do rejestru dyspozytora D, po czym zatrzymaj maszynę. Po ponownym uruchomieniu w zależności od manipulacji ręcznej albo zostaje wykonany rozkaz znajdujący się w rejestrze D, po którym maszyna pobierze rozkaz według wskazań licznika rozkazów L, albo maszyna pobierze kolejny rozkaz według wskazań licznika L (patrz Załącznik 2). Nie zmieniaj zawartości rejestrów N i W.

Kod operacji: 05 n.

Zapis symboliczny: stop n.