ROCZNIKI POLSKIEGO TOWARZYSTWA MATEMATYCZNEGO
Seria I: WIADOMOŚCI MATEMATYCZNE III (1959)

wmptm 1897

Ministerstwo Nauki i Szkolnictwa Wyższego

A. A. Lapunow i G. A. Szestopał (Moskwa)

Podstawowe informacje o rozwiązywaniu zadań na elektronowych maszynach cyfrowych (*)

Wstęp

Celem artykułu jest zapoznanie czytelnika z działaniem elektronowych automatycznych maszyn matematycznych cyfrowych (EMC). Zaznajomienie się z podstawowymi zasadami ich budowy i działania może okazać się pożyteczne dla specjalistów różnych dziedzin. Ograniczymy nasz wykład do opisu niektórych matematycznych i logicznych umiejętności EMC, nie interesując się zagadnieniem, przy pomocy jakich środków technicznych realizuje się je w konkretnych schematach i konstrukcjach maszyn cyfrowych.

Współczesna nauka i technika jest nierozerwalnie związana z wykorzystaniem EMC. Pojawienie się ich pozwoliło w zasadniczy sposób zmienić cały styl prac obliczeniowych, głównie dzięki ogromnemu zmniejszeniu czasu wykonywania poszczególnych operacji. Jeśli dotychczas na granicy możliwości znajdowały się prace obliczeniowe wymagające miliona operacji arytmetycznych, to teraz z pomocą EMC często rozwiązuje się zadania z miliardami działań arytmetycznych.

Współczesna EMC jest agregatem o bardzo złożonej budowie. Pierwszą charakterystyczną cechą jej budowy jest wykorzystanie elektroniki, umożliwiające wielokrotne zwiększenie szybkości pracy maszyny. W poprzednio konstruowanych maszynach liczących do wykonania poszczególnych operacji wykorzystywało się mechaniczne lub elektromechaniczne liczniki, których zasada działania opierała się na przemieszczeniach pewnych części (obrót kół zębatych, ruch kotwicy przekaźnika elektromagnetycznego itd.). W EMC wszystkie operacje sprowadzają się do zmian stanu urządzeń elektronowych zachodzących niemal błyskawicznie. Współczesna EMC potrafi przeprowadzać dziesiątki tysięcy działań na sekundę.

(*) Математическое просвещение 1 (1957), str. 57-74.

Ministerstwo Nauki i Szkolnictwa Wyższego

Schemat, według którego maszyna rozwiązuje problem obliczeniowy, przypomina schemat czynności człowieka. Jeżeli człowiek „ręcznie" rozwiązuje zadanie obliczeniowe, to zawczasu układa (w myśli lub na papierze — w zależności od stopnia złożoności zagadnienia) pewien plan: *program obliczeń*. Zgodnie z tym programem wykonuje on kolejno działania arytmetyczne — najpierw na danych początkowych, a potem na liczbach pojawiających się w procesie liczenia. Zwykle przy tym powstaje konieczność zapisywania wyników pośrednich. Pewna część tych wyników jest wykorzystywana bezpośrednio w następnym kroku obliczeń, a w dalszych okazuje się ona niepotrzebna; inną część natomiast trzeba zachować, by ją wykorzystać w innych etapach pracy (czasem tylko przy końcu obliczeń). Zdarza się też, że w zależności od uzyskanych w trakcie obliczeń wyników zmienia się metodę rozwiązywania (np. wykorzystuje się inne wzory itp.). Końcowym etapem rozwiązywania każdego zadania jest zapisanie ostatecznych wyników.

Aby wypełniać analogiczne funkcje, EMC powinna posiadać nie tylko urządzenie wykonujące działania arytmetyczne, ale i urządzenie „zapamiętujące" dane wstępne i niezbędne wyniki pośrednie.

Następna cecha EMC, ściśle związana z jej szybkim działaniem, polega na pełnej automatyzacji całego procesu rozwiązywania od początku do końca. Bez tej automatyzacji korzystanie z EMC byłoby bezcelowe.

Wyobraźmy sobie na chwilę, że maszyna może wykonać jedynie pewne serie podobnych obliczeń, zatrzymując się zawsze przed zmianą metody rozwiązywania lub przejściem do obliczania drugiej grupy wielkości itp. Jeżeli nawet pominiemy zagadnienie, w jaki sposób i jak szybko dostarcza się maszynie dane początkowe oraz ile czasu trzeba na jej uruchomienie, to czas potrzebny na odczytanie pośrednich wyników i obmyślenie, jakie należy wykonać następne działanie, jest nieporównanie większy od czasu, jaki EMC zużywa na wykonanie tej pracy. Takie wykorzystywanie EMC zniweczyłoby wszelkie zalety jej szybkiego działania.

Współczesna EMC może rozwiązywać bardzo skomplikowane zadania, składające się z ciągu obliczeń i analizy wyników, nie wymagając interwencji człowieka na żadnym etapie pośrednim. Cała praca EMC jest określona ułożonym uprzednio *programem* (dokładny sens tego terminu wyjaśnimy niżej).

EMC, po uruchomieniu, prowadzi zgodnie z programem proces obliczeń (automatycznie zmieniając jego bieg tam, gdzie to jest potrzebne), „podaje na zewnątrz" gotowy rezultat i zatrzymuje się. Przy tym maszyna traci kilka godzin, a nawet kilka minut na rozwiązanie zadania, które przy „ręcznych" rachunkach wymagałoby wielu lat pracy.

Możliwości EMC są bardzo szerokie i nie ograniczają się tylko do wykonywania obliczeń. Wykorzystuje się EMC do rozwiązywania rozmaitych zadań logicznych: automatycznego przekładu z jednego języka na drugi, rozliczeń buchalteryjnych, analizy ekonomiczno-statystycznej, a także wykorzystuje się je w celu automatycznego kierowania procesami produkcyjnymi, przelotami samolotów itp.

Uniwersalne zastosowanie EMC spowodowało konieczność rozwoju specjalnej stosowanej teorii algorytmów, mianowicie teorii racjonalnych metod układania programów dla EMC. Teoria ta stanowi teraz oddzielną gałąź matematyki stosowanej i nazywa się *programowaniem*.

Istnieje dzisiaj bardzo wiele różnych typów EMC. Toteż dla prostoty wykładu opiszemy pewną *umowną maszynę cyfrową* (UMC), która posiada podstawowe własności konkretnych EMC. Na przykładzie tej maszyny będą wyłożone podstawowe idee i sposoby programowania.

Należy zaznaczyć, że dla każdej konkretnej maszyny istnieje możliwość zastosowania specjalnych metod programowania, wykorzystujących jej indywidualne własności; oczywiście przy opisie UMC nie mogą być wyłożone takie specyficzne metody.

W pierwszej części artykułu podamy opis schematu i zasad działania UMC; druga część zawiera kilka prostych przykładów programowania zadań matematycznych i logicznych.

§ 1. Umowna maszyna cyfrowa (UMC)

Podstawowe części UMC. UMC (jak i każda maszyna cyfrowa) składa się z trzech części podstawowych: *pamięci* (P), wyposażonej w *urządzenie czytające i piszące* (Cz), *arytmometru* (A) i *urządzenia sterującego* (S). Prócz tego UMC posiada urządzenie wprowadzające dane początkowe (*wejście*) i wyprowadzające ostateczne rezultaty (*wyjście*).

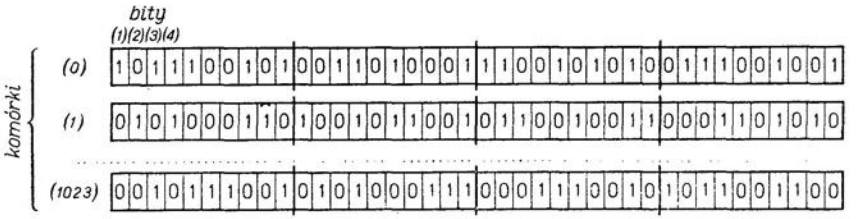
Opiszemy teraz przeznaczenie podstawowych części UMC. Pamięć ma za zadanie wprowadzenie, przechowanie i wyprowadzenie różnych informacji niezbędnych w pracy maszyny. Do informacji zaliczamy: wstępne dane liczbowe, wyniki częściowe, wyniki ostateczne oraz program pracy maszyny, będący spisem operacji ułożonych w pewnej kolejności. Informacja wprowadzona do pamięci może zawierać również inne dane (np. zasady gramatyki i słownik, co jest potrzebne do przekładu z jednego języka na drugi).

Zasada działania pamięci, zdolnej operować tak różnorodnymi informacjami, jest oparta na wykorzystaniu specjalnych kodów. Przy pomocy tych kodów najróżnorodniejsze pojęcia, jak liczby, nazwy działań itd. — wyrażone są przy pomocy kombinacji dwóch symboli „0”

Ministerstwo Nauki i Szkolnictwa Wyższego

i „1". W ten sposób zrealizowanie pamięci sprowadza się do stworzenia układu elementów zdolnych znajdować się w jednym z dwóch stanów(1). Stany te odpowiadają symbolom „0" i „1". Wprowadzenie elementu w odpowiedni stan (zapisanie „0" lub „1") wykonuje urządzenie czytające i piszące.

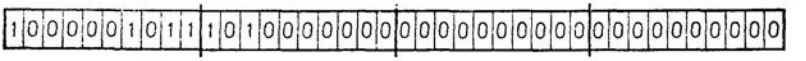
Każdy element pamięci nazywamy *bitem*. Bity są łączone w uporządkowane grupy, po 40 bitów w każdej, przy czym każdemu bitowi jest przyporządkowany numer od 1 do 40. Grupy te nazywamy *komórkami*



Rys. 1. Schemat budowy pamięci

pamięci. Przypuśćmy, dla ustalenia uwagi, że pamięć zawiera $2^{10} = 1024$ komórki raz na zawsze ponumerowane liczbami od 0 do 1023 (rys. 1). W ten sposób w każdej komórce pamięci może być zapisana kombinacja 40 symboli przyjmujących wartości 0 lub 1. Każdą taką kombinację nazywać będziemy *słowem*.

Rozpatrzymy najpierw budowę słowa przedstawiającego liczbę. Pierwszy bit takiego słowa wskazuje znak liczby: 0 oznacza +, a 1 oznacza —. Bity od 2-go do 10-go przedstawiają część całkowitą licz-



Rys. 2. Przedstawienie liczby —11,625

by zapisanej w układzie dwójkowym; bity od 11-go do 40-go przedstawiają część ułamkową liczby również w układzie dwójkowym(2). W ten sposób przecinek jest umieszczony między 10-tym a 11-tym bitem. Na przykład słowo na rysunku 2 przedstawia liczbę —11,625.

---

(1) Elementami takimi są np. przerzutniki, rdzenie ferromagnetyczne, odcinki taśmy magnetycznej lub bębna magnetycznego itd.

(2) Przypominamy, że przedstawienie liczby ξ w układzie dwójkowym $\xi = a_n a_{n-1} \dots a_1, \beta_1 \beta_2 \dots \beta_s$, gdzie cyfry α_i i β_j są równe 0 lub 1, ma sens następujący:

$$\xi = \sum_{i=1}^{n} a_i \cdot 2^{i-1} + \sum_{j=1}^{s} \beta_j \cdot 2^{-j}.$$

Ministerstwo Nauki i Szkolnictwa Wyższego

Podkreślamy raz jeszcze, że słowo może przedstawiać nie tylko liczbę, lecz każdą inną informację; przykłady takich słów podamy niżej. Słowo, znajdujące się w dowolnej komórce pamięci, przy pomocy urzą-

TABLICA 1

Operacje UMC

nr	Nazwa operacji	Symbol	Definicja	liczbowy kod operacji
		I. Operacje arytmetyczne[3]		
1	dodawanie	+	$x+y=z$	1[4]
2	odejmowanie	−	$x-y=z$	10
3	mnożenie	×	$x\times y=z$	11
4	dzielenie	:	$x:y=z$	100
		II. Operacje logiczne		
1	dodawanie logiczne	$\vee$	$x_k\vee y_k=\max(x_k,y_k)=z_k$	1000
2	mnożenie logiczne	$\wedge$	$x_k\wedge y_k=\min(x_k,y_k)=z_k$	1001
3	nierównoważność	$\nsim$	$x_k\nsim y_k=(\bar{x}_k\wedge y_k)\vee(x_k\wedge\bar{y}_k)=z_k$[5]	1010
4	przesłanie liczby	PL	$x=z$	1011
5	przesunięcie	$\rightarrow$	$x_k=z_{k+i}$	1100[6]
		III. Operacje zmiany sterowania		
1	bezwarunkowa zmiana sterowania	BS	Sterowanie przekazać wg III adresu: zawsze	10000
2	warunkowa zmiana sterowania I	WS_1	jeżeli $x<y$	10001
3	warunkowa zmiana sterowania II	WS_2	jeżeli $x\neq y$	10010
4	warunkowa zmiana sterowania III	WS_3	jeżeli $x=y$	10011
5	warunkowa zmiana sterowania IV	WS_4	jeżeli $\|x\|>\|y\|$	10100
		IV. Pozostałe operacje sterujące		
1	drukuj	DR	wydrukować z	11000
2	stop	ST	zatrzymać maszynę	11111

(3) x, y, z traktowane są jak liczby. Zakłada się zwykłe zaokrąglanie wyniku. Jeżeli wynik wychodzi poza przedziały komórki, maszyna zatrzymuje się.

(4) Piszemy 1 zamiast cyfr 0000000001 stojących w kodzie operacji.

(5) $\bar{u}$ oznacza negację u: jeżeli $u=1$, to $\bar{u}=0$, i odwrotnie. $z_k=x_k\nsim y_k$, tj. $z_k=1$, jeżeli $x_k\neq y_k$ i $z_k=0$, jeżeli $x_k=y_k$.

(6) Liczbowy kod wyraża tylko symbol operacji przesunięcia; w drugim adresie rozkazu koduje się o ile bitów ma być przesunięte słowo (niżej wyjaśnimy to dokładniej).

Ministerstwo Nauki i Szkolnictwa Wyższego

dzenia czytającego i piszącego (Cz) może być przeczytane i przekazane do którejkolwiek części maszyny: do arytmometru, urządzenia sterującego, do innej komórki pamięci lub do wyjścia. Przeczytane słowo w dalszym ciągu pozostaje w komórce. Jeżeli jednak Cz umieszcza słowo w jakiejś komórce, to automatycznie zostaje „wymazane" słowo, które się uprzednio w niej znajdowało.

Ministerstwo Nauki i Szkolnictwa Wyższego

Arytmometr (A). Cała praca maszyny składa się z szeregu taktów, z których każdy jest wykonaniem jakiejś elementarnej operacji na słowach. Te operacje elementarne wykonuje A. Każda EMC ma własny komplet operacji elementarnych określony przeznaczeniem i budową maszyny.

Spis elementarnych operacji UMC podzielony jest na cztery grupy (patrz tablica 1). Każda operacja elementarna jest dokonywana na jednym lub dwóch słowach i daje w rezultacie nowe słowo. Jeżeli oznaczymy słowa literami $x, y, z, \ldots$, to każdą operację elementarną można zapisać w postaci

$$z = \varphi(x) \quad \text{lub} \quad z = f(x, y),$$

gdzie φ i f są symbolami operacji. Zajmiemy się najpierw dwiema pierwszymi grupami: operacjami arytmetycznymi i logicznymi.

UMC wykonuje cztery operacje arytmetyczne: dodawanie, odejmowanie, mnożenie i dzielenie. Polegają one na wykonywaniu odpowiednich działań na słowach interpretowanych jako zapisy liczb. W przypadku gdy x i y rzeczywiście są liczbami, operacje arytmetyczne mają zwykły sens. Jeżeli jednak x i y są jakimikolwiek innymi informacjami, to można wykorzystać operacje arytmetyczne do przekształcania słów, którego konkretny sens zależy od treści słów x, y i z. Jak zobaczymy dalej, takie przekształcenia słów często stosuje się przy programowaniu.

Do przekształcania słów służą także operacje logiczne. Do operacji logicznych należą: dodawanie logiczne, mnożenie logiczne(7), nierównoważność, przesłanie liczby i przesunięcie.

Oznaczmy przez x_k, y_k, z_k k-te bity słów x, y, z ($k = 1, 2, \ldots, 40$). Mówimy, że słowo z otrzymujemy w wyniku *dodawania logicznego* słów x i y ($z = x \vee y$), gdy

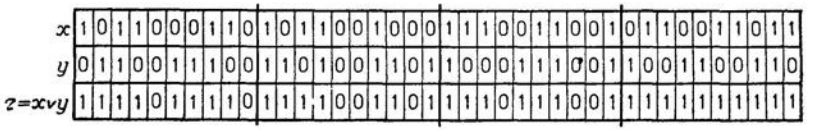
$$z_k = 0, \quad \text{jeżeli } x_k = 0 \text{ i } y_k = 0,$$

$$z_k = 1 \quad \text{w pozostałych przypadkach.}$$

(7) Pochodzenie tych terminów jest związane z tym, że jeżeli jedynką oznaczymy zdanie prawdziwe, a zerem fałszywe, to podane niżej reguły działań na bitach słów nie różnią się od reguł algebry zdań w logice matematycznej.

Inaczej mówiąc

$$z_k = \max(x_k, y_k) \quad \text{(patrz rys. 3).}$$



Rys. 3. Dodawanie logiczne

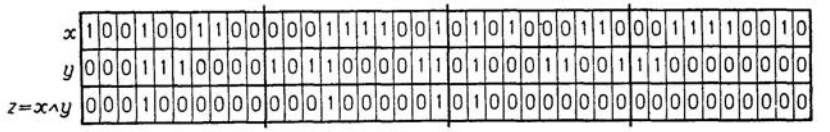
Słowo z nazywamy *iloczynem logicznym* słów x i y ($z = x \wedge y$), jeżeli

$$z_k = 1, \quad \text{gdy } x_k = 1,\ y_k = 1,$$

$$z_k = 0 \quad \text{w pozostałych przypadkach.}$$

Inaczej

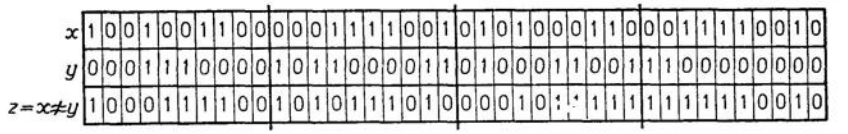
$$z_k = \min(x_k, y_k) \quad \text{(patrz rys. 4).}$$



Rys. 4. Mnożenie logiczne

Operacja *nierównoważności* ($x \nsim y$) (patrz rys. 5) jest określona przez warunki

$$\begin{cases} z_k = 1, & \text{jeżeli } x_k \neq y_k, \\ z_k = 0, & \text{jeżeli } x_k = y_k. \end{cases}$$



Rys. 5. Nierównoważność

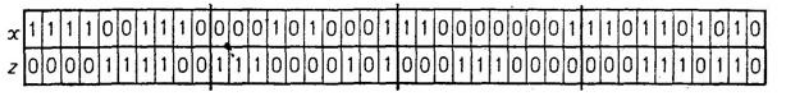
Przesłanie liczby jest operacją wykonywaną na jednym słowie: $z = x$. W końcu *przesunięcie* o i bitów (również operacja na jednym słowie) jest określone warunkiem

$$z_{k+i} = x_k,$$

przy czym jeżeli $k < 1$ lub $k > 40$, to $z_k = 0$. Liczba i może być zarówno dodatnia, jak i ujemna, przy czym jeżeli $i > 0$, to mamy przesunięcie w prawo, jeżeli $i < 0$, to przesunięcie w lewo (rys. 6).

Ministerstwo Nauki i Szkolnictwa Wyższego

Przy końcu artykułu opiszemy dokładniej mechanizm realizacji operacji przesunięcia w UMC. Inne operacje elementarne wykonywane przez A zostaną opisane w następnym paragrafie.



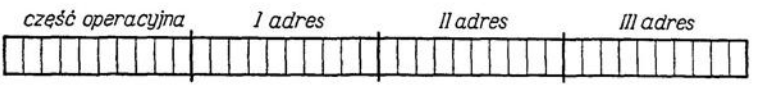
Rys. 6. Przesunięcie ($i = 4$)

Urządzenie sterujące (S) koordynuje całą pracę maszyny. Jego funkcje sprowadzają się do wykonywania następujących czynności: a) uruchomienie A i nastawienie go na wykonywanie określonej operacji; b) przesyłanie słowa z komórki pamięci do A; c) przesyłanie słowa z A do komórki pamięci; d) przesyłanie słowa z jednej komórki pamięci do drugiej; e) przesyłanie słowa z komórki pamięci do S; f) przesyłanie słowa z komórki pamięci do wyjścia; g) kierowanie kolejnością operacji.

Pracę S określają tzw. rozkazy, wychodzące z pamięci maszyny.

Rozkazy i program. *Rozkazem* nazywamy słowo przedstawiające operację elementarną na słowach: $z = \varphi(x)$ lub $z = f(x, y)$. Jeżeli φ lub f są operacjami arytmetycznymi, to i rozkaz nazywamy *arytmetycznym*. Oprócz rozkazów arytmetycznych istnieją też rozkazy *sterujące*.

Rozpatrzymy sposób zapisywania podstawowych rozkazów. Każde słowo przedstawiające rozkaz składa się z czterech grup dziesięciobitowych (rys. 7). Pierwsza z tych grup nazywa się *częścią operacyjną*. Znaj-



Rys. 7. Schemat rozkazu

duje się w niej numer wykonywanej operacji elementarnej. Trzy następne grupy bitów nazywamy *adresami*. Adres jest to numer komórki pamięci zapisany 10-ciobitową liczbą. Pierwszy adres (I) jest numerem komórki, w której znajduje się słowo x, drugi adres (II) — numer komórki ze słowem y, a trzeci (III) — numer komórki, w której ma być umieszczony wynik z. Czasem nie wszystkie adresy wykorzystuje się, np. nie używamy adresu II, jeżeli operacja w danym rozkazie ma postać $z = \varphi(x)$.

Wszystkie rozkazy, tak jak i pozostałe słowa, są przechowywane w pamięci maszyny. Każdemu rozkazowi jest przyporządkowany numer tej komórki, w której znajduje się on w pamięci. Ciąg ponumerowanych rozkazów nazywa się *programem*.

Ministerstwo Nauki i Szkolnictwa Wyższego

Ministerstwo Nauki i Szkolnictwa Wyższego

Prześledzimy, co dzieje się, gdy w S znajduje się rozkaz arytmetyczny. Część operacyjna zostaje przesłana do A i ustawia go na wykonywanie danej operacji, pierwszy i drugi adres (jeżeli $z = f(x, y)$) pozostaje w S. Następnie słowa x i y, przechowywane w komórkach pamięci, których numery podają adresy pierwszy i drugi, są przesyłane do A. Arytmometr realizuje operację $z = f(x, y)$. W końcu trzeci adres jest przekazany do Cz, dzięki czemu obliczone w A słowo z może być przesłane do komórki o numerze wskazanym przez trzeci adres.

Teraz zajmiemy się rolą rozkazów sterujących. Rozpatrzymy najpierw warunkowe rozkazy zmiany sterowania (WS). Podstawowe prawo kierowania pracą maszyny polega na tym, że po wykonaniu rozkazu o numerze a wykonuje się rozkaz $a+1$. Jednak w szeregu przypadków można odstąpić od tej zasady. Zdarza się to np. wtedy, gdy w procesie liczenia dojdziemy do miejsca, w którym wybór następnej operacji zależy od wyniku poprzedniej. Podczas układania programu takie „rozwidlenie" dróg dalszych obliczeń powinno być specjalnie omówione i wskazane zasady wyboru właściwej drogi. W języku programów oznacza to, że ze zbioru rozkazów powinien być wybrany jeden, ściśle określony rozkaz. Taki wybór jednego rozkazu spośród wielu zawsze może być sprowadzony do kolejnych wyborów jednego z dwóch rozkazów. Warunkowy rozkaz zmiany sterowania (WS) jest instrumentem, przy pomocy którego dokonuje się wyboru jednego z dwóch rozkazów.

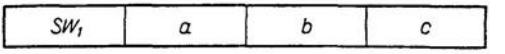
Rozkaz WS ma, mówiąc ogólnie, następującą treść: „jeżeli między słowami x i y, znajdującymi się w komórkach wskazanych pierwszym i drugim adresem rozkazu, zachodzi pewna relacja M określona w części operacyjnej rozkazu, to jako kolejny rozkaz należy wziąć ten, którego numer jest w trzecim adresie; jeżeli M nie zachodzi, to po danym rozkazie WS należy wykonywać rozkaz umieszczony w programie bezpośrednio po nim".

Istnieje wiele typów rozkazów WS w zależności od treści warunku M. Do spisu elementarnych operacji UMC włączyliśmy cztery typy rozkazów WS, chociaż zasadniczo można by się ograniczyć do jednego. (Takie ograniczenie doprowadziłoby do pewnego skomplikowania programów).

W rozkazie WS_1 badamy relację $x < y$, inaczej mówiąc, rozkaz WS_1 ma następującą treść: „jeżeli liczba x jest mniejsza od liczby y, to następnym rozkazem jest ten, którego numer wskazuje trzeci adres, jeżeli $x \geqslant y$, to należy wykonać kolejny rozkaz".

Schemat rozkazu WS_1 został pokazany na rys. 8, gdzie a oznacza numer komórki zawierającej słowo x, b — numer komórki zawierającej słowo y oraz c — numer odpowiedniego rozkazu.

Rozkaz WS_2 bada relację $x \neq y$, rozkaz WS_3 — relację $x = y$, a rozkaz WS_4 — relację $|x| > |y|$.



Rys. 8. Schemat rozkazu WS_1

Ministerstwo Nauki i Szkolnictwa Wyższego

Należy zaznaczyć, że podczas wszelkich badań relacji występujących w rozkazach WS słowa x i y są traktowane jako słowa-liczby. Jeżeli x i y są rzeczywiście liczbami, to takie porównywanie ma zwykły sens, jeżeli x i y zawierają inne informacje, to, w zależności od ich charakteru, badanie określone rozkazem WS może mieć inną konkretną treść. Badania relacji zachodzących między słowami x i y dokonuje się w A.

Oprócz rozkazów WS w programach mogą występować *bezwarunkowe rozkazy zmiany sterowania* (BS), spełniające następującą funkcję: „jeżeli w S znajduje się rozkaz BS, to bezpośrednio po nim do S wejdzie rozkaz, którego numer wskazuje trzeci adres rozkazu BS". (Nie wykorzystuje się w BS pierwszego i drugiego adresu).

Rozkazy zmiany sterowania stanowią trzecią grupę operacji elementarnych UMC.

Spośród pozostałych operacji sterujących, tworzących czwartą grupę operacji elementarnych, wyróżnimy rozkaz „stop" (ST), który zatrzymuje maszynę i rozkaz „drukuj" (DR), wyprowadzający na zewnątrz zawartość komórki pamięci wskazanej w pierwszym adresie tego rozkazu.

Zwrócimy uwagę na jeszcze jedną zasadę układania programów. Przy programowaniu dąży się zwykle do tego, by wykorzystywać tę samą część programu wielokrotnie, zmieniając tylko adresy. W tym celu do programów dołącza się tzw. *rozkazy przeadresowywania*. Rozkazy te zapewniają zmianę adresów rozkazów wykonywanych wcześniej, po to, by rozkazy te mogły być znów wykonywane, ale już na słowach zapisanych w komórkach o innych adresach. Tę własność programowania objaśnimy dokładniej na przykładach([8]).

Oprócz rozkazów arytmetycznych i sterujących programy zawierają jeszcze *słowa pomocnicze*: wstępne dane liczbowe oraz pewne stałe niezbędne w procesie liczenia. Na przykład, podczas rozwiązywania zadania metodą iteracyjną, rozkaz WS może porównywać słowo $|x|$, charakteryzujące

([8]) W UMC operacji przeadresowania można dokonać przy pomocy operacji dodawania i odejmowania. W wielu konkretnych maszynach cyfrowych liczby są przedstawiane inaczej niż w UMC i w związku z tym do przeadresowywania niezbędne są specjalne operacje.

dokładność rozwiązania, z daną liczbą dodatnią ε i w zależności od wyniku spowodować powtórzenie procesu iteracyjnego lub zatrzymać maszynę. Liczba ε jest w tym przypadku słowem pomocniczym.

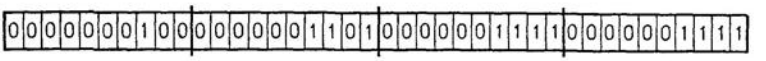
Na stronicy 133 podaliśmy spis wszystkich operacji elementarnych UMC. Symbole x i y oznaczają tam słowa przechowywane w komórkach o numerach zgodnych z pierwszym i drugim adresem rozkazu; z oznacza słowo powstałe w A i umieszczone w komórce, której numer wskazuje trzeci adres rozkazu; przez x_k, y_k, z_k rozumiemy odpowiednio k-te bity słów x, y, z.

Podczas programowania zapisuje się rozkazy według kolejności numerów. Zamiast kodu operacji pisze się zwykle jej symbol, a numery komórek w adresach podaje się w systemie dziesiętnym.

Niech na przykład czwarty rozkaz programu ma następującą postać: „słowo umieszczone w 13-tej komórce podziel przez słowo zapisane w 15-tej komórce i prześlij wynik do komórki 15-tej". Zapis tego rozkazu wygląda następująco:

4	:	13	15	15

Przed wprowadzeniem programu do pamięci rozkazy są kodowane według podanych wyżej zasad. Przytoczony wyżej rozkaz przyjmuje formę słowa przedstawionego na rysunku 9.



Rys. 9. Przedstawienie rozkazu 4 w postaci słowa

Po wprowadzeniu programu do pamięci UMC i przesłaniu pierwszego rozkazu do S następuje kolejne automatyczne wykonywanie dalszych rozkazów. Po rozkazie arytmetycznym następuje kolejny rozkaz programu. Po rozkazie WS jest wykonany albo rozkaz, którego numer znajduje się w trzecim adresie danego WS, albo rozkaz następny, bezpośrednio po nim umieszczony w programie. Gdy w S pojawi się rozkaz „drukuj", zostaje wyprowadzona na zewnątrz pewna liczba. Tak przebiega praca, dopóki w S nie pojawi się rozkaz „stop".

Możliwe jest także awaryjne zatrzymanie się maszyny. Zdarza się to wówczas, gdy operacja, jaką dyktuje S, okazuje się niemożliwa do wykonania. Sygnalizuje to albo błąd w programie, albo pomyłkę (uszkodzenie) w pracy maszyny.

§ 2. Przykłady programów

Rozpatrzmy kilka przykładów prostych programów.

PRZYKŁAD 1. Program obliczania kwadratów kolejnych liczb naturalnych od 1 do n (tablica 2).

Ministerstwo Nauki i Szkolnictwa Wyższego

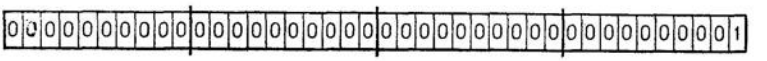
W programie tym muszą być przewidziane następujące czynności: a) obliczenie kwadratów n kolejnych liczb naturalnych, b) zgromadzenie tych liczb w n uprzednio zarezerwowanych komórkach pamięci, c) automatyczne zatrzymanie maszyny po zakończeniu tej pracy.

TABLICA 2(9)

1.	+	7	8	7
2.	×	7	7	10
3.	+	2	9	2
4.	WS_1	2	6	1
5.		ST		
6.	×	7	7	$10+n$
7.		„0"		
8.		„1"		
9.	0	0	0	1

W ten sposób program zajął dziewięć komórek pamięci: ostateczne wyniki (kwadraty liczb) będą gromadzić się w komórkach 10, 11, 12,..., $10+(n-1)$.

Zanim maszyna rozpocznie pracę, umieszczamy w 7-ej komórce liczbę 0, podczas liczenia będą się tam pojawiać i inne liczby. W 8-ej komórce umieszczamy liczbę 1; w 9-ej umieszczamy rozkaz, w którym wszystkie bity z wyjątkiem trzeciego adresu są zerami, a w trzecim adresie jest umieszczona liczba 1 (rys. 10).



Rys. 10. Dziewiąta komórka pamięci

Maszyna pracuje według tego programu w sposób następujący:

Wykonuje się rozkaz 1: „do słowa zapisanego w 7-ej komórce dodaj słowo z komórki 8-ej i wynik umieść w komórce 7-ej". W ten sposób w 7-ej komórce pojawia się liczba 1 ($0+1=1$).

Wykonujemy 2-gi rozkaz. W 10-tej komórce zostaje umieszczona liczba 1 ($1 \times 1 = 1$).

Wypełniamy 3-ci rozkaz. W 2-ej komórce pojawia się rozkaz:

×	7	7	11

Ponieważ część operacyjna i dwa pierwsze adresy 9-ej komórki zawierają zera, w komórce 2-ej zostały nie zmienione część operacyjna i dwa pierwsze adresy.

Rozkaz 3-ci jest rozkazem przeadresowania (operacja liczbowa nie na liczbach, a na rozkazach). Rozkaz ten tak przekształca rozkaz 2-gi, aby podczas jego kolejnego wykonywania wynik umieszczonej w nim operacji (obliczanie kwadratu liczby 2) był zapisany do nowej komórki. Użyte w tym celu dodawanie rozkazów należy rozumieć jak zwykłe dodawanie liczb.

(9) Umieszczone w 7-ej i 8-ej komórce znaki „0" i „1" oznaczają kody liczb 0 i 1.

Ministerstwo Nauki i Szkolnictwa Wyższego

Maszyna wykonuje rozkaz 4. Jest to rozkaz WS_1, porównuje się zawartość 2-ej i 6-ej komórki. Słowa znajdujące się w tych komórkach są traktowane jako liczby. Dopóki liczba przechowywana w 6-ej komórce jest większa od liczby z 2-ej komórki, następuje „skok" do komórki 1-ej. Porównywanie jest tu wprowadzone po to, by upewnić się, czy rozwiązywanie zadania nie zostało zakończone. Skok warunkowy umożliwił wykorzystanie programu do dalszego rozwiązywania zadania.

Następnie znów wykonuje się rozkazy 1-4. Rozkaz 1 służy teraz do obliczania nowej wartości liczby podnoszonej do kwadratu. Przy tym w 2-ej komórce pojawia się rozkaz:

2.	×	7	7	12

W komórce 7-ej znajduje się liczba 2, a kwadrat drugiej liczby naturalnej, 4, zostaje przesłany do 11-ej komórki.

Po rozkazie 4-tym znów następuje powrót do rozkazu 1-go. Po kolejnym wykonaniu rozkazów 1-4 tak wygląda zawartość zmienionych komórek:

2.	×	7	7	13
7.	„3"			
12.	„9"			

Proces ten będzie się powtarzał dopóty, dopóki w 2-ej komórce nie pojawi się rozkaz

2.	×	7	7	$n+1$

wówczas po rozkazie 4-tym nastąpi rozkaz 5-ty, który zatrzyma maszynę. W komórkach od 10 do $10+(n-1)$ znajdują się kwadraty n kolejnych liczb naturalnych.

Przykład 2. Program obliczania e^x (tablica 3).

Rozpatrzmy szereg

$$e^x = 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots$$

Będziemy obliczać sumy częściowe tego szeregu i prowadzić rachunek dotąd, dopóki kolejny wyraz szeregu nie będzie co do modułu mniejszy lub równy pewnej z góry zadanej liczbie $\varepsilon > 0$.

Przekształćmy szereg w sposób następujący

$$e^x = \sum_{n=0}^{\infty} u_n, \quad \text{gdzie} \quad u_0 = 1, \quad u_{n+1} = u_n x/(n+1).$$

Ministerstwo Nauki i Szkolnictwa Wyższego

TABLICA 3

1.	PL	10	—	13
2.	PL	10	—	15
3.	PL	12	—	14
4.	+	15	14	15
5.	+	13	12	13
6.	:	14	13	14
7.	×	14	11	14
8.	WS_4	14	16	4
9.		ST		

Zajmiemy następujące komórki: W komórkach od 1 do 9 umieścimy program rozwiązywania zadania; w komórce o numerze 10 umieścimy liczbę 0, w 11-tej — liczbę x, w 12-tej — liczbę 1, w 16-tej — liczbę ε. Komórki 13 i 14 zarezerwujemy na tzw. „komórki robocze", w których będą zapisywane częściowe wyniki. Wynik ostateczny znajdzie się w 15 komórce.

Objaśnimy działanie tego programu. W wyniku wykonania rozkazów 1-3 do komórek 13 i 15 zostaje przesłana liczba 0 (jest to niezbędne, ponieważ przed rozpoczęciem pracy mogłyby tam się znajdować jakieś liczby), a do komórki 14-tej liczba 1 równa wartości wyrazu u_0. Rozkaz 4-ty dodaje do zawartości komórki 15-tej (liczba 0) wartość u_0, teraz więc w 15-ej komórce będzie również znajdować się u_0 (początkowa wartość sumy szeregu). W 5-tym rozkazie do zawartości komórki 13 zostaje dodana 1, a więc zawartość komórki 13 jest równa 1. Rozkaz 6-ty przeprowadza dzielenie u_0 przez 1 i wynik zapisuje w 14-tej komórce; 7-my rozkaz oblicza pierwszy wyraz szeregu $u_1 = u_0 x/1$ i umieszcza go w 14-tej komórce; 8-my porównuje $|u_1|$ z liczbą ε: jeżeli $|u_1| > \varepsilon$, nakazuje powrót do rozkazu 4. Wówczas znów wykonuje się rozkazy od 4-go do 8-go. (Jeżeli $|u_1| \leqslant \varepsilon$, to zostaje wykonany rozkaz 9-ty i maszyna zatrzymuje się).

Przy powtórnym wykonywaniu rozkazów 4-8, w 15-tej komórce pojawia się liczba $u_0 + u_1$ (4-ty rozkaz), w 13-tej — liczba 2 (5-ty rozkaz), w 14-tej — najpierw liczba $u_1/2$ (6-ty rozkaz), a następnie liczba $u_1 x/2 = u_2$ (7-my rozkaz).

W ten sposób w następujących po sobie cyklach w komórce 14-tej będą się pojawiać kolejno wyrazy naszego szeregu: $u_0, u_1, u_2, \ldots$ W komórce 15-tej natomiast będą wartości kolejnych sum częściowych szeregu, będących przybliżonymi wartościami e^x. Maszyna dotąd będzie pracowała, dopóki (jak tego wymagają warunki zadania) kolejny wyraz szeregu nie będzie co do modułu mniejszy lub równy ε. Kiedy to nastąpi, maszyna zatrzyma się automatycznie. Ostateczny rezultat będzie się znajdował w 15-tej komórce pamięci.

Przykład 3. Program wykorzystujący operacje logiczne.

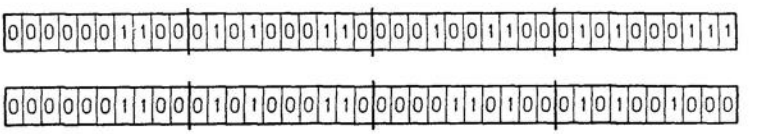
Przeanalizujemy strukturę rozkazu przesunięcia UMC. W pierwszym adresie tego rozkazu jest zapisany adres komórki zawierającej przesuwane słowo. W drugim adresie umieszcza się pewną całkowitą liczbę β. Liczba ta zapisana w układzie dwójkowym, mająca nie więcej niż 7 bitów różnych od zera, określa wielkość i kierunek przesunięcia. Zauważmy,

Ministerstwo Nauki i Szkolnictwa Wyższego

że zawsze $\beta = 2^6 + i$, gdzie $-2^6 \leqslant i \leqslant 2^6 - 1$. Przy tym przez $|i|$ rozumiemy ilość bitów, o którą ma być dokonane przesunięcie; $i > 0$ oznacza przesunięcie w prawo, a $i < 0$ — przesunięcie w lewo. Ponieważ i może zmieniać się w granicach od -39 do 39, każde przesunięcie może być zakodowane w powyższy sposób. (Nie wystarczyłaby nam ilość bitów mniejsza od 7).

Liczba $\beta = 2^6$ oznacza więc przesunięcie zerowe. Jedynka na najbardziej znaczącym (w tym przypadku 7-mym od strony prawej) miejscu świadczy o przesunięciu w prawo, a 0 na tym miejscu — o przesunięciu w lewo.

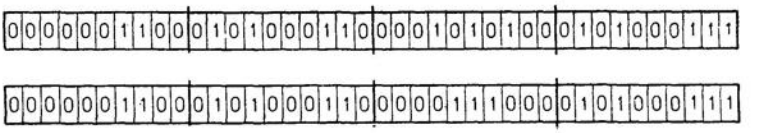
Przykłady na rysunku 11 przedstawiają komórki pamięci, zawierające rozkazy przesunięcia zawartości 326 komórki o 12 bitów w prawo (rys. 11a) i w lewo (rys. 11b).



Rys. 11. a) rozkaz przesunięcia zawartości komórki 326 o 12 bitów w prawo (wynik zapisuje się w komórce 327); b) rozkaz przesunięcia zawartości komórki 326 o 12 bitów w lewo (wynik zapisuje się w komórce 328)

Taki sposób kodowania rozkazu przesunięcia jest szczególnie wygodny dlatego, że pozwala zmieniać wielkość i kierunek przesunięcia wykorzystując operacje arytmetyczne: dodawanie i odejmowanie. Mianowicie, aby dokonać dodatkowego przesunięcia o j miejsc w prawo (w lewo), należy do drugiego adresu w komórce zawierającej rozkaz przesunięcia dodać (odjąć) liczbę j.

Tak na przykład, aby utworzyć uzupełnione przesunięcie zawartości komórki 326 (której pierwotne przesunięcie o 12 bitów w prawo jest pokazane na rys. 11a) o osiem bitów w prawo, wystarczy w komórce zawierającej rozkaz przesunięcia dodać do drugiego adresu liczbę 8; aby zbudować nowe uzupełnione przesunięcie o 28 bitów w lewo, wystarczy odjąć liczbę 28 od zawartości drugiego adresu w otrzymanej komórce. Rysunek 12 przedstawia zawartość komórek otrzymanych w wyniku tych operacji.



Rys. 12. a) uzupełniony rozkaz przesunięcia komórki 326 o 8 bitów w prawo; b) powtórnie uzupełniony rozkaz przesunięcia komórki 326 o 28 bitów w lewo

Ministerstwo Nauki i Szkolnictwa Wyższego

Zaprogramujemy teraz następującą „grę" z dwoma „graczami". Bierzemy pod uwagę dwa 40-bitowe słowa. Pierwsze słowo A wybiera się dowolnie i zapisuje w jakiejś komórce pamięci a. Wszystkie bity drugiego słowa C, oprócz k-tego, są równe zeru, k-ty bit równy jest jedynce. Słowo to zapisuje się w komórce z numerem c.

Każdy „ruch" gry polega na tym, że słowo C przesuwa się o pewną ilość bitów. Ruch ten określa się w następujący sposób. W słowie A wyodrębnia się siedem najmniej znaczących bitów w drugim adresie. Otrzymaną w ten sposób liczbę β przyjmuje się za kod żądanego przesunięcia. (Może to być przesunięcie w lewo lub w prawo — w zależności od tego, czy na najbardziej znaczącym miejscu znajduje się 0 czy 1). Przed następnym ruchem przesuwa się słowo A o jeden bit w prawo i wyodrębnia się na nowo (mówiąc ogólnie inne) siedem „najmłodszych" bitów drugiego adresu, które znów uważa się za kod przesunięcia C w następnym „ruchu" gry. W ten sposób następnym ruchem C będzie przesunięcie, mówiąc ogólnie, o inną liczbę bitów (i, być może, także w inną stronę). „Gra" kończy się wtedy, gdy w słowie C jedynka, początkowo stojąca na k-tym miejscu, wyjdzie za granice komórki. Jeżeli wyjdzie ona z prawej strony komórki, wygrywa pierwszy „gracz", jeżeli z lewej, wygrywa drugi.

Pokażemy program, według którego maszyna mogłaby przeprowadzić tę grę, wskazać wygrywającego i zatrzymać się, gdy gra się skończy.

Niech w komórce a będzie zapisane słowo A; w komórce b — liczba składająca się z jedynek na 7-miu najmłodszych miejscach drugiego adresu, a zer na pozostałych; w komórce c — liczba C; w d — liczba zero; w komórce $e+1$ — zakodowane zerowe przesunięcie, tj. w jej drugim adresie zapisana liczba 2^6, a na pozostałych miejscach zera. Komórka f ma budowę następującą

f	→	c	0	c

Ciąg rozkazów jest zapisany w tablicy 4.

TABLICA 4

1.	$\wedge$	a	b	e	W komórce e wyodrębnia się tylko drugi adres słowa A (na pozostałych miejscach pojawiają się zera).
2.	+	f	e	3	Po wykonaniu tego rozkazu w 3-ej komórce pojawia
3.					się rozkaz przesunięcia zawartości komórki c o wskazaną przez warunki gry liczbę bitów.
4.	→	a	2^6+1	a	Wykonuje się rozkaz przesunięcia słowa A o jeden bit w prawo.
5.	WS_1	d	c	1	Bada się, czy „gra" skończyła się, tj. czy słowo C jest zerem; jeżeli nie, to „grę" prowadzi się dalej i znów należy wykonać pierwszy rozkaz.

Ministerstwo Nauki i Szkolnictwa Wyższego

Rozkazy podane w tablicy 5 są wykonywane tylko po faktycznym zakończeniu „gry", przy ich pomocy określi się „wygrywającego".

TABLICA 5

6.	$\wedge$	3	$e+1$	h	Bada się kierunek ostatniego przesunięcia: jeżeli w prawo, to w komórce h pojawia się jedynka, jeżeli w lewo, to w komórce h pojawia się zero.
7.	DR	h			Wyprowadza się na zewnątrz zawartość komórki h, co pozwala ustalić, kto „wygrał"; jeżeli będzie wydrukowana 1, to wygrał pierwszy gracz, jeżeli 0, to drugi.
8.	ST				

Ministerstwo Nauki i Szkolnictwa Wyższego

Na zakończenie podamy bez szczegółowych komentarzy jeszcze jeden przykład programu.

Przykład 4. Program na obliczanie wartości wielomianu według schematu Hornera.

Weźmy wielomian

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 = \left(\ldots\left((0 + a_n)x + a_{n-1}\right)x + \ldots + a_1\right)x + a_0.$$

Będziemy literami oznaczać adresy niektórych komórek. W tablicy 6 jest podany program na obliczanie wartości wielomianu $f(x)$.

TABLICA 6

Nr komórki	Rozkazy			
	kod	I	II	III
1.	PL	c	—	a
2.	$\times$	a	d	a
3.	$+$	a	$c+1$	a
4.	$+$	3	$d+2$	3
5.	WS_2	3	$d+1$	2
6.	ST			

Nr komórki	Słowa pomocnicze			
	Liczby przechowywane			
c	a_n			
$c+1$	a_{n-1}			
. . . .	. . . . .			
$c+n$	a_0			
	kod	I	II	III
$d+1$	$+$	a	$c+n+1$	a
$d+2$	0	0	1	0

W komórkach $c, \ldots, c+n$ umieszczone są współczynniki wielomianu, w komórce d mieści się wartość argumentu x, do komórki a przesyła się szukaną wartość wielomianu $f(x)$.