ZASTOSOWANIA MATEMATYKI
V (1960)

S. PASZKOWSKI (Warszawa)

AUTOMATYCZNA ELEKTRONOWA MASZYNA LICZĄCA „STRIEŁA—4"

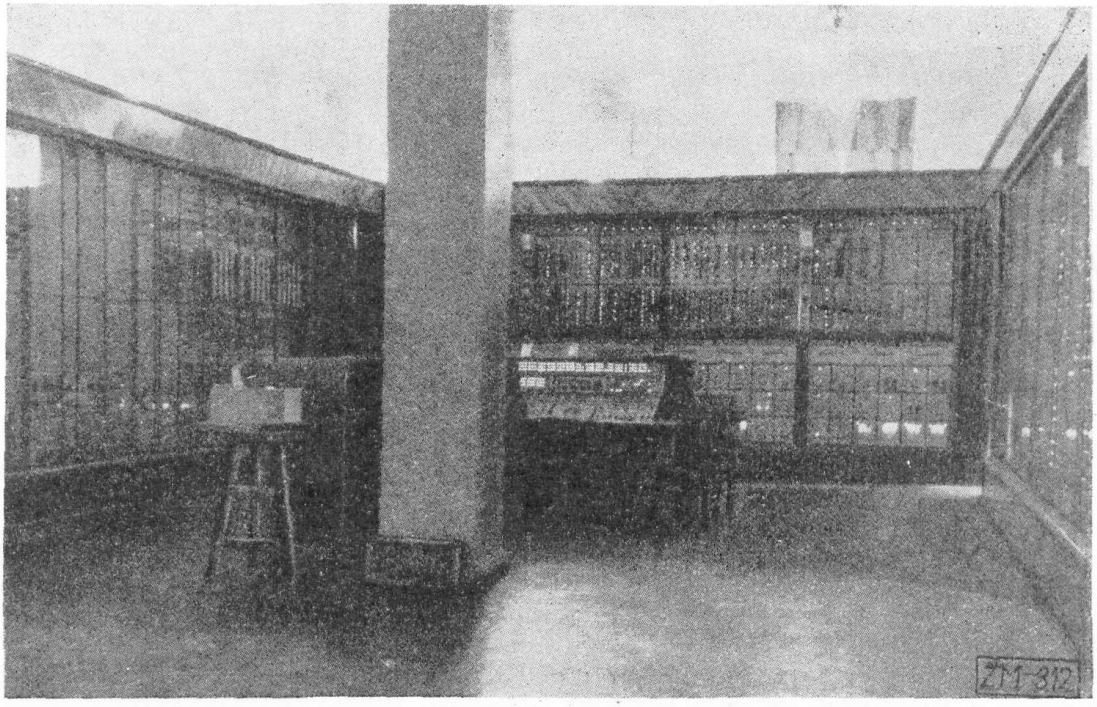
Autor tego artykułu w latach 1956-58 odbył kilkunastomiesięczną praktykę naukową w Ośrodku Rachunkowym Uniwersytetu Moskiewskiego(1). Artykuł zawiera opis automatycznej elektronowej maszyny liczącej typu „Strieła—4" („Стрела—4"), wchodzącej w skład wyposażenia tego ośrodka. Opis ogranicza się do spraw interesujących bezpośrednio matematyka korzystającego z takiej maszyny w pracy obliczeniowej i nie obejmuje konstrukcji maszyny.

Za zgodę na wydrukowanie tej pracy składam na tym miejscu jeszcze raz podziękowanie dyrektorowi Ośrodka Rachunkowego, prof. I. S. Bieriezinowi.

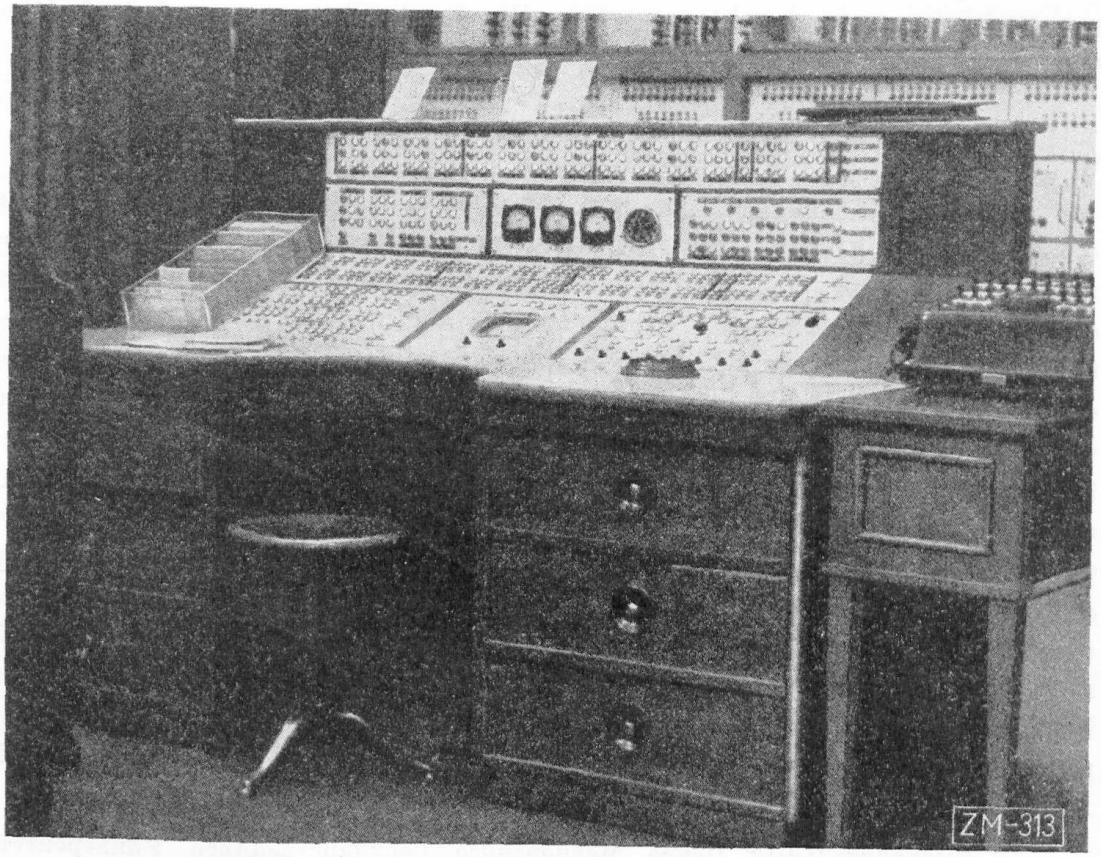
§ 1. Dane ogólne

Maszyna Strieła, jak ją będziemy krótko nazywać, zajmuje salę o powierzchni kilkuset metrów kwadratowych i kilka mniejszych pomieszczeń, w których znajdują się różne urządzenia pomocnicze. Główna część maszyny (urządzenie arytmetyczne wykonujące działania arytmetyczne i logiczne, pamięć, urządzenie sterujące) mieści się w trzech podwójnych, chłodzonych powietrzem szafach (zob. fot. 1) mających po około 10 m długości i 3 m wysokości. Lewa szafa zawiera pamięć magnetyczną i część urządzenia arytmetycznego wykonującą operacje standardowe (zob. § 5.4), środkowa szafa — pamięć wewnętrzną na tzw. potencjałoskopach, prawa zaś — resztę urządzenia arytmetycznego i urządzenie sterujące. Prócz tego na fot. 1 widać urządzenie wprowadzające dane i program obliczeń do pamięci maszyny (*wejście*), perforator (*wyjście*) i biurko wyposażone w liczne wskaźniki i przełączniki, za pomocą których operator może obserwować automatyczną pracę Strieły i w razie potrzeby zmieniać jej bieg. Biurko to jest pokazane z bliska na fot. 2.

(1) Вычислительный Центр Московского Государственного Университета им. М. В. Ломоносова.



Fot. 1. Maszyna „Strieła-4"



Fot. 2. Biurko operatora

Zużycie mocy przez maszynę (nie licząc jej chłodzenia) wynosi około 124 kW, ciężar maszyny — około 15 ton. Koszt budowy jednego z prototypów Strieły-4, stanowiącego obecnie własność Ośrodka Rachunkowego Uniwersytetu Moskiewskiego, wyniósł kilka milionów rubli. Z danych dotyczących szybkości pracy maszyny przytoczymy tu jedną: Strieła wykonuje około 2000 działań arytmetycznych na sekundę. O innych danych będzie mowa w dalszym ciągu.

§ 2. Postać liczb w Striele

Cyfrowe maszyny liczące dzielą się na takie, które operują liczbami należącymi do przedziału $(-1, 1)$ (maszyny ze stałym położeniem przecinka) i na takie, w których tego ograniczenia nie ma (maszyny ze zmiennym położeniem przecinka). Drugi typ, o bardziej skomplikowanej budowie, jest zwykle wygodniejszy od typu pierwszego dla matematyków korzystających z maszyny, gdyż ułożenie programu obliczeń tak, aby ich wszystkie pośrednie wyniki należały do przedziału $(-1, 1)$, jest na ogół bardzo kłopotliwe.

Strieła jest maszyną ze zmiennym położeniem przecinka. Wykonuje ona działania na liczbach przedstawionych w tzw. postaci *półlogarytmicznej*: $x = m2^r$, gdzie $\frac{1}{2} \leqslant |m| < 1$, $|r| \leqslant 63$; m nazywa się *mantysą* liczby x, a całkowita liczba r *rzędem* tejże liczby. Liczby m i r pisze się w układzie dwójkowym. Liczbę m podaje się z dokładnością do 35 cyfr układu dwójkowego po przecinku (przy czym jeśli tylko $x \neq 0$, to pierwsza cyfra jest równa 1, bo $|m| \geqslant \frac{1}{2}$). Liczba r, jak to wynika z nierówności $|r| \leqslant 63$, jest w układzie dwójkowym co najwyżej sześciocyfrowa; podaje się wszystkie sześć cyfr, z których kilka początkowych może być zerami. Dlatego obrazem dowolnej liczby $x \neq 0$ w pamięci maszyny jest układ 43 cyfr układu dwójkowego (tj. zer i jedynek), których znaczenie jest następujące:

1 cyfra — znak mantysy m (0, jeśli $m \geqslant 0$, i 1, jeśli $m < 0$),
2÷36 cyfry — 35 kolejnych cyfr ułamkowej mantysy m,
37 cyfra — znak rzędu r (0, jeśli $r \geqslant 0$, i 1, jeśli $r < 0$),
38÷43 cyfry — 6 kolejnych cyfr całkowitego rzędu r.

Natomiast obrazem liczby 0 jest układ 43 zer.

Z powyższego wynika, że np. symbole

(1) 110 111 010 001 000 000 000 000 000 000 000 000 0 001 000,
010 010 100 000 000 000 000 000 000 000 000 000 1 100 001

oznaczają liczby równe w układzie dwójkowym odpowiednio

$$-0{,}10\,111\,010\,001 \cdot 10^{1\,000}, \quad 0{,}10\,010\,1 \cdot 10^{-100\,001},$$

a w układzie dziesiętnym równe odpowiednio

$$-\left(\frac{1}{2}+\frac{1}{8}+\frac{1}{16}+\frac{1}{32}+\frac{1}{128}+\frac{1}{2048}\right)2^8 = -186{,}125,$$

$$\left(\frac{1}{2}+\frac{1}{16}+\frac{1}{64}\right)2^{-(32+1)} = 37\cdot 2^{-39} \approx 6{,}730\cdot 10^{-11}.$$

Z ograniczeń $\frac{1}{2} \leqslant |m| < 1$, $|r| \leqslant 63$ wynika, że maszyna może wykonywać działania na liczbach o wartości bezwzględnej z przedziału $\langle 2^{-64}, 2^{63}-2^{28}\rangle$, tj. z przedziału $\langle 0{,}542\cdot 10^{-19}, 0{,}922\cdot 10^{19}\rangle$ (nie bierzemy tu w rachubę zera). Choć przedział ten jest bardzo szeroki, może się zdarzyć, że wynik jakiegoś działania wykroczy poza ten przedział. Jeśli wartość bezwzględna otrzymanej liczby jest mniejsza od 2^{-64}, to zamiast niej maszyna pośle do pamięci bliskie tej liczbie zero i będzie rachować dalej, jeśli natomiast ta wartość jest większa od $2^{63}-2^{28}$, to zamiast niej maszyna pośle do pamięci liczbę $2^{63}-2^{28}$ i przerwie obliczenia, sygnalizując w ten sposób konieczność zmiany programu obliczeń. Zdarza się to jednak rzadko.

Zaletą Strieły jest — oprócz bardzo szerokiego przedziału, do którego mogą należeć dane wyjściowe i rezultaty działań — duża dokładność obliczeń. Określa ją dokładność, z jaką podaje się mantysę, a 35 cyfr tej ostatniej w układzie dwójkowym odpowiada 10 bądź 11 cyfrom w układzie dziesiętnym.

Chociaż zasadniczo dane początkowe wprowadza się do maszyny w układzie dziesiętnym (powiemy za chwilę w jakiej postaci) i w tym samym układzie otrzymuje się wyniki obliczeń, to jednak czasem trzeba pisać symbole typu (1). Ponieważ układ dwójkowy wymaga wielu cyfr dla wyrażenia liczb z daną dokładnością, więc zastępuje się go przez układ ósemkowy. Ściślej mówiąc, w 43-cyfrowym symbolu liczby łączy się w trójki cyfry od 1-ej do 36-ej i od 38-ej do 43-ej i zastępuje się te trójki przez cyfry układu ósemkowego według tablicy 1. 37-ą cyfrę (znak rzędu) pozostawia się bez zmiany zachowując również jej położenie między cyframi układu ósemkowego powstałymi z cyfr od 1-ej do 36-ej i takimiż cyframi powstałymi z cyfr od 38-ej do 43-ej. W szczególności liczby (1) zapiszemy w układzie ósemkowym tak:

TABLICA 1

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

6721 0000 0000 0 10, 2240 0000 0000 1 41.

Wszystko, o czym mówiliśmy dotąd w § 2, odnosiło się do tej postaci liczb, jaką mają one w czasie wykonywania obliczeń. Natomiast liczby

wprowadza się do maszyny i końcowe wyniki obliczeń otrzymuje się z reguły w układzie dziesiętnym, także w postaci półlogarytmicznej (sposób przejścia od jednego układu pozycyjnego do drugiego jest omówiony w § 5.4). Znaczy to, że liczbę $X \neq 0$ przedstawia się w postaci $X = M10^R$, gdzie $0{,}1 \leqslant M < 1$, $|R| \leqslant 19$, i w jej symbolu pisze się kolejno znak mantysy M, tj. znak samej liczby X, dziewięć kolejnych cyfr po przecinku mantysy (jeśli $X \neq 0$, to pierwsza z nich jest różna od zera), znak rzędu R i jego dwie cyfry przed przecinkiem, z których pierwsza może być tylko zerem lub jedynką. A więc np. liczby

$$-0{,}0041867692394, \qquad 136{,}258$$

napiszemy w postaci

(2) $$-418676924\ -02, \qquad +136258000\ +03$$

(pierwszą z nich trzeba było zaokrąglić). Liczbę 1 pisze się w postaci $+100000000\ +01$, a 0 w postaci $+000000000\ +00$.

TABLICA 2

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Aby wprowadzić liczbę napisaną w układzie dziesiętnym do pamięci maszyny, trzeba zastąpić cyfry tego układu przez kombinacje zer i jedynek. Odpowiedniość między cyframi i kombinacjami zer i jedynek może być w zasadzie dowolna, byle była wzajemnie jednoznaczna. W Striele jest ona wyznaczona przez tablicę 2.

Odnosi się to do wszystkich dziewięciu cyfr mantysy M i do drugiej cyfry rzędu R. Natomiast pierwsza cyfra (cyfra dziesiątek) rzędu R może się równać tylko 0 lub 1, więc jej postaci nie zmienia się. Dlatego symbol liczby X napisanej w układzie dziesiętnym składa się — w takiej postaci, z jaką ma do czynienia maszyna — z 43 cyfr układu dwójkowego oznaczających odpowiednio:

1 cyfra — znak mantysy M (0, jeśli $M \geqslant 0$, i 1, jeśli $M < 0$),
2÷5 cyfry — 1 cyfrę mantysy M,
6÷9 cyfry — 2 cyfrę mantysy M,
10÷13 cyfry — 3 cyfrę mantysy M,
14÷17 cyfry — 4 cyfrę mantysy M,
18÷21 cyfry — 5 cyfrę mantysy M,
22÷25 cyfry — 6 cyfrę mantysy M,
26÷29 cyfry — 7 cyfrę mantysy M,
30÷33 cyfry — 8 cyfrę mantysy M,
34÷37 cyfry — 9 cyfrę mantysy M,
38 cyfra — znak rzędu R (0, jeśli $R \geqslant 0$, i 1, jeśli $R < 0$),
39 cyfra — 1 cyfrę rzędu R,
40÷43 cyfry — 2 cyfre rzedu R.

Pierwsza z liczb (2) odpowiada więc następującemu układowi zer i jedynek:

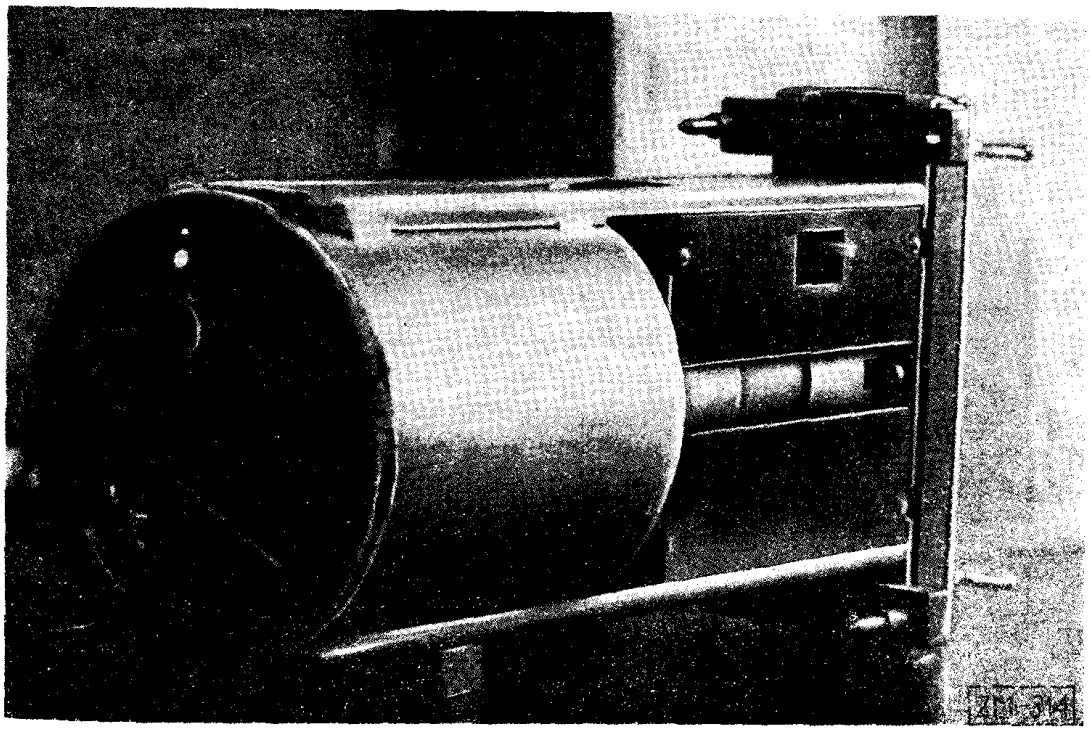
1 0100 0001 1000 0110 0111 0110 1001 0010 0100 1 0 0010.

§ 3. Pamięć Strieły

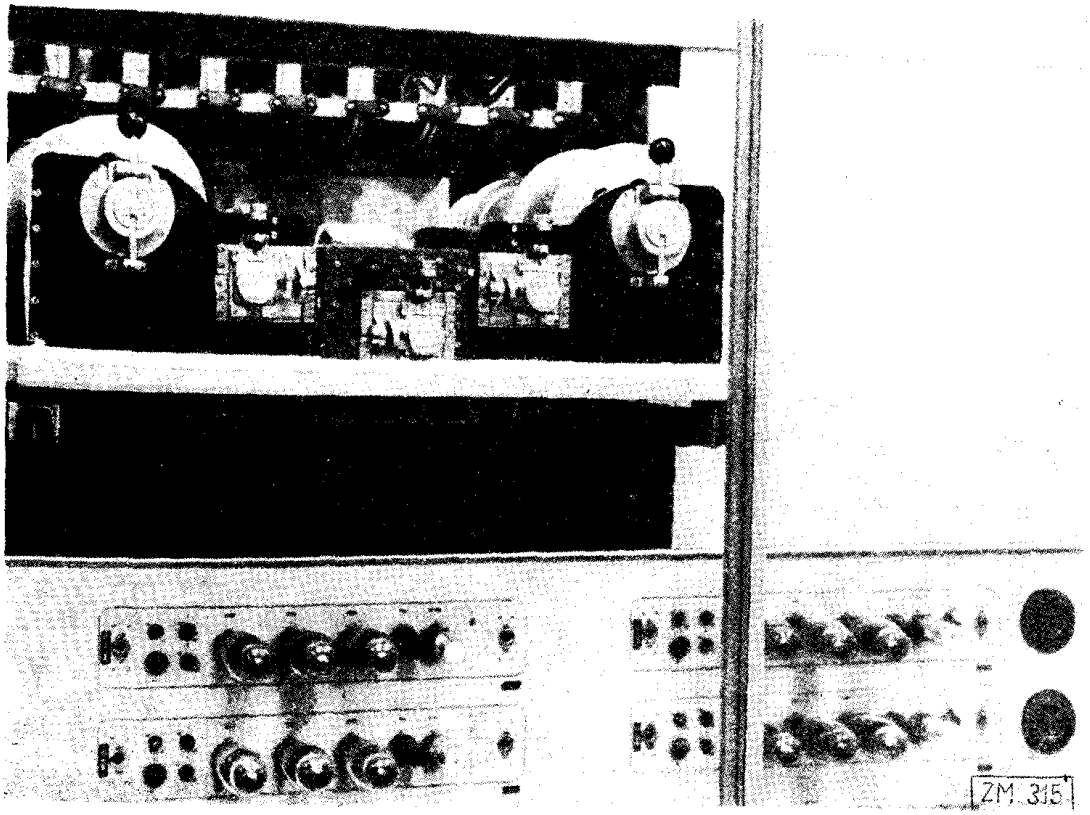
Pamięć, jak nazywa się zwykle tę część maszyny liczącej, która służy do przechowywania liczb i programu obliczeń, dzieli się w Striele na dwie części: na pamięć *wewnętrzną*, składającą się z tzw. potencjałoskopów (jeden z nich wyjęty z maszyny pokazano na fot. 3), i pamięć *zewnętrzną*, magnetyczną (fot. 4), w której liczby mieszczą się na taśmie magnetycznej. Nazwy obu wymienionych części pamięci (wewnętrzna, zewnętrzna) biorą się stąd, że tylko pierwsza z nich jest bezpośrednio wykorzystywana w czasie obliczeń, druga zaś może jedynie wymieniać swą zawartość z pierwszą częścią.

Potrzeba istnienia pamięci wewnętrznej i pamięci magnetycznej tłumaczy się tym, że ich własności wzajemnie się dopełniają. Pamięć wewnętrzna ma ze względów technicznych małą pojemność (około 2000 liczb), ale dużą szybkość działania (znacznie poniżej 1 msek), tj. wybieranie potrzebnych liczb z pamięci i posyłanie do niej wyników wykonywanych działań trwa bardzo krótko. Natomiast pamięć magnetyczna ma dużą pojemność (w Striele do 200000 liczb), ale mniejszą szybkość działania. Na czas potrzebny dla użycia pamięci magnetycznej składa się bowiem czas zużyty na podprowadzenie odpowiedniego odcinka taśmy pod głowicę odczytującą (zależny od poprzedniego położenia taśmy) i czas zużyty na samo pobranie liczb z taśmy lub ich utrwalenie na taśmie (około 1 m sek dla jednej liczby). Główną rolę gra oczywiście pierwsza część straconego czasu; zmniejsza się ją w ten sposób, że posyła się do pamięci lub z pamięci magnetycznej tylko duże grupy liczb, zawartości tzw. stref, o których powiemy za chwilę, liczące od 200 do 2000 liczb.

Pamięć jest podzielona na części i każdej z nich przyporządkowuje się numer będący jedną z liczb układu ósemkowego 0001÷3777, 4001÷÷4375, 5001÷5375, 7400÷7777. Numery 0001÷3777 są numerami tzw. *komórek* pamięci wewnętrznej. Każdemu z nich odpowiada jedna liczba (ściślej, jeden układ złożony z 43 zer i jedynek) mieszcząca się w tej komórce. Numery $4001 \geqslant 4375$ i $5001 \geqslant 5375$ są przyporządkowane całym odcinkom, tzw. *strefom*, odpowiednio pierwszej i drugiej taśmy magnetycznej. Długość dowolnej strefy, tj. ilość liczb, jaką można w niej zmieścić, nie jest wielkością niezmienną; ustala się ją oddzielnie dla każdego rozwiązywanego zadania zgodnie z warunkami podyktowanymi przez nie. Długość strefy nie może jednak przewyższać długości potrzebnej



Fot. 3. Potencjałoskop (element pamięci wewnętrznej)



Fot. 4. Pamięć magnetyczna

dla przechowania 2000 liczb. Na obu taśmach można łącznie umieścić do 200000 liczb. Konstrukcja maszyny pozwala na założenie jeszcze dwóch taśm, ale w rozwiązywanych dotąd na Striele zadaniach nie było to widocznie potrzebne.

Numery komórek 7400÷7777 odpowiadają pomocniczej pamięci na diodach, tzw. *martwej* pamięci, nazywanej tak dlatego, że zmiana zawartości tych komórek wymaga wymiany pewnych elementów konstrukcji maszyny. W martwej pamięci przechowuje się różne stałe, zarówno liczbowe (1, π itp.), jak i logiczne, służące do przekształcania rozkazów (zob. § 5.2).

W programach obliczeń spotyka się również numer 0000. Komórki o tym numerze w rzeczywistości nie ma, można jednak uważać, że jest to numer komórki zawierającej zero i należącej do pamięci martwej. Do tejże części pamięci możemy wreszcie zaliczyć programy operacji standardowych, służące między innymi do obliczania wartości funkcji elementarnych, lecz odpowiadają im nie numery komórek, a numery tych operacji.

Charakterystyczną cechą obu zasadniczych części pamięci — wewnętrznej i magnetycznej — jest to, że przesłanie liczby do jakiejś komórki pamięci wewnętrznej lub grupy liczb do strefy pamięci magnetycznej powoduje automatycznie starcie liczby lub grupy liczb, które przedtem zajmowały tę część pamięci. Pozwala to wykorzystywać pamięć do przechowywania coraz to nowych liczb i symboli, zmusza jednak do skupienia uwagi przy układaniu programu, gdyż musimy pamiętać o tym, że jeśli obliczone w danej chwili wielkości będą potrzebne w dalszym ciągu, to nie należy posyłać do zajmowanych przez nie komórek innych liczb.

O tym, w jaki sposób oba rodzaje pamięci mogą wymieniać między sobą zawartość i jak przesyła się do nich początkowe dane, a jak otrzymuje się wyniki, powiemy w § 4, a także — przy okazji omawiania operacji wykonywanych przez maszynę Strieła — w § 5.5.

§ 4. Wejście i wyjście. Urządzenia pomocnicze

Omówimy teraz urządzenia służące do przekazywania maszynie programu obliczeń i wydobywania z niej ich wyników, a także do transponowania symboli z ogólnie przyjętej postaci cyfrowej na postać „zrozumiałą" dla maszyny i odwrotnie — z tej ostatniej na postać cyfrową.

Mówiliśmy już, że liczby występujące w Striele są układami 43 zer i jedynek. Dotyczy to liczb podanych w układzie dwójkowym i liczb podanych w układzie dziesiętnym. Również dowolny rozkaz wykonania przez maszynę operacji arytmetycznej lub innej, stanowiącej cząstkę programu obliczeń, jest układem 43 zer i jedynek. Każde takie 43-cyfrowe

„słowo" — niezależnie od posiadanej przezeń treści — trzeba zaszyfrować w postaci, którą będzie mogła przyjąć maszyna licząca. Słowom w Striele odpowiadają układy otworków wyciętych w tzw. *kartach dziurkowanych.*

Karta dziurkowana jest kartonowym prostokątem o wymiarach 82,5×187,4 mm, który ma jeden róg ścięty, co zapobiega omyłkom przy sortowaniu kart. Karton, z którego jest zrobiona, powinien posiadać ściśle określone własności, takie np. jak właściwa grubość (0,18 mm) i twardość, mała wrażliwość na zmiany temperatury i wilgotności powietrza itp. Karta jest podzielona na 12 wierszy i 57 kolumn. W każdym z 684 segmentów utworzonych przez te wiersze i kolumny można wyciąć prostokątny otwór o wymiarach około 3×1,5 mm. 12 początkowych kolumn używa się dla zaznaczenia numeru karty i numeru zadania, do którego ta karta się odnosi. Kolumny 13 i 57 zawierają tzw. *markiery*, tj. otwory sygnalizujące, że symbol umieszczony w danym wierszu należy wprowadzić do pamięci maszyny. Pozwala to m. in. odróżnić zero od braku jakiegokolwiek symbolu w danym wierszu. 43 segmenty, znajdujące się w pozostałych 43 kolumnach (od 14-tej do 56-tej) i w dowolnym z 12 wierszy karty, służą do zanotowania 43-cyfrowego symbolu liczby lub rozkazu; otwór wycięty w k-tym segmencie oznacza, że k-tą cyfrą w tym symbolu jest 1, a pozostawienie tego segmentu nienaruszonym oznacza, że cyfrą tą jest 0. Na jednej karcie możemy więc umieścić 12 liczb lub rozkazów.

Urządzenie służące do przenoszenia na karty programu obliczeń i danych liczbowych potrzebnych do jego wykonania składa się z klawiatury, na której nastawia się cyfry (fot. 5), połączonej z urządzeniem sterującym elektrycznie pracą perforatora wycinającego otwory w kolejnych wierszach karty.

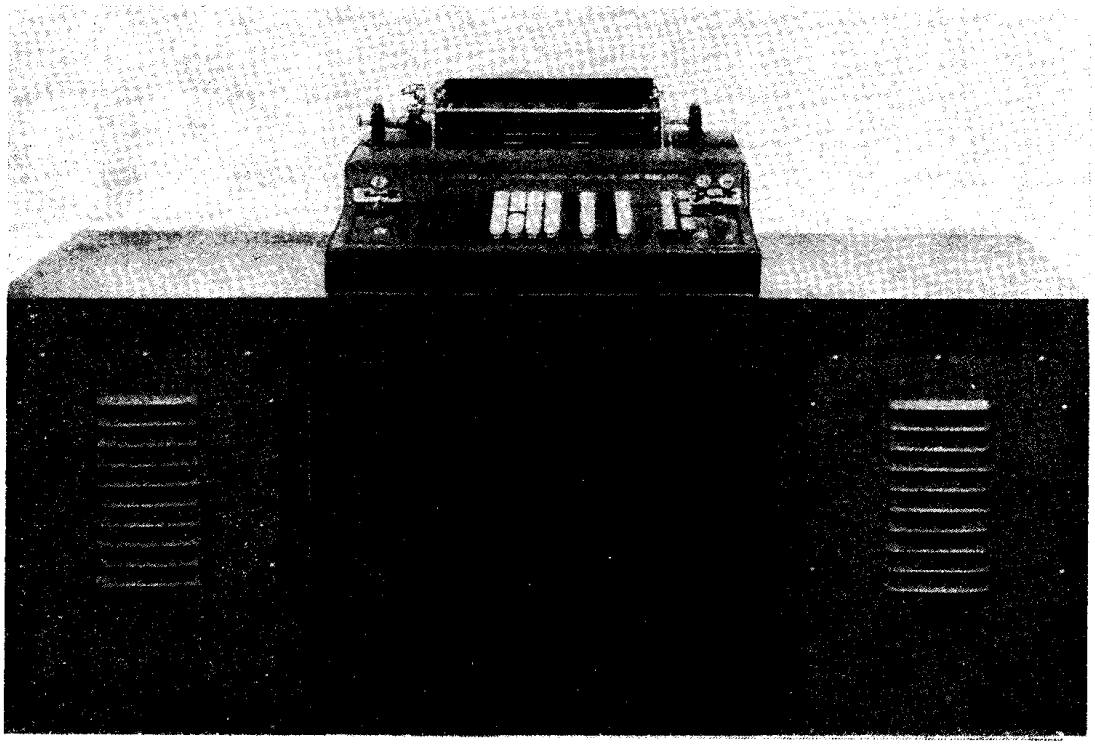
Gdy karty z programem obliczeń są już gotowe, należy sprawdzić, czy prawidłowo wycięto w nich otwory; błędy mogły powstać z winy człowieka lub maszyny. Kontrolę najłatwiej zrobić transponując na drukarce zawartość kart z powrotem na układy cyfr i porównując te ostatnie z danymi wyjściowymi.

Plik sprawdzonych i odpowiednio uporządkowanych kart zawierających program obliczeń wkłada się do urządzenia, które nazywamy krótko *wejściem*, ponieważ ono właśnie przekazuje program obliczeń z kart dziurkowanych do pamięci maszyny. Wejście pracuje z szybkością około 2000 liczb na minutę.

Strieła nie ma dotychczas urządzenia, które by bezpośrednio drukowało wyniki obliczeń na papierze za pomocą cyfr. Wyniki te otrzymujemy na kartach dziurkowanych we wspomnianym już perforatorze, który przy tej czynności jest sterowany przez samą maszynę liczącą. *Wyjście*

to pracuje z szybkością około 1000 liczb na minutę. Następnie transponuje się układy otworków wyciętych w kartach na postać cyfrową za pomocą drukarki pracującej z szybkością około 90 liczb na minutę.

Drukarka pracuje niezależnie od maszyny liczącej, dlatego mała szybkość jej działania nie pomniejsza w istotny sposób walorów Strieły. Decydującą natomiast rolę gra szybkość pracy wejścia i wyjścia, gdyż Strieła nie może wykonywać jednocześnie z przyjmowaniem programu



Fot. 5. Klawiatura perforatora

do swej pamięci lub wysyłaniem z niej wyników obliczeń żadnych innych czynności. Obecna szybkość pracy wejścia i wyjścia nie jest zadowalająca i personel techniczny zatrudniony w Ośrodku Rachunkowym pracuje nad jej dalszym zwiększeniem (zwiększono już ją dwukrotnie od czasu wprowadzenia maszyny do normalnej eksploatacji).

§ 5. Operacje elementarne

Aby ułożyć program obliczeń na automatycznej maszynie liczącej, trzeba wiedzieć przede wszystkim, jakie operacje ta maszyna wykonuje i w jakiej postaci należy podawać rozkazy ich wykonania. Ilość i różnorodność tych operacji jest — obok szybkości działania, dokładności obliczeń i pojemności pamięci — czwartym czynnikiem określającym jakość

maszyny. Ten właśnie czynnik jest niezmiernie istotny dla tzw. *programistów*, którzy układają programy obliczeń. Strieła odznacza się dużą ilością mądrze wybranych operacji, wygodną dla programistów ich postacią i możliwością wprowadzania nowych operacji bez istotnych zmian konstrukcyjnych.

Strieła jest maszyną *trójadresową*, tj. dla wykonania jednej operacji (np. jednego działania arytmetycznego) jest na ogół potrzebny jeden rozkaz zajmujący jedną komórkę pamięci wewnętrznej i mający postać $\alpha\,\beta\,\gamma\,S$, gdzie α (I adres) i β (II adres) są numerami komórek pamięci, zawierających liczby lub symbole, na których wykonuje się operację o numerze S (np. zawierających odjemną i odjemnik, jeśli S odpowiada odejmowaniu), a γ (III adres) jest numerem komórki, w której po wykonaniu operacji ma się znaleźć jej wynik (np. różnica). Dwa lub trzy adresy spośród α, β i γ mogą być sobie równe.

Ponieważ z pamięci magnetycznej nie korzysta się bezpośrednio w rachunku, więc α, β i γ są numerami komórek pamięci wewnętrznej albo pamięci martwej z tym rozróżnieniem, że do tej ostatniej może odnosić się tylko jeden z adresów α i β. W pewnych przypadkach symbole α, β i γ mają inne znaczenie, co przypomnimy w odpowiednim miejscu. Zawsze jednak te symbole i numery komórek pamięci, są liczbami 12-cyfrowymi w układzie dwójkowym. Numer, czyli tzw. *kod* S, operacji jest liczbą 6-cyfrową w tymże układzie. Łącznie symbol rozkazu $\alpha\,\beta\,\gamma\,S$ składa się z 15 cyfr układu ósemkowego (czyli 43 cyfr układu dwójkowego), mających następujące znaczenie:

1÷4 (1÷12) cyfry — I adres (numer komórki α),
5÷8 (13÷24) cyfry — II adres (numer komórki β),
9÷12 (25÷36) cyfry — III adres (numer komórki γ),
13 (37) cyfra — zawsze 0,
14÷15 (38÷43) cyfry — kod S operacji.

Dokładniej, 37 cyfra może być w pewnych przypadkach równa 1, nie będziemy jednak o nich mówić, i dlatego w symbolu $\alpha\,\beta\,\gamma\,S$ będziemy zawsze opuszczać zero stojące między γ i S.

Przy układaniu programu korzysta się zawsze z bardziej zwartego układu ósemkowego, ale w pamięci maszyny rozkazy są oczywiście reprezentowane przez układy 43 zer i jedynek.

Poniżej podajemy opis wszystkich operacji wykonywanych przez Striełę. Symbole $(\alpha), (\beta), (\gamma), \ldots$ używane przy tym oznaczają zawartość komórek o numerach $\alpha, \beta, \gamma, \ldots$, czyli, mówiąc prościej, komórek $\alpha, \beta, \gamma, \ldots$

5.1. Operacje arytmetyczne. Dodawanie. Kod 01. $(\gamma) = (\alpha) + (\beta)$. Równość ta oznacza np., że rozkaz 0000 1400 2001 01 powoduje dodanie

do siebie liczb zawartych w komórkach pamięci o numerach 0300 i 1400 oraz umieszczenie ich sumy w komórce o numerze 0001.

Odejmowanie. Kod 03. $(\gamma) = (\alpha) - (\beta)$.

Odejmowanie modułów. Kod 04. $(\gamma) = |(\alpha)| - |(\beta)|$.

Mnożenie. Kod 05. $(\gamma) = (\alpha) \cdot (\beta)$.

Zauważmy tutaj, że w Striele, jak zresztą i w wielu innych maszynach cyfrowych, nie ma operacji dzielenia. Trzeba ją złożyć z operacji mnożenia i opisanej w § 5.4 operacji obliczania odwrotności.

Oznaczmy teraz przez $m_\alpha, m_\beta, m_\gamma, r_\alpha, r_\beta, r_\gamma$ odpowiednio mantysy i rzędy liczb zawartych w komórkach α i β oraz wyniku operacji, który ma być posłany do komórki γ.

Dodawanie rzędów liczb. Kod 06. $m_\gamma = m_\alpha$, $r_\gamma = r_\alpha + r_\beta$. Operacja ta (a także następna) jest niesymetryczna i jej wynik nie zależy od mantysy liczby (β).

Odejmowanie rzędów liczb. Kod 07. $m_\gamma = m_\alpha$, $r_\gamma = r_\alpha - r_\beta$.

Przyswojenie znaku. Kod 10. $(\gamma) = |(\alpha)| \operatorname{sign}(\beta)$.

Wydzielenie części całkowitej. Kod 12. W tej operacji bierze udział tylko jedna liczba zawarta w komórce α. II adres, stanowiący w rozkazach wykonania poprzednio opisanych operacji numer komórki β, tu jest równy 0000. Do komórki γ posyła się liczbę $\lfloor |(\alpha)| \rfloor \operatorname{sign}(\alpha)$, gdzie $[x]$ oznacza część całkowitą liczby x. Jeśli w komórce α była liczba dodatnia, to rezultatem operacji 12 jest po prostu część całkowita tej liczby.

Dodawanie z cyklicznym przeniesieniem. Kod 17. Zawartość komórek α, β i γ traktuje się w tej operacji inaczej niż poprzednio, a mianowicie jak 43-cyfrowe liczby całkowite w układzie dwójkowym. Do komórki γ posyła się sumę liczb z komórek α i β, jeśli ta suma również jest liczbą 43-cyfrową, a sumę zmniejszoną o $2^{44} - 2^0$, jeśli jest ona liczbą 44-cyfrową. Innymi słowy, w drugim przypadku jedynka, która „nie mieści się" w komórce γ, zostaje dodana do rzędu jednostek sumy $(\alpha) + (\beta)$ — stąd nazwa operacji. Operację 17 stosuje się do kontroli bezbłędności wprowadzania liczb do pamięci maszyny, wydobywania z niej liczb lub ich przenoszenia z jednej części pamięci do drugiej (zob. opis operacji 60 w § 5.5).

5.2. Operacje na rozkazach. Operacje można wykonywać nie tylko na liczbach, ale i na rozkazach, tj. na symbolach typu $\alpha\ \beta\ \gamma\ S$. To właśnie jest przyczyną ogromnego znaczenia automatycznych maszyn liczących, ponieważ pozwala budować programy cykliczne (patrz § 6).

Niech w komórce α mieści się symbol $\delta\ \varepsilon\ \zeta\ S$, gdzie $\delta, \varepsilon, \zeta$ są liczbami 4-cyfrowymi w układzie ósemkowym, a S — liczbą dwucyfrową w tymże układzie. Niech w komórce β mieści się symbol $\eta\ \vartheta\ \iota\ T$, którego elementy

składowe mają podobne znaczenie, jak $\delta, \varepsilon, \zeta$ i S. Symbole te mogą w szczególności oznaczać rozkazy, ale nie muszą nimi być. Na przykład w komórce β może się znajdować symbol 0001 0000 0000 00, który na pewno nie jest rozkazem (ponieważ 00 nie jest kodem żadnej operacji), a jest tzw. stałą logiczną, służącą do zmiany rozkazów zawartych w innych komórkach pamięci.

Zmiana rozkazu przez dodawanie. Kod 02. Wynikiem tej operacji wykonanej na symbolach z komórek α i β, umieszczanym w komórce γ, jest symbol (w szczególności rozkaz) $\{\delta+\eta\}\{\varepsilon+\vartheta\}\{\zeta+\iota\}S$, gdzie dla liczby całkowitej ω w układzie ósemkowym określamy symbol $\{\omega\}$ następująco:

$$\{\omega\} = \begin{cases} \omega+10000, & \text{jeśli } \omega < 0, \\ \omega, & \text{jeśli } 0 \leqslant \omega < 10000, \\ \omega-10000, & \text{jeśli } \omega \geqslant 10000 \end{cases}$$

(10000 wszędzie tu także w układzie ósemkowym). Operacja 02 jest niesymetryczna i nie zależy od liczby T.

Dla wyjaśnienia sensu operacji 02 przypuśćmy, że w komórce 0001 mieści się rozkaz 2423 1547 2236 01 (rozkaz zsumowania liczb z komórek 2423 i 1547 i przesłania wyniku do komórki 2236), a w komórce 0002 — stała logiczna 1014 7452 1040 00. Wykonanie rozkazu 0001 0002 0003 02 spowoduje wtedy, że do komórki 0003 zostanie posłany symbol (rozkaz) 3437 1221 3276 01, ponieważ

$$\begin{aligned} 2423+1014 &= 3437, \\ 1547+7452 &= 1221+10000, \\ 2236+1040 &= 3276 \end{aligned}$$

(wszystkie dodawania w układzie ósemkowym).

Zmiana rozkazu przez odejmowanie. Kod 15. Wynikiem wykonania rozkazu $\alpha\,\beta\,\gamma$ 15, który to wynik ma być umieszczony w komórce γ, jest symbol $\{\delta-\eta\}\{\varepsilon-\vartheta\}\{\zeta-\iota\}S$.

Jeśli w komórkach 0001, 0002 mieściły się te same symbole, co w ostatnim przykładzie, to w wyniku wykonania rozkazu 0001 0002 0003 15 w komórce 0003 znajdzie się symbol (rozkaz) 1407 2075 1176 01, ponieważ

$$\begin{aligned} 2423-1014 &= 1407, \\ 1547-7452 &= -5703 = 2075-10000, \\ 2236-1040 &= 1176. \end{aligned}$$

W pozostałych operacjach omawianego w § 5.2 typu zawartość komórek α, β i γ interpretuje się nieco odmiennie, rozdzielając ją na poszczególne cyfry układu dwójkowego (zera i jedynki). Oznaczmy te cyfry odpowiednio przez $a_1, a_2, \ldots, a_{43}, b_1, b_2, \ldots, b_{43}$ i $c_1, c_2, \ldots, c_{43}$. Operacje będą określone przez podanie zależności między poszczególnymi cyframi a_i, b_i i c_i.

Dodawanie logiczne. Kod 13. $c_i = 0$ wtedy i tylko wtedy, gdy $a_i = b_i = 0$ $(i = 1, 2, \ldots, 43)$.

Mnożenie logiczne. Kod 11. $c_i = 1$ wtedy i tylko wtedy, gdy $a_i = b_i = 1$ $(i = 1, 2, \ldots, 43)$.

Porównanie. Kod 16. $c_i = 0$ wtedy i tylko wtedy, gdy $a_i = b_i$ $(i = 1, 2, \ldots, 43)$.

W podanej poniżej operacji, w komórce β ma znajdować się liczba, której rząd oznaczymy przez r_β.

Przesunięcie. Kod 14. $c_i = a_{i+r_\beta}$ dla i takich, że $1 \leqslant i \leqslant 43$, $1 \leqslant i + r_\beta \leqslant 43$, oraz $c_i = 0$ dla pozostałych i.

5.3. Operacje grupowe. W wielu obliczeniach należy wykonać kolejno grupę jednakowych działań na różnych liczbach. Na przykład iloczyn skalarny wektorów $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ jest sumą iloczynów postaci $x_i y_i$. Wprawdzie w żadnej automatycznej maszynie liczącej nie potrzeba dla obliczenia tego iloczynu skalarnego umieszczać w pamięci n rozkazów mnożenia x_i przez y_i $(i = 1, 2, \ldots, n)$, lecz w Striele łączenie wielkiej liczby podobnych do siebie rozkazów jest rozwiązane w szczególnie udany sposób. Służą do tego celu operacje grupowe. Rozkaz wykonania operacji grupowej zajmuje dwie sąsiednie komórki pamięci i ma postać

$$\begin{array}{lll} \text{(komórka } \delta) & 0000\ n\ 0000\ G, \\ \text{(komórka } \delta+1) & \alpha\ \ \beta\ \ \gamma\ \ S, \end{array} \tag{3}$$

gdzie S jest kodem dowolnej operacji spośród omówionych w §§ 5.1 i 5.2.

Kod G operacji grupowej jest jedną z liczb 30, 32, 33, 34, 35, 36 i 37, można więc przedstawić go w postaci $30 + 4i_1 + 2i_2 + i_3$, gdzie liczby i_1, i_2, i_3 są równe 0 lub 1. Wykonanie rozkazu (3) jest z definicji równoważne kolejnemu wykonaniu rozkazów

$$\begin{array}{llll} \alpha & \beta & \gamma & S, \\ \alpha + i_1 & \beta + i_2 & \gamma + i_3 & S, \\ \ldots & \ldots & \ldots & \ldots \\ \alpha + n i_1 & \beta + n i_2 & \gamma + n i_3 & S. \end{array}$$

W szczególności wykonanie rozkazu

$$\begin{matrix} 0000\ n\ 0000\ 30, \\ \alpha\ \ \beta\ \ \gamma\ \ S \end{matrix}$$

jest równoważne wykonaniu $n+1$ razy rozkazu $\alpha\,\beta\,\gamma\,S$, a wykonanie rozkazu

$$\begin{matrix} 0000\ n\ 0000\ 35, \\ \alpha\ \ \beta\ \ \gamma\ \ S \end{matrix}$$

jest równoważne kolejnemu wykonaniu $n+1$ rozkazów

$$\begin{matrix} \alpha\ \ \beta\ \gamma\ \ S, \\ \alpha+1\ \beta\ \gamma+1\ S, \\ \ldots\ldots\ldots \\ \alpha+n\ \beta\ \gamma+n\ S. \end{matrix}$$

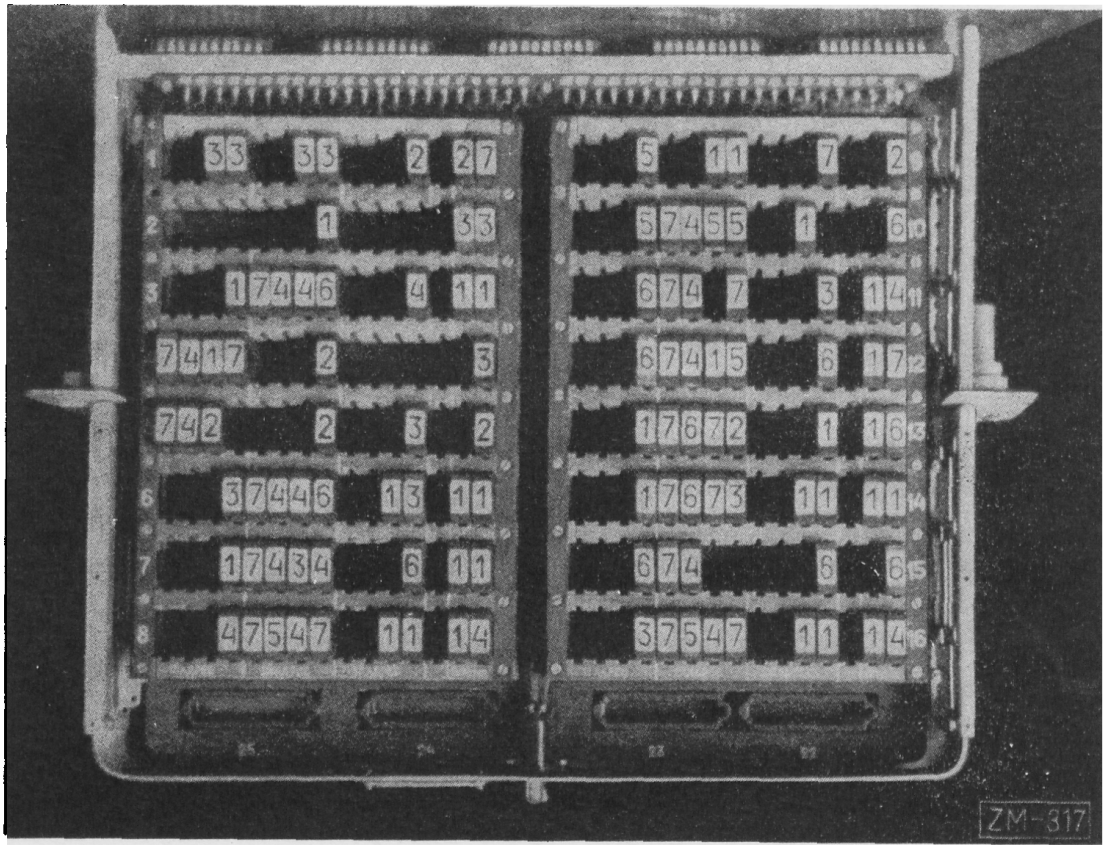
Operacje grupowe można stosować np. do zsumowania pewnej grupy liczb mieszczących się w sąsiednich komórkach, do wymnożenia parami dwóch układów liczb, również mieszczących się (w obrębie każdego układu) w sąsiednich komórkach, do odjęcia od wszystkich liczb pewnego układu jakiejś liczby itp. Zastosowanie operacji grupowej jest ograniczone tylko przez żądanie, by uczestniczące w niej grupy liczb (także wyniki operacji) były rozmieszczone w sąsiednich komórkach pamięci.

5.4. Operacje standardowe. Wygodną dla użytkowników cechą Strieły jest wykonywanie przez nią wielu dość skomplikowanych operacji, takich jak obliczanie wartości licznych funkcji elementarnych, tłumaczenie liczb z jednego układu pozycyjnego na drugi itd., bez wprowadzania do pamięci odpowiednich programów, które już znajdują się w maszynie, w odrębnej części jej pamięci. Fotografia 6 przedstawia wymienny element tej części, zawierający program jednej operacji. Widoczne na tej fotografii cyfry stanowią opis programu.

Rozkaz wykonania dowolnej operacji standardowej ma postać $\alpha\ 0000\ \gamma\ S$, gdzie α oznacza numer komórki zawierającej argument operacji, γ — numer komórki, do której należy posłać wynik operacji, S — kod operacji. W opisie operacji będziemy podawać liczbę taktów, tj. liczbę tych poprzednio opisanych prostszych operacji, których wykonanie składa się na wykonanie danej operacji standardowej. Jak już wiemy, każda z tych składowych operacji zajmuje około 0,5 msek.

Odwrotność. Kod 62. $(\gamma) = 1/(\alpha)$. 16 taktów. Ponieważ w Striele nie ma operacji dzielenia, więc dla obliczenia ilorazu liczb zawartych w komórkach α i β należy wykonać kolejno operacje β 0000 γ 62, α γ γ 05, jeśli iloraz ma się znaleźć w komórce γ.

Pierwiastek kwadratowy. Kod 63. $(\gamma) = \sqrt{(\alpha)}$. 23 do 25 taktów. Ubocznym rezultatem tej operacji jest odwrotność liczby $\sqrt{(\alpha)}$, znajdująca się w komórce 0002.



Fot. 6. Element pamięci martwej, odpowiadający programowi jednej operacji standardowej

Funkcja wykładnicza. Kod 64. $(\gamma) = e^{(\alpha)}$. 31 do 32 taktów.

Logarytm naturalny. Kod 66. $(\gamma) = \log(\alpha)$. 55 taktów.

Sinus. Kod 67. $(\gamma) = \sin(\alpha)$. 15 taktów, jeśli $-\frac{1}{2}\pi < (\alpha) < \frac{1}{2}\pi$ i 22 takty w pozostałych przypadkach.

Arc sin i arc cos. Kod 74. $(\gamma) = \arcsin(\alpha)$. 121 do 157 taktów. Liczba $\arccos(\alpha)$ znajduje się po wykonaniu tej operacji w komórce 0002.

Arc tg. Kod 73. $(\gamma) = \operatorname{arctg}(\alpha)$. 27 taktów, jeśli $|(\alpha)| < 1$ i 45 taktów w przeciwnym przypadku.

Tłumaczenie liczby z układu dziesiętnego na układ dwójkowy. Kod 72. 42 do 43 taktów. Jeśli w komórce α znajdowała się jakaś liczba przedstawiona w układzie dziesiętnym, w postaci określonej w § 2, to wynikiem operacji 72 posyłanym do komórki γ jest ta sama liczba (α), ale przedstawiona już w układzie dwójkowym. Operacja ta jest potrzebna do przekształcenia liczb wprowadzanych do pamięci maszyny do takiej postaci, w jakiej będą one używane w rachunku.

Tłumaczenie liczby z układu dwójkowego na układ dziesiętny. Kod 70. 60 do 77 taktów. Operacja odwrotna względem poprzedniej. Używa się jej do przekształcenia podanych w układzie dwójkowym wyników obliczeń na układ dziesiętny dla ich wyprowadzenia z pamięci maszyny na karty dziurkowane.

Może się zdarzyć, że jakaś operacja standardowa nie jest wykonalna (np. operacja 66 dla $(\alpha) \leqslant 0$). Maszyna sygnalizuje to przerywając w tym miejscu obliczenia. Zgodnie z ogólną zasadą podaną w § 2, maszyna zatrzymałaby się również wtedy, gdy wartość bezwzględna wyniku operacji przekraczałaby liczbę $2^{63}-2^{28}$. Może się to zdarzyć przy obliczaniu wartości funkcji wykładniczej.

We wszystkich dotąd opisanych operacjach standardowych drugi adres rozkazu ich wykonania jest równy 0000. Pozwoliło to uprościć postać rozkazu grupowego wykonania operacji standardowej. Zajmuje on mianowicie jedną komórkę pamięci i ma postać $\alpha\, n\, \gamma\, S$ (S — kod operacji), podczas gdy na podstawie § 5.3 powinien zajmować dwie komórki i mieć postać $0000\, n\, 0000\, 35$, $\alpha\, 0000\, \gamma\, S$. Wykonanie rozkazu $\alpha\, n\, \gamma\, S$ jest na mocy definicji równoważne kolejnemu wykonaniu $n+1$ rozkazów $\alpha\, 0000\, \gamma\, S$, $\alpha+1\, 0000\, \gamma+1\, S$, ..., $\alpha+n\, 0000\, \gamma+n\, S$. Jeśli obliczamy przy pomocy operacji grupowej pierwiastki kwadratowe (wtedy $S = 63$) lub wartości funkcji $\arcsin$ ($S = 74$), to spośród liczb

$$\frac{1}{\sqrt{(\alpha)}}, \frac{1}{\sqrt{(\alpha+1)}}, \ldots, \frac{1}{\sqrt{(\alpha+n)}}, \tag{4}$$

$$\arccos(\alpha), \arccos(\alpha+1), \ldots, \arccos(\alpha+n),$$

stanowiących wynik uboczny odpowiednio operacji 63 i 74, pozostanie w pamięci tylko liczba $1/\sqrt{(\alpha+n)}$ w pierwszym przypadku i $\arccos(\alpha+n)$ w drugim przypadku, ponieważ wszystkie liczby (4) posyłane były do tej samej komórki 0002, każda następna na miejsce poprzedniej. Znaczy to, że nie można stosować operacji grupowych w opisanej wyżej postaci do obliczania odwrotności pierwiastków kwadratowych i wartości funkcji $\arccos$.

Opis ostatniej operacji standardowej, posiadającej kod 60, odkładamy do § 5.5, ponieważ jest ona ściśle związana z opisanymi tam innymi operacjami.

Ważną cechą części pamięci zawierającej programy operacji standardowych jest wymienność jej elementów składowych. Można więc wyjąć z niej np. program obliczania wartości funkcji wykładniczej (kod operacji 64) i na jego miejsce włożyć np. program obliczania wartości funkcji Bessla $I_0(x)$, jeśli ten program był już opracowany. Po tej zamianie maszyna napotykając w obliczeniach na rozkaz typu $\alpha\ 0000\ \gamma\ 64$ będzie obliczać już nie $e^{(\alpha)}$, lecz $I_0((\alpha))$. Pozwala to rozszerzać możliwości maszyny, dopasowywać je do aktualnych potrzeb przez uzupełnianie kompletu operacji standardowych. Trzeba jednak zastrzec, że struktura takich programów musi spełniać dość silne warunki, które narzuca konstrukcja Strieły.

5.5. Operacje przesyłania. Zanim zaczniemy obliczenia na automatycznej maszynie cyfrowej musimy posłać do jej pamięci dane początkowe i program obliczeń. W czasie obliczeń trzeba często przesyłać liczby z jednej części pamięci do drugiej. Wreszcie po zakończeniu rachunku jego wyniki należy utrwalić na kartach dziurkowanych. Te wszystkie czynności wykonują operacje przesyłania i współdziałająca z nimi operacja standardowa o kodzie 60.

W maszynie są możliwe następujące rodzaje przesyłania liczb:

I. Z urządzenia czytającego (wejścia) do pamięci wewnętrznej.

II. Z pamięci wewnętrznej do perforatora.

III. Z jednej części pamięci wewnętrznej do innej części tejże pamięci.

IV. Z pamięci wewnętrznej do pamięci magnetycznej.

V. Z pamięci magnetycznej do pamięci wewnętrznej.

Operacja przesyłania z kontrolą. Kod 60. Rozkaz przesłania grupy n liczb, które tam, skąd je przesyłamy, zajmują kolejne miejsca (tj. kolejne komórki w pamięci wewnętrznej, kolejne wiersze na kartach dziurkowanych lub kolejne pozycje na taśmie magnetycznej) i które tam, dokąd je przesyłamy, również mają zajmować kolejne miejsca, ma postać $\alpha\ n\ \gamma\ 60$. I i III adres w tym rozkazie, tj. α i γ, mają następujące znaczenie: w przypadku I, $\alpha = 0000$, γ jest numerem komórki pamięci wewnętrznej (tj. jedną z liczb $0001 \div 3777$), do której należy przesłać pierwszą w kolejności liczbę; w przypadku II, α jest numerem komórki pamięci wewnętrznej zawierającej liczbę, od której należy rozpocząć przesyłanie, a $\gamma = 0000$; w przypadku III, α i γ oznaczają to samo, co odpowiednio w przypadku II i w przypadku I; w przypadku IV, α oznacza to samo, co w przypadku II, a γ jest numerem strefy pamięci

magnetycznej (tj. jedną z liczb 4001÷4375, 5001÷5375), do której przesyłamy liczby; w przypadku V, α jest numerem strefy pamięci magnetycznej, z której przesyłamy liczby, a γ oznacza to samo, co w przypadku I.

Dokładniej mówiąc (czego jednak matematyk programujący potrzebne mu rozwiązanie jakiegoś zadania może nie wiedzieć), operacja 60 spełnia jedynie rolę łącznika między właściwymi operacjami przesyłania — ich kody są równe w przypadkach I-V odpowiednio 41, 44, 45, 46 i 43 — a omówioną w § 5.1 operacją 17 służącą do kontroli przesyłania. Kontrola ta w przypadkach III-V polega na obliczeniu sumy z cyklicznym przeniesieniem (czyli sumy określonej jak w operacji 17) przesyłanej grupy liczb przed ich przesłaniem i sumy tejże grupy liczb w miejscu przeznaczenia, oraz na porównaniu obu sum. W przypadku I pierwsza z porównywanych sum kontrolnych musi być umieszczona na karcie dziurkowanej w wierszu bezpośrednio następującym po grupie symboli przesyłanych przy pomocy operacji 60. Jeśli obie wspomniane sumy kontrolne nie są sobie równe, co świadczy o omyłce powstałej przy przesyłaniu liczb, to maszyna się zatrzyma i przesyłanie trzeba będzie powtórzyć. W przypadku II perforator dziurkuje oprócz przesyłanej grupy symboli ich sumę kontrolną.

Dwie podane poniżej operacje są potrzebne przy korzystaniu z pamięci magnetycznej.

Podział pamięci magnetycznej na strefy. Kod 47. Operacja służy do podziału taśmy magnetycznej na strefy takiej długości, jaka w danym zadaniu jest najbardziej wygodna (wspominaliśmy już, że ten podział nie jest stały). Dla wyznaczenia strefy o numerze α mającej pomieścić $n+1$ liczb trzeba wykonać rozkaz α—4000 n 4000 47 lub rozkaz α—5000 n 5000 47 zależnie od tego, czy $4001 \leqslant \alpha \leqslant 4375$, czy $5001 \leqslant \alpha \leqslant 5375$.

Przesunięcie taśmy. Kod 25. Na rozkaz α 0000 0000 25 pod głowicę odczytującą zostaje podciągnięty początek strefy taśmy magnetycznej o numerze α. Wykonanie tego rozkazu związane z czynnościami mechanicznymi jest bardzo długie w porównaniu np. z wykonywaniem operacji arytmetycznych. Maszyna jednak zaraz po uruchomieniu bębna z taśmą zaczyna wykonywać następne rozkazy występujące w programie, wobec czego na ogół można tak ułożyć program, aby w chwili, gdy jakaś strefa pamięci magnetycznej będzie potrzebna dla odczytania z niej lub zanotowania na niej grupy symboli, operacja 25 została już zakończona i początek tej strefy znajdował się pod głowicą odczytującą.

5.6. Operacje kierujące obliczeniami. Gdy wykonujemy jakieś obliczenia, to — niezależnie od narzędzi, którymi się posługujemy —

spotykamy się często z faktem, że od rezultatów pewnych działań zależą nie tylko wyniki następnych działań, ale i algorytm lub wzór, który należy zastosować w dalszej części rachunku. Przykładem tego może być choćby rozwiązywanie równań algebraicznych trzeciego stopnia. Układając program obliczeń dla automatycznej maszyny cyfrowej uwzględniamy wspomniany fakt w ten sposób, że programujemy wszystkie warianty obliczeń, które zgodnie z naszymi przewidywaniami mogą okazać się potrzebne. Wielka szybkość obliczeń na takiej maszynie będzie jednak tylko wtedy naprawdę wykorzystana, gdy maszyna będzie wybierać właściwy wariant bez ingerencji człowieka w czasie rachunku. Do tego jest potrzebny nowy typ operacji — operacji kierujących obliczeniami.

W zasadzie Strieła wykonuje po kolei rozkazy zawarte w sąsiednich komórkach pamięci wewnętrznej z coraz to większymi numerami. Jeśli trzeba ten naturalny porządek zmienić — bo wariant rozwiązania, który okazał się potrzebny, jest umieszczony w innym miejscu pamięci — to musimy w programie obliczeń umieścić rozkaz wykonania jednej z dwóch operacji opisanych poniżej. Zanim je określimy, musimy uzupełnić ważnym szczegółem opis operacji arytmetycznych i operacji na rozkazach. Wykonaniu większości z nich towarzyszy powstanie tzw. sygnału ω. Można go uważać za 44-tą cyfrę wyniku operacji, sygnał ω nie jest jednak notowany w pamięci i istnieje tylko do chwili wykonania następnego rozkazu. Wartość sygnału ω zależy w następujący sposób od wyniku przesyłanego do komórki γ:

$$\text{dla operacji 01, 03 i 04} \quad \omega = \begin{cases} 0, & \text{jeśli } (\gamma) \geqslant 0, \\ 1, & \text{jeśli } (\gamma) < 0, \end{cases}$$

$$\text{dla operacji 05, 06 i 07} \quad \omega = \begin{cases} 0, & \text{jeśli } |(\gamma)| < 1, \\ 1, & \text{jeśli } |(\gamma)| \geqslant 1, \end{cases}$$

$$\text{dla operacji 11, 12 i 13} \quad \omega = \begin{cases} 0, & \text{jeśli } (\gamma) \neq 0, \\ 1, & \text{jeśli } (\gamma) = 0, \end{cases}$$

$$\text{dla operacji 16} \quad \omega = \begin{cases} 0, & \text{jeśli } (\gamma) = 0, \\ 1, & \text{jeśli } (\gamma) \neq 0. \end{cases}$$

Przy wykonywaniu pozostałych operacji sygnał ω nie powstaje, czyli, innymi słowy, jego wartość jest równa 0 niezależnie od wyników tych operacji.

Przejście warunkowe I typu. Kod 20. Wykonanie rozkazu $\alpha\,\beta\,\gamma\,20$ powoduje, że maszyna przejdzie do wykonania rozkazu zawartego w komórce α, jeśli sygnał ω, powstały przy wykonaniu rozkazu poprzedzającego bezpośrednio rozkaz $\alpha\,\beta\,\gamma\,20$, miał wartość 0, i do wy-

konania rozkazu zawartego w komórce β, jeśli ten sygnał miał wartość 1. Jednocześnie ściera się dotychczasową zawartość komórki γ, tj. posyła się do niej zero. Jeśli nie chcemy, aby to się stało, to w trzecim adresie rozkazu przejścia warunkowego powinniśmy napisać 0000 zamiast γ.

Nawet w tych obliczeniach, w których możliwy jest tylko jeden wariant, nie zawsze jest wygodne umieszczenie w pamięci maszyny rozkazów w takim porządku, w jakim mają one być wykonywane. Jeśli np. po wykonaniu rozkazu z komórki 0050 maszyna powinna przejść do wykonania rozkazu z komórki 3000, to w komórce 0051 umieścimy rozkaz przejścia bezwarunkowego 3000 3000 0000 20, w którym adresy pierwszy i drugi są sobie równe.

Przejście warunkowe II typu. Kod 27. Wynik wykonania rozkazu $\alpha\,\beta\,\gamma\,27$ *różni się od* wyniku wykonania rozkazu $\alpha\,\beta\,\gamma\,20$ tylko tym, że do komórki γ maszyna pośle nie 0, lecz rozkaz $\nu+1\;\nu+1\;\gamma\;20$, gdzie ν oznacza numer komórki zawierającej rozkaz $\alpha\,\beta\,\gamma\,27$. To powoduje, że kiedy w jakiś czas po wykonaniu operacji 27 maszyna będzie miała wykonać rozkaz z komórki γ, to będzie się w niej znajdował rozkaz bezwarunkowego przejścia do rozkazu następującego bezpośrednio w pamięci po rozkazie $\alpha\,\beta\,\gamma\,27$ oraz do starcia zawartości komórki γ.

Z operacji przejścia warunkowego II typu korzysta się wtedy, gdy w wielu miejscach rozwiązywanego zadania trzeba wykonać te same obliczenia (np. przy rozwiązywaniu równania różniczkowego $y' = f(x, y)$ metodą Rungego-Kutty trzeba dla obliczenia jednej wartości funkcji y kilka razy obliczać wartość funkcji f). Jeśli program tego, wiele razy spotykanego, fragmentu obliczeń jest zawarty np. w komórkach $2000 \div 2050$, to komórkę 2051 zostawimy pustą, a w zasadniczym programie wszędzie tam, gdzie jest potrzebne wykonanie tego fragmentu obliczeń, umieścimy rozkaz 2000 2000 2051 27. W rezultacie maszyna wykona rozkazy z komórek $2000 \div 2050$, a po nich wróci automatycznie do tego miejsca zasadniczego programu, w którym przerwała chwilowo jego realizację.

Do operacji kierujących obliczeniami zaliczamy jeszcze przerwanie obliczeń. Kod 40. Wykonanie rozkazu $\alpha\,\beta\,\gamma\,40$ powoduje przerwanie obliczeń, starcie zawartości komórki γ i ukazanie się na biurku operatora (w postaci rzędów świecących się lub nie świecących lampek) zawartości komórek α i β.

§ 6. Programowanie obliczeń

Programowaniem nazywamy układanie programu obliczeń, tj. zespołu tych rozkazów, których kolejne (zwykle wielokrotne) wykonanie składa się na te obliczenia. Programowanie jest pracą żmudną, wymaga największego skupienia uwagi i wielkiej cierpliwości, a nawet, być może,

specyficznych zdolności. Trudno wprawdzie skonstatować to na podstawie opisanych niżej paru programów, ale też są to właściwie tylko bardzo drobne części składowe programów rzeczywiście spotykanych w praktyce.

6.1. Załóżmy, że dla danych $n+1$ liczb $x_0, x_1, \ldots, x_n$ należy obliczyć ich średnią arytmetyczną

$$\bar{x} = \frac{1}{n+1} \sum_{k=0}^{n} x_k$$

i wariancję

$$s^2 = \frac{1}{n+1} \sum_{k=0}^{n} (x_k - \bar{x})^2 = \frac{1}{n+1} \sum_{k=0}^{n} x_k^2 - \bar{x}^2.$$

Przypuśćmy, że liczby te znajdują się w komórkach $\xi, \xi+1, \ldots, \xi+n$. Wielkości pomocnicze umieścimy w komórkach $\varrho_1, \varrho_2, \ldots$, a program obliczeń — w kolejnych komórkach $\alpha, \alpha+1, \ldots$

Aby znaleźć $\bar{x}$ musimy obliczyć sumę liczb $x_0, x_1, \ldots, x_n$. Można to zrobić dodając do zawartości komórki ϱ_1 (przy założeniu, że przed rozpoczęciem obliczeń było w niej zero) kolejno $x_0, x_1, \ldots, x_n$, tj. wykonując rozkazy

$$\begin{array}{l} \varrho_1\, \xi \quad \varrho_1\, 01, \\ \varrho_1\, \xi+1\ \varrho_1\, 01, \\ \ldots\ldots\ldots \\ \varrho_1\, \xi+n\ \varrho_1\, 01. \end{array} \tag{5}$$

Jak wynika z treści § 5.3, układ tych $n+1$ rozkazów można zastąpić przez jeden rozkaz wykonania operacji grupowej. Ponieważ rozkazy (5) różnią się między sobą tylko drugim adresem, więc — przy oznaczeniach § 5.3 — mamy $i_1 = 0$, $i_2 = 1$, $i_3 = 0$, tj. kod G operacji grupowej jest równy $30+4\cdot 0+2\cdot 1+1\cdot 0 = 32$ i rozkaz jej wykonania ma postać

$$\begin{array}{l} 0000\ n\ 0000\ 32, \\ \varrho_1\ \ \xi\ \ \varrho_1\ \ 01. \end{array}$$

Po wykonaniu tego rozkazu, w komórce ϱ_1 będzie się znajdowała suma $\sum_{k=0}^{n} x_k$, którą z kolei należy podzielić przez $n+1$. Przypuśćmy, że oprócz liczb x_k była również dana, np. w komórce $\xi+n+1$, liczba $(n+1)^{-1}$. Wtedy do obliczenia $\bar{x}$ potrzebny jest oprócz poprzedniego jeszcze jeden rozkaz $\varrho_1\ \xi+n+1\ \varrho_1\, 05$ — rozkaz pomnożenia $\sum_{k=0}^{n} x_k$ przez $(n+1)^{-1}$. Po jego wykonaniu, w komórce ϱ_1 znajdzie się średnia $\bar{x}$.

Na obliczenie wariancji s^2 składają się następujące czynności:

I. obliczenie kwadratów liczb x_k,
II. obliczenie sumy tych kwadratów,
III. pomnożenie tej sumy przez $(n+1)^{-1}$,
IV. obliczenie kwadratu średniej $\bar{x}$,
V. obliczenie różnicy wyników czynności III i IV.

Czynności I odpowiadają rozkazy

$$\begin{array}{l} \xi \quad \xi \quad \xi \quad 05, \\ \xi+1 \; \xi+1 \; \xi+1 \; 05, \\ \ldots\ldots\ldots\ldots \\ \xi+n \; \xi+n \; \xi+n \; 05. \end{array} \tag{6}$$

Po ich wykonaniu w komórkach $\xi, \xi+1, \ldots, \xi+n$ będą się mieścić liczby $x_0^2, x_1^2, \ldots, x_n^2$. Układ tych rozkazów jest równoważny rozkazowi

$$\begin{array}{l} 0000\; n\; 0000\; 37, \\ \xi \quad \xi \quad \xi \quad 05. \end{array}$$

Kod operacji grupowej równa się tu $30+4\cdot 1+2\cdot 1+1\cdot 1 = 37$, ponieważ rozkazy (6) różnią się między sobą wszystkimi adresami. Sumę $\sum_{k=0}^{n} x_k^2$ można by obliczyć tak samo, jak sumę liczb x_k, ale ponieważ zawartość komórki ξ nie będzie już dalej potrzebna, więc nie warto zajmować jeszcze jednej komórki — wystarczy do zawartości komórki ξ dodać kolejno zawartości komórek $\xi+1, \xi+2, \ldots, \xi+n$, tj. wykonać n rozkazów $\xi\; \xi+1\; \xi\; 01$, $\xi\; \xi+2\; \xi\; 01$, ..., $\xi\; \xi+n\; \xi\; 01$ równoważnych rozkazowi

$$\begin{array}{l} 0000\; n-1\; 0000\; 32, \\ \xi \quad \xi+1 \quad \xi \quad 01. \end{array}$$

Suma $\sum_{k=0}^{n} x_k^2$ znajdzie się wtedy w komórce ξ. Należy z kolei pomnożyć ją przez $(n+1)^{-1}$ (rozkaz $\xi\; \xi+n+1\; \xi\; 05$) i podnieść do kwadratu średnią $\bar{x}$ znajdującą się w komórce ϱ_1 (rozkaz $\varrho_1\; \varrho_1\; \xi+1\; 05$; ponieważ jedną z liczb szukanych jest średnia $\bar{x}$, więc nie możemy rezultatu mnożenia umieścić w komórce ϱ_1; posyłamy go np. do komórki $\xi+1$, której poprzednia zawartość przestała już być potrzebna). Wreszcie wykonując rozkaz $\xi\; \xi+1\; \xi\; 03$, tj. odejmując od $(n+1)^{-1}\sum_{k=0}^{n} x_k^2$ liczbę $\bar{x}^2$, maszyna obliczy wariancję s^2 i umieści ją w komórce ξ. Średnia $\bar{x}$ natomiast znalazła się już wcześniej w komórce ϱ_1.

W ten sposób ustaliliśmy ciąg rozkazów potrzebnych dla obliczenia wielkości $\bar{x}$ i s^2. Ponieważ dane wyjściowe $x_0, x_1, \ldots, x_n, (n+1)^{-1}$ wprowadza się do pamięci maszyny w układzie dziesiętnym, więc wspomniany ciąg należy poprzedzić rozkazem przetłumaczenia tych danych na układ dwójkowy, tj. rozkazem $\xi\ n+1\ \xi\ 72$ (zob. § 5.4), a tuż za tym ciągiem umieścić rozkazy przetłumaczenia wyników $\bar{x}$ i s^2 z powrotem na układ dziesiętny i wydziurkowania ich na karcie. Jeśli przyjąć, że $\varrho_1 = \xi-1$ (a nic nie stoi temu na przeszkodzie, bo komórka $\xi-1$ nie była używana do innych celów), to rozkazy te będą następujące:

$$\xi-1\ 0001\ \xi-1\ 70,$$

$$\xi-1\ 0002\ 0000\ 60.$$

Za nimi umieścimy jeszcze rozkaz zatrzymania się maszyny po wykonaniu wszystkich opisanych poprzednio czynności: $\xi-1\ \xi\ 0000\ 40$. Po jego wykonaniu na biurku operatora ukażą się liczby $\bar{x}$ i s^2 w układzie dziesiętnym.

Ostatecznie — przy założeniu, że przed przystąpieniem do rachunku dane wyjściowe znajdowały się już w pamięci — program obliczeń składa się z następujących rozkazów (w pierwszej kolumnie podano numery komórek, w których odpowiednie rozkazy mają się znajdować):

a	ξ	$n+1$	ξ 72	tłumaczenie na układ dwójkowy,
$a+1$	0000	n	0000 32	obliczenie $\sum_{k=0}^{n} x_k$,
$a+2$	$\xi-1$	ξ	$\xi-1$ 01	
$a+3$	$\xi-1$	$\xi+n+1$	$\xi-1$ 05	obliczenie $\bar{x}$,
$a+4$	0000	n	0000 37	obliczenie x_k^2,
$a+5$	ξ	ξ	ξ 05	
$a+6$	0000	$n-1$	0000 32	obliczenie $\sum_{k=0}^{n} x_k^2$,
$a+7$	ξ	$\xi+1$	ξ 01	
$a+10$	ξ	$\xi+n+1$	ξ 05	obliczenie $(n+1)^{-1}\sum_{k=0}^{n} x_k^2$,
$a+11$	$\xi-1$	$\xi-1$	$\xi+1$ 05	obliczenie $\bar{x}^2$,
$a+12$	ξ	$\xi+1$	ξ 03	obliczenie s^2,
$a+13$	$\xi-1$	0001	$\xi-1$ 70	tłumaczenie na układ dziesiętny,
$a+14$	$\xi-1$	0002	0000 60	dziurkowanie wyników na kartach,
$a+15$	$\xi-1$	ξ	0000 40	koniec rachunku.

Powiedzieliśmy, że program ten jest kompletny przy założeniu, iż dane wyjściowe znalazły się już w pamięci maszyny. Aby to się stało, trzeba właściwy program poprzedzić grupą rozkazów umieszczonych na oddzielnej karcie, tzw. *karcie wprowadzającej*. Po naciśnięciu odpowiedniego przycisku na biurku operatora maszyna posyła 12 rozkazów mieszczących się na karcie wprowadzającej do ustalonych komórek pamięci (od 0004 do 0017) i przechodzi automatycznie do wykonania tych rozkazów w ich naturalnej kolejności. One to właśnie służą do wprowadzenia właściwego programu obliczeń i danych pomocniczych do pamięci. W rozpatrywanym przykładzie na karcie wprowadzającej należy umieścić następujące rozkazy:

0004	0013 0013 0000 20,
0005÷0012	0000 0000 0000 00 (puste komórki),
0013	0000 0016 α 60,
0014	0000 $n+2$ ξ 60,
0015	α α 0000 20,
0016÷0017	0000 0000 0000 00 (puste komórki).

Pierwszy rozkaz powoduje jedynie przejście do wykonania rozkazu z komórki 0013. Jest to związane z tym, że komórki 0001÷0012 są komórkami roboczymi, używanymi przy wykonywaniu operacji standardowych (np. operacji przesyłania 60), że zatem ich ewentualna zawartość przeniesiona z karty wprowadzającej zostałaby starta przy pierwszym użyciu takiej operacji. Rozkazy z komórek 0013 i 0014 powodują (zob. § 5.5) wprowadzenie do pamięci wewnętrznej programu obliczenia $\bar{x}$ i s^2 oraz danych liczbowych, a rozkaz z komórki 0015 — przejście do wykonania tego programu.

Pozostaje tylko jeszcze jedna czynność związana z programowaniem obliczeń: zamiana symboli literowych ξ, α i n na ich konkretne wartości liczbowe. Ponieważ program zajmuje komórki $\alpha \div \alpha+15$, dane wyjściowe i wyniki obliczeń zajmują komórki $\xi-1 \div \xi+n+1$, a program wprowadzający — komórki 0004÷0017, więc wartości α i ξ należy ustalić tak, by przedziały $\langle \alpha, \alpha+15 \rangle$, $\langle \xi-1, \xi+n+1 \rangle$, $\langle 0004, 0017 \rangle$ nie zachodziły na siebie. Można więc np. przyjąć, że $\alpha = 0020$ i $\xi-1 = \alpha+16$, tj. $\xi = \alpha+17 = 0037$. Wartość n zależy oczywiście od konkretnych danych zadania.

Przypuśćmy, że $n = 207$ (w układzie ósemkowym). Wtedy, przy przyjętych powyżej wartościach na α i ξ, karta wprowadzająca i program obliczeń będą miały postać podaną w tablicy 3 (str. 92).

Rozpatrywany program był taki prosty, że można by nie używać symboli literowych dla oznaczania numerów komórek i od razu podawać ich konkretne wartości. Wtedy jednak, gdy program jest skomplikowany, posługiwanie się takimi symbolami staje się konieczne, gdyż trudno z góry przewidzieć, ile komórek zajmie program i pośrednie wyniki obliczeń, w jakiej kolejności powinny być umieszczone części programu itd.

TABLICA 3

0004	0013 0013 0000 20,
0005÷0012	0000 0000 0000 00,
0013	0000 0016 0020 60,
0014	0000 0211 0037 60,
0015	0020 0020 0000 20,
0016÷0017	0000 0000 0000 00;
0020	0037 0210 0037 72,
0021	0000 0207 0000 32,
0022	0036 0037 0036 01,
0023	0036 0247 0036 05,
0024	0000 0207 0000 37,
0025	0037 0037 0037 05,
0026	0000 0206 0000 32,
0027	0037 0040 0037 01,
0030	0037 0247 0037 05,
0031	0036 0036 0040 05,
0032	0037 0040 0037 03,
0033	0036 0001 0036 70,
0034	0036 0002 0000 60,
0035	0036 0037 0000 40.

6.2. Program opisany w § 6.1 nie jest typowy, ponieważ każdy rozkaz w nim zawarty maszyna wykonuje tylko raz. Gdyby tak miało być zawsze, to automatyczne maszyny cyfrowe nie miałyby wielkiego znaczenia, gdyż ułożenie programu trwałoby niewiele krócej niż jego wykonanie przy pomocy arytmometrów. Poza tym nie zawsze znamy z góry te operacje i ich ilość, które będą potrzebne dla rozwiązania zadania (przykład — rozwiązywanie równań metodą iteracji). Dzięki operacjom kierującym obliczeniami (§ 5.6) i operacjom na rozkazach (§ 5.2) jeden rozkaz zawarty w pamięci może być wykonywany wielokrotnie i w razie potrzeby zmieniany. Pokażemy to na przykładzie programu obliczania wartości wielomianu algebraicznego.

Załóżmy, że w komórkach $\gamma, \gamma+1, \ldots, \gamma+n$ dane są współczynniki $c_0, c_1, \ldots, c_n$ wielomianu $\varphi(t) = c_0 t^n + c_1 t^{n-1} + \ldots + c_{n-1} t + c_n$, a w komórce τ — wartość zmiennej t, dla której należy obliczyć wartość tego wielomianu. Najłatwiej to zrobić stosując schemat Hornera, tj. przedstawiając wielomian φ w postaci

$$\varphi(t) = \big(\ldots\big((c_0 t + c_1)t + c_2\big)t + \ldots + c_{n-1}\big)t + c_n.$$

Istotnie, można wtedy zauważyć, że $\varphi(t) = x_n$, gdzie $x_0 = c_0$ i $x_k = x_{k-1}t + c_k$ dla $k = 1, 2, \ldots, n$.

Przeznaczmy dla liczb x_k komórkę γ mieszczącą przed wykonaniem obliczeń współczynnik c_0, tj. początkowy wyraz ciągu $x_0, x_1, \ldots, x_n$. Możemy tak uczynić, ponieważ c_0 jest potrzebne tylko do określenia wartości x_0. W równości $x_{k-1}t + c_k = x_k$ liczby x_{k-1}, t, c_k mieszczą się zatem odpowiednio w komórkach $\gamma, \tau, \gamma+k$, a jej prawą stronę, tj. x_k, należy posłać do komórki γ. Dlatego, gdy x_{k-1} jest już znane, x_k oblicza

się przy pomocy rozkazów

$\gamma \quad \tau \quad \gamma\, 05$ (obliczenie $x_{k-1}t$ i posłanie wyniku do γ),

$\gamma\, \gamma+k\, \gamma\, 01$ (obliczenie $x_k = x_{k-1}t+c_k$ i posłanie wyniku do γ).

Pierwszy z wymienionych rozkazów jest dla wszystkich k jednakowy, a w drugim rozkazie wraz z k zmienia się tylko drugi adres $(\gamma+k)$. Stąd wynika, że aby z rozkazów

$$\begin{matrix} \gamma \quad \tau \quad \gamma\, 05, \\ \gamma\, \gamma+k\, \gamma\, 01 \end{matrix} \tag{7}$$

służących do obliczenia x_k otrzymać rozkazy

$$\begin{matrix} \gamma \quad \tau \quad \gamma\, 05, \\ \gamma\, \gamma+k+1\, \gamma\, 01 \end{matrix}$$

służące do obliczenia x_{k+1}, tj. następnego elementu w ciągu $x_0, x_1, \ldots, x_n$, wystarczy drugi adres drugiego rozkazu (7) zwiększyć o 1. Innymi słowy, jeśli rozkazy (7) znajdowały się w komórkach β i $\beta+1$, to bezpośrednio za nimi trzeba umieścić rozkaz $\beta+1\ 7424\ \beta+1\ 02$ (zob. § 5.2); 7424 jest numerem komórki pamięci martwej, zawierającej symbol 0000 0001 0000 00.

Brakuje jeszcze dwu elementów: 1° sprawdzenia, czy wartość wielomianu została już obliczona, tj. czy obliczono x_n; 2° rozkazu powtórnego wykonania rozkazów z komórek $\beta \div \beta+2$ (w zmienionej postaci), jeśli obliczono dopiero x_k dla $k < n$. Ustalić, czy $k < n$, można porównując rozkaz $\gamma\, \gamma+k\, \gamma\, 01$ z komórki $\beta+1$ z tą postacią, którą on przyjmie po obliczeniu x_n. Końcową postacią rozkazu $\beta+1$ jest rozkaz $\gamma\, \gamma+n+1\, \gamma\, 01$, ponieważ po obliczeniu x_n zgodnie z rozkazem

$$\gamma\, \gamma+n\, \gamma\, 01 \tag{8}$$

następujący po nim rozkaz z komórki $\beta+2$ spowoduje zwiększenie drugiego adresu (8) o 1. Zgodnie z tym, w komórce $\beta+3$ umieścimy rozkaz porównania zawartości komórki $\beta+1$ z symbolem $\gamma\, \gamma+n+1\, \gamma\, 01$, który przed rozpoczęciem obliczeń powinien się już znajdować w jakiejś ustalonej komórce ϱ. Rozkaz ten ma postać $\beta+1\ \varrho\ 0000\ 16$. Wynik operacji porównania, określonej w § 5.2, posyłamy do komórki 0000, która właściwie nie istnieje, więc wynik ten ginie. Interesuje nas natomiast wartość sygnału ω powstałego przy wykonywaniu tej operacji. Zgodnie z §§ 5.6 i 5.2 $\omega = 0$, jeśli porównywane symbole (w naszym przypadku zawartości komórek $\beta+1$ i ϱ) są identyczne, i $\omega = 1$, jeśli tak nie jest. Za rozkazem $\beta+1\ \varrho\ 0000\ 16$, tj. w komórce $\beta+4$, umieścimy rozkaz

$\beta+5\ \beta\ 0000\ 20$, którego wykonanie — zgodnie z własnościami operacji przejścia warunkowego I typu — spowoduje przejście do wykonania rozkazu z komórki $\beta+5$, jeśli $\omega = 0$, lub do wykonania rozkazu z komórki β, jeśli $\omega = 1$. Ponieważ porównujemy ze sobą pewien zmienny rozkaz i postać, jaką on przyjmie po obliczeniu wartości wielomianu φ, to $\omega = 0$, jeśli ta wartość jest już obliczona, i $\omega = 1$, jeśli ta wartość jeszcze nie jest obliczona. W pierwszym przypadku maszyna przejdzie do wykonania rozkazu (umieścimy go w komórce $\beta+5$) tłumaczenia wartości wielomianu na układ dziesiętny, w drugim przypadku natomiast — do obliczenia x_{k+1} (pierwszy rozkaz w komórce β).

Wypiszmy raz jeszcze wszystkie już ustalone rozkazy:

β	$\gamma \quad \tau \quad \gamma \quad 05,$
$\beta+1$	$\gamma \quad \gamma+k \quad \gamma \quad 01,$
$\beta+2$	$\beta+1\ 7424\ \beta+1\ 02,$
$\beta+3$	$\beta+1 \quad \varrho \quad 0000\ 16,$
$\beta+4$	$\beta+5 \quad \beta \quad 0000\ 20,$

i przypomnijmy ich rolę. Rozkazy β i $\beta+1$ powodują obliczenie liczby x_k. Rozkaz $\beta+2$ przekształca rozkaz $\beta+1$ do postaci, jaką ten ostatni powinien mieć przy obliczaniu x_{k+1}. Rozkaz $\beta+3$ sprawdza czy $k = n$, czy $k < n$, a rozkaz $\beta+4$ zleca wykonanie odpowiedniego rozkazu w zależności od wyniku tego sprawdzenia.

Podobnie jak to było w przykładzie z § 6.1, grupę rozkazów $\beta \div \beta+4$ poprzedzamy rozkazem tłumaczenia danych liczbowych $c_0, c_1, \ldots, c_n, t$ na układ dwójkowy. Jeśli przyjmiemy, że $\tau = \gamma+n+1$, to rozkaz ten będzie miał postać

$$\beta-1 \mid \gamma\ n+1\ \gamma\ 72.$$

W komórkach $\beta+5, \ldots$ umieścimy rozkaz tłumaczenia wyniku $\varphi(t)$ znajdującego się w komórce γ na układ dziesiętny, rozkaz wydziurkowania tego wyniku na karcie i rozkaz zatrzymania się maszyny:

$\beta+5$	$\gamma \quad 0000 \quad \gamma \quad 70,$
$\beta+6$	$\gamma \quad 0001\ 0000\ 60,$
$\beta+7$	$0000\ 0000\ 0000\ 40.$

Ostatecznie więc program zajmuje komórki od $\beta-1$ do $\beta+7$, w komórce ϱ znajduje się symbol $\gamma\ \gamma+n+1\ \gamma\ 01$, a komórki $\gamma \div \gamma+n+1$ są zajęte przez dane liczbowe. Dlatego możemy przyjąć, że np. $\beta-1 = 0020$, tj. $\beta = 0021$, $\varrho = \beta+10 = 0031$, $\gamma = \varrho+1 = 0032$. Wtedy

dla jakiejś konkretnej wartości n, na przykład dla $n = 10$, program obliczenia wartości wielomianu oraz rozmieszczenie tego programu i danych pomocniczych w pamięci będą następujące:

0020	0032 0011 0032 72	program obliczeń,
0021	0032 0043 0032 05	
0022	0032 0033 0032 01	
0023	0022 7424 0022 02	
0024	0022 0031 0000 16	
0025	0026 0021 0000 20	
0026	0032 0000 0032 70	
0027	0032 0001 0000 60	
0030	0000 0000 0000 40	
0031	0032 0043 0032 01	stała pomocnicza,
0032÷0042	$c_0, c_1, \ldots, c_n$	dane liczbowe
0043	t	

(w rozkazie z komórki $\beta+1 = 0022$ przyjęliśmy teraz — pisząc program w jego ostatecznej postaci — $k = 1$, gdyż maszyna powinna zacząć od obliczenia x_1).

Opisany program ma już zasadniczą cechę programów istotnie spotykanych w praktyce — cykliczność. Polega ona na tym, że pewną grupę rozkazów zawartych w pamięci maszyna wykonuje — zmieniając je, jeśli to jest potrzebne — wielokrotnie i że sama maszyna sprawdza, kiedy powtarzanie rozkazów tej grupy powinna zakończyć i przejść do wykonania następnej części programu.

§ 7. Zakończenie

Celem autora nie było, rzecz jasna, pokazanie zasadniczych metod programowania; trudno zresztą byłoby to zrobić w krótkiej pracy. W § 6 pokazano jedynie na najprostszych przykładach, jak wykorzystuje się różne rodzaje operacji, które może wykonywać maszyna, jak z drobnych elementów powstaje program. Opis Strieły zakończymy kilkoma uwagami o szybkości jej pracy.

Powiedzieliśmy już, że Strieła wykonuje około 2000 operacji na sekundę. Trzeba jednak pamiętać o tym, że oprócz działań arytmetycznych niezbędnych dla wykonania obliczeń i wtedy, gdy posługujemy się prostszymi urządzeniami rachującymi, maszyna musi wykonać wiele pomocniczych operacji — operacji zmieniających rozkazy, operacji kierujących obliczeniami itp. (w przykładzie z § 6.2 odpowiadają im rozkazy 0020, 0023÷0030). Frakcją tych pomocniczych operacji często przekracza $\frac{3}{4}$. Aby pokazać efektywną szybkość maszyny, przytoczymy przykład obliczeń, które były rzeczywiście wykonane na Striele. Dla

potrzeb teorii aproksymacji okazało się konieczne rozwiązanie w przedziale $\langle 0, 1\rangle$ układu równań różniczkowych

$$\frac{dc_k}{dp} = \frac{c_k^2-1}{(1-pc_k)\left[2+(1-p^2)\sum_{i=1}^{n}(1-pc_i)^{-1}\right]} \qquad (k = 1, 2, \ldots, n)$$

z warunkami początkowymi $c_k(0) = -\cos k\pi/(n+1)$. Układ rozwiązano metodą Rungego-Kutty 4-go rzędu z krokiem ustalanym tak, aby zapewnić wystarczającą dokładność rozwiązania. W metodzie tej dla obliczenia jednego układu wartości funkcji $c_1, c_2, \ldots, c_n$ trzeba 11 razy obliczać prawe strony podanych wyżej równań różniczkowych. Dla każdego n obliczono 100 takich układów, a n zmieniało się od 1 do 13 co jeden. Znalezienie wszystkich 9100 wartości funkcji c_k trwało około 3 godzin.

INSTYTUT MATEMATYCZNY POLSKIEJ AKADEMII NAUK

Praca wpłynęła 13. 12. 1958

С. ПАШКОВСКИЙ (Варшава)

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА „СТРЕЛА-4"

РЕЗЮМЕ

В работе дано описание советской электронной цифровой вычислительной машины „Стрела-4". Приводится вид чисел, с которыми оперирует машина, и вид команд, типы памяти, устройства ввода и вывода численного материала, а также подробное описание всех операций, выполняемых машиной.

S. PASZKOWSKI (Warszawa)

ELECTRONIC DIGITAL COMPUTER STRELA-4

SUMMARY

The author describes the Soviet electronic digital computer Strela-4 (Arrow-4). He gives the form of the numbers on which the machine performs operations, the form of the orders, the kinds of memory of the machine, the input and the output and a detailed description of all operations.
