ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
VIII (1965)

T. KAMBURELIS (Wrocław)

MASZYNA CYFROWA ODRA 1003

Maszyna cyfrowa ODRA 1003, opracowana i produkowana już seryjnie we Wrocławskich Zakładach Elektronicznych, jest pierwszą polską maszyną cyfrową skonstruowaną całkowicie na elementach półprzewodnikowych i wykonującą automatycznie operacje zmiennoprzecinkowe.

Jest to maszyna o małych wymiarach (1640 mm × 670 mm × 1235 mm) i zwartej konstrukcji (patrz zdjęcia). Jest ona pomyślana jako mała i stosunkowo tania maszyna, przeznaczona głównie do obliczeń techniczno-naukowych i do zastosowań w procesach technologicznych. Może również rozwiązywać pewne problemy ekonomiczne i administracyjne, chociaż nie jest specjalnie przystosowana do tych celów.

Maszyna cyfrowa ODRA 1003 dzięki swoim walorom technicznym i organizacyjnym, a także dzięki specjalnym opracowaniom matematycznym (bogata biblioteka programów i wygodny system automatycznego programowania) nadaje się do szerokiego szkolenia w wyższych uczelniach.

Maszyna ODRA 1003 została zainstalowana, do końca 1964 roku, w dwunastu ośrodkach obliczeniowych. Są to między innymi:

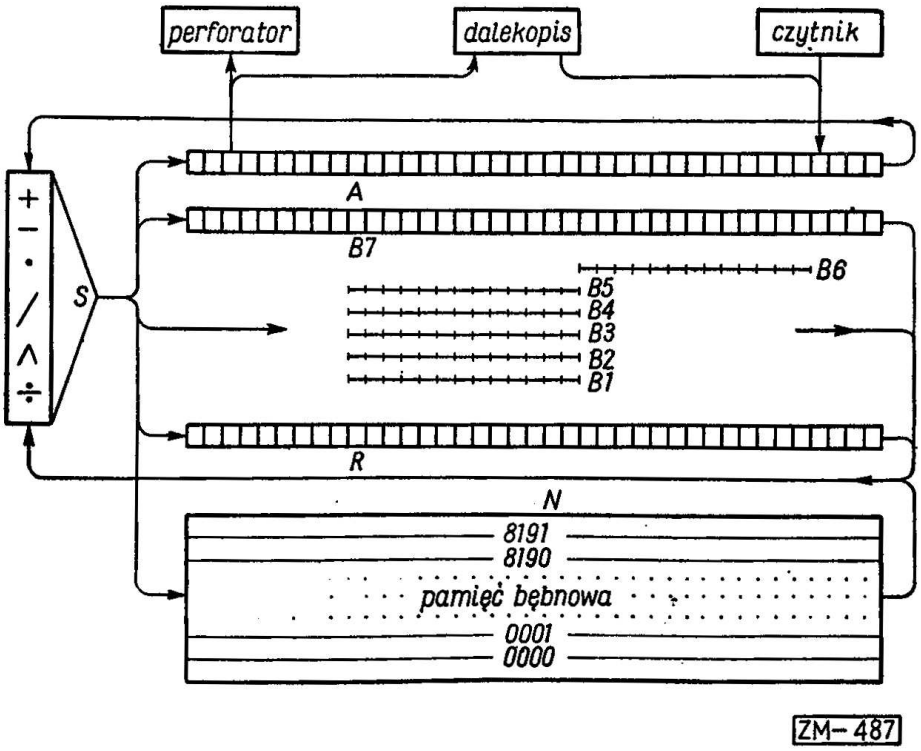
— Huta im. Lenina w Nowej Hucie,
— Instytut Automatyki Systemów Energetycznych we Wrocławiu,
— Instytut Metalurgii Żelaza w Gliwicach,
— Instytut Zootechniki w Krakowie,
— Zakłady Energetyczne Okręgu Południowego w Katowicach,
— Zakłady Energetyczne Okręgu Wschodniego w Radomiu,
— Wytwórnia Sprzętu Komunikacyjnego w Rzeszowie.

Liczba instalacji maszyn ODRA 1003 dokonanych w roku 1965 zostanie zwielokrotniona. Znajdą się one między innymi w wyższych uczelniach, w instytutach naukowych, w biurach projektowych, w hutach, w zakładach energetycznych.

1. Ogólna organizacja maszyny

W maszynie ODRA 1003 wyróżnia się (z punktu widzenia programowania) następujące bloki:

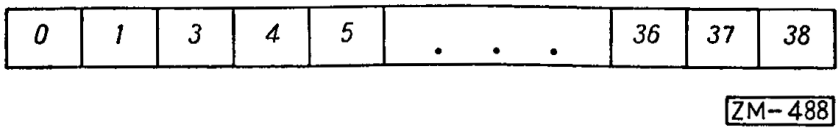
- *arytmometr*, zawierający rejestry $A, B1, \ldots, B7$ (patrz rys. 1) oraz zespół S, w którym odbywają się operacje arytmetyczne i logiczne;
- *sterowanie* z rejestrem R, do którego skierowywane są rozkazy do wykonania;
- *pamięć* bębnowa z 8192 komórkami ponumerowanymi od 0 do 8191 (oznacza się je ogólnie przez N);
- *wejście*, zawierające czytnik 5-kanałowy taśmy papierowej i dalekopis, dla wprowadzania informacji do maszyny;
- *wyjście*, zawierające perforator 5-kanałowy taśmy papierowej i dalekopis (ten sam, który stanowi wejście), dla wyprowadzania informacji z maszyny;
- *pulpit sterowania*, zawierający klawisze i lampki sygnalizacyjne.



Rys. 1. Schemat blokowy ODRY 1003

W rejestrach maszyny przechowuje się informacje zwane *słowami*. Słowo maszyny ODRA 1003, jest to ciąg cyfr dwójkowych (*bitów*), ma

39 pozycji ponumerowanych od 0 do 38 (patrz rys. 2). Słowo maszyny może oznaczać liczbę lub rozkaz.

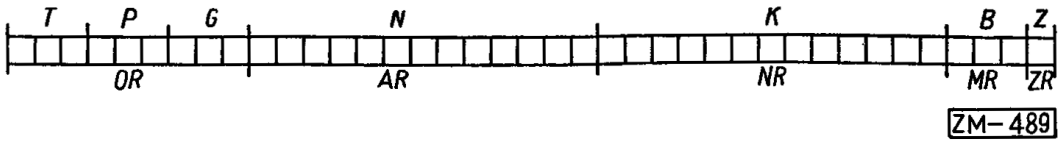


Rys. 2. Słowo maszyny

1.1. Sposób pracy maszyny. Maszyna ODRA 1003 rozwiązuje zadania obliczeniowe lub logiczne, wykonując zadane ciągi operacji elementarnych. Operacje te są przedstawiane w maszynie za pomocą *rozkazów*. Wszystkie możliwe operacje są jednoznacznie ponumerowane od 0 do 511 (w praktyce używa się ósemkowych numerów operacji od 000 do 777). Ciągi rozkazów są zapisywane w pamięci bębnowej maszyny, skąd pojedyncze rozkazy są brane w pewnej kolejności do sterowania (do rejestru R), a następnie wykonywane. Kolejność wykonywania poszczególnych rozkazów ciągu (programu) jest na ogół w maszynach cyfrowych *sekwencyjna*; oznacza to, że rozkazy ciągu są wykonywane w kolejności ich występowania w pamięci. Kolejność ta może być zmieniona za pomocą specjalnych rozkazów sterujących (skokowych). Natomiast w maszynie ODRA 1003 kolejność ta jest dowolna; oznacza to, że po wykonaniu pewnego rozkazu ciągu można przejść do wykonania dowolnego (a więc niekoniecznie następnego) rozkazu ciągu.

Operacje maszyny są jedno- lub dwuargumentowe. Argumentami operacji mogą być słowa zapisane w komórkach pamięci bębnowej (N) lub w rejestrach A, B i R(1). Wynik operacji może być zapamiętany w komórce N lub w rejestrach A, B i R. Informacje o typie operacji (rozkazu), o miejscach argumentów i wyniku operacji oraz o tym, który rozkaz ma być wykonany jako następny, są zawarte w wykonanym rozkazie.

1.2. Struktura słowa rozkazowego. Podział 39-bitowego słowa maszynowego interpretowanego jako rozkaz pokazuje rysunek 3.



Rys. 3. Struktura rozkazu

Pierwsze 9 bitów, część *OR*, określa rodzaj (*kod*) operacji, która ma być wykonana. Część *OR* została podzielona na trzy trójki bitów (T, P, G),

(1) Symbole wszystkich rejestrów podane są w § 2.

czyli 3 cyfry ósemkowe. Zatem w części *OR* rozkazu mogą być zapisane numery ósemkowe od 000 do 777.

Następne 13 bitów, część *AR*, określa numer (*adres*) komórki pamięci bębnowej, w której znajduje się argument operacji (lub do której ma być zapisany wynik operacji) bądź też podaje parametr operacji (np. liczba przesunięć). Liczbę zapisaną w tej części oznacza się literą *N* i nazywa się ją *pierwszym adresem* rozkazu.

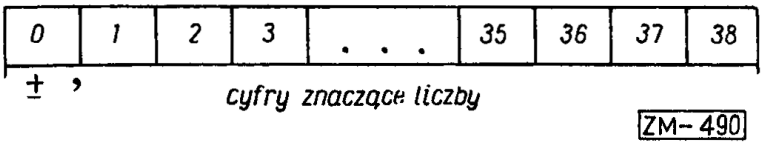
Część *NR*, mająca także 13 bitów, określa adres (*K*) komórki pamięci, w której znajduje się kolejny rozkaz do wykonania. Adres ten nazywa się *drugim adresem* rozkazu.

Część *MR* (3 bity) określa numer rejestru *B*.

Część *ZR* (1 bit) wskazuje, czy rozkaz ten ma być przed wykonaniem zmodyfikowany (patrz § 1.4) czy też nie. Jest to tzw. *znak modyfikacji rozkazu*.

1.3. Słowa liczbowe. Słowo maszynowe może także oznaczać liczbę. Maszyna może wykonywać technicznie działania arytmetyczne na liczbach *stałoprzecinkowych* lub *zmiennoprzecinkowych*.

1.3.1. Liczby stałoprzecinkowe. Znak liczby zapisany jest w bicie zerowym słowa (patrz rys. 4). Jedynka w tym bicie oznacza liczbę ujemną, a zero dodatnią. Cyfry znaczące liczby zapisane są w bitach od 1 do 38. Maszyna traktuje przecinek liczby jako umiejscowiony między pozycją zerową a pierwszą. Zatem jedynka w n-tej pozycji słowa dodatniego oznacza dla maszyny wartość 2^{-n}.



Rys. 4. Liczba stałoprzecinkowa

Liczby przedstawione są w arytmetyce uzupełnieniowej według wzoru

$$x_{uz} = \begin{cases} x & \text{dla} \quad x \geqslant 0, \\ 2+x & \text{dla} \quad x < 0. \end{cases} \tag{1}$$

Liczba x przyjmuje wartości z przedziału $[-1, 1-2^{-38}]$. A więc są to liczby stałoprzecinkowe ułamkowe.

W programach przyjmuje się często umownie inne stałe położenia przecinka liczby niż pokazuje rysunek 4, np. po pozycji 38 słowa liczbowego. Wówczas liczby są traktowane jako całkowite, należące do przedziału $[-2^{38}, 2^{38}-1]$, lecz wtedy po mnożeniu lub dzieleniu konieczna jest korekta położenia przecinka wyniku.

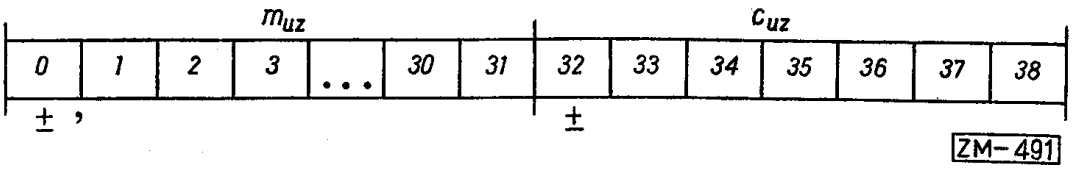
1.3.2. Liczby zmiennoprzecinkowe. W maszynie istnieje również drugi sposób przedstawienia liczby (oprócz opisanego w § 1.3.1), który zawiera informację o miejscu przecinka w liczbie. Sposób ten polega na przedstawieniu liczby x w postaci pary (m, c), gdzie

$$x = m \cdot 2^c; \tag{2}$$

c jest liczbą całkowitą z przedziału $[-64, +63]$, zaś m liczbą spełniającą warunek

$$-1 \leqslant m \leqslant 1-2^{-31}.$$

Liczby c i m są nazwane odpowiednio *cechą* i *mantysą* liczby x. Liczba x wyrażona w postaci (2) nazywa się liczbą zmiennoprzecinkową.



Rys. 5. Liczba zmiennoprzecinkowa

Strukturę liczby zmiennoprzecinkowej w maszynie przedstawia rysunek 5. Bit zerowy określa znak mantysy (0 — plus, 1 — minus), a bity od 1 do 31 określają cyfry mantysy po przecinku. Bit 32 określa znak cechy (0 — plus, 1 — minus), zaś bity od 33 do 38 określają kolejne cyfry znaczące cechy. Zarówno mantysa m jak i cecha c liczby są przedstawione w maszynie w arytmetyce uzupełnieniowej według wzorów:

$$m_{uz} = \begin{cases} m & \text{dla} \quad m \geqslant 0, \\ 2+m & \text{dla} \quad m < 0, \end{cases} \tag{3}$$

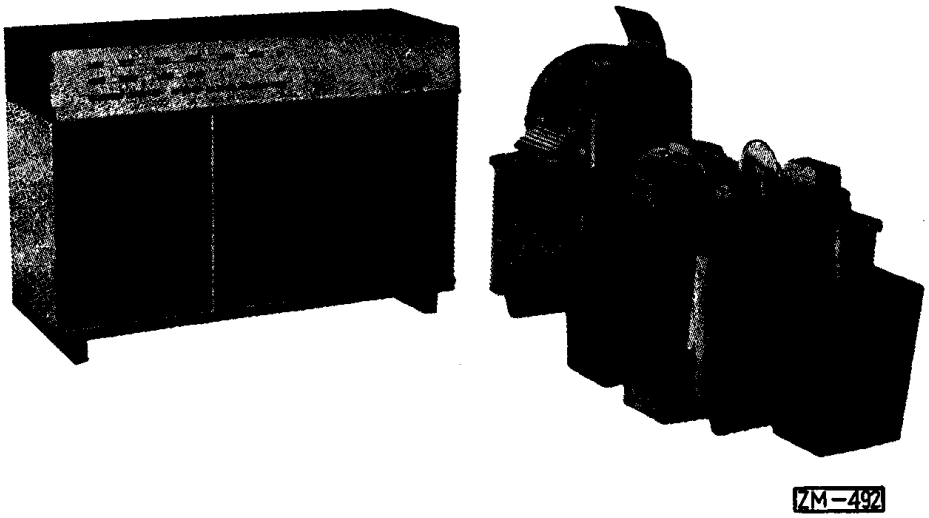
$$c_{uz} = \begin{cases} c & \text{dla} \quad c \geqslant 0, \\ 2^7+c & \text{dla} \quad c < 0, \end{cases} \tag{4}$$

Z przyjętej struktury liczby zmiennoprzecinkowej x wynika, że należy ona do przedziału $[-2^{63}, 2^{63}-2^{32}]$.

1.4. Modyfikacja rozkazów. W maszynie istnieje 7 specjalnych rejestrów zwanych rejestrami B, przeznaczonych głównie do *modyfikacji* rozkazu. Modyfikacja rozkazu r, pobranego z pamięci bębnowej do wykonania, polega na dodaniu do niego liczby zawartej we wskazanym (przez część MR rozkazu) rejestrze B. Jeżeli zawartość (stan) rejestru B oznaczy się przez b, to zmodyfikowany rozkaz r' wyraża się wzorem

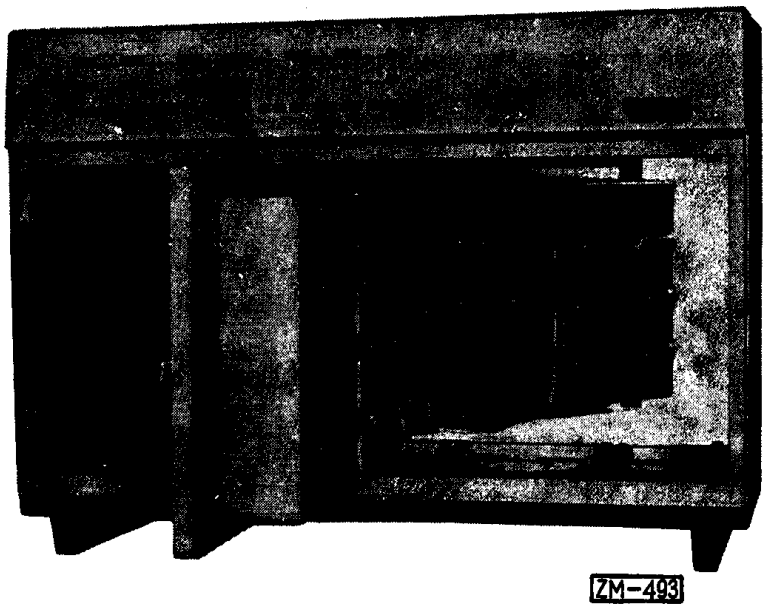
$$r' = r+b.$$

Maszyna wykonuje rozkaz r (niezmodyfikowany) lub r' (zmodyfikowany) w zależności od stanu bitu ZR omawianego rozkazu. Jeśli $ZR = 0$, to wykonuje ona rozkaz r, w przeciwnym przypadku r'. W obu przypadkach stan wskazanego rejestru B i rozkaz r w pamięci nie zmieniają się.



Rys. 6. Maszyna cyfrowa ODRA 1003. Widok ogólny maszyny wraz z urządzeniami zewnętrznymi

Rejestry od $B1$ do $B6$ mają po 13 bitów (patrz rys. 1), a rejestr $B7$ ma 39 bitów. Rejestr $B7$ oznacza się także przez M. Zawartością re-

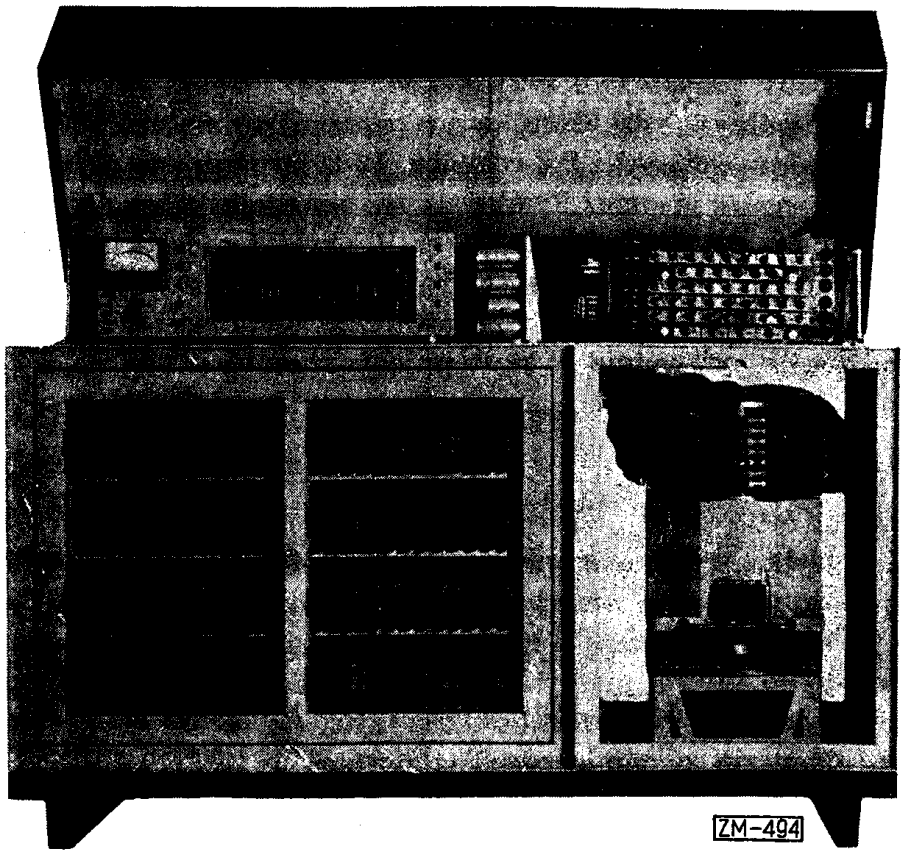


Rys. 7. Szafa maszyny otwarta (widok z przodu)

jestru $B7$ można modyfikować dowolną część rozkazu (OR, AR, NR, MR) lub wszystkie jego części. Zawartością jednego z pierwszych pięciu

rejestrów *B* można modyfikować tylko część *AR* modyfikowanego rozkazu, zaś zawartością rejestru *B*6 jego część *NR*.

Automatyczna modyfikacja rozkazu pozwala na wykonanie tego samego ciągu operacji elementarnych na różnych argumentach. A więc można sprawnie organizować programy cykliczne.



Rys. 8. Szafa maszyny z podniesioną górną pokrywą (widok z tyłu)

2. Lista rozkazów maszyny

Przy omawianiu listy rozkazów będziemy używali następujących skrótów:

A — rejestr akumulatora (39 bitów),

M — rejestr mnożnika (lub rej. *B*7, 39 bitów),

AM — rejestr długi, składający się z 39 bitów rej. *A* (od 0 do 38) i z 38 bitów rej. *M* (od 1 do 38),

R — rejestr rozkazów (39 bitów),

B — rejestry *B* (od *B*1 do *B*7),

N — pierwszy adres rozkazu (po ewentualnej modyfikacji),

K — drugi adres rozkazu (po ewentualnej modyfikacji),

Z — znak modyfikacji rozkazu (część ZR).

Małe litery: a, m, am, r, b, n oznaczają odpowiednio zawartości rejestrów: A, M, AM, R, B, N.

Litery: a', m', am', r', b', n' oznaczają zawartości odpowiednich rejestrów po wykonaniu operacji.

2.1. Zasady przyporządkowania kodów rozkazom. Jak już wspomniano, poszczególnym rozkazom są przyporządkowane trójcyfrowe kody ósemkowe zapisane w części OR rozkazu. W tym przyporządkowaniu lewa cyfra kodu, oznaczona przez T, określa *typ* operacji, która ma być wykonana. Środkowa cyfra, oznaczona przez P, określa *podstawienie*, jakie ma być dokonane w operacji. Prawa cyfra oznaczona przez G, określa *grupę* operacji. Zatem rozkaz łącznie z pozostałymi jego częściami (patrz § 1.2) może być napisany ogólnie w następujący sposób:

$$TPG\ N\ K\ BZ$$

Na przykład rozkaz

412 2500 1305 20

oznacza, że będzie wykonana operacja o kodzie 412 ($T = 4, P = 1, G = 2$), że jeden z argumentów operacji znajduje się w komórce 2500 ($N = 2500$), i że w operacji wymieniony jest rejestr B o numerze 2 ($B = 2$). Rozkaz ten będzie wykonany w maszynie bez modyfikacji ($Z = 0$) a jako następny będzie wykonany rozkaz z komórki 1305 ($K = 1305$).

Znajdujący się w pamięci rozkaz postaci

412 2500 1305 21

oznacza tę samą operację co poprzedni rozkaz, z tą tylko różnicą, że rozkaz ten zostanie zmodyfikowany ($Z = 1$) o zawartość rejestru $B2$ przed wykonaniem. Jeśli na przykład w rejestrze $B2$ jest zapisana liczba 15, to zostanie wykonany rozkaz postaci

412 2515 1305 20.

W tym przypadku jeden z argumentów operacji zostanie pobrany z komórki 2515.

Operacje wykonywane w maszynie są, w większości przypadków, operacjami dwuargumentowymi.

Jeden z tych argumentów jest argumentem *bezadresowym*, tzn. że w rozkazie nie podaje się informacji (adresu) o jego lokalizacji. Jest nim zawartość akumulatora A, czyli a.

Drugi argument operacji jest adresowany i może nim być zawartość dowolnej komórki pamięci bębnowej (o adresie N), zawartość dowolnego

rejestru B (o adresie B), dowolny adres N, albo też zero. Jeśli drugi argument operacji oznaczy się przez x, to większość operacji można napisać w postaci

$$s = a \oplus x \quad \text{albo} \quad s = x \oplus a,$$

gdzie $\oplus$ jest znakiem operacji arytmetycznej lub logicznej, a s wynikiem tej operacji.

W dalszej części niniejszego opisu zostanie podana treść, jaką przypisano poszczególnym rozkazom maszyny.

2.2. Lewa cyfra kodu (T). Wartość cyfry ósemkowej T określa rodzaj operacji (arytmetycznej lub logicznej), wykonanej na argumentach a i x według poniższej tabeli:

T	operacja
0	x
1	$-x$
2	$\lvert a \rvert$
3	$a-x$
4	$a+x$
5	$x-a$
6	$a \wedge x$
7	$a \div x$

gdzie znaki $+$ i $-$ oznaczają tutaj odpowiednio dodawanie i odejmowanie na liczbach stałoprzecinkowych. Symbole $\wedge$ i $\div$ oznaczają operacje logiczne — koniunkcję i różnicę symetryczną.

2.3. Środkowa cyfra kodu (P). Cyfra P określa, co jest argumentem x oraz czy umieszcza się wynik s operacji określonej cyfrą T w akumulatorze A:

P	podstawienie
0	$x = 0$
1	$x = n$
2	$x = b$
3	$x = N$
4	$x = 0$ i $a' = s$
5	$x = n$ i $a' = s$
6	$x = b$ i $a' = s$
7	$x = N$ i $a' = s$

Jeśli na przykład $T = 3$ i $P = 5$, to zostaną wykonane w maszynie następujące funkcje:

$$s = a - x, \qquad \text{ponieważ } T = 3,$$

$$x = n \text{ i } a' = a - n, \qquad \text{ponieważ } P = 5,$$

czyli różnica zawartości akumulatora A i komórki o adresie N zostanie wpisana do akumulatora.

W przypadku $T = 0$ i $P = 7$ wykonana zostanie operacja:

$$s = x, \qquad \text{ponieważ } T = 0,$$

$$x = N \text{ i } a' = N, \qquad \text{ponieważ } P = 7,$$

czyli adres N zostanie wpisany do akumulatora.

Uwaga. Adres N oraz zawartości pierwszych sześciu rejestrów B (13-bitowe) są podstawiane do operacji zawsze jako argumenty dodatnie, mające: 9 zer, po nich 13-bitowy adres (lub zawartość rejestru B) i znów 17 zer.

2.4. Prawa cyfra kodu (G). Wartość cyfry G decyduje o umieszczeniu wyniku s (operacji określonej cyfrą T) w komórce o adresie N, albo w rejestrze B, albo w rejestrze rozkazów R, albo określa grupę operacji (np. mnożenia, przesuwania):

G	operacja grupowa
0	—
1	$n' = s$
2	$b' = s$
3	$r' = s$
4	$am' = a \cdot x$ albo $am' = s \cdot x$
5	$a' = a/x$ albo $a' = s/x$

Jeśli więc $G = 0$, to wynik s nie jest kierowany do żadnego z rejestrów maszyny (chyba że $P \geqslant 4$, wtedy jest $a' = s$). Jeśli $G = 1, 2, 3$, to wynik s zostaje przesłany odpowiednio do komórki N, do rejestru B, do rejestru rozkazów R. W tym ostatnim przypadku wynik s traktuje się w rejestrze R jako rozkaz i wykonuje się go. A więc jest to inny sposób modyfikacji rozkazu, szczególnie przydatny w programach organizacyjnych. Jeśli $G = 4, 5$ to wykonuje się odpowiednio mnożenie lub dzielenie na liczbach stałoprzecinkowych postaci:

$$am' = a \cdot x \quad \text{lub} \quad a' = a/x \quad \text{dla} \quad P < 4,$$

bądź

$$am' = s \cdot x \quad \text{lub} \quad a' = s/x \quad \text{dla} \quad P \geqslant 4.$$

Przykłady mnożeń dla $P < 4$:

TPG	mnożenie
−04	$am' = a \cdot 0$
−14	$am' = a \cdot n$
−24	$am' = a \cdot b$
−34	$am' = a \cdot N$

Znak — oznacza tutaj, że wartość cyfry T jest dowolna. Przykłady mnożeń dla $P \geqslant 4$, np. $P = 5$:

TPG	mnożenie
054	$am' = n \cdot n$
154	$am' = -n \cdot n$
254	$am' = \lvert a \rvert \cdot n$
354	$am' = (a - n) \cdot n$
454	$am' = (a + n) \cdot n$
554	$am' = (n - a) \cdot n$
654	$am' = (a \wedge n) \cdot n$
754	$am' = (a \div n) \cdot n$

Analogiczne przykłady można zapisać także dla dzieleń stałoprzecinkowych podstawiając $G = 5$.

Operacje grupy $G = 6$ i $G = 7$ zostaną opisane w następnych paragrafach, tutaj zaś podane będą dotychczas opisane rozkazy ujęte w tabelę:

	Kod rozkazu		
	T	P	G
0	x	$x = 0$	—
1	$-x$	$x = n$	$n' = s$
2	$\lvert a \rvert$	$x = b$	$b' = s$
3	$a - x$	$x = N$	$r' = s$
4	$a + x$	$x = 0$ i $a' = s$	$am' = a \cdot x$
5	$x - a$	$x = n$ i $a' = s$	$a' = a/x$
6	$a \wedge x$	$x = b$ i $a' = s$	patrz część II
7	$a \div x$	$x = N$ i $a' = s$	

Tabela rozkazów, część I

Powyższa tabela obowiązuje tylko wówczas, gdy wartość cyfry G jest mniejsza od 6. W przeciwnym przypadku obowiązuje część II tabeli rozkazów opisana niżej.

2.5. Rozkazy przesuwania. Słowo znajdujące się w akumulatorze A lub w rejestrze długim AM może być przesunięte w lewo lub w prawo o N pozycji. Przesunięcia mogą być: arytmetyczne, naturalne lub cykliczne. Dla wszystkich rozkazów przesuwania zachodzi $P = 1$, a $G = 6$. Zatem kod dowolnego rozkazu przesuwania ma postać $T16$.

2.5.1. Arytmetyczne przesunięcie w lewo (Al).

Symbol: 216 N K BZ,
funkcja: $a' = a \cdot 2^N$,
treść: zawartość akumulatora A zostaje pomnożona N razy przez dwa.

2.5.2. Arytmetyczne przesunięcie w prawo (*Ap*).

Symbol: 316 *N K BZ*,
funkcja: $a' = a/2^N$,
treść: zawartość akumulatora *A* zostaje podzielona *N* razy przez dwa.

2.5.3. Naturalne przesunięcie w lewo (*Lw*).

Symbol: 016 *N K BZ*,
funkcja: $a' = a \cdot 2_N$,
treść: zawartość akumulatora *A* zostaje przesunięta naturalnie w lewo o *N* miejsc. Gubi się *N* lewych bitów akumulatora, a w prawej jego części wpisuje się *N* zer.

2.5.4. Naturalne przesunięcie w prawo (*Pr*).

Symbol: 116 *N K BZ*,
funkcja: $a' = a/2_N$,
treść: zawartość akumulatora zostaje przesunięta naturalnie w prawo o *N* miejsc. Gubi się *N* prawych bitów akumulatora, a w lewej jego części wpisuje się *N* zer.

2.5.5. Cykliczne przesunięcie w prawo (*Cp*).

Symbol: 416 *N K BZ*,
funkcja: $a' = a * N$,
treść: zawartość akumulatora *A* zostaje przesunięta cyklicznie w prawo o *N* miejsc. Bit z pozycji 38 przechodzi do pozycji 0, a bity z pozostałych pozycji przechodzą do sąsiednich.

2.5.6. Arytmetyczne przesunięcie długie w lewo (*Ald*).

Symbol: 516 *N K BZ*,
funkcja: $am' = am \cdot 2^N$,
treść: zawartość rejestru *AM* zostaje pomnożona *N* razy przez dwa.

2.5.7. Arytmetyczne przesunięcie długie w prawo (*Apd*).

Symbol: 616 *N K BZ*,
funkcja: $am' = am/2^N$,
treść: zawartość rejestru *AM* zostaje podzielona *N* razy przez dwa.

2.6. Rozkazy wejścia i wyjścia. Informacje programowe zapisane w postaci *rządków* dziurek na 5-kanałowej taśmie papierowej, przekazywane są za pośrednictwem czytnika do akumulatora *A*. Jeśli pojawi się odpowiedni rozkaz (*We*1), to pojedynczy rządek taśmy wprowadza się do pięciu prawych pozycji akumulatora. Wprowadzanie większej ilości rządków taśmy i kompletowanie ich w słowa maszynowe odbywa się za pomocą specjalnych programów *wprowadzających*.

Czytnik może wprowadzać znaki z szybkością 300 zn/sek.

Pięciobitowe znaki (cyfry lub litery) można także wprowadzać do maszyny z klawiatury dalekopisu (rozkazem *We*2). Dalekopis ten pracuje w międzynarodowym kodzie Nr 2.

W maszynie przewidziana jest również możliwość podłączenia dodatkowego wejścia, zawierającego:

— *selektor*, który wybiera punkt pomiarowy;
— *konwerter* analogowo-cyfrowy, który zamienia formę sygnału pomiarowego, określonego przez selektor, z analogowej na cyfrową;
— *zegar*, który podaje czas realny;
— *pulpit* dodatkowy, na którym nastawia się dziesiętne instrukcje zewnętrzne; na pulpicie tym mieści się także sygnalizacja dziesiętna pomiaru danego punktu.

W maszynie są zrealizowane rozkazy czytania pomiaru, instrukcji zewnętrznej i czasu realnego oraz rozkaz wybierania punktu. Rozkazy te mają wspólny kod 226 (*WeK*).

Zawartość pięciu lewych bitów akumulatora (jeden znak) może być wyprowadzona w postaci rządka dziurek na pięciokanałowej taśmie papierowej przez perforator (rozkaz *Wy*1) albo też wydziurkowana na dalekopisie (rozkaz *Wy*2). Perforator może pracować z szybkością 150 rządków na sekundę, zaś dalekopis z szybkością do 10 znaków na sekundę.

Rozkazy wejścia i wyjścia mają kod postaci *T*26 (patrz część II tabeli rozkazów). W tej grupie znajdują się także rozkazy:

— czytania klawiatury do akumulatora (rozkaz *CzK*),
— zaokrąglenia wyniku zawartego w akumulatorze (rozkaz *Okr*),
— zatrzymania pracy maszyny (rozkaz *Stop*).

2.7. Skoki warunkowe. Opisując poszczególne rozkazy w poprzednich paragrafach, podawaliśmy tylko *podstawową* funkcję rozkazu. Na przykład: $a' = a + n$.

Oprócz podstawowej funkcji rozkazu, w maszynie wykonuje się pewne pomocnicze funkcje, które pozwalają określić znak (lub wartość względem zera) wyniku operacji. Te dodatkowe informacje są zawsze zapisywane w specjalnych jednobitowych rejestrach: *U*, *Z*, *D*, *Nd*. Do rejestru *U* wpisuje się 1, gdy wynik wykonanej operacji jest ujemny, w przeciwnym zaś przypadku wpisuje się 0. Do rejestru *Z* wpisuje się 1, gdy wynik jest równy zeru. Do rejestru *D* wpisuje się 1, gdy wynik jest dodatni. Do rejestru *Nd* wpisuje się 1, gdy wynik wykonanej operacji nie należy do przedziału $[-1, 1-2^{-38}]$ dla liczb stałoprzecinkowych lub do przedziału $[-2^{63}, 2^{63}-2^{32}]$ dla liczb zmiennoprzecinkowych (grupa działań zmiennoprzecinkowych będzie omówiona w paragrafie 2.8).

W maszynie istnieją specjalne rozkazy (skoki), przy pomocy których można się dowiedzieć, jaki jest stan rejestrów *U*, *Z*, *D*, *Nd*. W zależności od stanu określonego rejestru można wybrać jedną z dwóch możliwych

dróg programu wskazanych przez pierwszy adres (N) i przez drugi adres (K) rozkazu skokowego. Rozkazy skoków warunkowych mają kod ogólny postaci $T46$.

2.7.1. Skok przy zerze (*SkZ*).

Symbol: 046 N K BZ,

treść: jeżeli zawartość rejestru Z jest 1, to przejdź do wykonania rozkazu z komórki o adresie N, w przeciwnym przypadku przejdź do wykonania rozkazu z komórki o adresie K.

2.7.2. Skok przy ujemnej wartości (*SkU*).

Symbol: 146 N K BZ,

treść: jeżeli zawartość rejestru U jest 1, to przejdź do wykonania rozkazu z komórki o adresie N, w przeciwnym przypadku z komórki K.

2.7.3. Skok przy dodatniej wartości (*SkD*).

Symbol: 246 N K BZ,

treść: jeżeli zawartość rejestru D jest 1, to przejdź do wykonania rozkazu z komórki o adresie N, w przeciwnym przypadku z komórki K.

2.7.4. Skok przy nadmiarze (*SkNd*).

Symbol: 346 N K BZ,

treść: jeżeli zawartość rejestru Nd jest 1, to przejdź do wykonania rozkazu z komórki o adresie N, w przeciwnym przypadku z komórki K.

2.7.5. Koniec cyklu z minusem ($KC-$).

Symbol: 546 N K BZ,

treść: jeżeli zawartość rejestru B jest różna od zera, to przejdź do wykonania rozkazu z komórki N, w przeciwnym przypadku z komórki K. W obu przypadkach odejmij 1 od zawartości tegoż rejestru B.

2.7.6. Koniec cyklu z plusem ($KC+$).

Symbol: 646 N K BZ,

treść: jeżeli zawartość rejestru B jest różna od zera, to przejdź do wykonania rozkazu z komórki N, w przeciwnym przypadku z komórki K. W obu przypadkach dodaj 1 do zawartości tegoż rejestru B.

2.7.7. Skok ze śladem (*SkS*).

Symbol: 746 N K BZ,

funkcja: $n' = 000\ N\ K\ B$ oraz $N+2$ do R,

treść: prześlij do N-tej komórki pamięci słowo rozkazowe 000 N K B, po czym przejdź do wykonania rozkazu z komórki o adresie $N+2$.

Rozkaz *SkS* służy do organizacji wywołania podprogramu rozpoczynającego się od komórki $N+2$ i automatycznego powrotu do głównego programu (do rozkazu z komórki K). *Ślad*, czyli informacja powrotu, jest zostawiony w komórce o adresie N. Dlatego komórka N nie może być zajęta przez dany podprogram. Natomiast komórka $N+1$ może być zajęta przez podprogram.

2.8. Rozkazy zmiennoprzecinkowe. Arytmetyczne operacje zmiennoprzecinkowe są bardzo ważne dla programowania, gdyż w praktyce obliczeniowej używa się częściej liczb zmiennoprzecinkowych. Ważny jest również fakt, że zakres liczb zmiennoprzecinkowych jest bardzo duży $[-2^{63}, 2^{63}-2^{32}]$. Arytmetyczne operacje zmiennoprzecinkowe wykonywane są na argumencie a i argumencie y (traktowane jako liczby zmiennoprzecinkowe). Wynik tych operacji zostawia się zawsze w akumulatorze; przy czym przez akumulator rozumie się tutaj dwa rejestry: Am i Ac. W pierwszym z tych rejestrów pamięta się mantysę argumentu a (po operacji mantysę wyniku), w drugim cechę argumentu a (po operacji cechę wyniku).

Wynik operacji zmiennoprzecinkowych może być *znormalizowany* lub nie, a także *zaokrąglony* logicznie lub nie. Gdy wynik operacji zmiennoprzecinkowej ma cechę większą od $+63$, maszyna zostaje automatycznie zatrzymana. Jeśli natomiast cecha wyniku operacji jest mniejsza od -64, to przyjmuje się jako wynik zero zmiennoprzecinkowe. Przez zero zmiennoprzecinkowe rozumie się taką liczbę x, która ma mantysę 0 a cechę -64; czyli parę 0; -64. Podobnie jak i przy poprzednich rozkazach, wartość T określa typ operacji zmiennoprzecinkowej a cyfra P określa podstawienie, jakie ma być dokonane w operacji, poza tym precyzuje wynik operacji: czy ma być ewentualnie znormalizowany, czy zaokrąglony. Cyfra G jest stale równa 7 dla operacji zmiennoprzecinkowych, a funkcję cyfry P podaje poniższa tabelka:

P	funkcja
0	$y = n$ i BON
1	$y = n$ i BO
2	$y = n$ i BN
3	$y = n$
4	$y = b$ i BON
5	$y = b$ i BO
6	$y = b$ i BN
7	$y = b$

Jeśli jest więc $P = 0$, to za argument y podstawia się zawartość komórki o adresie N. Wynik zaś operacji zmiennoprzecinkowej nie jest znormalizowany ani też zaokrąglony, czyli jak się przyjęło mówić — jest *zablokowana* normalizacja i zaokrąglenie (BON). Podobnie skróty BO i BN oznaczają odpowiednio blokadę zaokrąglenia i blokadę normalizacji.

Gdy $P = 3$ lub $P = 7$, wówczas wynik operacji jest zawsze znormalizowany i zaokrąglony.

Funkcję cyfry T przedstawia poniższa tabelka:

T	operacja zmiennoprzecinkowa	
0	—	rezerwa
1	$a' \underset{z}{=} y$	pobieranie zmiennoprzecinkowe
2	$y' \underset{z}{=} a$	przesyłanie zmiennoprzecinkowe
3	$a' \underset{z}{=} a+y$	dodawanie zmiennoprzecinkowe
4	$a' \underset{z}{=} a-y$	odejmowanie zmiennoprzecinkowe proste
5	$a' \underset{z}{=} y-a$	odejmowanie zmiennoprzecinkowe odwrotne
6	$a' \underset{z}{=} a \cdot y$	mnożenie zmiennoprzecinkowe
7	$a' \underset{z}{=} a/y$	dzielenie zmiennoprzecinkowe

2.8.1. Pobieranie zmiennoprzecinkowe.

Symbol: 137 N K BZ,

funkcja: $a' \underset{z}{=} n$,

treść: wyzeruj akumulator mantysy i akumulator cechy, po czym wpisz do Am mantysę, do Ac cechę liczby z komórki o adresie N. Nieznormalizowaną mantysę znormalizuj, a od Ac odejmij odpowiednią poprawkę(²). Mantysę wyniku zaokrąglij.

Uwaga: Bity 0-31 słowa liczbowego (mantysa) są zapisywane w Am (w pozycjach $A0$-$A31$), zaś bity 32-38 (cecha) są zapisywane w Ac. W pozycjach $A32$-$A38$ akumulatora mantysy zapisywane są zera.

2.8.2. Przesyłanie zmiennoprzecinkowe.

Symbol: 237 N K BZ,

funkcja: $n' \underset{z}{=} a$,

treść: prześlij zawartość akumulatora mantysy i akumulatora cechy do komórki o adresie N. Akumulator bez zmian.

Uwaga: Zawartość pierwszych 32 pozycji Am zapamiętana jest w pozycjach 0-31 komórki, zaś zawartość Ac w pozycjach 32-38 tej komórki.

2.8.3. Dodawanie zmiennoprzecinkowe.

Symbol: 337 N K BZ,

funkcja: $a' \underset{z}{=} a+n$,

(²) Przyjęto i opisano przykład pobierania zmiennoprzecinkowego, w którym $P = 3$. Czytelnik sam łatwo dopowie treść tego rozkazu dla innych wartości P. Również następne rozkazy zmiennoprzecinkowe zostaną opisane dla przypadku $P = 3$.

treść: do liczby zmiennoprzecinkowej zapisanej w akumulatorze mantysy i w akumulatorze cechy dodaj liczbę zmiennoprzecinkową zapisaną w komórce o adresie N. Wynik dodawania zmiennoprzecinkowego umieść w akumulatorze. Gdy wynik jest nieznormalizowany, to znormalizuj go, a od Ac odejmij odpowiednią poprawkę. Mantysę wyniku zaokrąglij.

2.8.4. Odejmowania zmiennoprzecinkowe.

a) Odejmowanie proste

Symbol: 437 N K BZ,

funkcja: $a' \underset{z}{=} a - n$,

treść: od liczby zmiennoprzecinkowej w akumulatorze odejmij liczbę zmiennoprzecinkową zapisaną w komórce o adresie N. Wynik umieść w akumulatorze.

b) Odejmowanie odwrotne

Symbol: 537 N K BZ,

funkcja: $a' \underset{z}{=} n - a$,

treść: od liczby zmiennoprzecinkowej zapisanej w komórce o adresie N odejmij liczbę zmiennoprzecinkową zapisaną w akumulatorze. Wynik umieść w akumulatorze.
Normalizacja i zaokrąglenie odbywają się podobnie jak przy dodawaniu zmiennoprzecinkowym.

2.8.5. Mnożenie zmiennoprzecinkowe.

Symbol: 637 N K BZ,

funkcja: $a' \underset{z}{=} a \cdot n$,

treść: pomnóż liczbę zmiennoprzecinkową w akumulatorze przez liczbę zmiennoprzecinkową zapisaną w komórce o adresie N. Iloczyn mantys 77-bitowy wpisz do rejestru AM, a sumę cech do Ac. Znormalizuj iloczyn zapisany w akumulatorze mantysy, a od Ac odejmij odpowiednią poprawkę. Wynik w Am po normalizacji zostaje zaokrąglony.

2.8.6. Dzielenie zmiennoprzecinkowe.

Symbol: 737 N K BZ,

funkcja: $a' \underset{z}{=} a/n$,

treść: podziel liczbę zmiennoprzecinkową, zapisaną w akumulatorze, przez liczbę zmiennoprzecinkową zapisaną w komórce o adresie N. Iloraz mantys umieść w Am, różnicę cech w Ac. Nieznormalizowany iloraz znormalizuj, a od Ac odejmij odpowiednią poprawkę. Wynik w Am po normalizacji zostaje także zaokrąglony. Mantysa dzielnika zostaje zachowana w rejestrze M.

Uwaga. Gdy mantysa dzielnika jest równa zeru, to maszyna zatrzyma się.

Obecnie podamy tabelę rozkazów grupy 6 i 7:

T								P		G
0	1	2	3	4	5	6	7			
Lw	*Pr*	*Al*	*Ap*	*Cp*	*Ald*	*Apd*	*DzD*	1		6
We1	*We2*	*WeK*	*CzK*	*Okr*	*Wy1*	*Wy2*	*Stop*	2		
SkZ	*SkU*	*SkD*	*SkNd*	—	*KC−*	*KC+*	*SkS*	4		
—	$a' \underset{z}{=} y$	$y' \underset{z}{=} a$	$a' \underset{z}{=} a+y$	$a' \underset{z}{=} a-y$	$a' \underset{z}{=} y-a$	$a \underset{z}{=} a \cdot y$	$a' \underset{z}{=} a/y$	*BON* 0	$y=n$	7
								BO 1		
								BN 2		
								3		
								BON 4	$y=b$	
								BO 5		
								BN 6		
								7		

Tabela rozkazów, część II

3. Ogólne dane techniczne

3.1. Organizacja wewnętrzna.
3.1.1. Rodzaj pracy: szeregowy, niesekwencyjny.
3.1.2. System liczenia: dwójkowy, kod uzupełnieniowy.
3.1.3. Długość słowa: 39 bitów plus 1 bit techniczny.
3.1.4. Adresy: jeden plus jeden.
3.1.5. Przecinek: stały i zmienny.
3.1.6. Ilość rozkazów: około 460 (ujęte w tabelę funkcjonalną).
3.1.7. Ilość rejestrów modyfikacji: 7
3.1.8. Szybkość operacji:

stały przecinek

czas dodawania — 0,68 msek,
czas mnożenia — 4,12 msek,
czas dzielenia — 7,74 msek,

zmienny przecinek

czas dodawania — 1,20 msek,
czas mnożenia — 3,78 msek,
czas dzielenia — 8,08 msek,
rozkazy sterujące — 0,68 msek bądź 0,34 msek.

3.2. Pamięć: bęben magnetyczny o pojemności 8192 słów 40-bitowych, średni czas dostępu 11 msek([3]).
3.3. Wejście: — czytnik taśmy (300 rządków/sek) produkcji ELWRO,
— dalekopis (10 znaków/sek) firmy Lorenz,
— przystawka zawierająca konwerter analogowo-cyfrowy (produkcji ELWRO),
3.4. Wyjście: — perforator taśmy (150 rządków/sek) firmy Facit,
— dalekopis.
3.5. Technika: tranzystorowo-dynamiczna.
3.6. Częstotliwość podstawowa: 250 kHz.
3.7. Zasilanie: 3×220/380 V, 50 Hz.
3.8. Pobór mocy: 700 VA.
3.9. Gabaryty: 1640×670×1235.
3.10. Ciężar: 400 kg.

4. Biblioteka programów

Maszyna cyfrowa ODRA 1003 posiada trzy opracowane i udokumentowane zewnętrzne języki programowania.

4.1. Podstawowy Język Zewnętrzny. W Podstawowym Języku maszyny cyfrowej ODRA 1003 [2] można pisać: rozkazy (z adresami bezwzględnymi lub względnymi), pseudorozkazy, liczby całkowite i zmiennoprzecinkowe, teksty.

Podstawowy Język pozwala na napisanie sprawnych i dobrze zoptymalizowanych programów ściśle związanych z językiem wewnętrznym maszyny. Pozwala on również w łatwy sposób włączać podprogramy do programów głównych. W języku tym napisane są programy biblioteczne oraz programy translatorów innych języków. Programy wprowadzająco-wyprowadzające tego języka zajmują stosunkowo niewiele miejsca w pamięci maszyny (komórki o adresach od 0 do 639) i dlatego pozwalają na wprowadzanie do maszyny dużych sekwencji rozkazów i danych. Podstawowy program wprowadzania jest tak zorganizowany, że pozwala na łatwe (programowe) jego modyfikacje. Można wprowadzać nowe symbole do języka przypisując im nową interpertację (na przykład przy opracowywaniu różnych programów interpretacyjnych).

4.2. Język Adresów Symbolicznych. W Języku Adresów Symbolicznych można pisać te same informacje, co w Języku Podstawowym, a ponadto można pisać adresy symboliczne (do siedmiu liter) i etykiety.

([3]) Istnieje także wersja maszyny wyposażonej dodatkowo w pamięć ferrytową o pojemności 256 słów 40-bitowych z czasem cyklu zapisu i odczytu 8 mikrosekund.

Język ten daje duże ułatwienia przy opracowywaniu oddzielnych bloków programów (np. przez różne osoby). Redukuje on poza tym błędy w programach oraz uwalnia programistę od konieczności pamiętania numerycznych adresów (argumentów operacji i następnych rozkazów do wykonania).

4.3. Autokod MOST 1. Polecenia zapisywane w Języku Podstawowym lub w Języku Adresów Symbolicznych są przetłumaczone na słowa maszyny w stosunku jeden do jednego. Natomiast w Języku MOST 1 [3] można pisać ogólniejsze polecenia dla maszyny, które zostają przetłumaczone na słowa maszyny, przez translator, w stosunku jeden do kilku (lub kilkunastu).

Zatem autokod MOST 1 jest systemem automatycznego programowania, w którym czas i trud potrzebny do przygotowania programów dla ODRY 1003 skraca się wielokrotnie w stosunku do poprzednich systemów programowania. Program napisany w autokodzie MOST 1 jest bardzo czytelny i zwięzły. Osoby z wykształceniem matematycznym, technicznym lub ekonomicznym mogą praktycznie opanować programowanie w autokodzie MOST 1 w ciągu kilkunastogodzinnego kursu.

4.4. Programy biblioteczne. Dla często spotykanych obliczeń maszyna posiada opracowane programy, których taśmy i opisy przekazuje się użytkownikom wraz z maszyną.

Biblioteka programów jest sukcesywnie rozszerzana; do stycznia 1965 opracowano i udokumentowano około 110 programów. Między innymi maszyna posiada programy dla następujących zagadnień:

Programy organizacyjne.

- Programy wprowadzająco-wyprowadzające dla Języka Podstawowego.
- Programy wprowadzająco-wyprowadzające dla Języka Adresów Symbolicznych.
- Translator MOST 1 i podprogramy pomocnicze.
- Programy wprowadzające i tworzące taśmy binarne.
- Programy wyprowadzania programów i danych (*post mortem*).
- Programy kopiowania.

Programy obliczeniowe.

- Obliczanie pierwiastka kwadratowego i sześciennego, funkcji wykładniczej e^x, logarytmu naturalnego, funkcji trygonometrycznych i odwrotnych do nich, funkcji hiperbolicznych.
- Rozwiązywanie układów równań liniowych (metodą Gaussa z wyborem maksymalnego elementu).
- Rozwiązywanie symetrycznych układów równań liniowych (metodą Gaussa).
- Rozwiązywanie układów równań liniowych dobrze uwarunkowanych.

— Przybliżone rozwiązywanie układu równań liniowych (metodą najmniejszych kwadratów).
— Obliczenie całki metodą Gaussa i metodą Simpsona.
— Rozwiązywanie równania różniczkowego pierwszego i drugiego rzędu (metodą Rungego-Kutty).
— Rozwiązywanie układu równań różniczkowych (metodą Adamsa i metodą Rungego-Kutty).
— Aproksymacja funkcji wielomianami (metodą najmniejszych kwadratów).
— Wyznaczanie pierwiastka równania przestępnego (metodą Newtona).
— Wyznaczanie wielomianu interpolacyjnego (metodą Lagrange'a i metodą Newtona, podwójna i pojedyncza dokładność).
— Obliczanie pierwiastków wielomianu.
— Obliczanie pierwiastka kwadratowego z dowolną dokładnością.
— Różniczkowanie numeryczne funkcji.
— Obliczanie wartości funkcji gamma Eulera i funkcji Bessla.
— Obliczanie wyznacznika, mnożenie macierzy, odwracanie i transponowanie macierzy, dodawanie i odejmowanie macierzy.
— Obliczanie wartości własnych macierzy.
— Działania na liczbach zespolonych wraz z programem interpretacyjnym, mnożenie macierzy o elementach zespolonych.
— Podprogramy wprowadzająco-wyprowadzające i działań elementarnych dla liczb podwójnie długich dwudziestocyfrowych).
— Analiza drogi krytycznej (metoda PERT, na 850 i 2300 czynności).
— Metoda simplex i problem transportowy programowania liniowego.
— Podstawowe działania na krakowianach oraz rozwiązywanie układów równań liniowych metodą krakowianów.
— Obliczenia geodezyjne (24 programy).
— Obliczenia ze statystyki.
— Obliczenia ze statyki.
— Obliczenia układów optycznych.

Prace cytowane

[1] *Opis ogólny maszyny cyfrowej ODRA 1003*, Wrocławskie Zakłady Elektroniczne ELWRO, Wrocław 1964.

[2] *Instrukcja programowania maszyny cyfrowej ODRA 1003*, Wrocławskie Zakłady Elektroniczne ELWRO, Wrocław 1964.

[3] *Programowanie w autokodzie MOST 1 dla maszyny cyfrowej ODRA 1003*, Wrocławskie Zakłady Elektroniczne ELWRO, Wrocław 1965.

WROCŁAWSKIE ZAKŁADY ELEKTRONICZNE ELWRO

Praca wpłynęła 7. 1. 1965

Т. КАМБУРЭЛИС (Вроцлав)

ВЫЧИСЛИТЕЛЬНАЯ ЦИФРОВАЯ МАШИНА ODRA 1003

РЕЗЮМЕ

В статье дается обзор организации вычислительной цифровой машины ODRA 1003, разработанной и продуцированной вроцлавским заводом Wrocławskie Zakłady Elektroniczne ELWRO.

Приведена общая функциональная схема машины, структура комманд (один плюс один адрес) и структура чисел с постоянной и плавущей запятой. Детально представлен перечень комманд машины. Сводкой общих технических параметров машины и краткой информацией о разработанной библиотеке стандартных программ статья заканчивается.

T. KAMBURELIS (Wrocław)

DIGITAL COMPUTER ODRA 1003

SUMMARY

The paper presents the organizational structure of the digital computer ODRA 1003, designed and produced by Wrocławskie Zakłady Elektroniczne ELWRO. A general functional computer scheme and the built-up of the commands (one plus one address) and of fixed and floating point numbers are given. There is also a detailed list of the commands of the computer. Some technical data and brief information on the library of programmes end the paper.
