Witold Komorowski

MOMIK 8B - PIERWSZY POLSKI MINIKOMPUTER

1. Minikomputer – co to jest?

Słowo „minikomputer" używane jest obecnie na określenie komputerów jednopłytkowych, czyli mieszczących się na jednej płytce z obwodem drukowanym (*single-board computers*), przeznaczonych do celów edukacyjnych, dla hobbystów lub do stosowania w mikrosterownikach wbudowanych. Np. bestselerowa seria produktów brytyjskiej fundacji Raspberry Pi była zapoczątkowana w 2012 r. prostym dydaktycznym modelem Pi 1 przeznaczonym dla uczniów w krajach rozwijających się. Sukces tej konstrukcji doprowadził do tego, że w marcu 2018 r. łączna sprzedaż wszystkich, ciągle rozbudowywanych modeli osiągnęła liczbę 19 milionów sztuk. Model Raspberry Pi Zero z 2015 r. był sprzedawany w cenie 5 USD, najnowszy model Raspberry Pi Zero 3B+ kosztuje 35 USD, ale ma już 4-rdzeniowy procesor taktowany zegarem 1,4 GHz, 3 GB pamięci RAM, zasilanie przez MicroUSB, a waży tylko 45 g [6]. Inny współczesny minikomputer, Intel Compute Stick (4 GB pamięci RAM, 64 GB pamięci masowej, 64-bitowy Windows 10) w postaci „batonika" o długości 11,5 cm, również zasilany przez USB, można podłączyć do gniazda HDMI dowolnego monitora uzyskując wysokowydajny komputer [2]. Przedrostek „mini" odnosi się w tych urządzeniach nie tyle do funkcji, ile rozmiarów przyrównywanych w reklamach do karty kredytowej czy paczki gumy do żucia.

Na początku lat 60. XX wieku, kiedy pojawiła się po raz pierwszy nazwa minikomputer (*minicomputer*), również chodziło o szokujące wówczas zmniejszenie gabarytów. Miały to być – jak pisano – maszyny o rozmiarach porównywalnych z lodówką i cenie poniżej 25 tys. dolarów przy zachowaniu funkcji „normalnych", czyli dużych komputerów (*mainframe*) kosztujących setki tysięcy dolarów i wymagających klimatyzowanych pomieszczeń i specjalnego zasilania. Pierwszym minikomputerem, który odniósł wielki sukces rynkowy był 12-bitowy PDP-8 firmy DEC (*Digital Equipment Corp.*) zastąpiony w latach 70. przez 16-bitowy PDP-11. W latach 1970 -1990 powstało kilkanaście modeli tej doskonałej maszyny, a jej klony produkowane w ZSSR pod nazwą SM – *Система миникомпютеров* pojawiły się też w Polsce [4]. Minikomputery, dzięki umiarkowanej cenie i braku wymagań instalacyjnych spowodowały upowszechnienie techniki cyfrowej przez rozszerzenie grona użytkowników na

małe przedsiębiorstwa, biura i uczelnie. Wpływ minikomputerów nie ograniczał się tylko do zastosowań, dotyczył również sfery projektowania samych komputerów. Miedzy innymi miały one strukturę szynową umożliwiającą zunifikowaną komunikację między różnymi funkcjonalnie modułami i taka koncepcja stała się niebawem standardową w rodzących się systemach mikroprocesorowych (pierwszy mikroprocesor, 4-bitowy Intel 4004 pojawił się w 1971 r., a 16-bitowy Intel 8086, który stał się jądrem IBM-owskiego „peceta" – w 1978 r.). Nb. twórcą architektury PDP-11 był Gordon C. Bell, późniejszy autor notacji ISP zastosowanej w tym artykule do formalnego opisu procesora.

Pierwszy polski minikomputer, nazwany Momik 8b, zaprojektowano w 1973 r. w Instytucie Maszyn Matematycznych w Warszawie i stał się on „jednostką centralną" komputerów serii MERA 300 produkowanych seryjnie od 1974 roku przez Zakład Systemów Minikomputerowych MERA [7].

2. Architektura komputera Momik 8b

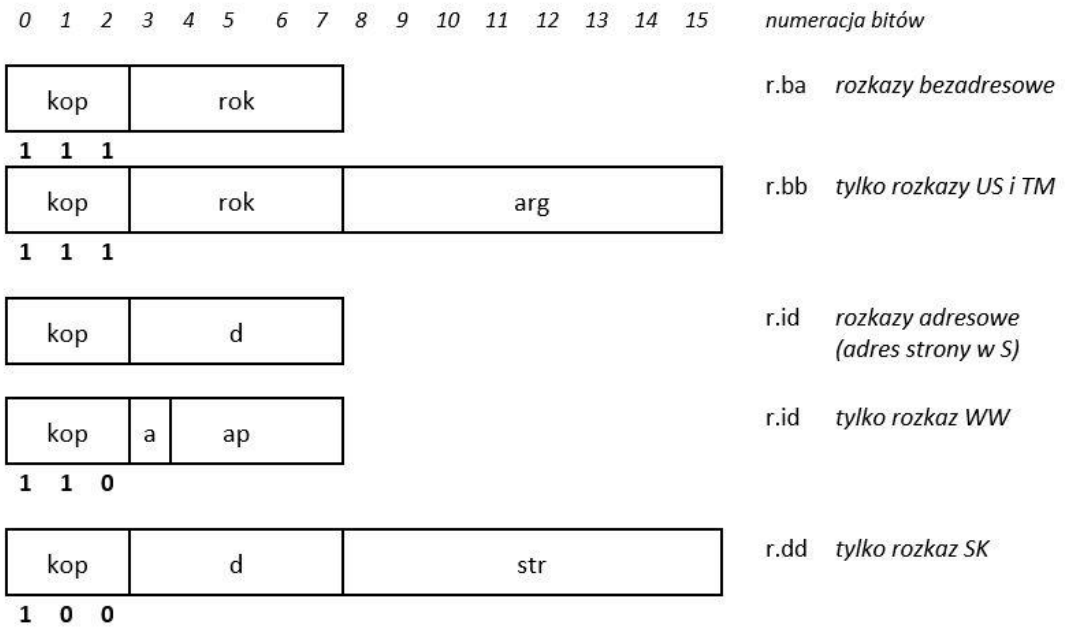
Momik 8b był komputerem 8-bitowym, zbudowanym na układach scalonych TTL (*Transistor-Transistor Logic*) i wyposażonym pierwotnie w 8-kilobajtową pamięć o czasie cyklu 1,8 μs zrealizowaną na rdzeniach ferrytowych. System przerwań przewidywał możliwość 128 zgłoszeń zgrupowanych w 4 klasy. Komunikacja z urządzeniami zewnętrznymi odbywała się przez magistralę we/wy z dwoma 8-bitowymi szynami danych [5]. Całość była zawarta w standardowym module 19" z płytą czołową służącą jako pulpit operatora.

Niektóre rozwiązania architektury tego minikomputera wydają się obecnie nieco dziwne, choć wówczas były stosowane w wielu innych maszynach. Np. ze względu na brak stosu obsługa przerwań wymagała zdefiniowania miejsca w pamięci służącego jako przechowalnia stanu – podobnie jak to miało miejsce w dominujących wówczas na rynku komputerach IBM System/360. Brak stosu komplikował też organizację podprogramów, szczególnie ich zagnieżdżania; standardowy obecnie schemat *CALL – RET* musiał być zastąpiony sekwencją rozkazów „zapamiętaj stan" (PS) i „skok warunkowy" (SK), a powrót – rozkazem „odtwórz stan" (PO). Skok warunkowy istniał tu tylko w jednej wersji za to było kilka rozkazów operujących na znacznikach powodujących efektywność skoku; np. sekwencja rozkazów: „zerowanie wskaźnika CI" (ZC) i rozkazu adresowego „skok warunkowy" (SK) była równoważna skokowi bezwarunkowemu. Osobliwością był rozkaz „wykonanie rozkazu z pamięci" (WP), działający jak jednorozkazowy podprogram (analogiczne rozwiązanie było też w Systemie/360 – rozkaz *Execute*). Znamiennym reliktem wczesnych realizacji

komputerów było istnienie klawiatury operatora, której stan można wpisywać do akumulatora rozkazem KL.

Krótkie, 8-bitowe słowo maszynowe, a z drugiej strony 13-bitowa długość adresu utrudniały kodowanie rozkazów i były przyczyną ograniczenia do sześciu (!) liczby rozkazów adresowych (w tym tylko jeden rozkaz arytmetyczny: „dodawanie" (DS) i jeden logiczny: "iloczyn logiczny" (ML)). Wyliczenie pełnego adresu efektywnego uzyskano przez złożenie 8-bitowego adresu strony przechowywanego w specjalnym rejestrze S i 5-bitowego przesunięcia (*displacement*) zapisanego w słowie rozkazowym. Tak więc, pamięć operacyjną można było traktować jako podzieloną na 256 stron po 32 bajty na stronie. Strona zerowa była uprzywilejowana, gdyż dostęp do niej można było wymusić ustawiając wskaźnik Z.

Ponieważ Momik był maszyną jednoakumulatorową, bez żadnych dodatkowych rejestrów procesora, problem adresacji pośredniej rozwiązano przez zarezerwowanie ośmiu komórek pamięci jako miejsc służących równocześnie do autoindeksacji argumentów – każde użycie adresu z takiego miejsca powodowało zwiększenie adresu o 1. Ściśle rzecz biorąc – w tych miejscach (na stronie zerowej) przechowywany był tylko 5-bitowy numer przesunięcia; numer strony pobierano z rejestru stron (S), który w przypadku wyzerowaniu przesunięcia również był inkrementowany.



Formaty rozkazów procesora Momik 8b.

Momik 8b – kodowanie rozkazów adresowych

kop rop (dwójkowo)	Oryginalny mnemonik	kop rop (dziesiętnie)	Funkcja	Typ
000 d	DS	**0** d	dodawanie w AK	id
001 d	PZ	**1** d	zapamiętanie AK	id
010 d	ML	**2** d	iloczyn logiczny w AK	id
011 d	DP	**3** d	zwiększenie bajta w pamięci o 1	id
100 d	SK	**4** d	skok warunkowy	dd
101 d	WP	**5** d	wykonanie rozkazu z pamięci	id
110 a^ap	WW	**6** a^ap	wejście/wyjście do/z AK	id

Momik 8b – kodowanie rozkazów bezadresowych

kop rop (dwójkowo)	Oryginalny mnemonik	kop rop (dziesiętnie)	Funkcja	Typ
111 00000	PR	**7** **0**	przerwanie programowe	ba
111 00001	ZA	**7** **1**	zerowanie AK	ba
111 00010	ZC	**7** **2**	zerowanie CI	ba
111 00011	NN	**7** **3**	brak działania	ba
111 00100	PW	**7** **4**	powrót z przerwania	ba
111 00101	PS	**7** **5**	zapamiętanie stanu	ba
111 00110	PO	**7** **6**	odtworzenie stanu	ba
111 01000	ZZ	**7** **8**	zerowanie Z	ba
111 01001	LZ	**7** **9**	ustawienie Z	ba
111 01010	AS	**7** **10**	wpisanie numeru strony	ba
111 01011	SA	**7** **11**	pobranie numeru strony do AK	ba
111 01100	KA	**7** **12**	pobranie z klawiatury do AK	ba
111 01101	PP	**7** **13**	badanie przeniesienia	ba
111 01110	SD	**7** **14**	stan Czekaj	ba
111 01111	SS	**7** **15**	zwiększenie numeru strony o 1	ba
111 10000	NA	**7** **16**	negacja AK	ba
111 10001	DA	**7** **17**	zwiększenie AK o 1	ba
111 10010	DN	**7** **18**	warunkowe zwiększenie AK o 1	ba
111 10011	LC	**7** **19**	rotacja AK w lewo	ba
111 10100	AL	**7** **20**	przesunięcie AK w lewo	ba
111 10101	AP	**7** **21**	przesunięcie AK w prawo	ba
111 10110	AZ	**7** **22**	badanie zera w AK	ba
111 10111	AD	**7** **23**	badanie znaku AK	ba
111 11000	US	**7** **24**	ustawienie numeru strony	bd
111 11001	UW	**7** **25**	pobranie maski przerwań	ba
111 11010	TM	**7** **26**	różnica symetryczna w AK	bd
111 11100	SP	**7** **28**	stop	ba

Lista rozkazów zawierała 34 pozycje: 27 rozkazów bezadresowych, 6 rozkazów adresowych i jeden rozkaz we/wy; 5 kodów operacyjnych pozostawało niewykorzystanych. Słowa rozkazowe były jedno- i dwubajtowe w 4 podstawowych formatach BA, BB, ID i DD.

Dwubajtowy format BB miały tylko dwa rozkazy, a format DD – tylko jeden rozkaz (skoku warunkowego). W formacie ID jeden rozkaz (wejścia/wyjścia) miał odmienną od innych rozkazów tej grupy interpretację bitów <3:7>, mianowicie bit <3> wskazywał kierunek transmisji. Oznaczenia stosowane w dalszych tabelach i opisie formalnym są pokazane na rysunku i nawiązują do oryginalnych, używanych przez autorów projektu [5]. Zachowano również oryginalne mnemoniki rozkazów i podano, stosowane tam, dziesiętne wartości kodów.

3. Język formalny opisu architektury

Poniżej opisano szczegółowo architekturę procesora Momik 8b stosując formalizm będący modyfikacją notacji ISP (*Instruction Set Processor*) wprowadzonej w książce G.C.Bell'a i A.Newell'a "Computer Structures" [1] i używanej w różnych wersjach publikacyjnych. Formalizm ten nadaje się szczególnie do opisu architektury na poziomie listy rozkazów (ISA - *Instruction-Set Architecture*) i był z powodzeniem zastosowany do opisu kilku znanych i bardzo różniących się między sobą architektur w pracy [4], gdzie przedstawiono również dokładniej jego składnię i semantykę. Podany tu formalny opis architektury mikrokomputera Momik 8b oparto na z tekście [3] z 1978 r. przygotowanym na podstawie informacji dostępnych wówczas w wydawnictwie „Momik 8b. Zasady działania. Cz.I" (wyd. IMM, Warszawa 1972).

W notacji ISP można podać model procesora lub innego bloku funkcjonalnego opisujący zachowanie się tego bloku (jego architekturę) na poziomie elementów i sygnałów dwustanowych. W takim opisie każdy blok funkcjonalny zawiera dwie części: deklaracje zmiennych (operatorem definicji jest tu znak :=) oraz zapis możliwych zmian stanu. Zwykle dodaje się komentarz, choć w zasadzie nie jest on potrzebny dla zrozumienia działania (zachowania się) układu w każdej sytuacji określonej przez dowolny stan jego elementów zdefiniowany z dokładnością do pojedynczego bitu..

Zmiany stanu polegają na nadaniu nowych wartości zmiennym, co formalnie opisuje się jako przesłanie między elementami pamięciowymi (operator ←). Działania te są zawsze zapisywane jako warunkowe wg formuły $\Delta \Rightarrow \Omega$, gdzie Δ jest zmienną (lub funkcją) binarną (przyjmującą wartości 0 lub 1) albo relacją (przyjmującą wartości "fałsz" lub "prawda") zaś Ω oznacza sekwencję akcji realizowaną gdy $\Delta = 1$ (albo Δ = "prawda"). Poszczególne akcje są oddzielane średnikiem (;), a jeżeli istotna jest ich kolejność, wówczas separatorem jest operator **next**. W przypadku gdy trzeba podać postać binarną zmiennej (np. reprezentującej fizyczny rejestr), numery bitów zapisywane są w nawiasach kątowych; np. LR<0:12> oznacza 13-bitową

zmienną LR, w której bity są numerowane od lewej strony; LR<5:12> oznacza 8-bitową mniej znaczącą część zmiennej LR. Pamięć można traktować jako tablicę bajtów, których adresy zapisuje się w nawiasach prostokątnych; np. 8192-elementowa pamięć nazwana M ma zapis M[0:8191]<0:7>. W Momiku wygodnie jest traktować pamięć operacyjną jako 256-elementową tablicę tablic 32-elementowych M[0:255][0:31]<0:7>; zapis M[LR] oznacza wówczas bajt w pamięci określony przez adres zapisany w LR; a M[0][d] oznacza bajt o numerze d na stronie zerowej.

Tworzenie łańcuchów binarnych ze zdefiniowanych wcześniej części zapisuje się z użyciem operatora konkatenacji (^). Np. P^C^Z^LR<8:12> jest opisem 8-bitowego łańcucha 1-bitowych zmiennych P, C i Z oraz 5-bitowego fragmentu zmiennej LR i jeżeli taka struktura ma być nazwana PSW, to należy ją zdefiniować jako PSW<0:7> := P^C^Z^LR<8:12> .

Rozkazy są zdefiniowane przez ich kody operacyjne, ale dla wygody zapisano je z użyciem mnemoników, np. rozkaz KL („pobranie z klawiatury do AK") ma w grupie rozkazów bezadresowych (dla których kop = 7) rozszerzenie kodu rop = $01100_2 = 12_{10}$, co zapisuje się KL(:=12).

Poniżej wypisano alfabetycznie oryginalne mnemoniki i nazwy rozkazów procesora Momik 8b wg oznaczeń przyjętych przez autorów projektu w [5], w nawiasach podany jest typ słowa rozkazowego. W tym wykazie zwraca uwagę, dziś raczej nie stosowany, tryb rozkazujący użyty w nazwach:

AD - Badaj znak w akumulatorze (BA)
AL - Przesuń akumulator w lewo (BA)
AP - Przesuń akumulator w prawo (BA)
AS - Prześlij akumulator do rejestru strony (BA)
AZ - Badaj zero w akumulatorze (BA)
DA - Dodaj jeden do akumulatora (BA)
DN - Dodaj przeniesienie do akumulatora (BA)
DP - Dodaj jeden do pamięci (ID)
DS - Dodaj do akumulatora (ID)
KA - Czytaj klucze (BA)
LC - Przesuń akumulator w lewo cyklicznie (BA)
LZ - Zapal wskaźnik strony zerowej (BA)
ML - Mnóż logicznie akumulator (ID)
NA - Neguj akumulator (BA)
NN - Nic nie rob (BA)
PO - Wróć według śladu (BA)
PP - Badaj przeniesienie (BA)
PR - Przerwanie programowe (BA)
PS - Pamiętaj ślad (BA)

PW - Wróć z przerwania (BA)
PZ - Pamiętaj akumulator (ID)
SA - Prześlij zaw. rej. strony do akumulatora (BA)
SD - Czekaj (BA)
SK - Skocz warunkowo (DD)
SP - Stop (BA)
SS - Zwiększ zawartość rej. strony o jeden (BA)
TH - Testuj z maską (BB)
US - Ustaw rejestr strony (BB)
UW - Ustaw rejestr maski (BA)
WP - Wykonaj rozkaz (ID)
WW - Rozkazy wejścia/wyjścia (ID)
ZA - Zeruj akumulator (BA)
ZC - Zeruj wskaźnik warunku (BA)
ZZ - Zagaś wskaźnik strony zerowej (BA)

4. Ginące ślady przeszłości

W 45 lat od powstania pierwszego polskiego minikomputera (1973 – 2018) można by pomyśleć o stworzeniu emulatora tej tak egzotycznej – z dzisiejszego punktu widzenia – architektury. Polskie osiągnięcia w tej dziedzinie (np. „wrocławskie" maszyny serii Odra) nie mają szczęścia do dobrego i nowoczesnego udokumentowania (w wielu przypadkach są w ogóle obawy o istnienie dokumentacji „papierowej"), skompletowania egzemplarzy muzealnych, nie mówiąc o zasłużonym uznaniu ich twórców. Wyjątkiem zdaje się być duży projekt emulatora systemu MERA-400 i sprzętowej (!) reimplementacji jednostki centralnej prowadzony przez grono entuzjastów pod skrzydłami Polskiego Towarzystwa Informatycznego [8]. We wszystkich tego rodzaju działaniach odtworzeniowych znajomość wszystkich szczegółów architektury jest konieczna, co okazuje się dowodnie już przy próbie formalizacji jej opisu – takiej jak przedstawiona poniżej.

5. Formalny opis architektury Momik 8b.

W poniższym zapisie architektury procesora Momik 8b formalny opis (w notacji ISP) zredagowano czcionką Courier New, natomiast komentarze – w zasadzie opcjonalne – zawierające objaśnienia lub/i interpretację formuł zapisano obok – kursywą. Charakter komentarza mają również tytuły poszczególnych fragmentów opisu.

Pamięć

M[0:8191]<0:7>/M[0:255][0:31]<0:7>

pamięć operacyjna 8 kB (256 stron 32-bajtowych)
15 bajtów zarezerwowanych na stronie 0:
M[0] ... M[3] – przechowalnia „starego stanu" przy przerwaniu
M[4] – maska przerwań (bity <4:7>)
M[5] ... M[6] – przechowalnia śladu
M[16] ... M[23] – 8 bajtów do adresacji pośredniej z autoindeksacją
10 bajtów zarezerwowanych na stronie 1:
M[32] ... M[41] – 5 adresów programów obsługi („wektorów")

Rejestry

AK<0:7>	*rejestr akumulatora*
LR<0:12>	*licznik rozkazów*
W<0:3>	*rejestr maski przerwań*

Wskaźniki

P	*lewe wydłużenie AK (wskaźnik przeniesien*a)
Z	*wskaźnik strony zerowej (M[0] ... M[31])*
CI	*warunek skoku*
ND	*wskaźnik nadmiaru*
PRACA	*wskaźnik stanu Praca/Stop*
CZEKAJ	*wskaźnik stanu Czekaj/Przetwarzanie*

Pulpit operatora

KL<0:7>	*rejestr 8 klawiszy stabilnych (na płycie czołowej)*

Magistrala we/wy

BA<0:3>	*linie adresowe*
A	*linia kierunku transmisji*
BY<0:7>	*linie wyjściowe*
BE<0:7>	*linie wejściowe*

Adres efektywny

```
e<0:12> :=  (                                   adres efektywny
         Z and (15<d<24) => (                   warunek autoindeksacji
          (M[0][d]<3:7>=0) => S ← S+1; next
                                                kolejna strona, jeżeli przepełnienie
          r.id => S^M[0][d]<3:7>;               adresacja pośrednia dla r.id
          r.dd => str^M[0][d]<3:7>; next
                                                adresacja pośrednia dla r.dd
          M[0][d] ← M[0][d] + 1                 zwiększenie adresu pośredniego
                              )
         (not Z) or (not (15<d<24)) => (
                                                warunek adresacji bezpośredniej
               r.id => S^d;                     względem rejestru strony
               r.dd => str^d )                  adres bezwzględny w rozkazie
              )
```

Wykonanie rozkazów

```
wykonanie := (

DS (:= kop=0) => P^AK ← AK + M[e]; next (ND=1)=> CI ← 0;
PZ (:= kop=1) => M[e] ← AK; next AK ← 0;
ML (:= kop=2) => AK and M[e];
DP (:= kop=3) => M[e] ← M[e] + 1; next (M[e]=0)=> CI ← 1;
                 (M[e]><0)=> CI ← 0;
SK (:= kop=4) => CI => LR ← LR +1;
                 not CI => LR ← e;

WP (:= kop=5) => r<0:7> ← M[e]; next e:= S^M[e]<3:7>;
                 next wykonanie
WW (:= kop=6) => BA ← r<4:7>; next
                 a => A ← 1; BY ← AK;
                 not a => A ← 0; AK ← BE; next CI ← 0;

(kop=7) => rozkazy bezadresowe
              )
```

Przerwania

```
przerwanie                    wystąpienie sygnału w niezamaskowanej klasie
klasa                         numer klasy przerwania (1 ...4)
kodp                          numer przerwania w klasie (0 ... 31)

przerwanie and PRACA => (
     W ← 0; next                                  zamaskowanie zgłoszeń
     M[0]<0:2> ← P^CI^Z;                          zapamiętanie wskaźników
     M[1]<0:7>^M[0]<3:7> ← LR<0:12>;              zapamiętanie LR
     M[2] ← AK; M[3] ← S;                         zapamiętanie AK i S
     AK ← kodp; CI ← 0; Z ← 0;                    wpisanie nr przerwania do AK
     LR<0:12> ← M[32+2*klasa]<0:7>^M[32+2*klasa]<3:7>;
                                                  skok do programu obsługi
     CZEKAJ => CZEKAJ ← 0)                        powrót do stanu "Przetwarzanie"
```

Formaty rozkazów

```
rozkaz/r<0:15>                słowo rozkazowe (8- lub 16-bitowe)
     kop<0:2> := r<0:2>            kod operacyjny
     rok<0:4> := r<3:7>            rozszerzenie kodu op.
     d<0:4> := r<3:7>              adres na stronie (przesunięcie)
     str<0:7> := r<8:15>           numer strony
     arg<0:7> := r<8:15>           argument bezpośredni
     a := r<3>                     kierunek transmisji we/wy
     ap<0:3> := r<4:7>             adres rejestru we/wy („portu")


r.id := (kop><4) and (kop><7)      rozkazy adresowe
r.dd := (kop=4)                    rozkaz z adresem bezpośrednim (SK)
r.bb := (kop=7) and ((rok=24) or (rok=26))
                                   rozkazy z argumentem bezpośrednim (US i TM)
r.ba := (kop=7 ) and (not r.bb)    rozkazy bezadresowe
```

```
rozkazy bezadresowe := (

PR (:= rok=0) => W ← 0;
          M[1]<0:7>^M[0]<3:7> ← LR <0:12>;
          M[0}<0:2> ← P^CI^Z
          M[2] ← AK; M[3] ← S; next
          CI ← 0; Z ← 0; LR<0:12> ← M[33]<0:7>^M[32]<3:7>
ZA (:= rok=1) => AK ← 0;
ZC (:= rok=2) => CI ← 0;
NN (:= rok=3) => ;
PW (:= rok=4) => LR<0:12> ← M[1]<0:7>^M[0]<3:7>;
            P^CI^Z ← M[0}<0:2>;
PS (:= rok=5) => M[6]<0:7>^M[5]<3:7> ← LR<0:12>;
           M[5]<0:2> ← P^CI^Z;
PO (:= rok=6) => LR<0:12> ← M[6]<0:7>^M[5]<3:7>;
            P^CI^Z ← M[5}<0:2>; next LR ← LR + 1
ZZ (:= rok=8) => Z ← 0;
LZ (:= rok=9) => Z ← 1;
AS (:= rok=10) => S ← AK;
SA (:= rok=11) => AK ← S;
KA (:= rok=12) => AK ← KL;
PP (:= rok=13) => CI ← not P;
SD (:= rok=14) => CZEKAJ ← 1;
SS (:= rok=15) => S ← S + 1;
NA (:= rok=16) => AK ← not AK;
DA (:= rok=17) => P^AK ← AK + 1; next ND => CI ← 0;
DN (:= rok=18) => P^AK ← AK + P; next ND => CI ← 0;
LC (:= rok=19) => AK ← rol AK;
AL (:= rok=20) => P^AK ← shl AK;
AP (:= rok=21) => AK ← shr (P^AK); next P ← 0;
AZ (:= rok=22) => ((AK=0) => CI ← 0; (AK≠0) => CI ← 1);
AD (:= rok=23) => CI ← AK<0>;
US (:= rok=24) => S ← M[LR]; next LR ← LR + 1;
UW (:= rok=25) => W ← M[4]<4:7>;
TM (:= rok=26) => AK ← (AK xor M[LR]);
              next ((AK=0) => CI ← 0; (AK≠0) => CI ← 1);
              next LR ← LR + 1;
SP (:= rok=28) => PRACA ← 0
               )
```

LITERATURA

1. BELL C.G., NEWELL A. Computer Structures: Readings and Examples. McGraw Hill, 1971.
2. Intel Compute Stick. https://www.intel.com/content/www/us/en/products/boards-kits/compute-stick/stk1a32sc.html (*dostęp 27.10.2018*)
3. KOMOROWSKI W. Procesor Momik 8b – Opis architektury. [w: ZETO we Wrocławiu. Informacje i komunikaty. Nr 1 (39), 1978.]
4. KOMOROWSKI W. Instrumenta computatoria. Helion, Gliwice 2000.
5. POPKO J., ROMANIUK W. Minikomputer MOMIK 8b. [w: IMM Warszawa. ETO Nowości. Nr 2., 1974.] https://historiainformatyki.pl/historia/momik-8b (*dostęp 1.05.2017*)
6. Raspberry Pi. https://en.wikipedia.org/wiki/Raspberry_Pi#Pi_Zero (*dostęp 27.10.2018*)
7. Seria komputerów MERA 300. http://www.wikiwand.com/pl/Mera_300 (*dostęp 1.05.2017*)
8. System MERA400. https://mera400.pl/ (*dostęp 30.10.2018*)

6-11-2018