Zaktualizowany i nieco zmodyfikowany tekst zamieszczony pierwotnie
p.t. „Procesor Momik 8b – Opis architektury”
w biuletynie ZETO Wrocław. „Informacje i komunikaty” Nr 1 (39), 1978.

Witold Komorowski

Momik 8b - pierwszy polski minikomputer

W 1973 r. w Instytucie Maszyn Matematycznych w Warszawie zaprojektowano pierwszy polski minikomputer nazwany Momik 8b, który stał się „jednostką centralną” komputerów serii MERA 300 produkowanych seryjnie od 1974 roku przez Zakład Systemów Minikomputerowych MERA [4].

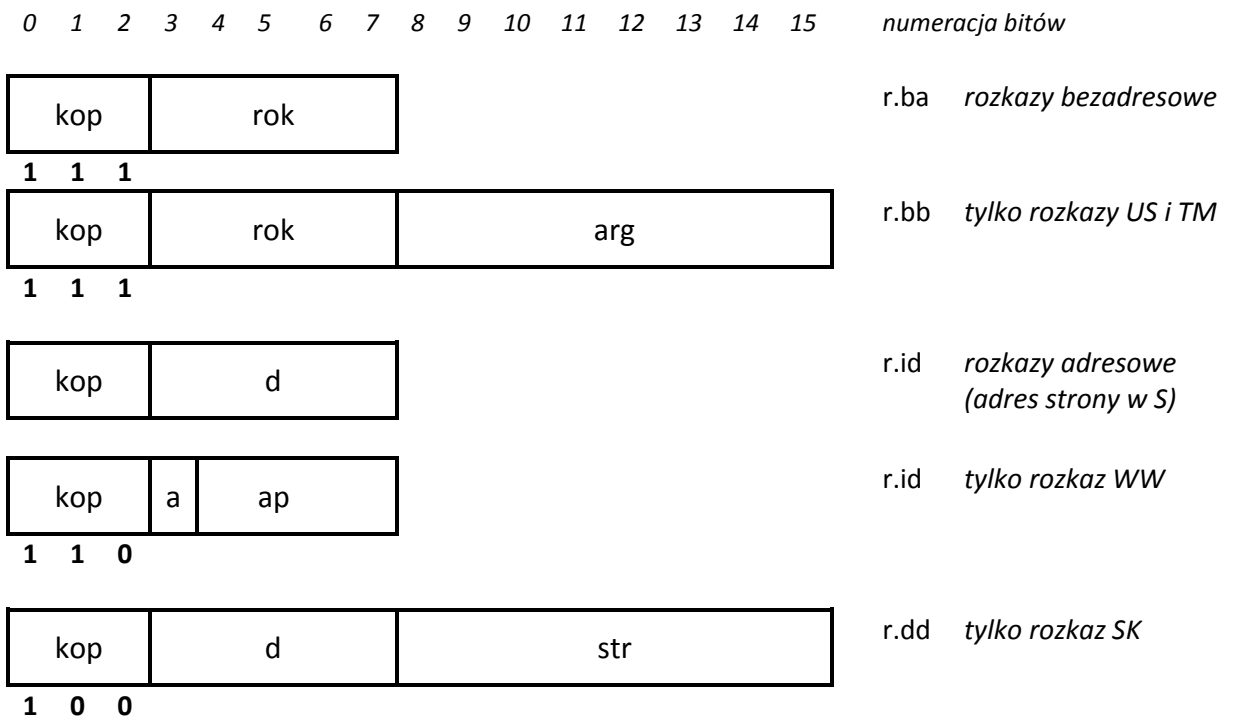
Momik 8b był komputerem 8-bitowym, zbudowanym na układach scalonych TTL (*Transistor-Transistor Logic*) i wyposażonym pierwotnie w 8-kilobajtową pamięć o czasie cyklu 1,8 µs zrealizowaną na rdzeniach ferrytowych. System przerwań przewidywał możliwość 128 zgłoszeń zgrupowanych w 4 klasy. Komunikacja z urządzeniami zewnętrznymi odbywała się przez magistralę we/wy z dwoma 8-bitowymi szynami danych [3]. Całość była zawarta w standardowym module 19” z płytą czołową służącą jako pulpit operatora.

Niektóre rozwiązania architektury tego minikomputera wydają się obecnie nieco egzotyczne, 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 6(!) liczby rozkazów adresowych (w tym 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 Momik 8.

Lista rozkazów zawierała 34 pozycje – 27 rozkazów bezadresowych, 6 rozkazów adresowych i jeden rozkaz we/wy; 5 kodów operacyjnych pozostawało niewykorzystanych. Oznaczenia pól i formatów pokazane na rysunku nawiązują do oryginalnych, używanych przez autorów projektu [3]; oznaczenia te są stosowane w dalszych tabelach i opisie formalnym. Zachowano również oryginalne mnemoniki rozkazów.

Momik 8b – rozkazy adresowe

kop	rop	Oryginalny mnemonik	kop	rop	Funkcja	Typ
(dwójkowo)			(dziesiętnie)			
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 – rozkazy bezadresowe

kop	rop	Oryginalny mnemonik	kop	rop	Funkcja	Typ
(dwójkowo)			(dziesiętnie)			
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**	nic nie rób	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

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" (1971) i często używanej w różnych wersjach publikacyjnych. Formalizm ten nadaje się szczególnie do opisu architektury na poziomie listy rozkazów (ISA - *Instruction-Set Architecture*) i np. był z powodzeniem zastosowany do opisu kilku znanych i bardzo różniących się między sobą architektur w pracy [2], gdzie przedstawiono również dokładniej jego składnię i semantykę (rozd.9). Podany tu formalny opis architektury mikrokomputera Momik 8b był pierwotnie zamieszczony w [1] na podstawie informacji zawartych 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.

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 Δ => Ω, 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 Δ = 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 (kop=7) rozszerzenie kodu rop=01100, co zapisuje się KL(:=12).

Momik 8b - oryginalne nazwy I mnemoniki rozkazów (wg [3])

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)

Źródła:

1. W.Komorowski: Procesor Momik 8b – Opis architektury. [w: ZETO we Wrocławiu. Informacje i komunikaty. Nr 1 (39), 1978.]
2. W.Komorowski: Instrumenta computatoria. Helion, Gliwice 2000.
3. J.Popko, W.Romaniuk: Minikomputer MOMIK 8b. [w: IMM Warszawa. ETO Nowości. Nr 2., 1974.] https://historiainformatyki.pl/historia/momik-8b (*dostęp 1.05.2017*)
4. Seria komputerów MERA 300. http://www.wikiwand.com/pl/Mera_300 (*dostęp 1.05.2017*)

Formalny opis architektury Momik 8b.

Komentarze (po prawej stronie formuł ISP) zapisano kursywą.

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 przeniesieni*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*

Przerwania

przerwanie *wystąpienie sygnału przerwania w niezamaskowanej klasie*
klasa ***n****umer 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
```

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
           )

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
                        )
```