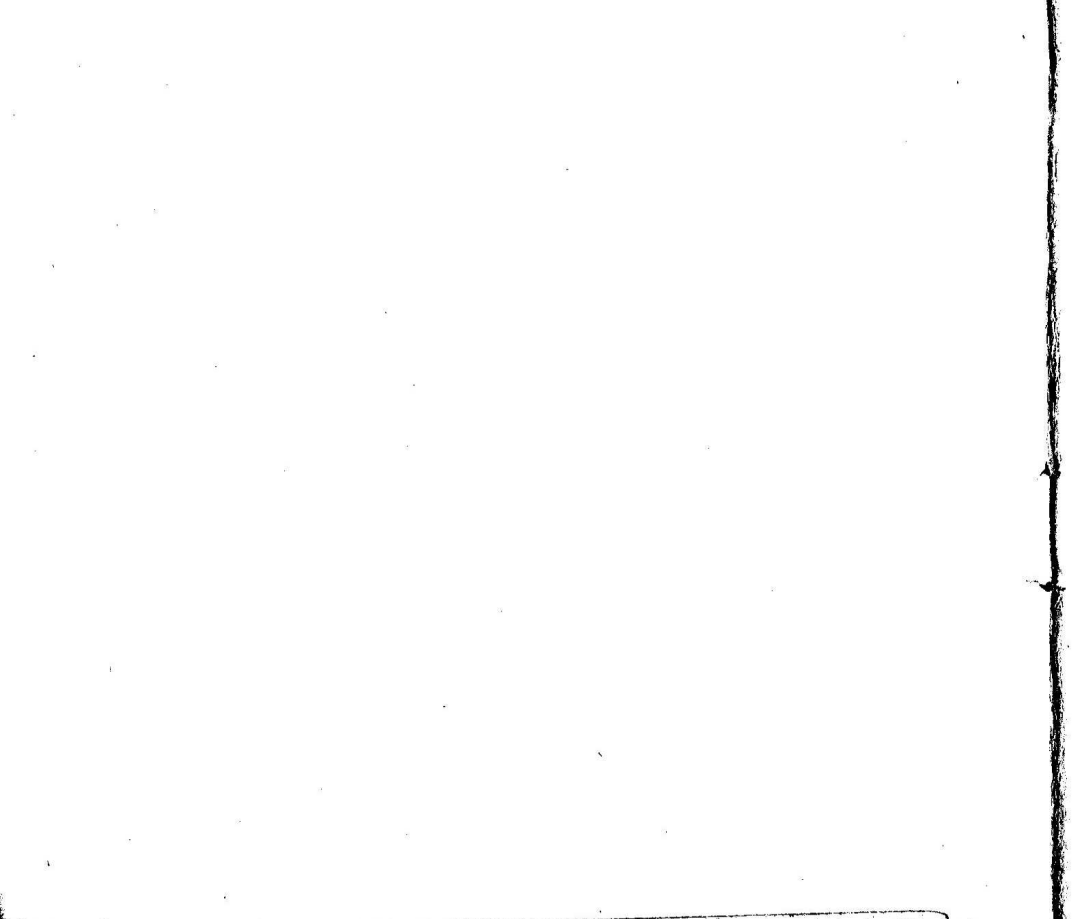
Systémové programování na počítačích

SHARP MZ-800

OK VTČM Česká Lípa
a KUOP SHARP Svazarm Nový Bor



S y s t é m o v é p r o g r a m o v á n í
na počítačích SHARP MZ-800

Autor: RNDr. Jiří Bárta, Vladimír Nevoral
Povoleno: ONV Česká Lípa, odbor kultury pod čj. 1025/88
Povolovací číslo: o 350109888
Vydává: OK VTČM Česká Lípa a KUOP SHARP - Svazarm N. Bor
Tisk: Severočeské dřevařské závody Česká Lípa
Náklad: 100 ks

SYSTÉMOVÉ PROGRAMOVÁNÍ

na počítačích SHARP MZ-800.

RNDr. Jiří Bárta
Vladimír Nevoral

Klub uživatelů osobních počítačů SHARP - SVAZARM Nový Bor

Okresní klub vědeckotechnické činnosti mládeže Česká Lípa

Dodatek D:

Adresy periferních obvodů

adresa	směr přenosu	registr	strana
CCH	O	WRITE format registr (WF)	5-3
CDH	O	READ format registr (RF)	5-2
CEH	I/O	registr módu zobrazení (DMD)	3-1
1CFH	O	registr rolování SOF (dolní část)	5-7
2CFH	O	registr rolování SOF (horní část)	5-7
3CFH	O	registr rolování SW	5-7
4CFH	O	registr rolování SSA	5-7
5CFH	O	registr rolování SEA	5-7
6CFH	O	registr barvy okraje obrazovky (border)	5-8
D0H (E000H)	O	Obvod 8255 port A	6-1
D1H (E001H)	I	Obvod 8255 port B	6-1
D2H (E002H)	I/O	Obvod 8255 port C	6-1
D3H (E003H)	O	Obvod 8255 řídicí registr	6-1
D4H (E004H)	I/O	Obvod 8253 - čítač 0	6-7
D5H (E005H)	I/O	Obvod 8253 - čítač 1	6-7
D6H (E006H)	I/O	Obvod 8253 - čítač 2	6-7
D7H (E007H)	I/O	Stavový a řídicí registr obvodu 8253	6-7
E0H	I/O	bankování paměti	4-1, 4-3, 5-1
E1H	I/O	"	
E2H	O	"	
E3H	O	"	
E4H	O	"	
E5H	O	"	
E6H	O	"	
F0H	I	ovladač 1	6-3
F0H	O	registr palet	5-6
F1H	I	ovladač 2	6-3
F2H	O	programovatelný zvukový generátor	7-1
FCH	I/O	Obvod Z80 PIO - řídicí a stavový registr brány A	6-7
FDH	I/O	Obvod Z80 PIO - řídicí a stavový registr brány B	6-7
FEH	I/O	Obvod Z80 PIO - datový registr brány A	6-7
FFH	O	Obvod Z80 PIO - datový registr brány B	6-7

```
;
; STRING NA DISPLEJ
;
STM:DEFS 80         ;paměť pro 80 znaků stringu
;
STR:
     LD HL, STM
STRL:LD A, (HL)
     OR A           ;test konce stringu (0)
     RET Z
     PUSH HL
     CALL CHAR
     POP HL
     INC HL
     JR STRL
;
     END
```

OBSAH:

1.	Kurs programování ve strojovém jazyce	1-1
1.1	Programovací medium	1-1
1.2	Dvojková soustava	1-1
1.3	Hexadecimální soustava	1-2
1.3.1	Binárně - hexadecimální přepočet	1-2
1.4	Bit a bajt	1-3
1.5	Mikroprocesor Z80	1-3
1.5.1	Registry	1-3
1.5.1.1	Akumulátor - A	1-3
1.5.1.2	Další osmibitové registry	1-3
1.5.1.3	Příznaky	1-3
1.5.1.3.1	Příznak Carry	1-4
1.5.1.3.2	Příznak Subtraction	1-4
1.5.1.3.3	Příznak Halfcarry	1-4
1.5.1.3.4	Příznak Parity/Overflow	1-4
1.5.1.3.5	Příznak Zero	1-4
1.5.1.3.6	Příznak Sign	1-4
1.5.1.4	Registr R	1-4
1.5.1.5	Registr I	1-5
1.5.1.6	Registr PC	1-5
1.5.1.7	Registr SP	1-5
1.5.1.8	Registr HL	1-5
1.5.1.9	Registry BC a DE	1-5
1.5.1.10	Registry IX a IY	1-5
1.5.1.11	Záložní registry	1-5
1.5.2	Mnemonika a operační kód	1-5
1.5.3	Struktura paměti	1-6
1.5.4	Způsoby adresování	1-6
1.5.4.1	Výpočet indexového adresování	1-7
1.6	Binární logika	1-7
1.7	Aritmetické operace	1-8
1.7.1	ADD - součet	1-8
1.7.2	ADC - součet s přenosem	1-8
1.7.3	SUB - rozdíl	1-8
1.7.4	SBC - rozdíl s přenosem	1-8
1.7.5	INC - inkrement	1-8
1.7.6	DEC - dekrement	1-9
1.7.7	CP - porovnání	1-9
1.8	Zpracování jednotlivých bitů	1-9
1.8.1	SET - nastavení bitu	1-9
1.8.2	RES - nulování bitu	1-9
1.8.3	BIT - test bitu	1-9
1.9	Instrukce rotací	1-9
1.10	Instrukce pro práci s bloky dat	1-9
1.10.1	Instrukce pro prohledávání bloků dat	1-9
1.10.2	Instrukce pro přesuny datových bloků	1-10
1.11	Instrukce skoků	1-10
1.11.1	Absolutní skoky	1-10
1.11.2	Relativí skoky	1-10
1.12	Volání podprogramů a návrat z nich	1-11
1.13	Práce se zásobníkem	1-11
1.14	Instrukce DJNZ	1-12
1.15	Instrukce pro řízení vstupů a výstupů	1-12
2.	Příklady programů	2-1
2.1	Doporučení	2-1
2.2	Výpis na obrazovku	2-1
2.2.1	Výpis textu	2-1
2.2.2	Výpis znaku	2-1
2.3	Obsluha klávesnice	2-1
2.4	Dimenzování ve strojovém jazyce	2-1
2.5	Cyklus	2-2

2. 6	Dodatečné instrukce Z80	2-3
2. 7	Závěrem kapitoly 1 a 2	2-3
3.	Registr módu zobrazení (display mode registr - DMD)	3-1
3. 1	Několik slov o organizaci VideoRAM v různých režimech	3-1
3. 1. 1	Rozdělení VideoRAM bez rozšíření	3-1
3. 1. 2	Rozdělení VideoRAM s rozšířením	3-1
3. 2	Volba zobrazovacího režimu	3-2
4.	Režim MZ-700	4-1
4. 1	Stavba znakové paměti	4-1
4. 2	Stavba paměti barev	4-2
5.	Režim MZ-800	5-1
5. 1	Operace Single Write (jednoduchý zápis)	5-3
5. 2	Operace Exclusive - Or (EXOR)	5-3
5. 3	Operace OR	5-4
5. 4	Operace RESET	5-4
5. 5	Operace REPLACE	5-4
5. 6	Operace PSET	5-5
5. 7	Další poznámky k používání WF a RF registrů	5-5
5. 8	Hardwareové rolování	5-7
5. 9	Nastavení barvy okraje (border)	5-8
6.	I/O operace v režimech MZ-700 a MZ-800	6-1
6. 1	Obvod 8255	6-1
6. 1. 1	Řízení ovladačů	6-3
6. 1. 2	Ukládání dat na magnetofon	6-4
6. 2	Obvod 8253	6-7
6. 3	Obvod Z80-PIO	6-7
7.	Programovatelný zvukový generátor	7-1
7. 1	Nastavení frekvence tónu	7-1
7. 2	Řízení generátoru šumu	7-2
7. 3	Nastavení útlumu tónu nebo šumu	7-2
8.	Seznam použité litertatury	8-1
Dodatek A	Tabulka kódu ASCII	A-1
Dodatek B	Tabulka kódu Z80	B-1
	Tabulka operací s příznaky	B-2
	Instrukční soubor Z80	B-3
Dodatek C	Příklad programu v assembleru (GRAFIK)	C-1
Dodatek D	Adresy periferních obvodů	D-1

SEZNAM TABULEK:

Tab. 3. 1	Význam 2. a 3. bitu registru DMD	3-2
Tab. 3. 2	Význam 0. a 1. bitu registru DMD	3-2
Tab. 4. 1	Bankování paměti v režimu MZ-700	4-1
Tab. 4. 2	Obsah znakové VideoRAM	4-2
Tab. 4. 3	Bankování znakových generátorů	4-3
Tab. 5. 1	Bankování paměti v režimu MZ-800 výstupem na port	5-1
Tab. 5. 2	Bankování paměti v režimu MZ-800 vstupem z portu	5-1
Tab. 5. 3	Operace mezi ukládanými a uloženými daty při zápisu do VideoRAM	5-3
Tab. 6. 1	Adresy obvodu 8255	6-1
Tab. 6. 2	Jednotlivé bity obvodu 8255	6-1
Tab. 6. 3	Zapojení konektorů ovladačů	6-3
Tab. 6. 4	Adresy obvodu 8253	6-7
Tab. 7. 1	Adresování registrů PSG	7-2


```
      LD DE,200
      XOR A
      SBC HL,DE      ;test Y<200?
      RET M
NEDSP:POP DE         ;oprava návratové adresy
      EI
      RET

; ZNAK NA DISPLEJ
;  tento podprogram předpokládá, že na adrese 0 až FFFh je
;  ROM MONITOR, protože využívá jeho tabulku videokódu
      DEFS 17H
;
CURSOR:DEFW 0        ;pozice kursoru
MCHAR:DEFB           ;znak pro zobrazení

CHARS:LD A,(MCHAR)
;
CHAR:LD HL,(CURSOR)
      INC HL         ;oprava pozice kurzoru
      LD (CURSOR),HL
      DEC HL
      DI
      PUSH HL
      LD HL,0A92H    ;tabulka videokódu (v ROM)
      LD E,A
      LD D,0
      ADD HL,DE      ;pozice znaku v tabulce
      LD A,(HL)      ;videokód znaku
      POP HL
; uložení videokódu do osmi 'mikrořádek' ve VRAM
      EXX
      LD H,0
      LD L,A
      ADD HL,HL
      ADD HL,HL
      ADD HL,HL
      SET 4,H        ;nastavení adresy c CG-ROM
      EXX
      LD B,8         ;počet řádek
      LD DE,0050H    ;aditivní konstanta
      IN A,(E0H)
ROW:
      EXX
      LD A,(HL)      ;vzor z CG-ROM
      INC HL
      EXX
      LD (HL),A      ;do VRAM
      ADD HL,DE
      DJNZ ROW
      IN A,(E1H)
      EI
      RET
```

```
      XOR A
      SBC HL,DE
      JR NZ,LOOP0    ;Y2>Y - opakuj
      RET

; X2=X - svislý vektor

DD4:  LD HL,(Y2)
      LD DE,(Y)
      XOR A
      SBC HL,DE
      JP P,DD5       ;Y2>Y
      JR Z,DD5
      LD A,80H
DD5:  LD (SGY),A
DD8:  CALL PLOT
      LD A,(SGY)
      RLA
      LD HL,(Y)
      JR C,DD6
      INC HL         ;Y:=Y+1
      JR DD7
DD6:  DEC HL         ;Y:=Y-1
DD7:  LD (Y),HL
      LD DE,(Y2)
      XOR A
      SBC HL,DE
      JR NZ,DD8
      RET
      DEFS 35H


;  test hranic VRAM pro PLOT
MEZE:
      XOR A
      LD (PRES),A
      LD HL,(X)
      LD BC,0000
      XOR A
      SBC HL,BC      ;test X>0?
      JP M,NEDSP
      LD DE,640
      XOR A
      SBC HL,DE      ;test X<640?
      JP M,MEZE1
      JR NEDSP
MEZE1:
      LD HL,(Y)
      XOR A
      SBC HL,BC      ;test Y>0?
      JP M,NEDSP
```

Úvod.

Cílem této publikace je seznámit uživatele osobních počítačů SHARP MZ-800 s některými vlastnostmi počítače, které nenajde v uživatelské příručce. Kniha nepodává úplný a vyčerpávající popis všech jeho obvodů. Nejvíce pozornosti je věnováno popisu zákaznického obvodu, který obstarává zobrazování na obrazovce a poskytuje některé systémové služby jako je volba režimů, bankování paměti a podobně.

Pro úplné začátečníky jsme zařadili kapitoly 1 a 2, ve kterých budou uvedeni do tajů strojového programování. Na závěr knihy jsou uvedeny dodatky A a B, které obsahují nejzákladnější pomůcky programátora - tabulku kódu ASCII a a tabulku kódu Z80. V dodatku C čtenář najde rozsáhlejší příklad programu v assembleru.

Věříme, že kniha bude dobrou pomůckou pro všechny uživatele, kteří si chtějí bez BASICovského kabátu vlastnoručně osahat obvody svého počítače.

1. Kurs programování ve strojovém jazyce.

Pro každého uživatele osobního počítače se dříve nebo později stane nutností alespoň částečná znalost strojového jazyka. Ale není to tak jednoduché - prostě jen zasednout a začít programovat.

Nejdříve se pokusme o srovnání s jazykem BASIC. V BASICu existuje obrovský komfort, který ulehčuje obsluhu. Např. operátory + - * / . Většina interpretů BASICu má spoustu dalších funkcí (SIN, COS, TAN, EXP,...). Oproti tomu ve strojovém jazyce už obyčejné násobení celých čísel dá značnou práci. Podíváme-li se však na problém z hlediska rychlosti výpočtu, žádný interpret BASICu před strojovým jazykem neobstojí. Tento faktor dokonce způsobuje, že některé problémy v BASICu nelze řešit vůbec. Podobně dopadne i srovnání velikosti paměťového prostoru, který zabírají programy napsané v BASICu a ve strojovém jazyce. Stejný problém lze ve strojovém jazyce vyřešit na jedné desetině paměťového prostoru oproti BASICu. Navíc je vždy v paměti nezbytná přítomnost vlastního interpretu BASICu, bez něhož žádný BASICový program nepoběží (pokud nepoužijeme kompilátor BASICu). A nyní přistupme k tajům strojového jazyka.

1.1 Programovací medium.

Tak jako nelze programovat v BASICu bez interpretu BASICu, nelze ani ve strojovém jazyce programovat bez překladače, pokud nechceme programovat přímo ve strojovém kódu z Monitoru. Tomuto překladači říkáme Assembler. Nejlepšími dostupnými prostředky na počítači SHARP MZ-800 jsou programy AREM, GENS3M2 a ASM V1/700. Je vhodné mít k dispozici další prostředky pro ladění jako např. MON V4.T nebo MS-700.

1.2 Dvojková soustava.

Pro toho, kdo zná strojový jazyk, je tato kapitola zbytečná. Avšak pro úplného začátečníka je klíčem k pochopení toho, jak vlastně procesor vykonává instrukce.

Jak sám název naznačuje, jedná se o soustavu, ve které existují pouze dvě číslice 0 a 1. Jim v elektronice odpovídají stavy vypnuto a zapnuto. Princip je podobný "našemu" desítkovému systému. Pro ty, kteří náš desítkový systém ještě dobře neznají jeden příklad.

Pozorujme číslo 15326. Máme rozdělit toto číslo na jednotky, desítky, stovky, tisíce a desetitisíce. Vznikne následující obrázek:

```
15326 =   1 desetitisíc  =  1*10^4
        + 5 tisíců       =  5*10^3
        + 3 sta          =  3*10^2
        + 2 desítky      =  2*10^1
        + 6 jednotek     =  6*10^0   (10^0=1)
```

Přesně tak je tomu i u dvojkové soustavy. Desítková soustava má za základ číslo deset a dvojková dvě. U desítkové soustavy je největší číslice 9, u dvojkové 1. Máme tedy opravdu jen dvě číslice - 0 a 1.

Jako příklad přepočítáme desítkové číslo 39 do dvojkové soustavy. K tomu je třeba znát násobky dvou. Tak jako násobky deseti jsou 1, 10, 100, 1000,..., jsou násobky dvou 1, 2, 4, 8, 16, 32, 64,..... . Číslo 39 se dá rozdělit na násobky dvou následovně:

```
39 = 32 + 4 + 2 + 1  nebo
39 = 2^5 + 2^2 + 2^1 + 2^0
```

Nejlépe je seřadit tato čísla do tabulky:

128	64	32	16	8	4	2	1	
0	0	1	0	0	1	1	1	= 39

Dekadické číslo 39 tedy vypadá ve dvojkovém (binárním) tvaru takto:

00100111.


```
      INC A
DD3:  LD (TG),A      ;příznak sklonu vektoru
      CALL PLOT      ;XD,YD zobrazit
;
LOOPD:LD HL,(D)
      LD DE,0000
      XOR A
      SBC HL,DE      ;test znaménka D
      JP M,DDV       ;D<0
      LD DE,(PY)
      XOR A
      SBC HL,DE      ;D-PY
      LD (D),HL
      LD A,(SGX)
      RLA            ;test SIGNUM(PX)
      LD HL,(X)
      JR C,SGNM      ;<0
      INC HL         ;>0 => XD:=XD+1
      JR STX
SGNM: DEC HL         ;XD:=XD-1
STX:  LD (X),HL
      LD A,(TG)
      OR A
      JR NZ,DTEST    ;opatření proti nadbytečným bodům
      CALL PLOT
      JR DTEST
;
DDV:  LD DE,(PX)
      ADD HL,DE      ;D<0 => D:=D+PX
      LD (D),HL
      LD A,(SGY)
      RLA            ;test SIGNUM(PY)
      LD HL,(Y)
      JR C,SGNMY
      INC HL         ;Y:=Y+1
      JR STY
SGNMY:DEC HL         ;Y:=Y-1
STY:  LD (Y),HL
      CALL PLOT
;
DTEST:LD HL,(X)
      LD DE,(X2)
      XOR A
      SBC HL,DE
      JR NZ,LOOPD    ;X2>X - opakuj
      LD HL,(Y)
      LD DE,(Y2)
```

```
; V naznačeném postupu se pro názornost předpokládá, že X2>X
; a Y2>Y

     DEFS 4
X2:  ENT
     DEFW 0
Y2:  ENT
     DEFW 0
;
PX:  DEFW 0
PY:  DEFW 0
D:   DEFW 0
SGX: DEFB 0
SGY: DEFB 0
TG:  DEFB 0

;
LINE:ENT
     LD HL,(X2)
     LD DE,(X)
     XOR A
     SBC HL,DE      ;X2-X
     JP Z,DD4       ;X2=X
     JP P,DD1
     EX DE,HL
     XOR A
     LD HL,0000
     SBC HL,DE      ;ABS(PX)
     LD A,80H
DD1: LD (PX),HL
     LD (SGX),A     ;znaménko PX
;
     LD HL,(Y2)
     LD DE,(Y)
     XOR A
     SBC HL,DE      ;Y2-Y
     JP P,DD2
     JR Z,DD2
     EX DE,HL
     XOR A
     LD HL,0000
     SBC HL,DE      ;ABS(PY)
     LD A,80H
DD2: LD (PY),HL
     LD (SGY),A     ;znaménko PY
;
     LD HL,0000
     LD (D),HL      ;D:=0
     LD HL,(PY)
     LD DE,(PX)
     XOR A
     SBC HL,DE      ;test sklonu vektoru
     JP M,DD3
```


Existuje ještě jedna metoda jak určit, jestli se má za příslušný násobek dvou dosadit 1 nebo 0. Předvedeme to na dekadickém čísle 111. Nejdříve najdeme násobek dvou, který je nejbližší tomuto číslu, ale je menší než toto číslo. Číslo 128 je větší než 111, ale 64 již naší podmínce vyhovuje. Zapíšeme tedy pod 64 jedničku a odečteme ho od daného čísla (111-64=47). Se získaným číslem 47 zacházíme opět stejně jako s původním číslem 111. V tomto případě bude 32 nejbližší násobek 2. Zapíšeme i 1 pod 32 a znovu odečteme (47-32=15). Nejbližší násobek 2 je pro 15 číslo 8, jedničku napíšeme pod 8 a po odečtení zbyde 7. Z hlavy již vidíme násobky 2, které jsou obsaženy v čísle 7 (7=4+2+1). Tímto postupem jsme získali číslo 01101111. Čím větší je převáděné dekadické číslo, tím větší násobek čísla 2 musíme použít na začátku převodu a tím větší počet číslic bude mít výsledné dvojkové číslo.

Máme-li dvojkové číslo a chceme-li je převést do dekadické soustavy, jednoduše sečteme násobky 2, u kterých byla dosazena 1.

01101111 = 0+64+32+0+8+4+2+1 = 111

1.3 Hexadecimální soustava.

Hexadecimální soustava se může začátečníkovi zdát ještě hrůznější než dvojková, ale je to základ pro učení strojového jazyka a velmi logická soustava pro programátora. Možná že se ptáte, proč nebyl jako základ pro počítače použit dvojkový nebo desítkový systém. Počítač sám pracuje ve dvojkové soustavě, ale pro programátora je tato soustava nevýhodná vzhledem k velkému počtu míst již u malých čísel. Desítkový systém je pro počítače těžko zpracovatelný a přepočítávání z desítkového do dvojkového systému je časově náročné. Oproti tomu přepočet z hexadecimálního (šestnáctkového) systému do dvojkového je rychlý, protože 16 je násobek 2. Zkusme si to ověřit. Jak již název napovídá, základem tohoto systému je číslo 16. Násobky 16-ti jsou 1, 16, 256, 4096 = 16^0, 16^1, 16^2, 16^3. Tyto násobky potřebujeme k adresování paměti až do adresy 65535.

Určitě se nabízí otázka, jak se označují číslice v této soustavě, když máme k dispozici pouze deset číslic a potřebujeme jich šestnáct. Na základě dohody bylo stanoveno, že se k doplnění použijí písmena A-F. Tedy A=10, B=11, C=12, D=13, E=14, F=15. Přepočítávání do této soustavy se dělá stejným způsobem jako do dvojkové. Zkusme přepočítat číslo 83. Napíšeme si známou tabulku, tentokrát se šestnáctkovými násobky. Postupujeme analogicky jako u binárního systému. Nejdříve najdeme nejbližší nižší násobek 16. Je to číslo 16. V 83 je tato hodnota obsažena pětkrát a proto pod 16 napíšeme 5. Zbytek má hodnotu 3 a zapíšeme ho přímo pod číslo 1.

4096	256	16	1	
0	0	5	3	= 83

Zkusíme převést větší číslo, např. 15311. Nejbližší násobek je 4096 a je obsažen třikrát. Po odečtení hodnoty 3*4096=12288 zbývá 3023. Nejbližší násobek má hodnotu 256 a je obsažen jedenáctkrát. Zbývá ještě 207, v němž je hodnota 16 obsažena dvanáctkrát. Nyní nám zbývá 15.

15311 = 3*4096 + 11*256 + 12*16 + 15

15311 = 3*16^3 + 11*16^2 + 12*16^1 + 15*16^0

Získané hodnoty musíme vyjádřit hexadecimálními číslicemi. Tedy 3, 11, 12, 15 se převede na 3, B, C, F. Hexadecimální číslo má tvar 3BCF.

Hexadecimální čísla se označují písmenem H, aby se rozlišila od dekadických. Podobně se dvojková čísla odlišují písmenem B.

Volný paměťový prostor počítače MZ-800 má rozsah 64 KB, což představuje 65536 různých adres. Nejvyšší adresa má dekadickou hodnotu 65535, což lze vyjádřit hexadecimálním číslem FFFFH.

1.3.1 Binárně - hexadecimální přepočet.

Ve strojovém programování je potřeba převádět binární hodnoty do hexadecimální soustavy a naopak. K tomu existuje několik pomůcek: všimněme si osmibitového čísla - např. 0111 1111. Rozdělíme-li ho na dvě poloviny, dostaneme 0111 a 1111. To jsou čísla 7 a 15, čemuž v hexadecimální soustavě odpovídá 7 a F. Číslo tedy bude 7FH. Obdobným způsobem postupujeme i při opačném převodu.

1.4 Bit a bajt.

S těmito základními pojmy se každý čtenář jistě setkal. Pro ty, kteří opravdu nevědí o co se jedná: bit je nejmenší jednotka informace, kterou počítač rozezná. Bajt se skládá z osmi bitů. Bity nabývají hodnot pouze 0 a 1, bajt může nabývat 2^8=256 různých hodnot. Obsah bajtu se většinou vyjadřuje právě dvojciferným hexadecimálním číslem.

1.5 Mikroprocesor Z80.

Mikroprocesor je srdcem a mozkem mikropočítače. Z80 je osmibitový procesor, který umí vykonávat oficiálně asi 400 instrukcí. Má však ještě mnoho dalších skrytých instrukcí. Mikroprocesor může pracovat s pamětí o rozsahu 64 KB.

1.5.1 Registry.

Registry jsou zvláštností procesoru. V podstatě jsou prostředníkem mezi aritmeticko - logickou jednotkou procesoru a pamětí. Většinu aritmetických, logických a jiných operací provádí procesor s obsahy těchto registrů a teprve výsledky těchto operací se ukládají do paměti.

Registry jsou jednobajtové a dvoubajtové. Dvoubajtové (16 - bitové) registry jsou výhodné zejména pro matematické operace. Z80 má oproti jiným procesorům velmi bohatý soubor registrů:

8 - bitové registry	16 - bitové registry
A (akumulátor)	PC (programový čítač)
B	BC (složen z reg. B a C)
C	
D	DE (složen z reg. D a E)
E	
H	HL (složen z reg. H a L)
L	
F (příznaky)	SP (ukazatel zásobníku)
HX	IX (složen z reg. HX a LX)
LX	
HY	IY (složen z reg. HY a LY)
LY	
Náhradní sada registrů	
A, B, C, D, E, F, H, L.	
R (registr obnovení paměti)	
I (registr vektoru přerušení)	

1.5.1.1 Akumulátor - A.

Akumulátor má zvláštní postavení mezi registry. Pouze pro akumulátor existují logické instrukce AND, OR, XOR, aritmetické instrukce sčítání a odečítání a instrukce rotací a některé další instrukce. Operace se provádějí mezi akumulátorem a ostatními registry nebo akumulátorem a pamětí, přičemž výsledek se vždy ukládá do akumulátoru.

1.5.1.2 Další osmibitové registry.

Jedná se o registry B, C, D, E, H, L. Tyto registry se zúčastňují operací s akumulátorem a operací přesunů dat.

1.5.1.3 Příznaky.

Jsou to jednobitové významové příznaky, které se sdružují do registru F. Obsah registru F je následující:


```
;  Výmaz bodu na obrazovce
;  Program nastaví příznak PRES a vzávislosti na jeho obsahu
;  se v předchozím programu modifikuje registr WF i způsob
;  zápisu dat do VRAM.


;  RESET

PRES:ENT
     DEFB 0
RESET:ENT
;
     DI
     CALL MEZE
     LD A,1
     LD (PRES),A
     JP PL
;
PPY: DEFW BE3OH
OLX: DEFW 0
PHL: DEFW 0
PPIX:DEFW
OLY: DEFW 0


;  Zobrazení vektoru podle zadaného počátečního bodu
;  X,Y (z programu PLOT) a koncového bodu X2,Y2.
;
;   Analýza vektoru:

;      PX:=X2-X;
;      PY:=Y2-Y;
;      D:=X*PY-Y*PX;
;      XD:=X;
;      YD:=Y;

;     repeat
;     plot(XD,YD);
;     if D>0 then
;      begin
;       D:=D-PY;
;       XD:=XD+1
;      end
;      else
;      begin
;       D:=D+PX;
;       YD:=YD+1
;      end;
;     until (XD=X2) or (YD=Y2);
;
```

```
      SRL D
      RR E
      SRL D
      RR E
      SRL D
      RR E          ;X div 8
      LD HL, (PPY)
      ADD HL, DE
      LD B, A
      INC B
      XOR A
      SCF
BITA: RL A
      DJNZ BITA     ;výpočet bitu
;
DISP:
      LD B, A
      IN A, (0E0H)  ;otevřít VRAM
      LD A, (PRES)
      OR A          ;set nebo reset?
      JR NZ, DISP1
      LD (HL), B    ;set bit
      JR DISP2
DISP1:LD A, B
      CPL
      LD B, (HL)
      AND B         ;výmaz příslušného bitu
      LD B, A
      LD A, 81H
      OUT (0CCH), A ;při reset je WF-REPLACE
      LD (HL), B
      LD A, 41H
      OUT (0CCH), A ;po operaci návrat OR
DISP2:
      IN A, (E1H)   ;zavřít VRAM
      EI
      RET
```


bit	7	6	5	4	3	2	1	0
	I S	I Z	I -	I H	I -	I P/O	I N	I C I

Označení jednotlivých příznaků:

S	Sign-Flag	- znaménkový příznak
Z	Zero-Flag	- příznak nulového výsledku operace
H	Halfcarry	- poloviční přenos
P/O	Parity/Overflow	- parita/přetečení
N	Subtraction-Flag	- příznak odečítání
C	Carry-Flag	- příznak přenosu

Příznaky slouží k vyhodnocování výsledků aritmetických a logických operací. Podle jejich stavu se dá posoudit výsledek operace a modifikovat další činnost programu.

1.5.1.3.1 Příznak Carry.

Tento příznak signalizuje přenos ze sedmého (při šestnáctibitové operaci z patnáctého) bitu. Carry si můžeme představit jako fiktivní devátý (sedmnáctý) bit. Pokud k přenosu dojde, bude Carry = 1, v ostatních případech = 0.

Příklad: V registru A je 83H v registru B je 90H. Součet A a B dává 113H, což se pochopitelně do A registru nevejde. Výsledek bude takový, že v registru A bude po operaci 13H a Carry bude rovno jedné.

1.5.1.3.2 Příznak Subtraction.

Tento příznak se vyhodnocuje pouze při speciálních dekadicko - aritmetických operacích procesorem a to vnitřně, bez možnosti zásahu operátora.

1.5.1.3.3 Příznak Halfcarry.

Tento příznak má stejnou funkci jako Carry ale při přenosu ze třetího do čtvrtého bitu. Používá se při dekadicko - aritmetických operacích a programátor opět nemá možnost kontrolovat obsah tohoto bitu.

1.5.1.3.4 Příznak Parity/Overflow.

Tento příznak má dva odlišné významy:

a - Při logických operacích určuje paritu dat v registru. Je-li parita lichá je P/O = 0, jinak je P/O = 1.

b - Při aritmetických operacích signalizuje přetečení operandu. Jinými slovy, sleduje chybné nastavení znaménkového bitu (bit 7). Dojde-li k přetečení je P/O = 1, jinak P/O = 0.

1.5.1.3.5 Příznak Zero.

Příznak Zero slouží k indikaci nulového výsledku operace. Je-li výsledek roven nule, bude Z = 1, jinak je Z = 0.

1.5.1.3.6 Příznak Sign.

Tento příznak je ovlivňován znaménkem výsledku operace. Bity v registru se dělí na sedm datových bitů (bity 0 - 6) a jeden znaménkový bit (bit 7). Rozsah osmibitového registru je tedy -128 až +127. Když se bit 7 rovná jedné, což znamená, že obsah registru je záporný, má S hodnotu 1.

1.5.1.4 Registr R.

Registr R slouží interně k obnovování obsahu dynamických pamětí. Obsah tohoto registru lze číst, čehož se dá využít při generování pseudonáhodných čísel.

1.5.1.5 Registr I.

Tento registr plní svou funkci při práci v přerušovacím režimu procesoru a pro začátečníka nemá velký význam. Do registru lze programově nastavit hodnotu potřebnou při zpracování přerušení.

1.5.1.6 Registr PC.

Tento registr se nazývá programový čítač. Svým obsahem určuje adresu v operační paměti na které procesor získává další istrukci ke zpracování. Obsah PC nelze přímo číst a nastavovat.

1.5.1.7 Registr SP.

Registr SP (ukazatel zásobníku) obsahuje šestnáctibitovou adresu paměti, kterou nazýváme zásobník. Na toto místo paměti se ukládají data zvláštními instrukcemi jak uvidíme dále (kapitola 1.5.8.8).

1.5.1.8 Registr HL.

HL je šestnáctibitový registr složený z registrů H a L, přičemž H tvoří jeho horní bajt. K šestnáctibitovému adresování je nutno hned zpočátku poznamenat, že v paměti je nejprve uložena dolní část (dolní bajt) a pak teprve horní část (horní bajt) adresy. Obsahuje-li H například 06H a L hodnotu F4H, pak HL obsahuje 06F4H. Uložíme-li HL do paměti, pak v ní budou data v pořadí F4H a 06H, tedy zdánlivě obráceně. Registr HL je instrukčním souborem předurčen pro aritmetické operace. Je možno říci, že určitým způsobem představuje akumulátor mezi šestnáctibitovými registry.

1.5.1.9 Registry BC a DE.

Tyto registry jsou podobné registru HL. Oba se skládají analogicky z osmibitových registrů B, C, D a E, avšak s těmito registry pracuje daleko méně instrukcí než s HL. Pochopitelně osmibitové části těchto registrů jsou identické s osmibitovými registry stejného označení. Tedy pokud měníme obsah registru B, mění se i obsah registru BC. Pouze některé intrukce chápou BC jako jeden celek a jiné B a C jako samostatné registry. Toto platí přirozeně i pro všechny ostatní šestnáctibitové registry.

1.5.1.10 Registry IX a IY.

Tyto registry jsou příbuzné k registru HL. Jejich názvy byly zvoleny trochu nešťastně. V žádném případě nemají nic společného s registrem I. Části těchto registrů jsou LX, HX, LY a HY. Programování těchto registrů není popsáno v žádné příručce procesoru Z80, avšak my se později k těmto utajeným instrukcím vrátíme.

Registry IX a IY se nazývají indexregistry. Jejich instrukce se podobají instrukcím registru HL, avšak většina z nich zabírá v paměti tři až čtyři bajty (viz kapitola 1.5.4).

1.5.1.11 Záložní registry.

Zvláštností procesoru Z80 je, že registry A, B, C, D, E, F, H a L mají své náhradníky A', B', C', D', E', F', H' a L'. Speciálními instrukcemi se normální registry dají rychle uschovat a nahradit záložními.

1.5.2 Mnemonika a operační kód

Další důležitou kapitolou je vazba mezi symboly a významem instrukcí ve strojovém kódu. Každá instrukce má svůj hexadecimální kód. Například instrukce NOP má kód 00H. Jakmile procesor nalezne tento kód, tedy hodnotu 00H, provede se instrukce NOP (No Operation). Nalezne-li kód 09H, budou se sčítat registry HL a BC. Mnemonický kód této instrukce je ADD HL,BC. Mnemonický kód je srozumi-


```
;
; Zobrazeni bodu na obrazovce
;
;   Souřadnice bodu jsou uloženy jako celá čísla v paměťových buňkách
; X a Y. Počátek souřadnic pro zobrazení je v levém dolním rohu obrazovky.
; Program nejprve zkontroluje, zda čísla uložená v X a Y nejsou menší než
; 0 nebo větší než 639 ve směru X a 199 ve směru Y. Je-li tomu tak, zobra-
; zení se neprovede. Pak se porovnají souřadnice X a Y se souřadnicemi
; OLX a OLY, které byly zobrazovány při předchozím volání tohoto programu.
; Jsou-li stejné, nebo liší-li se o 1, nemusí se provádět výpočet adresy
; ve VRAM, což výrazně urychlí zobrazování vektorů. Adresa ve VRAM se vypo-
; čítá podle vztahu:
;       BE30H-Y*80+X DIV 8
; Zbytek po celočíselném dělení X osmi je číslo bitu na příslušné adrese
; počítáno od nejnižšího bitu a myšleno zleva doprava na obrazovce.


X:ENT
     DEFW 0         ;souřadnice X
Y:ENT
     DEFW 0         ;souřadnice Y
;
PLOT:ENT
     DI
     CALL MEZE      ;Kontrola hranic obrazovky
PL:ENT
     LD BC,(Y)
     LD HL,(OLY)
     LD (OLY),BC
     XOR A
     SBC HL,BC      ;porovnání staré souřadnice s novou
     JR Z,PLX
     LD DE,50H      ;činitel pro násobení
     LD A,H
     OR A
     JR NZ,DLO
     LD A,L
     LD HL,(PPY)
     CP 1
     JR NZ,ZMY
     ADD HL,DE      ;OLY-Y=1
     JR STPY
ZMY: CP -1
     JR NZ,DLO
     JR SBY         ;OLY-Y=-1
DLO: LD B,8
     LD HL,0000
MULT:SRL C
     JR NC,NOADD
     ADD HL,DE
NOADD:SLA E
     RL D
     DJNZ MULT      ;Y*50H
     EX DE,HL
     LD HL,BE30H
SBY: XOR A
     SBC HL,DE      ;BE30H-Y*50H
STPY:LD (PPY),HL
PLX:
     LD DE,(X)
     LD A,E
     AND 7
```

```
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      PUSH HL
      DEC BC
      LD A,C
      OR B
      JR NZ,LOOPC
      IN A,(0E1H)    ;zavřít VRAM
      LD A,41H
      OUT (0CCH),A   ;WF - OR
      LD SP,(SP)     ;návrat zasobniku
      EI
      RET
;
DATA: DEFB 1
      DEFB 2FH
      DEFB 16H
      DEFB 3FH       ;data pro registr palet
SP:   DEFW 0
```


telný programátorovi a jeho strojový ekvivalent vyjádřený hexadecimálním číslem je srozumitelný procesoru. Rozpoznat funkci programu z hexadecimálních čísel je zejména pro začínajícího programátora velmi obtížné. Mnemonický kód je snadno čitelný, neboť vychází ze slovního popisu instrukce. K převodu mnemonického kódu do strojového tvaru nám slouží Assembler. Tedy napíšeme-li např. INC A, Assembler tuto instrukci převede na 3CH. Opakem Assembleru je Disassembler, který převádí strojový kód do mnemonického tvaru. To je výhodné zejména při ladění programu, který je v této fázi v paměti ve strojovém tvaru.

1.5.3 Strukturą paměti.

Napsaný a přeložený program je umístěn v paměti na určité adrese. Paměť začíná adresou 0000H a končí adresou FFFFH. Normálně použitelná paměť u počítače MZ-800 začíná na adrese 1200H. Na každé adrese v paměti je určitý obsah, který může být chápán jako programový nebo datový bajt.

1.5.4. Způsoby adresování

Na způsoby adresování není procesor Z80 tak bohatý jako například procesor 6502. Mnemonika Z80 se dělí na tři části:

a - instrukce
b - cílová adresa nebo registr
c - zdrojová adresa nebo registr

Objasníme si to na příkladu:

```
LD A,B
```

Je to instrukce přesunu dat při kterém se obsah registru B přesune do registru A. Instrukce LD tedy znamená přesun, cílový registr je A a zdrojový registr je B. Tato instrukce je příkladem adresování registru. Složitější situace je v případě instrukce LD A,(HL). Jedná se opět o přesun a akumulátor je opět cílovým registrem. Ale co je zdrojem dat? Závorky symbolizují, že zde je adresována paměť. Uvnitř závorek je registr HL. Jedná se o šestnáctibitový registr, který svým obsahem ukazuje na určité místo v paměti a právě z tohoto místa se do akumulátoru přenesou data. Bude-li obsah HL 0000H, bude se do akumulátoru přenášet hodnota z adresy 0000H. Adresování paměti se v tomto případě provádí nepřímo obsahem registru HL a říkáme mu proto nepřímé adresování.

Ještě jeden příklad:

```
LD (1000H),A
```

Opět se jedná o přesun a závorky naznačují, že cílem operace je paměť. Hodnota, která se nahraje na adresu 1000H se vezme z akumulátoru. Jedná se o přímé adresování, neboť adresa paměti je součástí instrukce. Složitější situace nastane u instrukce LD (1000H),HL. Na adresu 1000H se má uložit obsah 16 - bitového registru HL. Jelikož se jedná o 16 - bitovou hodnotu, uloží se nejen na adresu 1000H ale i na adresu 1001H. Na adresu 1000H se uloží dolní část registru HL. (reg. L) a na adresu 1001H se uloží horní část registru HL (reg. H). Je-li obsah HL = 12F5H, uloží se F5H na adresu 1000H a 12H na adresu 1001H.

Nakonec obrátíme pozornost na již zmíněné indexové adresování pomocí registrů IX a IY. Jak už bylo řečeno, jsou IX i IY blízcí příbuzní registru HL. Pro IX platí v instrukcích rozpoznávací kód DDH, pro IY FDH. Pak je možné zařadit kódy které platí pro HL. Například:

```
INC HL = 23H
INC IX = DDH 23H
INC IY = FDH 23H
```

Toto platí jen pro instrukce, u kterých se nevyskytují operandy zdroje a cíle, neboť pak se v instrukci nachází ještě hodnota indexu. Ta se přidává za operační kód jako další bajt. Příklad:

```
LD A,(IX+05)
```

Tato instrukce má mnoho společného s již zmíněnou instrukcí LD A,(HL). V závorce je nyní indexový registr IX a konstanta 5. Obsahuje-li IX hodnotu 2000H, bude se do A přesouvat hodnota z adresy 2000H + 5 = 2005H.

1.5.4.1 Výpočet indexového adresování.

Rozhodující je výpočet ofsetu indexové hodnoty za operačním kódem. Rozsah hodnot ofsetu indexů je od -128 do +127. Uvažujme například instrukci:

LD A,(IX+F0H)

U hodnoty F0H je nastaven bit 7 do jedničky. To znamená, že hodnota má záporné znaménko. Absolutní hodnotu zjistíme tak, že vytvoříme dvojkový doplněk k číslu F0H (vyměníme všechny nuly za jedničky a naopak a přičteme jedničku). Vyjde nám hodnota 10H. Je-li IX rovno 2000H, bude se hodnota do A přesouvat z adresy 2000H - 10H = 1FF0H. Příklady:

IX	ofset	vypočtená adresa
3000H	10H	3010H
3000H	00H	3000H
3000H	C0H	2FC0H

Nezkušený programátor by měl, dříve než se pustí do programování s indexovými registry, znát všechny možnosti operací s HL.

1.6 Binární logika.

Rovněž binární logika je ve strojovém jazyce velmi důležitá. Většinou se jedná o operace s akumulátorem, neboť instrukční soubor nám umožňuje s akumulátorem velmi dobře pracovat.

Existují následující logické operace:

a - AND - logický součin
b - OR - logický součet
c - XOR - neekvivalence

Nejdříve něco k podstatě operací. Vždy se jedná o interakci mezi dvěma bity podle logického schématu.

Schéma pro operaci OR vypadá takto:

bit 1	bit 2	výsledek
0	0	0
1	0	1
0	1	1
1	1	1

Vysvětlení: Má-li bit 1 n e b o bit 2 (nebo oba) hodnotu 1, má výsledek hodnotu 1. Ve strojové jazyce probíhá tato operace vždy mezi akumulátorem a dalším registrem nebo pamětí a výsledek se ukládá do akumulátoru. Příklad:

```
A = 45H
B = 09H
OR A,B
```

Rozepíšme si operaci v binární formě.

```
A:    0 1 0 0 0 1 0 1   = 45H
B:    0 0 0 0 1 0 0 1   = 09H
      -----------------------
výsledek: 0 1 0 0 1 1 0 1   = 4DH
```

Po vykonání operace se výsledek 4DH objeví v akumulátoru.

Schéma pro operaci AND je následující:

bit 1	bit 2	výsledek
0	0	0
1	0	0
0	1	0
1	1	1

Dodatek C:

Příklad programu v assembleru.

Následující příklad ukazuje na několika podprogramech, jak pracovat s grafikou s vysokým rozlišením. Podprogramy umožňují naprogramovat zákaznický obvod do režimu MZ-800 s rozlišením 640/200, zobrazit nebo vymazat na obrazovce bod, zobrazit vektor, znak nebo řetězec znaků. Program může být umístěn v paměti počítače kdekoliv kromě místa, kam se přepíná VRAM v režimu MZ-800 (t.j. 8000H až BFFFH).

```
;************
; G R A F I K
;************



;  Inicializace do modu 640/200 a výmaz obrazovky
;
;    Zákaznický obvod naprogramujeme do režimu 640/200 a obrazovku
;  (tedy příslušnou část VRAM) vymažeme snadno a rychle například
;  použitím instrukce PUSH HL.

INIT:ENT
     DI
     LD A,4          ;DMD 640/200
     OUT (0CEH),A
     LD A,1
     OUT (0CDH),A    ;RF
     LD A,40H
     OUT (0F0H),A
     LD HL,DATA
     LD B,4
     LD C,0F0H
     OTIR            ;Nastaveni palet
     LD BC,190H      ;počet bytu
     LD HL,BE80H     ;konec VRAM
     LD (SP),SP      ;úklid SP
     LD SP,HL
     LD HL,0000      ;obsah pro výmaz
     LD A,81H
     OUT (0CCH),A    ;WF-REPLACE
     IN A,(0E0H)     ;otevřít VRAM
LOOPC:PUSH HL
     PUSH HL
     PUSH HL
     PUSH HL
     PUSH HL
```

I STROJOVÝ KÓD	INSTRUKCE		OPERACE
I 96	SUB	(HL)	Rozdíl akumulátoru a operandu
I DD9605	SUB	(IX+d)	
I FD9605	SUB	(IY+d)	
I 97	SUB	A	
I 90	SUB	B	
I 91	SUB	C	
I 92	SUB	D	
I 93	SUB	E	
I 94	SUB	H	
I 95	SUB	L	
I D620	SUB	n	
I AE	XOR	(HL)	Exklusive "OR" akumulátoru a operandu
I DDAE05	XOR	(IX+d)	
I FDAE05	XOR	(IY+d)	
I AF	XOR	A	
I A8	XOR	B	
I A9	XOR	C	
I AA	XOR	D	
I AB	XOR	E	
I AC	XOR	H	
I AD	XOR	L	
I D620	XOR	n	

Hodnoty použitých příkladů:

nn=0584H
d=5
n=20H
e=30H

Příklad operace AND A,B se stejnými operandy jako v předcházejícím případě:

```
A:        0 1 0 0 0 1 0 1    = 45H
B:        0 0 0 0 1 0 0 1    = 09H
          -----------------------
výsledek: 0 0 0 0 0 0 0 1    = 01H
```

Po vykonání operace se výsledek 01H objeví v akumulátoru.

Schéma pro operaci XOR je následující:

bit 1	bit 2	výsledek
0	0	0
1	0	1
0	1	1
1	1	0

Příklad operace XOR A,B se stejnými operandy jako v předcházejícím případě:

```
A:        0 1 0 0 0 1 0 1    = 45H
B:        0 0 0 0 1 0 0 1    = 09H
          -----------------------
výsledek: 0 1 0 0 1 1 0 0    = 4CH
```

Po vykonání operace se výsledek 4CH objeví v akumulátoru.

1.7 Aritmetické operace.

1.7.1 ADD - součet.

Jedná se o jednoduchý aritmetický součet dvou registrů nebo registru a paměti. Příklad:

```
HL = 3245H
BC = AF53H
ADD HL,BC
```

Výsledek operace v HL bude D998H.

1.7.2 ADC - součet s přenosem.

ADC je také aritmetický součet dvou registrů, ale navíc se k výsledku přičítá hodnota příznaku Carry. Příklad:

```
ADC HL,BC      (HL a BC stejné jako v 1.7.1, C = 1)
```

Výsledek operace v HL bude D999H.

1.7.3 SUB - rozdíl.

SUB odečítá dva osmibitové registry. SUB A,B tedy znamená A-B do A.

1.7.4 SBC - rozdíl s přenosem.

SBC funguje stejně jako SUB, navíc se od výsledku odečte Carry. Narozdíl od SUB však SBC lze použít i pro 16 - bitové registry.

1.7.5 INC - inkrement.

INC jednoduše přičte k operandu jedničku.
Příklad:

```
HL = 3223H
INC HL
```

Výsledek: HL = 3224H.

1.7.6 DEC - dekrement.

DEC zmenší obsah operandu o jedničku.

1.7.7 CP - porovnání.

CP umožňuje porovnat operand s obsahem akumulátoru. Porovnává se, je-li hodnota operandu menší, stejná nebo větší než akumulátor a podle toho se nastavují příznaky Z a C. Obsahy registrů se nemění.

1.8 Zpracování jednotlivých bitů

Určitá část instrukčního souboru umožňuje nastavovat, nulovat nebo testovat jednotlivé bity operandu.

1.8.1 SET - nastavení bitu.

Instrukce SET umožňuje nastavit určený bit v registru nebo v paměti.
Příklad:

```
SET 7,A
```

nastaví bit 7 v akumulátoru na hodnotu 1.

1.8.2 RES - nulování bitu.

Instrukce umožňuje vynulovat určený bit v registru nebo v paměti.
Příklad:

```
A = 23H
RES 5,A
```

Výsledek: A = 03H

1.8.3 BIT - test bitu.

Instrukce BIT umožňuje otestovat libovolný bit v registru nebo v paměti.
Příklad:

```
BIT 0,A
```

testuje nulty bit v akumulátoru. Komplement hodnoty uvedeného bitu je v příznaku Z. Je-li tedy v bitu nula akumulátoru hodnota 0, je Z=1 a naopak.

1.9 Instrukce rotací.

Tyto instrukce umožňují rotovat obsah registrů. Při rotaci vlevo se bit 6 přesune do bitu 7, bit 5 do bitu 6, bit 4 do bitu 5 atd. Bit 7 se přesune do bitu 0. Při rotaci vpravo se bity posouvají obráceně. Instrukce rotací se liší pouze vztahem k podmínkovým bitům. Nejčastěji se setkáváme s instrukcemi RRCA a RLCA. Tyto instrukce provádějí rotace přes příznakový bit C. Zde je potřeba poukázat na schopnost těchto instrukcí násobit a dělit čísla dvěmi. Pozorujme číslo 52 (00110100). Provedeme-li na něm istrukci RLCA, změní se na číslo 01101000, což je 104. Došlo k jeho vynásobení dvěmi.

1.10 Instrukce pro práci s bloky dat.

Zvláštností procesoru Z80 jsou velmi výkonné instrukce pro práci s bloky dat. Rozdělují se na dva druhy:

a - instrukce pro prohledávání bloků dat
b - instrukce pro přesuny bloků dat

1.10.1 Instrukce pro prohledávání bloků dat.

Tyto instrukce si představíme na příkladu instrukce CPIR (compare-inkrement-repeat; porovnej-zvětši-opakuj). Tato instrukce hledá hodnotu, kterou ob-

I STROJOVÝ KÓD	INSTRUKCE		OPERACE
I CBFF	SET	7,A	
I CBF8	SET	7,B	
I CBF9	SET	7,C	
I CBFA	SET	7,D	
I CBFB	SET	7,E	
I CBFC	SET	7,H	
I CBFD	SET	7,L	
I CB26	SLA	(HL)	Aritmetický posun doleva
I DDCB0526	SLA	(IX+d)	
I FDCB0526	SLA	(IY+d)	
I CB27	SLA	A	
I CB20	SLA	B	
I CB21	SLA	C	
I CB22	SLA	D	
I CB23	SLA	E	
I CB24	SLA	H	
I CB25	SLA	L	
I CB2E	SRA	(HL)	Aritmetický posun doprava
I DDCB052E	SRA	(IX+d)	
I FDCB052E	SRA	(IY+d)	
I CB2F	SRA	A	
I CB28	SRA	B	
I CB29	SRA	C	
I CB2A	SRA	D	
I CB2B	SRA	E	
I CB2C	SRA	H	
I CB2D	SRA	L	
I CB3E	SRL	(HL)	Logický posun doleva
I DDCB053E	SRL	(IX+d)	
I FDCB053E	SRL	(IY+d)	
I CB3F	SRL	A	
I CB38	SRL	B	
I CB39	SRL	C	
I CB3A	SRL	D	
I CB3B	SRL	E	
I CB3C	SRL	H	
I CB3D	SRL	L	

```
I STROJOVY KOD      INSTRUKCE                OPERACE
I CBD1              SET      2,C
I CBD2              SET      2,D
I CBD3              SET      2,E
I CBD4              SET      2,H
I CBDE              SET      3,(HL)
I DDCB05DE          SET      3,(IX+d)
I FDCB05DE          SET      3,(IY+d)
I CBDF              SET      3,A
I CBD8              SET      3,B
I CBD9              SET      3,C
I CBDA              SET      3,D
I CBDB              SET      3,E
I CBDC              SET      3,H
I CBDD              SET      3,L
I CBE6              SET      4,(HL)
I DDCB05E6          SET      4,(IX+d)
I FDCB05E6          SET      4,(IY+d)
I CBE7              SET      4,A
I CBE0              SET      4,B
I CBE1              SET      4,C
I CBE2              SET      4,D
I CBE3              SET      4,E
I CBE4              SET      4,H
I CBE5              SET      4,L
I CBEE              SET      5,(HL)
I DDCB05EE          SET      5,(IX+d)
I FDCB05EE          SET      5,(IY+d)
I CBEF              SET      5,A
I CBE8              SET      5,B
I CBE9              SET      5,C
I CBEA              SET      5,D
I CBEB              SET      5,E
I CBEC              SET      5,H
I CBED              SET      5,L
I CBF6              SET      6,(HL)
I DDCB05F6          SET      6,(IX+d)
I FDCB05F6          SET      6,(IY+d)
I CBF7              SET      6,A
I CBF0              SET      6,B
I CBF1              SET      6,C
I CBF2              SET      6,D
I CBF3              SET      6,E
I CBF4              SET      6,H
I CBF5              SET      6,L
I CBFE              SET      7,(HL)
I DDCB05FE          SET      7,(IX+d)
I FDCB05FE          SET      7,(IY+d)
```


sahuje akumulátor v určitém bloku dat v paměti. Počáteční adresu tohoto bloku určuje obsah HL registru, délku bloku, v němž se má uskutečnit prohledávání, určuje obsah BC. Bude-li hodnota v bloku nalezena, prohledávání se ukončí a nastaví se příznak Z. Registr HL pak určuje místo, kde se hledaná hodnota nachází. Nebude-li hodnota v celém bloku nalezena, hledání se ukončí na konci bloku a příznak Z bude vynulován. Příklad: zkoumáme výskyt hodnoty 66H v rozsahu 3000H až 4000H.

```
LD A,66H      ;vzorek
LD HL,3000H   ;začátek bloku
LD BC,1000H   ;délka bloku
CPIR          ;prohledávání bloku
JP Z,XXXX     ;pokračování při nalezení hodnoty
```

1.10.2 Instrukce pro přesuny datových bloků.

Tyto instrukce představíme na instrukci LDIR (load-increment-repeat; přenes-zvětši-opakuj). Tato instrukce přesune blok dat z jednoho místa paměti na jiné. Počáteční adresa původního bloku dat je v HL, počáteční adresa cílového bloku je v DE a délka bloku je v BC. Příklad: máme přenést data z paměti v rozsahu 2000H až 3000H na adresy počínající hodnotou C000H.

```
LD HL,2000H     ;začátek zdrojového bloku
LD DE,C000H     ;začátek cílového bloku
LD BC,1000H     ;délka bloku
LDIR            ;přenos bloku
```

1.11 Instrukce skoků.

Instrukce skoků lze rozdělit podle dvou různých kritérií do několika skupin. Jedno z kritérií je dělí na podmíněné a nepodmíněné skoky. Zatímco nepodmíněné skoky se vykonávají vždy, když na ně procesor v programu narazí, je pro vykonání podmíněných skoků nutné splnění nějaké podmínky. Podmínku tvoří stav zvoleného příznakového bitu (ať již jeho nastavení nebo snulování) a skok je vykonán pouze tehdy, je-li tato podmínka splněna.

Druhé kritérium dělí skoky na absolutní a relativní.

1.11.1 Absolutní skoky.

Do této skupiny patří skoky, jejichž cílová adresa je již nějakým způsobem obsažena v instrukci. Tato cílová adresa zde může být obsažena v operačním kódu buď přímo hodnotou, nebo nepřímo - prostřednictvím některého registru.
Příklad:

```
JP 3000H        ;absolutní nepodmíněný skok na adresu 3000H
JP NZ,3000H     ;absolutní podmíněný skok na adresu 3000H,
                ;vykoná se pouze je-li příznak Z = 0
```

1.11.2 Relativní skoky.

Relativní skoky jsou poněkud komplikovanější než skoky absolutní. Dělí se rovněž na podmíněné a nepodmíněné, ale liší se způsobem určení adresy cíle skoku. V mnemonickém kódu relativního skoku se neudává hodnota adresy cíle skoku, ale pouze hodnota určující o kolik bajtů vpřed nebo vzad se bude skákat. Tato hodnota může ležet v rozsahu -128 až +127. Výhodou tohoto typu skoků je, že jejich operační kód je o jeden bajt kratší než u skoků absolutních a že programový blok obsahující pouze relativní skoky lze libovolně přemisťovat uvnitř celé adresovatelné paměti a to beze změn operačních kódů. Nevýhodou naproti tomu je omezení rozsahu skoku, což také začátečníkům způsobuje určité komplikace.
Příklad:

```
JR C,+16H ;(Jump relative if Carry)
          ;relativní podmíněný skok, který se vykoná
          ;je-li nastaven příznak C a který způsobí
          ;přesun vpřed o 22 bajtů (16H = 22)
```

Zvláštností jsou skoky typu JP (HL). Tato intrukce znamená, že se má skákat na adresu, která je obsažena v registru HL.

1.12 Volání podprogramů a návrat z nich

Volání podprogramů se provádí pomocí instrukcí typu CALL. Tyto instrukce provádějí skok na adresu, na níž začíná volaný podprogram. Narozdíl od instrukcí skoků však předem připravují cestu pro návrat z volaného podprogramu. Tato příprava spočívá v nahrání adresy instrukce následující za instrukcí CALL do takzvaného zásobníku. Zásobníkem rozumíme specializovanou oblast paměti adresovanou pomocí registru SP, s jejíž funkcí se seznámíme v následující kapitole. Návrat z podprogramu se provádí pomocí instrukcí typu RET, které tuto adresu vyzvednou ze zásobníku a předají na ni řízení.

Instrukce typu CALL i instrukce typu RET jsou opět podmíněné nebo nepodmíněné, stejně jako instrukce skoků.

1.13 Práce se zásobníkem

Zvláštní část paměti adresovaná registrem SP se nazývá zásobník. Tento zásobník slouží k přechodné úschově návratových adres podprogramů nebo obsahů registrů. Existují i jiné způsoby využití zásobníku, například pro předávání parametrů mezi hlavním programem a podprogramem a lze najít i jiná použití.

Pro práci se zásobníkem existuje dvojice instrukcí PUSH a POP. Instrukce PUSH ukládá do zásobníku jeden celý šestnáctibitový registr (BC, DE, HL, IX, IY) nebo obsah akumulátoru a příznakových registrů současně (AF). Instrukcí POP naopak tyto registry naplníme.

Kam se vlastně registry nebo návratové adresy (v případě instrukce CALL) ukládají? Na volnou adresu zásobníku ukazuje registr SP. Jakmile dojde při instrukci PUSH nebo CALL k ukládání na zásobník, uloží se hodnota na tuto volnou adresu a obsah registru SP se automaticky sníží. Registr SP opět ukazuje na volnou adresu zásobníku a dojde k uložení druhého bajtu dat. Opět dojde k automatickému snížení obsahu registru SP, který tedy neustále ukazuje na počátek volné oblasti zásobníku. Při instrukcích POP nebo RET dochází k opačnému ději. Obsah registru SP se nejprve automaticky zvýší a obsah adresované paměti se přesune do příslušného registru. Tento děj se zopakuje pro druhý bajt šestnáctibitových dat. Registr SP nyní opět ukazuje na volnou adresu zásobníku, neboť data na této adrese umístěna byla přemístěna do některého registru.

Je zřejmé, že při vícenásobném použití instrukcí PUSH nebo při vnořeném volání podprogramů musíme hodnoty ze zásobníku získávat v opačném pořadí, než v jakém byly ukládány. Hodnota uložená jako poslední je přístupná jako první. Příklad:

```
        .           ;SP = 2000H
PUSH BC             ;SP = 1FFEH
PUSH DE             ;SP = 1FFCH
PUSH HL             ;SP = 1FFAH
        .
        .
POP HL              ;SP = 1FFCH
POP DE              ;SP = 1FFEH
POP BC              ;SP = 2000H
        .
        .
```

Instrukce, které pracují s registrem SP nám dovolují umístit zásobník do libovolného místa v paměti. Záleží tedy jenom na vás, zda bude zásobník umístěn tak, aby nevadil ostatním částem vašeho programu. Rovněž je potřeba v programu zajistit, aby každá instrukce CALL měla odpovídající instrukci RET a každá instrukce PUSH odpovídající instrukci POP. Jakmile toto nezabezpečíte, dojde zákonitě ke katastrofě. Zásobník bleskurychle zaplní celou paměť a smaže váš program.

I STROJOVÝ KÓD	INSTRUKCE		OPERACE
I C7	RST	00H	Restart
I CF	RST	08H	
I D7	RST	10H	
I DF	RST	18H	
I E7	RST	20H	
I EF	RST	28H	
I F7	RST	30H	
I FF	RST	38H	
I DE20	SBC	A, n	Odečtení operandu od A s carry
I 9E	SBC	A, (HL)	
I DD9E05	SBC	A, (IX+d)	
I FD9E05	SBC	A, (IY+d)	
I 9F	SBC	A, A	
I 98	SBC	A, B	
I 99	SBC	A, C	
I 9A	SBC	A, D	
I 9B	SBC	A, E	
I 9C	SBC	A, H	
I 9D	SBC	A, L	
I ED42	SBC	HL, BC	
I ED52	SBC	HL, DE	
I ED62	SBC	HL, HL	
I ED72	SBC	HL, SP	
I 37	SCF		Nastavení C=1
I CBC6	SET	0, (HL)	Nastavení bitu b v lokaci nebo registru
I DDCB05C6	SET	0, (IX+d)	
I FDCB05C6	SET	0, (IY+d)	
I CBC7	SET	0, A	
I CBC0	SET	0, B	
I CBC1	SET	0, C	
I CBC2	SET	0, D	
I CBC3	SET	0, E	
I CBC4	SET	0, H	
I CBC5	SET	0, L	
I CBCE	SET	1, (HL)	
I DDCB05CE	SET	1, (IX+d)	
I FDCB05CE	SET	1, (IY+d)	
I CBCF	SET	1, A	
I CBC8	SET	1, B	
I CBC9	SET	1, C	
I CBCA	SET	1, D	
I CBCB	SET	1, E	
I CBCC	SET	1, H	
I CBCD	SET	1, L	
I CBD6	SET	2, (HL)	
I DDCB05D6	SET	2, (IX+d)	
I FDCB05D6	SET	2, (IY+d)	
I CBD7	SET	2, A	
I CBD0	SET	2, B	

STROJOVY KOD	INSTRUKCE		OPERACE
CB06	RLC	(HL)	Kruhová rotace doleva
DDCB0506	RLC	(IX+d)	
FDCB0506	RLC	(IY+d)	
CB07	RLC	A	
CB00	RLC	B	
CB01	RLC	C	
CB02	RLC	D	
CB03	RLC	E	
CB04	RLC	H	
CB05	RLC	L	
07	RLCA		Kruhová rotace akumulátoru vlevo
ED6F	RLD		Rotace doleva a doprava mezi A (HL)
CB1E	RR	(HL)	Rotace vpravo přes carry
DDCB051E	RR	(IX+d)	
FDCB051E	RR	(IY+d)	
CB1F	RR	A	
CB18	RR	B	
CB19	RR	C	
CB1A	RR	D	
CB1B	RR	E	
CB1C	RR	H	
CB1D	RR	L	
1F	RRA		Rotace akumulátoru vpravo přes carry
CB0E	RRC	(HL)	Kruhová rotace doprava
DDCB050E	RRC	(IX+d)	
FDCB050E	RRC	(IY+d)	
CB0F	RRC	A	
CB08	RRC	B	
CB09	RRC	C	
CB0A	RRC	D	
CB0B	RRC	E	
CB0C	RRC	H	
CB0D	RRC	L	
0F7	RRCA		Kruhová rotace akumulátoru vpravo
ED67	RRD		Rotace doprava a doleva mezi A (HL)

1.14 Instrukce DJNZ.

Instrukce DJNZ je speciální instrukce skoku, která plní funkci programové smyčky. Přitom je registr B použit jako počítadlo cyklu. Jedná se o relativní skok, který se vykonává tak dlouho, dokud nebude registr B nulový. Před každým vykonáním instrukce DJNZ se obsah registru B sníží a otestuje. Nejlépe si to objasníme na příkladu:

```
1200    LD B,6          ;počítadlo cyklu
1202    CALL 003EH      ;volání podprogramu
1205    DJNZ 1202H      ;smyčka
```

Tento krátký program způsobí šestinásobné vyvolání podprogramu na adrese 003EH.

1.15 Instrukce pro řízení vstupů a výstupů.

Mikroprocesor Z80 umožňuje vysílat a přijímat data přes vstupní a výstupní brány pomocí celé skupiny instrukcí. Příjem dat ze vstupních bran se provádí pomocí instrukcí typu IN a vysílání dat přes výstupní brány zabezpečují instrukce typu OUT. Pomocí všech těchto instrukcí spolupracuje procesor s připojenými zařízeními (tiskárna, magnetofon, pružný disk, ale i klávesnice, reproduktor ...). V počítači MZ-800 se tyto instrukce používají i k různým systémovým službám, jak uvidíme později.

2. Příklady programů.

2.1 Doporučení.

Následující příklady programů se dají napsat a spustit přímo v Monitoru počítače MZ-800 pomocí příkazů M a J. Převod mnemokódů na strojní kódy lze provádět pomocí převodní tabulky. Pro lepší komfort je však výhodnější používat některý dostupný ladicí prostředek.

2.2 Výpis na obrazovku.

2.2.1 Výpis textu.

Při výpisu textu na obrazovku použijeme podprogram z Monitoru. Příslušný text musíme nejprve umístit do paměti počítače tak, aby si ho podprogram mohl vyzvednout a zpracovat. Na adresu 2000H proto zapíšeme čísla 41H, 42H, 43H a 0DH, což odpovídá znakům A, B, C a [CR] (ukončovač textu). Vlastní program bude vypadat takto:

```
        LD DE,2000H     ;adresa textu
        CALL 0015H      ;podprogram pro výpis
        JP 00ADH        ;návrat do Monitoru
```

Do registru DE nahrajeme počáteční adresu textu a podprogram začínající na adrese 15H vypíše na obrazovku "ABC".

2.2.2 Výpis znaku.

V předcházejícím případě můžeme na obrazovce zobrazit libovolně dlouhý text. Nyní budeme chtít zobrazit pouze jeden znak. Pochopitelně to můžeme udělat stejně jako v předchozím případě. Existuje ale možnost využít jiný podprogram Monitoru umístěný na adrese 0012H. Tento podprogram požaduje, aby znak, který chceme zobrazit, byl v akumulátoru. Chceme-li zobrazit "0", napíšeme:

```
        LD A,30H  ;kód nuly
        CALL 0012H
        JP 00ADH
```

2.3 Obsluha klávesnice.

Obsluha klávesnice je důležitým programovým prvkem. V Monitoru opět existuje podprogram (GET - adresa 001BH), který získá kód stisknuté klávesy. Zavoláme-li tento podprogram a bude-li ve stejném okamžiku stisknuta klávesa, bude akumulátor po návratu z podprogramu obsahovat kód této klávesy. Nebyla-li stisknuta žádná klávesa, bude akumulátor po návratu z podprogramu obsahovat hodnotu 0. Následující program sejme kód stisknuté klávesy a zobrazí ho na obrazovce:

```
1200H   CALL 001BH      ;podprogram GET
1203H   JR Z,1200H      ;nebyla stisknuta klávesa - znovu GET
1205H   CALL 0012H      ;zobrazení
1208H   JP 1200H        ;návrat do Monitoru
```

2.4 Dimenzování ve strojovém jazyce.

Při práci s daty potřebuje každý programátor vyhradit v paměti určitou oblast pro uložení dat. V BASICu k tomu slouží příkaz DIM, ale ve strojovém jazyce musíme tuto úlohu vyřešit sami. Zkusíme realizovat "dimenzování" pomocí HL registru a provedeme srovnání s BASICem. Nejprve příklad v BASICu:

```
10 DIM A (4)
20 A (0) = 10000 : A (1) = 10001 : A (2) = 10003 : A (3) = 10005 : A (4) = 0
30 X = 1 : PRINT A (X)
```


I STROJOVÝ KÓD	INSTRUKCE		OPERACE
I CBB6	RES	6,(HL)	
I DDCB05B6	RES	6,(IX+d)	
I FDCB05B6	RES	6,(IY+d)	
I CBB7	RES	6,A	
I CBB0	RES	6,B	
I CBB1	RES	6,C	
I CBB2	RES	6,D	
I CBB3	RES	6,E	
I CBB4	RES	6,H	
I CBB5	RES	6,L	
I CBBE	RES	7,(HL)	
I DDCB05BE	RES	7,(IX+d)	
I FDCB05BE	RES	7,(IY+d)	
I CBBF	RES	7,A	
I CBB8	RES	7,B	
I CBB9	RES	7,C	
I CBBA	RES	7,D	
I CBBB	RES	7,E	
I CBBC	RES	7,H	
I CBBD	RES	7,L	
I C9	RET		Návrat z podprogramu
I D8	RET	C	Návrat z podprogramu, jeli podmínka
I F8	RET	M	splněna
I D0	RET	NC	
I C0	RET	NZ	
I F0	RET	P	
I E8	RET	PE	
I E0	RET	PO	
I C8	RET	Z	
I ED4D	RETI		Návrat z interruptu
I ED45	RETN		Návrat z nemaskovatelného interruptu
I CB16	RL	(HL)	Rotace vlevo přes carry
I DDCB0516	RL	(IX+d)	
I FDCB0516	RL	(IY+d)	
I CB17	RL	A	
I CB10	RL	B	
I CB11	RL	C	
I CB12	RL	D	
I CB13	RL	E	
I CB14	RL	H	
I CB15	RL	L	
I 17	RLA		Rotace akumulátoru vlevo přes carry

I STROJOVY KOD	INSTRUKCE	OPERACE
I FDCB058E	RES	1, (IY+d)
I CB8F	RES	1, A
I CB88	RES	1, B
I CB89	RES	1, C
I CB8A	RES	1, D
I CB8B	RES	1, E
I CB8C	RES	1, H
I CB8D	RES	1, L
I CB96	RES	2, (HL)
I DDCB0596	RES	2, (IX+d)
I FDCB0596	RES	2, (IY+d)
I CB97	RES	2, A
I CB90	RES	2, B
I CB91	RES	2, C
I CB92	RES	2, D
I CB93	RES	2, E
I CB94	RES	2, H
I CB9E	RES	3, (HL)
I DDCB059E	RES	3, (IX+d)
I FDCB059E	RES	3, (IY+d)
I CB9F	RES	3, A
I CB98	RES	3, B
I CB99	RES	3, C
I CB9A	RES	3, D
I CB9B	RES	3, E
I CB9C	RES	3, H
I CB9D	RES	3, L
I CBA6	RES	4, (HL)
I DDCB05A6	RES	4, (IX+d)
I FDCB05A6	RES	4, (IY+d)
I CBA7	RES	4, A
I CBA0	RES	4, B
I CBA1	RES	4, C
I CBA2	RES	4, D
I CBA3	RES	4, E
I CBA4	RES	4, H
I CBA5	RES	4, L
I CBAE	RES	5, (HL)
I DDCB05AE	RES	5, (IX+d)
I FDCB05AE	RES	5, (IY+d)
I CBAF	RES	5, A
I CBA8	RES	5, B
I CBA9	RES	5, C
I CBAA	RES	5, D
I CBAB	RES	5, E
I CBAC	RES	5, H
I CBAD	RES	5, L

Tento program zobrazí číslo 10001 na obrazovce.

Ve strojovém jazyce odpovídá proměnné A určitá (nepojmenovaná) oblast paměti. Její počáteční adresu musíme určit sami a to tak, aby oblast paměti pro uložení dat nepřekrývala oblast využívanou v počítači jiným způsobem. Tato oblast musí být dostatečně velká pro uložení všech dat. Dále je třeba zvolit i tvar dat, s nimiž budeme pracovat. Pro hodnoty použité v předchozím případě lze použít šestnáctibitové vyjádření čísel. Každá hodnota bude v paměti uložena ve dvou následujících bajtech a hodnoty budou následovat těsně jedna za druhou. Počátek paměti dat určíme 3000H a musíme sem uložit hodnoty proměnných.

```
3000   10 27 11 27 13 27 15 27
3008   00 00
```

Toto jsou hodnoty proměnných A(0) až A(4) v hexadecimálním tvaru. K určení příslušné proměnné potřebujeme dvě hodnoty:

	BASIC	strojový jazyk
a/	proměnná (A)	počáteční adresa paměti dat
b/	index (X)	posun vůči počátku paměti

Posun vůči počátku paměti dat musí mít dvojnásobnou hodnotu než index v BASICu, neboť data jsou ukládána v dvoubajtovém tvaru. Program, který bude mít stejnou funkci jako předchozí, má tvar:

```
        LD A, 1          ;X=1  .
        LD HL, 3000H     ;adresa počátku paměti dat
        ADD A            ;výpočet ofsetu - 2 * X
        LD B, 0
        LD C, A
        ADD HL, BC       ;výpočet adresy žádané hodnoty
        LD E, (HL)
        INC HL
        LD D, (HL)       ;přenos dat do registru DE
        EX DE, HL        ;výměna obsahu registru DE a HL
        CALL 03BAH       ;zobrazení obsahu HL
        JP 00ADH         ;návrat do monitoru
```

Změníme-li X v prvním řádku programu na jinou hodnotu, zobrazíme jiný prvek pole A.

V tomto programu je několik velmi často používaných rutin, na které se blíže podíváme. Nejprve nastavíme oba parametry a pak provádíme instrukci ADD A. Máme-li pomocí parametru 1 (X) adresovat šestnáctibitové hodnoty dat z pole A (parametr 2), které leží na adresách 3000H, 3002H, 3004H, ..., musíme přičítat jako ofset adresy dvojnásobnou hodnotu parametru 1. Toto vynásobení se provede právě instrukcí ADD A. Nyní musíme hodnotu ofsetu přičíst k počáteční adrese paměti dat. Proto musíme sečíst hodnoty registru HL a A. Toto sečtení nelze provést přímo a obsah akumulátoru musíme proto převést do registru BC. Chybějící bajt nahradíme hodnotou 0. Dva šestnáctibitové registry BC a HL již můžeme sečíst. Obsahem registru HL je po této operaci adresa žádaných dat. Nyní tato data potřebujeme přenést opět do registru HL. To však nelze provést přímo, ale pouze s využitím dalšího šestnáctibitového registru (DE). Data nejprve přeneseme do tohoto registru a potom vyměníme obsahy registrů DE a HL. Nakonec obsah registru HL zobrazíme.

2.5 Cyklus.

Cyklus je velmi důležitou programovací technikou a my si ukážeme jak vytvořit cyklus, jehož parametr je větší než 255 a nelze tedy použít instrukci DJNZ. Napíšeme program, který nechá tisíckrát zaznít tón zvukového generátoru (prakticky to raději nezkoušejte, mějte ohled na své okolí):

```
1200 LD BC,03E8H     ;=1000D
     PUSH BC         ;uschovej obsah BC
     CALL 003BH      ;podprogram Monitoru pro vytvoření tónu
     POP BC          ;obnovení obsahu BC
     DEC BC          ;snížení obsahu o jednu
     LD A,B
     OR C
     JP NZ,1200H     ;test konce cyklu
     RET
```

BC registr je použit jako řídící proměnná cyklu. Proto musíme jeho obsah před voláním podprogramu uložit, neboť nemáme jistotu, že tento podprogram registr BC nevyužívá. Po návratu z podprogramu jeho obsah obnovíme, snížíme o jednotku a zjistíme, není-li již nulový. Pro zjištění nulové hodnoty registru BC musíme použít akumulátor a využít logickou operaci OR. Výsledek této logické operace je nulový jen tehdy, jsou-li obě vstupní veličiny nulové. Proto jednu veličinu (obsah registru B) přesuneme do akumulátoru přímo a druhou (obsah registru C) prostřednictvím instrukce OR. Není-li výsledek nulový, opakujeme celý cyklus.

2.6 Dodatečné instrukce Z80.

Dodatečné instrukce Z80 většinou nenajdete v žádné knize. Celý soubor instrukcí procesoru Z80 totiž obsahuje přibližně 1000 instrukcí a ne pouhých 400, které se využívají. Nepopsané instrukce jsou většinou určeny pro práci s registry HX, HY, LX a LY. Operační kódy jsou příbuzné s operačními kódy pro práci s částmi registru HL. Příklad:

```
26 77          LD H,77
DD 26 77       LD HX,77
```

Lze tedy jednoduše napsat DDH (pro HX, HY) nebo FDH (pro LX, LY) před již známý kód instrukce pro práci s H nebo L registrem. Žádný z dostupných programovacích prostředků neumí dodatečné instrukce přímo využívat a proto se jejich používání příliš nerozšířilo.

2.7 Závěrem kapitoly 1 a 2.

Závěrem těchto úvodních kapitol je třeba upozornit na skutečnost, že to co bylo v těchto kapitolách uvedeno, zdaleka nestačí k pochopení celé problematiky programování ve strojovém kódu. Jedná se spíše o první seznámení s tímto způsobem programování. Obecně lze říci, že nejlepší učebnicí programování ve strojovém kódu je komentovaný výpis programu, který vytvořil některý zkušený programátor. V dodatku C této publikace je pro tento účel uveden výpis programu pro obsluhu grafiky v módu MZ-800. Další přílohy tvoří seznam instrukcí strojového kódu, doplněný dalšími potřebnými informacemi a tabulka kódu ASCII.

Hlavním účelem těchto úvodních kapitol byla nutná příprava pro pochopení dále uváděných informací o vnitřním uspořádání počítače MZ-800 a o způsobech jeho efektivního využití.

I	STROJOVÝ KÓD	INSTRUKCE		OPERACE
I	ED8B	OTDR		Přesun do výstupního portu (C)
I				z lokace (HL), dekrement HL a B
I				a opakovat dokud B=0
I	ED83	OTIR		Přesun do výstupního portu (C)
I				z lokace (HL), inkrement HL,
I				dekrement B
I				a opakovat až bude B=0
I	ED79	OUT	(C),A	Přesun do výstupního portu (C) z reg.
I	ED41	OUT	(C),B	
I	ED49	OUT	(C),C	
I	ED51	OUT	(C),D	
I	ED59	OUT	(C),E	
I	ED61	OUT	(C),H	
I	ED69	OUT	(C),L	
I	D320	OUT	(n),A	Přesun do výstupního portu (n) z A
I	EDAB	OUTD		Přesun do výstupního portu (C)
I				z lokace (HL), dekrement HL a B
I	EDA3	OUTI		Přesun do výstupního portu (C)
I				z lokace (HL), inkrement HL a
I				dekrement B
I	F1	POP	AF	Přesun obsahu (SP) do lokace
I	C1	POP	BC	
I	D1	POP	DE	
I	E1	POP	HL	
I	DDE1	POP	IX	
I	FDE1	POP	IY	
I	F5	PUSH	A	Přesun lokace do (SP)
I	C5	PUSH	BC	
I	D5	PUSH	DE	
I	E5	PUSH	HL	
I	DDE5	PUSH	IX	
I	FDE5	PUSH	IY	
I	CB86	RES	0,(HL)	Nulování bitu b v lokaci nebo registru
I	DDCB0586	RES	0,(IX+d)	
I	FDCB0586	RES	0,(IY+d)	
I	CB87	RES	0,A	
I	CB80	RES	0,B	
I	CB81	RES	0,C	
I	CB82	RES	0,D	
I	CB83	RES	0,E	
I	CB84	RES	0,H	
I	CB85	RES	0,L	
I	CB8E	RES	1,(HL)	
I	DDCB058E	RES	1,(IX+d)	

I STROJOVY KOD	INSTRUKCE		OPERACE
I FD2A8405	LD	IY, (nn)	
I FD218405	LD	IY, nn	
I 6E	LD	L, (HL)	
I DD6E05	LD	L, (IX+d)	
I FD6E05	LD	L, (IY+d)	
I 6F	LD	L, A	
I 68	LD	L, B	
I 69	LD	L, C	
I 6A	LD	L, D	
I 6B	LD	L, E	
I 6C	LD	L, H	
I 6D	LD	L, L	
I 2E20	LD	L, n	
I ED4F	LD	R, A	
I ED7B8405	LD	SP, (nn)	
I F9	LD	SP, HL	
I DDF9	LD	SP, IX	
I FDF9	LD	SP, IY	
I 318405	LD	SP, nn	
I EDA8	LDD		Přesun z lokace (HL) do lokace (DE), dekrement HL, DE a BC
I EDB8	LDDR		Přesun z lokace (HL) do lokace (DE), dekrement HL, DE a BC a opakovat, dokud BC=0
I EDA0	LDI		Přesun z lokace (HL) do lokace (DE), inkrement HL a DE a dekrement BC
I EDB8	LDDR		Přesun z lokace (HL) do lokace (DE), dekrement HL a DE a dekrement BC opakovat dokud BC=0
I ED44	NEG		Negace A (dvojkový doplněk)
I 00	NOP		Prázdná instrukce.
I B6	OR	(HL)	Logické "OR" operandu a akumulátoru
I DDB605	OR	(IX+d)	
I FDB605	OR	(IYX+d)	
I B7	OR	A	
I B0	OR	B	
I B1	OR	C	
I B2	OR	D	
I B3	OR	E	
I B4	OR	H	
I B5	OR	L	
I F620	OR	n	

3. Registr módu zobrazení (display mode registr - DMD) - adresa CEH.

MZ-800 může pracovat ve dvou různých režimech:

a) režim MZ-700, ve kterém se počítač chová přesně jako počítač MZ-700.

b) režim MZ-800, ve kterém si počítač uchovává všechny své vlastnosti.

Přepnutí mezi oběma režimy se neděje systémovým přepínačem na zadní straně počítače, jak se uvádí v příručce uživatele, ale právě přes registr DMD. Systémový přepínač nemá žádný přímý vliv na režim, ve kterém se počítač nachází. Jeho poloha se otestuje po spustění MONITORU a program sám podle výsledku testu nastaví příslušný režim.

Systémový přepínač si může uživatel bez problémů otestovat přes port CEH. Je-li v poloze MZ-700, je druhý bit (bit 1) na tomto portu v 0.

Volba zobrazovacího, popřípadě pracovního režimu se děje právě přes port CEH. Principielně lze volit mezi režimy MZ-700 a MZ-800. Přitom každý zobrazovací režim, který využívá grafiku s vysokým rozlišením, je režimem MZ-800.

3.1 Několik slov o organizaci VideoRAM v různých režimech:

3.1.1 Rozdělení VideoRAM bez rozšíření.

V režimu MZ-800 je k dispozici 16kB VideoRAM. Paměť je organizována podle režimu následujícím způsobem:

Při rozlišení 640/200 bodů existuje jeden grafický blok o velikosti 16kB. V něm jeden nastavený bit reprezentuje jeden svítící grafický bod (paleta 1) a jeden vynulovaný bit odpovídá jednomu zhasnutému grafickému bodu (paleta 0).

Komplikovaně situace vyhlíží při rozlišení 320/200 bodů. V tomto případě existují dva grafické bloky po 8kB, které vytváří překrytím odpovídající kód palety. Bit, který není nastaven v rovině 1 ani v rovině 2, je zhasnutý grafický bod (paleta 0). Je-li bit nastaven pouze v rovině 1, je to rozsvícený grafický bod z palety 1. Bit nastavený pouze v rovině 2 reprezentuje svítící grafický bod z palety 2. Je-li bit nastaven v obou rovinách, kóduje rozsvícený grafický bod z palety 3.

3.1.2 Rozdělení VideoRAM s rozšířením.

S grafickým rozšířením má uživatel k dispozici 32kB VideoRAM s následující organizací:

V režimu 640/200 bodů je zobrazení buď jednobarevné, tak jako bez grafického rozšíření, přičemž je možné přepínat dva různé grafické obrazy, nebo ve čtyřech barvách, přičemž spojení bitů pro kódy palet je stejné jako v režimu 320/200 bodů bez rozšíření.

Také v režimu 320/200 existují dvě možnosti. Buďto můžeme pracovat ve čtyřech barvách s možností využít přepínání dvou obrazů, nebo se šestnáctibarevným zobrazením. V tomto případě existují čtyři grafické roviny. Jejich kombinace vytváří kódy palet.

3.2 Volba zobrazovacího režimu.

Datový výstup na registr DMD (bity 0-3) určuje zobrazovací režim podle následujících tabulek.

Bit 3	Bit 2	Zobrazovací režim
0	0	320/200, MZ-800
0	1	640/200, MZ-800
1	0	MZ-700
1	1	nedovolená kombinace

Tab. 3.1 - Význam 2. a 3. bitu registru DMD.

Bit 1	Bit 0	Zobrazení grafických bloků	
		320/200	640/200
0	0	Obraz A, blok I+II	Obraz A, blok I
0	1	Obraz B, blok III+IV	Obraz B, blok III
1	0	16 barev, bloky I-IV	4 barvy, bloky I+III

Tab. 3.2 - Význam 0. a 1. bitu registru DMD.

Jak je vidět z tabulky 3.2, nejsou v základní verzi dovoleny obě spodní kombinace bitů 0 a 1, protože se nedostává paměť pro bloky III a IV. Pokud je uživatel přesto zvolí, nemůže si být jist výsledkem.

STROJOVÝ KÓD	INSTRUKCE	OPERACE
4F	LD	C, A
48	LD	C, B
49	LD	C, C
4A	LD	C, D
4B	LD	C, E
4C	LD	C, H
4D	LD	C, L
0E20	LD	C, n
56	LD	D, (HL)
DD5605	LD	D, (IX+d)
FD5605	LD	D, (IY+d)
57	LD	D, A
50	LD	D, B
51	LD	D, C
52	LD	D, D
53	LD	D, E
54	LD	D, H
55	LD	D, L
1620	LD	D, n
ED5B8405	LD	DE, (nn)
118405	LD	DE, nn
5E	LD	E, (HL)
DD5E05	LD	E, (IX+d)
FD5E05	LD	E, (IY+d)
5F	LD	E, A
58	LD	E, B
59	LD	E, C
5A	LD	E, D
5B	LD	E, E
5C	LD	E, H
5D	LD	E, L
1E20	LD	E, n
66	LD	H, (HL)
DD6605	LD	H, (IX+d)
FD6605	LD	H, (IY+d)
67	LD	H, A
60	LD	H, B
61	LD	H, C
62	LD	H, D
63	LD	H, E
64	LD	H, H
65	LD	H, L
2620	LD	H, n
2A8405	LD	HL, (nn)
218405	LD	HL, nn
ED47	LD	I, A
DD2A8405	LD	IX, (nn)
DD218405	LD	IX, nn

I STROJOVY KOD	INSTRUKCE	OPERACE
I FD7705	LD	(IY+d),A
I FD7005	LD	(IY+d),B
I FD7105	LD	(IY+d),C
I FD7205	LD	(IY+d),D
I FD7305	LD	(IY+d),E
I FD7405	LD	(IY+d),H
I FD7505	LD	(IY+d),L
I FD360520	LD	(IY+d),n
I 328405	LD	(nn),A
I ED438405	LD	(nn),BC
I ED538405	LD	(nn),DE
I 228405	LD	(nn),HL
I DD228405	LD	(nn),IX
I FD228405	LD	(nn),IY
I ED738405	LD	(nn),SP
I 0A	LD	A,(BC)
I 1A	LD	A,(DE)
I 7E	LD	A,(HL)
I DD7E05	LD	A,(IX+d)
I FD7E05	LD	A,(IY+d)
I 3A8405	LD	A,(nn)
I 7F	LD	A,A
I 78	LD	A,B
I 79	LD	A,C
I 7A	LD	A,D
I 7B	LD	A,E
I 7C	LD	A,H
I ED57	LD	A,I
I 7D	LD	A,L
I 3E20	LD	A,n
I ED5F	LD	A,R
I 46	LD	B,(HL)
I DD4605	LD	B,(IX+d)
I FD4605	LD	B,(IY+d)
I 47	LD	B,A
I 40	LD	B,B
I 41	LD	B,C
I 42	LD	B,D
I 43	LD	B,E
I 44	LD	B,H
I 45	LD	B,L
I 0620	LD	B,n
I ED4B8405	LD	BC,(nn)
I 018405	LD	BC,nn
I 4E	LD	C,(HL)
I DD4E05	LD	C,(IX+d)
I FD4E05	LD	C,(IY+d)

4. Režim MZ-700.

Režim MZ-700 je způsob zobrazování, ve kterém může uživatel nejjednodušším způsobem zobrazovat texty na obrazovce. Díky programovatelnému znakovému generátoru (PCG) existují i v tomto režimu jisté grafické možnosti.

V režimu MZ-700 se paměť počítače skládá v podstatě ze tří částí:

- 0000H - 0FFFH .... ROM (MONITOR) nebo RAM
- 1000H - CFFFH .... vždy RAM
- D000H - FFFFH .... ROM (MONITOR) a VideoRAM nebo RAM

Počítač typu MZ-800 obsahuje 64kB paměti RAM k volnému použití, nejméně 16kB VideoRAM, 12kB paměti ROM (MONITOR) a 4kB paměti ROM se znakovým generátorem (PCG). Dohromady činí paměťová kapacita nejméně 96kB. Jelikož procesor Z80 může přímo adresovat pouze 64kB paměti, musí existovat nějaký náhradní způsob adresování dalších částí paměti. Toto se děje metodou bankování paměti. Obsah určité paměťové oblasti je možné vyměnit, takže procesor dosahuje na stejné adrese do dvou různých paměťových oblastí.

Bankovací příkazy v režimu MZ-700 jsou kompatabilní s bankovacími příkazy počítače MZ-700, takže systémové programy MZ-700 je možné bez potíží využívat. Přepínaní bank se provádí adresováním určitého portu podle tabulky 4.1 .

Port	0000H - 0FFFH	D000H - FFFFH
E0H	RAM	-----
E1H	-----	RAM
E2H	ROM	-----
E3H	-----	VideoRAM, ROM
E4H	ROM	VideoRAM, ROM
E5H	-----	Přístup zakázán
E6H	-----	Stav před zákazem

Tab. 4.1 - Bankování paměti v režimu MZ-700.

VideoRAM v režimu MZ-700 se skládá ze dvou částí:

- znakové paměti, ve které jsou uloženy znaky zobrazené na obrazovce
- paměti barvy, která obsahuje příslušné barvy zobrazených znaků.

4.1 Stavba znakové paměti.

Znaková paměť se nachází na adresách D000H - D7FFH. Z tohoto rozsahu se na obrazovku zobrazuje pouze obsah adres D000H - D3E7H včetně. Zbytek paměti je opět volně k použití, například pro Editor, který obsluhuje dvě obrazovkové stránky a může mezi nimi rolovat a listovat tak, jak je tomu např. u BASICu MZ-700. Na obrazovku se však vždy přenáší jen rozsah paměti D000H - D3E7H. Rovněž zde neexistuje žádná možnost přepínání stránek jako v režimu MZ-800.

Obrazovka obsahuje 40x25 znaků. Ve VideoRAM jsou umístěny postupně zleva nahoře dolů doprava. K zakódování znaku je potřeba jeden bajt. Kódování se provádí podle videokódu.

	0	1	2	.....	38	39
0	D000H	D001H	D002H		D026H	D027H
1	D028H	D029H	D02AH			
.						
.						
.						
25	D3C0H	D3C1H	D3C2H		D3E6H	D3E7H

Tab. 4.2 - Obsah znakové VideoRAM.

MONITOR počítače MZ-800 pracuje vždy v režimu MZ-700 s přístupnou VideoRAM.

4.2 Stavba paměti barev.

Paměť barev leží na adresách D800H - DFFFH a přiřazení je následující:
- adresa D800H určuje barvu znaku na adrese D000H.
- adresa D801H určuje barvu znaku na adrese D001H atd..

Poslední adresa paměti barev, která se zobrazuje na obrazovce je D3E7H a odpovídá znaku na adrese D3E7H. Zbytek paměti je opět volně k použití.

V režimu MZ-700 se ze šestnácti možných barev využívá pouze osm. A sice, ačkoliv by se to nezdálo, těch světlejších, tedy barev 9 - 15. Pouze černá barva patří do skupiny tmavých, má kód 0. V režimu MZ-700 tedy barevný kód i odpovídá kódu 9 v režimu MZ-800 atd.. K zakódování barvy stačí pouze tři bity.

Barva je ve VideoRAM zakódována takto:
- dolní část (nibble) bajtu obsahuje barvu pozadí.
- horní část (nibble) bajtu obsahuje barvu popředí.
- horní bit dolní části bajtu (bit 3) nemá význam.
- horní bit horní části bajtu (bit 7) přepíná znakové generátory.

Z výše uvedených skutečností je zřejmé, že například kód 56H znamená znak azurové barvy (5) na žlutém pozadí (6) a kód 71H představuje bílý znak (7) na modrém pozadí (1) atd..

Přepínání mezi dvěma znakovými generátory umožňuje využívat ne pouze 256, ale dokonce 512 různých znaků. Je-li bit 7 v 0, využívá se první (základní) soubor znaků, je-li v 1, zobrazuje se podle druhého. Znaky které se zobrazují v MONITORU patří do prvního souboru. Druhý soubor obsahuje převážně znaky pro hry a jiné aplikace.

Všechny dosud popsané příkazy pro bankování a organizaci paměti existují i v počítači MZ-700. Počítač MZ-800 však nabízí v režimu MZ-700 určité grafické možnosti. Jedná se o takzvanou PCG grafiku, tedy o možnost volně programovat znakový generátor. Prakticky to znamená, že uživatel má možnost libovolně měnit znakovou předlohu zobrazitelných symbolů. Tyto grafické možnosti však nelze zaměňovat s grafikou s vysokým rozlišením (HRG).

Grafika s vysokým rozlišením znamená, že můžeme na obrazovce přímo rozsvěcovat nebo zhášet jednotlivé body. PCG grafika naproti tomu znamená, že je možno libovolně předefinovat znakový vzor každého symbolu. Změní-li se například vzor písmene A tak, že se místo něho má zobrazit UFO, objeví se po změně vzoru UFO na obrazovce všude tam, kde předtím bylo písmeno A. K realizaci PCG grafiky existují v počítači dvě paměti, které je možno bankováním připojit do hlavní paměti: CG - ROM a CG - RAM.

CG - RAM obsahuje vždy vzory, podle kterých se právě zobrazuje na obrazovce. V ní se také musí provádět všechny definice nových symbolů. CG - ROM obsahuje původní znakové vzory, které platí po zapnutí počítače nebo po stisknutí tlačítka RESET. Jak CG - ROM, tak i CG - RAM zabírají 4kB paměti. Pro stanovení znakové předlohy jednoho symbolu je zapotřebí osm bajtů. Jelikož máme 512 znaků, musí být pro ně k dispozici 512 x 8 = 4096 bajtu (4kB).

Každý jednotlivý symbol se skládá z 8 x 8 bodů. Klíčování se děje shora dolů, přičemž matrice jedné řádky je v jednom bajtu. Uvnitř bajtu se klíčuje zleva, přičemž bit 0 leží zcela vlevo.

STROJOVÝ KÓD	INSTRUKCE		OPERACE
DB20	IN	A, (n)	Vstup do A ze zařízení s V/V adresou n
EDAA	IND		Vstup do lokace (HL) ze zařízení s V/V adresou (C), dekrement HL a B
EDBA	INDR		Vstup do lokace (HL) ze zařízení s V/V adresou (C), dekrement HL a B a opakování, dokud B=0
EDA2	INI		Vstup do lokace (HL) ze zařízení s V/V adresou (C), inkrement HL a dekrement B
EDB2	INIR		Vstup do lokace (HL) ze zařízení s V/V adresou (C), inkrement HL a dekrement B a opakování, dokud B=0
C38405	JP	nn	Nepodmíněný skok na lokaci
E9	JP	(HL)	
DDE9	JP	(IX)	
FDE9	JP	(IY)	
DA8405	JP	C, nn	Skok na lokaci, je-li splněna podmínka
FA8405	JP	M, nn	
D28405	JP	NC, nn	
C28405	JP	NZ, nn	
F28405	JP	P, nn	
EA8405	JP	PE, nn	
E28405	JP	PO, nn	
CA8405	JP	Z, nn	
382E	JR	C, e	Relativní skok na PC+e, je-li splněna podmínka
302E	JR	NC, e	
202E	JR	NZ, e	
282E	JR	Z, e	
182E	JR	e	Nepodmíněný skok na PC+e
02	LD	(BC), A	Přesun obsahu ze zdrojového operandu do cílového operandu
12	LD	(DE), A	
77	LD	(HL), A	
70	LD	(HL), B	
71	LD	(HL), C	
72	LD	(HL), D	
73	LD	(HL), E	
74	LD	(HL), H	
75	LD	(HL), L	
3620	LD	(HL), n	
DD7705	LD	(IX+d), A	
DD7005	LD	(IX+d), B	
DD7105	LD	(IX+d), C	
DD7205	LD	(IX+d), D	
DD7305	LD	(IX+d), E	
DD7405	LD	(IX+d), H	
DD7505	LD	(IX+d), L	
DD360520	LD	(IX+d), n	

STROJOVÝ KÓD	INSTRUKCE		OPERACE
102E	DJNZ	e	Dekrement B a relativní skok, když B<>0
FB	EI		Povolení přerušení
E3	EX	(SP),HL	Výměna lokace a (SP)
DDE3	EX	(SP),IX	
FDE3	EX	(SP),IY	
08	EX	AF,AF'	Výměna obsahů AF a AF'
EB	EX	DE,HL	Výměna obsahů DE a HL
D9	EXX		Výměna obsahů BC,DE a HL s BC',DE',HL'
76	HALT		Zastavení programu - čekání na reset nebo přerušení
ED46	IM	0	Nastavení modu přerušení
ED56	IM	1	
ED5E	IM	2	
ED78	IN	A,(C)	Vstup ze zařízení s V/V adresou (C)
ED40	IN	B,(C)	
ED48	IN	C,(C)	
ED50	IN	D,(C)	
ED58	IN	E,(C)	
ED60	IN	H,(C)	
ED68	IN	L,(C)	
34	INC	(HL)	Inkrement operandu
DD3405	INC	(IX+d)	
FD3405	INC	(IY+d)	
3C	INC	A	
04	INC	B	
03	INC	BC	
0C	INC	C	
14	INC	D	
13	INC	DE	
1C	INC	E	
24	INC	H	
23	INC	HL	
DD23	INC	IX	
FD23	INC	IY	
2C	INC	L	
33	INC	SP	

A nyní k umístění CG - ROM a CG - RAM. CG - ROM leží na adresách 1000H až 1FFFH, CG - RAM na adresách C000H až CFFFH. Zapojení dvou paměťových oblastí se děje bankovacími příkazy. Narozdíl od jiných bankovacích příkazů se tentokrát jedná o vstupy z V/V portů.

Příkaz	1000H - 1FFFH	C000H - CFFFH
IN A,(E0H)	CG - ROM	CG - RAM
IN A,(E1H)	RAM	RAM

Tab. 4.3 - Bankování znakových generátorů.

Jak je vidět z tabulky, přepíná se CG - ROM a CG - RAM vždy současně.

Uspořádání znaků v CG - ROM a v CG - RAM je následující:
- pořadí znaku odpovídá videokódu znaku, to znamená, že znak s videokódem 2 následuje za znakem s videokódem 1 a ten opět následuje po znaku s videokódem 0.
- každý znak potřebuje pro uložení 8 bajtů.

Uložení v paměti je pak následující:

```
1000H - 1007H     kód   0 [mezera]     1. soubor
1008H - 100FH     kód   1 [znak A]     1. soubor
1010H - 1017H     kód   2 [znak B]     1. soubor
    .
    .
17F8H - 17FFH     kód 255              1. soubor
1800H - 1807H     kód   0              2. soubor
    .
    .
1FF8H - 1FFFH     kód 255              2. soubor
```

Je zřejmé, že znaky následují v paměti jeden za druhým bez mezer.
Organizace CG - RAM je stejná, začíná však na adrese C000H. Inicializace CG - RAM se provede okopírováním obsahu CG - ROM do CG - RAM.

Příklad: Předefinování znaku 2.

```
        LD HL,TBL           ;HL na startovací adresu tabulky
        LD DE,C010H         ;DE na začátek vzoru v CG - RAM
        LD BC,8             ;délka tabulky
        IN A,(E0H)          ;bankování (zapnutí) paměti CG
        LDIR                ;přesun tabulky
        IN A,(E1H)          ;bankování (vypnutí) CG
        JP EA5EH            ;teplý start MONITORU (každý studený start
                            ;MONITORU inicializuje CG - RAM podle CG - ROM
TBL:    DEFB 33H,33H,CCH,CCH,CCH,CCH,33H,33H
                            ;tabulka vzoru
```

Zvláštností režimu MZ-700 je, že komunikace s některými částmi počítače se neděje přes porty, ale prostřednictvím paměťového mapovaní I/O. Je to pozůstatek ze systému MZ-80K. Nevýhodou tohoto systému je, že odpovídající paměťový prostor (E000H - E008H) se musí bankovat s VideoRAM a MONITOREM MZ-800, takže např. části programů umístěné na adresách od F000H nemohou komunikovat s klávesnicí.

5. Režim MZ-800.

Jak je vidět již z kapitoly 3, je počítač v režimu MZ-800 vybaven všemi prostředky pro práci s grafikou s vysokým rozlišením. V tomto režimu pracuje počítač zcela jinak než v režimu MZ-700.

Hlavní paměť se skládá z následujících úseků:

- 0000H - 0FFFH .... MONITOR ROM nebo volná RAM
- 1000H - 1FFFH .... CG - ROM nebo volná RAM
- 2000H - 7FFFH .... volná RAM
- 8000H - BFFFH .... VideoRAM (HRG - RAM) nebo volná RAM
  Při bankování se délka VideoRAM mění následovně:
  - mód 320/200 ... 8000H - 9FFFH VideoRAM
    A000H - BFFFH volná paměť RAM
  - mód 640/200 ... 8000H - BFFFH VideoRAM
- C000H - DFFFH .... vždy volná RAM
- E000H - FFFFH .... MONITOR ROM nebo volná RAM

Bankování se provádí výstupem na port (OUT (n),A). Jsou možné následující bankovací příkazy:

Port	0000H - 0FFFH	1000H - 1FFFH	8000H - BFFFH	E000H - FFFFH
E0H	volná RAM	volná RAM	-----	-----
E1H	-----	-----	-----	volná RAM
E2H	ROM	-----	-----	-----
E3H	-----	-----	-----	ROM
E4H	ROM	CG - ROM	VideoRAM	ROM
E5H	-----	-----	-----	zakázaný přístup
E6H	-----	-----	-----	stav před zákazem

Tab. 5.1 - Bankování paměti v režimu MZ-800 výstupem na port.

Dále je možno provádět bankování vstupem z portu (IN A,(n)) podle následující tabulky:

Port	1000H - 1FFFH	8000H - BFFFH
E0H	CG - ROM	VideoRAM
E1H	volná RAM	volná RAM

Tab. 5.2 - Bankování paměti v režimu MZ-800 vstupem z portu.

Tak jako v režimu MZ-700, je i zde lhostejné, jaká hodnota se na port předává. Samotné adresování portu stačí k přepnutí paměti. V následujícím textu bude objasněna práce s HRG. Jak víme už z kapitoly 1, představuje barevné rozlišení (kód palety) každého grafického bodu překrytí několika obrazových bloků. Dalo by se tedy očekávat, že různé bankovací příkazy přepínají nejdříve blok I, pak blok II atd.. Takový přístup by měl pro uživatele celou řadu nevýhod. Kdyby měl například při rozlišení 320/200 v šestnácti barvách nastavit bod v paletě 15, musel by tento bod nastavit ve všech čtyřech grafických blocích. Taková práce s HRG by asi byla dosti pomalá.

STROJOVÝ KÓD	INSTRUKCE		OPERACE
3F	CCF		Komplement příznaku carry
BE	CP	(HL)	Porovnání operandu s A
DDBE05	CP	(IX+d)	
FDBE05	CP	(IY+d)	
BF	CP	A	
B8	CP	B	
B9	CP	C	
BA	CP	D	
BB	CP	E	
BC	CP	H	
BD	CP	L	
FE20	CP	n	
EDA9	CPD		Porovná lokaci (HL) s A, dekrementuje HL a BC
EDB9	CPDR		Porovná lokaci (HL) s A, dekrementuje HL a BC a opakuje dokud BC=0
EDA1	CPI		Porovná lokaci (HL) s A, inkrementuje HL a BC
EDB1	CPIR		Porovná lokaci (HL) s A, inkrementuje HL a BC a opakuje dokud BC=0
2F	CPL		Jednotkový doplněk A
27	DAA		Dekadické nastavení A
35	DEC	(HL)	Dekrement operandu
DD3505	DEC	(IX+d)	
FD3505	DEC	(IY+d)	
3D	DEC	A	
05	DEC	B	
0B	DEC	BC	
0D	DEC	C	
15	DEC	D	
18	DEC	DE	
1D	DEC	E	
25	DEC	H	
2B	DEC	HL	
DD2B	DEC	IX	
FD2B	DEC	IY	
2D	DEC	L	
3B	DEC	SP	
F3	DI		Zákaz přerušení

STROJOVÝ KÓD	INSTRUKCE		OPERACE
CB5C	BIT	3,H	
CB5D	BIT	3,L	
CB66	BIT	4,(HL)	
DDCB0566	BIT	4,(IX+d)	
FDCB0566	BIT	4,(IY+d)	
CB67	BIT	4,A	
CB60	BIT	4,B	
CB61	BIT	4,C	
CB62	BIT	4,D	
CB63	BIT	4,E	
CB64	BIT	4,H	
CB65	BIT	4,L	
CB6E	BIT	5,(HL)	
DDCB056E	BIT	5,(IX+d)	
FDCB056E	BIT	5,(IY+d)	
CB6F	BIT	5,A	
CB68	BIT	5,B	
CB69	BIT	5,C	
CB6A	BIT	5,D	
CB6B	BIT	5,E	
CB6C	BIT	5,H	
CB6D	BIT	5,L	
CB76	BIT	6,(HL)	
DDCB0576	BIT	6,(IX+d)	
FDCB0576	BIT	6,(IY+d)	
CB77	BIT	6,A	
CB70	BIT	6,B	
CB71	BIT	6,C	
CB72	BIT	6,D	
CB73	BIT	6,E	
CB74	BIT	6,H	
CB75	BIT	6,L	
CB7E	BIT	7,(HL)	
DDCB057E	BIT	7,(IX+d)	
FDCB057E	BIT	7,(IY+d)	
CB7F	BIT	7,A	
CB78	BIT	7,B	
CB79	BIT	7,C	
CB7A	BIT	7,D	
CB7B	BIT	7,E	
CB7C	BIT	7,H	
CB7D	BIT	7,L	
DC8405	CALL	C,nn	Volání podprogramu na lokaci nn,
FC8405	CALL	M,nn	je-li splněna podmínka
D48405	CALL	NC,nn	
C48405	CALL	NZ,nn	
F48405	CALL	P,nn	
EC8405	CALL	PE,nn	
E48405	CALL	PO,nn	
CC8405	CALL	Z,nn	
CD8405	CALL	nn	Nepodmíněné volání podprogramu

V módu MZ-800 se proto postupuje jinou cestou. Ve dvou registrech READ-formát registru (RF) a WRITE-formát registru (WF) uživatel stanoví jakým způsobem se čtecí a zapisovací operace s VideoRAM budou provádět. Vlastní nastavení nebo nulování bodu v určitém grafickém bloku bude vykonávat hardware.

Uživatel nepíše například při operaci typu:

```
LD A,FFH
LD (8000H),A
```

přímo do VideoRAM, tak jak tomu bylo v režimu MZ-700, nýbrž jen udá adresu VideoRAM, se kterou se má jeho předložený bitový vzor spojit. Způsob spojení závisí na obsahu WRITE-formát registru.

Podobně je to při čtecí operaci, neboť

```
LD A,(8000H)
```

neudá obsah VideoRAM na této adrese. Navrácená hodnota je mohoznačně závislá na obsahu registru RF. V závislosti na obsahu tohoto registru je možno z té samé adresy získat různé hodnoty.

Výhoda tohoto postupu je nasnadě; přístup hardwareově realizovanými příkazy jako PSET, RESET atd. je podstatně rychlejší. Navíc existuje možnost přenášet bez problémů obsah z jednoho grafického bloku do druhého, protože RF a WF registry jsou odděleny.

Obsah RF registru je následující:

```
bit     7     6     5     4     3     2     1     0          Port CDH
      +-----+-----+-----+-----+-----+-----+-----+-----+
      I R/S I     I     I A/B I IV  I III I II  I  I  I
      +-----+-----+-----+-----+-----+-----+-----+-----+
```

Římské číslice označují grafické bloky, kterých se týká čtecí operace. Je-li odpovídající bit nastaven, je dotazován příslušný grafický blok.

Vztah mezi kombinacemi grafických bloků a kódů palet je následující. Kód palety, která je představována určitou kombinací grafických bloků, je takové číslo, které se získá, když se vytváří bitový vzor pro dolní část (nibble) registru.

Příklad: Který kód palety určují bloky I + III ?
Odpověď: Podle výše uvedeného vzoru se získá kombinace 0101B = 5D. Ty body, které jsou nastavené jen v blocích I a III tedy představují paletu 5.
Příklad: Je třeba čtecí operací získat body palety 2. Jakým způsobem je třeba nastavit dolní část čtecího registru ?
Odpověď: Číslo 2 odpovídá binárnímu vzoru 0010B. Musíme tedy dolní část READ-formát registru nastavit kombinací 0010B.

Bit A/B určuje v režimu grafického rozlišení dostupný obraz. Je-li vynulován, je dostupný obraz A, je-li nastaven, je dostupný obraz B.
POZOR: bez grafického rozšíření musí být tento bit vždy vynulován.

Bit R/S rozlišuje čtení nebo vyhledávání (Read/Search). Je-li tento bit vynulován, bude se číst hodnota jednoho určitého grafického bloku. Tato možnost však v mnoha případech způsobuje problémy. Jelikož například paleta 3 je určena kombinací bloků I + III, bude bod nastavený v paletě 3 nastaven v grafickém bloku I i III. Budeme-li se nyní dotazovat na paletu i v bloku I, získáme i všechny body palety 3, protože se v bloku I vyskytují. Aby se všechny tyto falešné body vytřídily, muselo by se provést srovnání s ostatními grafickými bloky. Tuto práci vykoná zákaznický obvod sám, bude-li 7. bit registru READ-formát nastaven. Dotáže-li se nyní počítač na grafickou rovinu I, zákaznický obvod pozná, že se nejedná o grafický blok I, ale o paletu i a všechny body, které jsou nastaveny i v jiných grafických blocích automaticky neuvažuje.

Dříve, než začneme mluvit o registru WF, je nutné říci několik slov o organizaci VideoRAM. Obrazovka v režimu HRG je navržena jako bitová mapa. To znamená, že jeden nastavený bit ve VideoRAM reprezentuje jeden rozsvícený bod na obrazovce. Obsah obrazovky každého grafického bloku je na adresách 8000H až 9FFFH (640/200 : 8000H až BFFFH) a je adresován zleva nahoře doprava dolů. Adresa 8000H obsahuje tedy prvních osm grafických bodů se souřadnicemi 0/0 - 7/0.

Uvnitř jednoho bajtu se klíčuje každá osmice bodů zleva doprava, přičemž bit 0 je vždy nalevo.

Nyní se blíže podívejme na WF registr. Jak již bylo řečeno, tento registr určuje, jakým způsobem budou probíhat operace mezi daty ukládanými procesorem na určitou adresu a daty, která již na této adrese ve VideoRAM leží.

Obsah WF registru je následující:

bit	7	6	5	4	3	2	1	0	Port CCH
	WM2	WM1	WM0	A/B	IV	III	II	I	

Dolní část (nibble) WF registru opět určuje grafické bloky. Kódování je shodné s RF registrem.

A/B opět rozlišuje obrazy v režimech, ve kterých jsou dva obrazy k dispozici.
POZOR: Bez grafického rozšíření musí být tento bit vždy vynulován.

Důležitou roli hrají bity WM0 až WM2. Určují druh operace mezi daty dodávanými procesorem a daty již uloženými ve VideoRAM. Jsou to následující operace:

WM2	WM1	WM0	Operace
0	0	0	Single Write (jednoduchý zápis)
0	0	1	Exclusive - Or (EXOR)
0	1	0	OR
0	1	1	RESET
1	0	?	REPLACE
1	1	?	PSET

Tab. 5.3 - Operace mezi ukládanými a uloženými daty při zápisu do VideoRAM (? nabrazuje libovolnou hodnotu bitu WM0).

5.1 Operace Single Write (jednoduchý zápis).

Operace probíhá tak, že data předávaná programem do VideoRAM se skutečně uloží na specifikovaný datový blok a ostatní bloky zůstanou beze změny.

Příklad: Zápis

```
LD A,00000010B
OUT (CCH),A
LD A,FFH
LD (8000H),A
```

způsobí, že data FFH se uloží na adresu 8000H do druhého grafického bloku.

5.2 Operace Exclusive - Or (EXOR).

Provede se výlučný součet mezi daty předávanými programem do VideoRAM a daty, která leží ve specifikovaném grafickém bloku. Takto vzniklá data se do tohoto grafického bloku uloží. Ostatní bloky zůstanou beze změny.

Příklad: Zápis

```
LD A,00100011B
OUT (CCH),A
LD A,0FH
LD (8000H),A
```

způsobí, že se provede výlučný součet hodnoty 0FH s daty přečtenými z grafická-

STROJOVÝ KÓD	INSTRUKCE		OPERACE
A6	AND	(HL)	Logický součin operandu a akumulátoru
DDA605	AND	(IX+d)	
FDA605	AND	(IY+d)	
A7	AND	A	
A0	AND	B	
A1	AND	C	
A2	AND	D	
A3	AND	E	
A4	AND	H	
A5	AND	L	
E620	AND	n	
CB46	BIT	0, (HL)	Test bitu b v lokaci nebo registru
DDCB0546	BIT	0, (IX+d)	
FDCB0546	BIT	0, (IY+d)	
CB47	BIT	0,A	
CB40	BIT	0,B	
CB41	BIT	0,C	
CB42	BIT	0,D	
CB43	BIT	0,E	
CB44	BIT	0,H	
CB45	BIT	0,L	
CB4E	BIT	1, (HL)	
DDCB054E	BIT	1, (IX+d)	
FDCB054E	BIT	1, (IY+d)	
CB4F	BIT	1,A	
CB48	BIT	1,B	
CB49	BIT	1,C	
CB4A	BIT	1,D	
CB4B	BIT	1,E	
CB4C	BIT	1,H	
CB4D	BIT	1,L	
CB56	BIT	2, (HL)	
DDCB0546	BIT	2, (IX+d)	
FDCB0546	BIT	2, (IY+d)	
CB57	BIT	2,A	
CB50	BIT	2,B	
CB51	BIT	2,C	
CB52	BIT	2,D	
CB53	BIT	2,E	
CB54	BIT	2,H	
CB5E	BIT	3, (HL)	
DDCB055E	BIT	3, (IX+d)	
FDCB055E	BIT	3, (IY+d)	
CB5F	BIT	3,A	
CB58	BIT	3,B	
CB59	BIT	3,C	
CB5A	BIT	3,D	
CB5B	BIT	3,E	

Instrukční soubor Z-80

I STROJOVÝ KÓD	INSTRUKCE		OPERACE
I 8E	ADC	A, (HL)	Sečti operand a carry s akumulátorem
I DD8E05	ADC	A, <IX+d)	
I FD8E05	ADC	A, (IY+d)	
I 8F	ADC	A, A	
I 88	ADC	A, B	
I 89	ADC	A, C	
I 8A	ADC	A, D	
I 8B	ADC	A, E	
I 8C	ADC	A, H	
I 8D	ADC	A, L	
I CE20	ADC	A, n	
I ED4A	ADC	HL, BC	Sečti operand a carry s HL
I ED5A	ADC	HL, DE	
I ED6A	ADC	HL, HL	
I ED7A	ADC	HL, SP	
I 86	ADD	A, (HL)	Sečti operand s akumulátorem
I DD8605	ADD	A, (IX+d)	
I FD8605	ADD	A, (IY+d)	
I 87	ADD	A, A	
I 80	ADD	A, B	
I 81	ADD	A, C	
I 82	ADD	A, D	
I 83	ADD	A, E	
I 84	ADD	A, H	
I 85	ADD	A, L	
I C620	ADD	A, n	
I 09	ADD	HL, BC	Sečti registrový pár s HL
I 19	ADD	HL, DE	
I 29	ADD	HL, HL	
I 39	ADD	HL, SP	
I DD09	ADD	IX, BC	
I DD19	ADD	IX, DE	
I DD29	ADD	IX, IX	
I DD39	ADD	IX, SP	
I FD09	ADD	IY, BC	
I FD19	ADD	IY, DE	
I FD29	ADD	IY, IY	
I FD39	ADD	IY, SP	

ho bloku I a výsledek se do tohoto bloku opět uloží. Stejná operace se provede v grafickém bloku II. Výsledek bude:

blok I: (8000H):=(8000H) EXOR 0FH
blok II: (8000H):=(8000H) EXOR 0FH

5.3 Operace OR.

Provede se logický součet mezi daty předávanými programem do VideoRAM a daty, která leží ve specifikovaném grafickém bloku. Takto vzniklá data se do tohoto grafického bloku uloží. Ostatní bloky zůstanou beze změny.

Příklad: Zápis
```
LD A,01000011B
OUT (CCH),A
LD A,F0H
LD (8000H),A
```

způsobí, že se provede logický součet hodnoty F0H s daty přečtenými z grafického bloku I a výsledek se do tohoto bloku opět uloží. Stejná operace se provede v grafickém bloku II. Výsledek bude:

blok I: (8000H):=(8000H) OR F0H
blok II: (8000H):=(8000H) OR F0H

5.4 Operace RESET.

Provede se nulování jednotlivých bitů ve specifikovaném grafickém bloku. Všechny bity nastavené v předávaných datech budou v tomto bloku vynulovány. Ostatní bity nebudou ovlivněny.

Příklad: Zápis
```
LD A,01100011B
OUT (CCH),A
LD A,00010001B
LD (8000H),A
```

způsobí, že bity 0 a 4 budou v grafických blocích I a II vynulovány. Všechny ostatní bity v těchto blocích a ostatní grafické bloky nebudou ovlivněny.

5.5 Operace REPLACE.

Tato operace umožňuje zápis grafických dat, jejichž barva je určena číslem palety. Předávaná data se zapíší do specifikovaných grafických bloků, ostatní bloky jsou na této adrese vynulovány.

Příklad: Zápis
```
LD A,10000010B
OUT (CCH),A
LD A,11001100B
LD (8000H),A
```

způsobí, že obsah adresy 8000H v grafickém bloku I bude vynulován a na stejnou adresu grafického bloku II budou zapsána předávaná data. V závislosti na zobrazovacím režimu se mohou nulovat i ostatní bloky (v režimu 320/200 - 16 barev se navíc nulují obsahy adres 8000H v grafických blocích III a IV. Rozhodující je skutečnost, které bloky jsou programově přístupné.

5.6 Operace PSET.

Tato operace umožňuje nastavit jednotlivé body, jejichž barva je určena číslem palety. Nastavené bity předávaných dat se zapíší do specifikovaných grafických bloků a v ostatních blocích se tyto bity vynulují. Bity, které nejsou v předávaných datech nastaveny, zůstanou beze změny.

Příklad: Zápis

```
LD A,11000010B
OUT (CCH),A
LD A,00001000B
LD (8000H),A
```

způsobí, že bit 3 na adrese 8000H bude v bloku II nastaven a v ostatních blocích vynulován.

5.7 Další poznámky k používání WF a RF registrů.

- grafické bloky v režimu 640/200 - 4 barvy se číslují I a III, nikoli I a II. Zapsat v tomto režimu do palety 3 tedy znamená zapsat do bloků I a III.
- v režimech, ve kterých jsou k dispozici dva obrazy (A a B), mají grafické bloky svá normální označení.

  320/200 - 4 barvy: obraz A - bloky I, II
  obraz B - bloky III, IV

  640/200 - 1 barva: obraz A - blok I
  obraz B - blok III

  Má-li se například zapsat do obrazu B v režimu 320/200 paletou 3, musí být nastaven bit A/B a pracuje se s bloky III a IV.
- v režimu MZ-700 se musí WF a RF registr také naprogramovat. To vyplývá ze skutečnosti, že grafický blok I má dvojí uplatnění. V režimu MZ-700 obsahuje CG - RAM, znakovou paměť a paměť barev (atributů). V režimu MZ-800 pracuje jako grafický blok. Proto je třeba oba tyto registry nastavit v režimu MZ-700 na hodnoty 01H.

Potud teorie a nyní trochu praxe.

Napíšeme program, který vymaže obrazovku a do levého horního rohu zapíše znak A. Zobrazovací režim bude 320/200 - 4 barvy a znak se zobrazí v paletě 3.

```
        ORG 5000H
        LD A,0
        OUT (CEH),A         ;mód 320/200
        LD A,3
        OUT (CCH),A         ;WF registr
        OUT (CDH),A         ;RF registr
        LD HL,8000H         ;začátek VideoRAM
        LD DE,8001H         ;HL+1
        LD BC,1F3FH         ;délka VideoRAM
        IN A,(E0H)          ;bankování VideoRAM
        LD (HL),0
        LDIR                ;výmaz obrazovky
        LD HL,8000H
        LD DE,1008H         ;adresa vzoru písmene A v CG - ROM
        LD BC,40H           ;řádkové posunutí
        EXX
        LD B,8              ;8 bajtů
        LD A,83H            ;operace REPLACE
        OUT (CCH),A         ;do WF registru
LBL1:   EXX
        LD A,(DE)           ;vzor do akumulátoru
        LD (HL),A           ;a do VideoRAM
        INC DE              ;připrav další vzor
```


Tabulka operací s příznaky:

INSTRUKCE	C	Z	IP/V	S	N	H	POZNÁMKY
ADD A,s; ADC A,s	:	:	V	:	0	:	8-bit. sčítání nebo sčítání s C
SUB s; SBC A,s	:	:	V	:	1	:	8-bit. odečítání, odec. s C
CP s; NEG	:	:	V	:	1	:	Komparace a negace akumulátoru
AND s	0	:	P	:	0	1	Logické operace
OR s; XOR s	0	:	P	:	0	0	AND nastavuje flagy jinak
INC s	*	:	V	:	0	:	8-bit. inkrement
DEC m	*	:	V	:	1	:	8-bit. dekrement
ADD DD,ss	:	*	*	*	0	X	16-bit. sčítání
ADC HL,ss	:	:	V	:	0	X	16-bit. sčítání s C
SBC HL,ss	:	:	V	:	1	X	16-bit. odečítání s C
RLA;RLCA;RRA;RRCA	:	*	*	*	0	0	rotace akumulátoru
RL m;RLC m;RR m	:	:	P	:	0	0	rotace a posuvy lokace m
RRC m;SLA m;SRA m							
SRL m							
RLD;RRD		*	P	:	0	0	rotace čísla doleva a doprava
DAA	:	*	P	:	*	:	dekadické nastavení akumulátoru
CPL	*	*	*	*	1	1	komplement akumulátoru
SCF	1	*	*	*	0	0	nastavení carry
CCF	:	*	*	*	0	X	komplement carry
IN r,(C)	*	:	P	:	0	0	nepřímý vstup do registru
INI;IND;OUTI;OUTDI	*	:	X	X	1	X	vstup/výstup bloku
INIR;INDR;	*	1	X	X	1	X	Z=0 když B=0, jinak Z=1
OTIR;OTDR							
LDI;LDD	*	X	:	0	0	0	instrukce přenosu bloku
LDIR;LDDR	*	X	0	X	0	0	P/V=1 když BC<>0, jinak P/V=0
CPI;CPIR;CPD;CPDR	*	:	:	:	1	X	instrukce prohledávání bloku
							Z=1 když A=(HL) jinak Z=0
							P/V=1 když BC<>0 jinak P/V=0
LD A,I; LD A,R	*	:	IFF	:	0	0	obsah klopného obvodu povolení
							přerušení se nahraje do P/V
BIT b,s	*	:	X	X	0	1	komplement bitu b lokace s
							se nahraje do flagu Z
NEG	:	:	V	:	1	:	negace akumulátoru

Dodatek B:

Tabulka kódu Z-80.

Použité notace:

C	Carry flag. C=1, jestliže operace produkuje přenos z nejvyššího bitu operandu výsledku.
Z	Zero flag. Z=1 je-li výsledek operace nulový.
S	Sign flag. S=1 jeli nejvyší bit výsledku 1.
P/V	Parity nebo overflow. Logické operace ovlivňují tento bit paritou výsledku a aritmetické operace s přetečením (overflow) výsledku. Jde-li o paritu, je P/V=1, když je parita výsledku sudá. Jde-li o přetečení, je P/V=1, když operace produkuje přetečení.
H	Half-carry (poloviční carry). H=1, když sčítání nebo odčítání produkuje přenos do 4. bitu nebo výpůjčku (borrow) ze 4. bitu akumulátoru.
N	Příznak sčítání/odečítání. N=1, když předchozí operace byla odečítání. H a N se používají ve spojení s instrukcí DAA - dekadické nastavení akumulátoru pro správné korigování výsledků do formátu BCD
:	příznak je ovlivněn podle výsledku operace
*	příznak je operací neovlivněn
0	příznak je operací vynulován
1	příznak je operací nastaven
X	nastavení příznaku je nezajímavé
V	P/V je ovlivňován přetečením výsledku operace
P	P/V je ovlivňován paritou výsledku operace
r	libovolný registr A, B, C, D, E, H, L
s	libovolná 8-bitová lokace pro všechny adresní režimy
ss	libovolná 16-bitová lokace pro všechny adresní režimy
ii	indexní registr IX nebo IY
R	registr obnovení
n	8-bitové číslo <0-255>
nn	16-bitové číslo <0-65535>
m	libovolná 8-bitová lokace pro všechny adresní režimy dovolené pro instrukci
b	jeden z osmi bitů registru


```
        ADD HL,BC           ;a adresu jeho umístění
        EXX
        DJNZ LBL1
        LD A,11H            ;PAL 1 na modrou barvu
        OUT (FOH),A
        LD A,22H            ;PAL 2 na červenou barvu
        OUT (FOH),A
        LD A,3FH            ;PAL 3 na bílou barvu
        OUT (FOH),A
        XOR A               ;PAL 0 na černou barvu
        OUT (FOH),A
        HALT
```

Až dosud se mluvilo o vytváření kódů palet, ale nikoliv o tom jak jednotlivým kódům přiřadit různé barvy. Jak je známo, mohou se v módu MZ-800 přiřadit jednotlivým kódům palet libovolné barvy.

Přiřazení barev ke kódům palet zprostředkovává port FOH. Rozdělení bitů tohoto portu je závislé na hodnotě bitu 6 v předávaných datech. Je-li tento bit vynulován, provádí operace s portem FOH přiřazení zvolených barev v rámci předem definované skupiny palet. Význam jednotlivých bitů je pak následující:

```
bit    7   6    5    4   3   2   1   0                    Port FOH
     +---+---+----+----+---+---+---+---+
     I   I 0 I Si I SO I I I G I R I B I
     +---+---+----+----+---+---+---+---+
```

Bity SO a Si určují číslo barvy v rámci zvolené skupiny palet. Dolní část (nibble) dat určuje barvu složenou ze složek RGBI. Každá skupina palet obsahuje čtyři barvy.

Volba skupiny, v níž bude probíhat nastavování jednotlivých barev se provádí opět voláním portu FOH. V tomto případě však musíme bit 6 v předávaných datech nastavit. Význam jednotlivých bitů je pak následující:

```
bit    7   6   5   4   3   2    1     0                   Port FOH
     +---+---+---+---+---+---+-----+-----+
     I   I 1 I   I   I   I . I SWi I SWO I
     +---+---+---+---+---+---+-----+-----+
```

Bity SWO a SWi nyní určují číslo palety v níž bude možno výše popsaným způsobem nastavovat barvy. Číslování barev probíhá ve všech skupinách palet vždy v rozsahu 0 až 3.

Příklad: Zápis

```
        LD A,36H
        OUT (FOH),A
```

způsobí nastavení palety 3 na barvu 6 (žlutá).

Příklad: Zápis

```
        LD A,40H
        OUT (FOH),A
```

způsobí připojení nulté čtveřice palet a je tedy možno paletám 0 až 3 přiřadit barvy.

Příklad: Má se nastavit paleta 5 na barvu 15. Paleta 5 je druhá paleta druhé skupiny, tedy paleta 1 skupiny 1. Program bude vypadat takto:

```
        LD A,41H            ;skupina 1
        OUT (FOH),A
        LD A,1FH            ;barva 15, paleta 1 z 1. skupiny
        OUT (FOH),A
```

5.8 Hardwareové rolování.

Další zajímavou vlastností počítače MZ-800 je, že jeho řadič zobrazení má technické prostředky pro rolování obrazu. Možnosti tohoto rolování jsou předvedeny v ukázkovém programu DEMO 800. Libovolný úsek obrazovky může být rolován nahoru i dolů. Řízení tohoto rolování se provádí prostřednictvím portů. Pro rolování jsou důležité následující registry:

SSA - (scroll start registr) - určuje od kterého obrazového řádku (včetně) se bude rolovat. Obrazovými řádky se rozumí celé znakové řádky (0 - 24). Před předáním dat do SSA se musí jejich hodnota ještě vynásobit pěti.
Příklad: Má-li se rolovat od řádku 0 do registru SSA se nahraje hodnota 0 x 5 = 0.

SEA - (scroll end registr) - určuje od kterého obrazového řádku (včetně) se rolování již nebude provádět. Tak jako u SSA se zde musí hodnota vynásobit pěti. Zadávají se řádky 1 - 25 (25 tehdy, když se má rolovat včetně řádku 24)
Příklad: Má-li se rolovat do 22. řádku včetně do SEA registru se nahraje hodnota 23 x 5 = 115.

SW - (scroll width) - do tohoto registru se nahrává rozdíl hodnot SEA a SSA.
Příklad: Je-li SSA = 5 a SEA = 115, do SW se nahrává hodnota 115 - 5 = 110.

SOF - (scroll offset) - tímto registrem se zadává vlastní příkaz pro rolování. Rolování se provede tehdy, když se změní obsah SOF. Zvýšení hodnoty způsobí rolování nahoru, snížení hodnoty rolování dolů. I zde se musí data zadávat s krokem pět. Změna hodnoty SOF o pět způsobí rolování o jeden mikrořádek. Je-li změna hodnoty menší než pět roluje se vodorovně po pětinách obrazovky.

Zápis do registru se děje prostřednictvím nepřímé instrukce výstupu OUT (C),A. Adresy portů jsou následující:

01CFH	SOF	(dolních 8 bitů)
02CFH	SOF	(horních 8 bitů)
03CFH	SW	(7 bitů)
04CFH	SSA	(7 bitů)
05CFH	SEA	(7 bitů)

Příklad: Napišme program, který bude pomalu rolovat obrazovkou nahoru. Rolování bude probíhat po řádcích HRG. Hodnota SOF se tedy musí vždy měnit o 5. Kdybychom chtěli rolovat o dva řádky HRG musela by se hodnota SOF měnit o 10.

Je tu ještě jeden problém. SOF se nemůže zvyšovat stále, protože by došlo k přetečení. Protože rolování obsah obrazovky nemění (řádky, které se nahoře ztrácí se dole znova objevují a naopak), lze přetečení zabránit nahrazením rolování o jeden řádek nahoru, rolováním o 199 řádků dolu. SOF bude mít opět nízkou hodnotu a na obrazovce nebude patrný žádný rozdíl. Program bude vypadat následovně:

```
        ORG 5000H
        XOR A
        OUT (CEH),A         ;320/200 - 4 barvy
LBL1:   CALL WAIT           ;zpoždění
        CALL SCROLL         ;rolování
        JP LBL1
WAIT:   LD HL,1000H
LBL2:   DEC HL
        LD A,H
        OR L
        JP NZ,LBL2
        RET
SCROLL: LD HL,(SOF)
        LD DE,5             ;offset
        ADD HL,DE
        PUSH HL             ;SOF na zásobník
        LD DE,1000          ;nový SOF = 200 x 5 = 1000
        AND A               ;reset carry flag
```


Dodatek A:

Tabulka kódu ASCII

		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	-	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	=	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	;
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	VS	/	?	O	_	o	DEL

Seznam použité literatury:

1. Personal Computer MZ-800 Owner's Manual - SHARP Corporation Osaka, Japan
2. SHARP Service Manual - Personal Computer Model MZ-800
3. Rodnay Zaks: Programming Z80, Sybex
4. Torsten Ziegler: Systemprogramierung auf MZ-800, MZ-Verlag
5. Bialke, Berendsen, Gliszczynski: Alles uber den MZ-800, BBG Software
6. Výpis Monitoru MZ-700
7. Steve Harrington: Computer Graphics, McGraw Hill


```
        SBC HL,BE
        POP HL
        JR NZ,LBL3          ;přeskoč, není-li třeba SOF snižovat
        LD HL,0             ;SOF = 0
LBL3:   LD (SOF),HL
        LD HL,SOF
        LD BC,06CFH
        OTDR
        RET
SOF:    DEFW 0              ;offset
        DEFB 125            ;SW
        DEFB 0              ;SSA
        DEFB 125            ;SEA
        END
```

5.9 Nastavení barvy okraje (border).

Počítač MZ-800 umožňuje nastavení barvy okraje obrazovky (border). Děje se to prostřednictvím čtyřbitového registru, jehož obsah je následující:

bit 7 6 5 4 3 2 1 0 Port 6CFH

I	I	I	I	I	G	R	B

Význam jednotlivých bitů je stejný jako u registru palet a je zřejmé, že je k dispozici 16 barev okraje obrazovky. Zápis do registru se děje prostřednictvím nepřímé instrukce výstupu OUT (C),A. Po zapnutí počítače nebo po resetu jsou všechny bity tohoto registru nastaveny na hodnotu 0 (černá barva).

6. I/O operace v režimech MZ-700 a MZ-800.

6.1 Obvod 8255.

Počítač MZ-800 používá obvod 8255 ke komunikaci s klávesnicí, dvěma ovladači a kazetovým magnetofonem. V závislosti na režimu probíhá datový přenos mezi obvodem a procesorem buď jako paměťově adresovaný nebo jako periferně adresovaný.

MZ-700	MZ-800	Funkce
E000H	D0H	Port A - výstup
E001H	D1H	Port B - vstup
E002H	D2H	Port C - vstup/výstup
E003H	D3H	Řídící registr

Tab. 6.1 Adresy obvodu 8255

Port	Bit	Aktivní	I/O	Funkce
PA	0-3	1	O	Vzorkování klávesnice
	4	0	O	Vzorkování ovladače 1
	5	0	O	Vzorkování ovladače 2
	7	0	O	Nulování časovače blikání kursoru
PB	0-7	0	I	Data z klávesnice
PC	0	0	O	Blokování tónového výstupu z 8253
	1	-	O	Data pro zápis na magnetofon
	2	0	O	Zákaz přerušení z časovače
	3	0-1-0	O	Řízení motoru magnetofonu
	4	1	I	Testování motoru magnetofonu
	5	-	I	Data čtená z magnetofonu
	6	-	I	Časovač kursoru
	7	-	I	Vertikální zatemnění

Tab. 6.2 Jednotlivé bity obvodu 8255

Bity 0-3 portu A slouží k vzorkování klávesnice. Těmito bity se udává číslo prohledávaného sloupce matice spínačů klávesnice. Platí pouze hodnoty 0 - 9.

Bity 4 a 5 portu A slouží ke vzorkování ovladačů. Vzorkování je aktivní v případě, že příslušný bit je vynulován.

Bit 7 portu A slouží k nulování časovače blikání kursoru.

Bity 0-7 portu B slouží k přebírání stavu vzorkovaného sloupce klávesnice.

Vynulování bitu 0 portu C způsobí zablokování tónového výstupu obvodu 8253.

Přes bit 1 portu C se předávají data do magnetofonu.

Nulováním bitu 2 portu C se zamezí přerušení dvanáctihodinového časovače pro přepnutí AM/PM (dopoledne/odpoledne).

Prostřednictvím bitu 3 portu C se provádí ovládání motoru magnetofonu. Vytvořením impulsu přechodem mezi stavy 0-1-0 na tomto výstupu se mění stav motoru magnetofonu; otáčel-li se pak se zastaví, a naopak.

Prostřednictvím bitu 4 portu C se testuje stav motoru magnetofonu. Je-li tento bit nastaven, pak se motor otáčí.

Přes bit 5 portu C se získávají data z magnetofonu.

Bit 6 portu C zobrazuje stav časovače blikání kursoru. Tento stav se periodicky mění mezi 0 a 1, tak jak bliká kursor.

Bit 7 portu C umožňuje testovat aktivní stav signálu vertikalního zatemně-

bit	7	6	5	4	3	2	1	0
	1	1	adresa registru		u3	u2	u1	u0

Parametry u0 až u3 tvoří binární hodnotu, která po vynásobení 2 určuje hodnotu útlumu výstupního signálu příslušného generátoru v dB. Je-li tato hodnota nulová, má výstupní signál maximální hlasitost. Při hodnotě 15 dojde k vypnutí výstupu příslušného generátoru.

Výstupní signál má obdélníkový průběh se střídou 1 :1. Hodnota N určuje dělicí poměr vnitřního děliče a tím i kmitočet výstupního signálu. Protože se jedná o desetibitovou hodnotu, je nutné ji rozdělit do dvou bajtů. Toto dělení se provádí následovně:

bit	7	6	5	4	3	2	1	0
1. bajt	1	1	adresa registru		n3	n2	n1	n0
2. bajt	1	0	1	-	n9	n8	n7	n6	n5	n4

Parametry n0 až n9 představují jednotlivé bity dělicího poměru N pro vnitřní dělič kmitočtu.

7.2 Řízení generátoru šumu.

Pro řízení generátoru šumu postačuje jeden bajt dat. Jeho vnitřní složení je následující:

bit	7	6	5	4	3	2	1	0
	1	1	1	1	1	0	-	T	Fi	F0

Parametr T určuje typ šumu. Má-li hodnotu nula, jedná se o synchronní šum, jinak se jedná o takzvaný bílý šum. Parametry F0 a Fi určují frekvenci šumu podle následující tabulky:

Fi	F0	frekvence
0	0	6,93 kHz
0	1	3,47 kHz
1	0	1,73 kHz
1	1	řízena výstupem tónového generátoru 2

Tab. 7.2 - Řízení frekvece šumu.

Synchronní šum má tvar obdélníkového signálu se střídou 1 : 16, kdežto průběh bílého šumu má střídu a frekvenci proměnnou, ovlivňovanou generátorem náhodných čísel. Je-li frekvence šumu řízena výstupem tónového generátoru 2, musí být útlum tohoto kanálu nastaven na maximální hodnotu (vypnut).

7.3 Nastavení útlumu tónu nebo šumu.

Nastavení útlumu je pro všechny zvukové generátory prováděno stejným způsobem. Hodnota útlumu se předává do příslušného registru pomocí jednoho bajtu dat, který má následující složení:

ní. V praktickém provozu se tento bit nevyužívá; je to pozůstatek z počítače MZ-80K.

Následující část popisu se týká obsluhy klávesnice a magnetofonu.

Příklad: Program má v režimu MZ-700 zjistit, zda je stisknuto tlačítko [CTRL]. Je-li stisknuto, zobrazí se na obrazovce C, není-li stisknuto, pak se zobrazí mezera. Progam skončí při stisknutí tlačítka [F1].

```
        ORG 5000H
LBL1:   LD A,8              ;CTRL je ve sloupci 8
        LD (E000H),A
        LD A,(E001H)        ;vzorkování klávesnice
        BIT 6,A             ;CTRL je v řádku 6
        LD A,67             ;znak C
        JR Z,LBL2           ;test stisknutí
        LD A,32             ;mezera
LBL2:   CALL 0012H          ;výpis pomocí podprogramu z monitoru
        LD A,9              ;F1 je ve sloupci 9
        LD (E000H),A
        LD A,(E001H)
        BIT 7,A             ;F1 je v řádku 7
        JP Z,EA5EH          ;návrat do monitoru
        JP LBL1
```

Příklad: Následující podprogram má zastavit motor magnetofonu, pokud se otáčí. Jelikož to nejde provést jednoduchým příkazem, musí se nejprve testovat zda se motor točí a pouze v tomto případě ho zastavovat.

```
        ORG 5000H
        LD A,(E002H)        ;čtení portu C
        BIT 4,A             ;test chodu motoru
        RET Z               ;ne; konec
        LD HL,E002H         ;nastavení adresy portu C do HL
        RES 3,HL
        SET 3,HL
        RES 3,HL            ;impuls pro zastavení
        RET
```

Poznámka: podprogram je nutné volat instrukcí CALL, nelze ho spustit přímo vzhledem k zakončení instrukcí RET.

Příklad: Program má provádět zvukové echo dat čtených z magnetofonu na reproduktor. Bude pracovat podle následující myšlenky: Bude se testovat bit 5 portu C, jehož prostřednictvím se získávají informace z magnetofonu a podle stavu tohoto bitu se buď tón spustí nebo zastaví. Tímto způsobem lze reprodukovat obsah magnetofonové pásky. Pro vlastní tvorbu tónu se využívají podprogramy z monitoru počítače. Vytváří se nejvyšší možný tón, protože při nižších frekvencích je větší ztráta informace.

```
        ORG 5000H
        LD DE,E002H         ;adresa portu C
LBL1:   LD HL,0100H         ;výška tónu
        LD A,(DE)           ;čtení portu C
        BIT 5,A             ;tón L/H ?
        JR Z,LBL2
        CALL 02AEH          ;tón spustit
        JP LBL1
LBL2:   CALL 0047H          ;tón zastavit
        JP LBL1
```

Poznámka: tento program lze přerušit pouze stiskmutím tlačítka RESET.
V monitoru počítače MZ-800 existují dva podprogramy pro řízení zvukového generátoru:
0044H - start tónu jehož frekvence je určena obsahem adres 11A1H/11A2H.
0047H - zastavení tónu.
V uvedeném příkladu je místo podprogramu 0044H použit jiný s adresou 02AEH, který získává frekvenci z obsahu HL registru.

6.1.1 Řízení ovladačů.

Jak je možné se dočíst v uživatelské příručce, pro počítač MZ-800 se dá použít ovladač (joystick) kompatabilní s ovladačem ATARI. MZ-800 však umožňuje připojení ovladačů ATARI pouze jako jednoho z možných zařízení, přičemž se nevyužívá všech možností, které jsou k dispozici. Ovladač ATARI totiž využívá pouze pět vývodů konektoru, kterých je však k dispozici devět. Jsou to následující vývody:

Vývod	Signál	Funkce	Poznámka
1	FWD	Tlačítko pro směr vpřed	D0
2	BACK	Tlačítko pro směr zpět	D1
3	LEFT	Tlačítko pro směr vlevo	D2
4	RIGHT	Tlačítko pro směr vpravo	D3
5	+5V	Napájení externího zařízení	
6	TRIG1	Pomocné tlačítko 1	D4
7	TRIG2	Pomocné tlačítko 2	D5
8	COM	Vzorkovací signál	PA4/PA5
9	GND	Napájení externího zařízení	

Tab. 6.3 - Zapojení konektorů ovladačů.

Ovladač ATARI využívá pouze vývody 1, 2, 3, 4, 6, 8. I v případě, že tento ovladač obsahuje dvě pomocná tlačítka, nelze je využívat samostatně, protože obě jsou zapojena na vývod 6. Signál COM (A/B) slouží ke vzorkování tlačítek. Vzorkovací signály pro oba ovladače jsou vytvářeny obvodem 8255 a to v případě signálu COM A vývodem PA4 a v případě signálu COM B vývodem PA5.

Z tabulky je zřejmé, že je možno připojit i jiné typy ovladačů. Tyto ovladače mohou obsahovat dvě samostatná pomocná tlačítka, nemusí používat vzorkování signálem COM (tlačítka mohou být připojena na signál GND) a mohou obsahovat i aktivní obvody napájené z vývodu 5. Navíc je nutno konstatovat, že vhodnou programovou obsluhou je možno dosáhnout i odlišné funkce pro připojení jiných typů vnějších zařízení.

Programové ovládání signálu COM A, COM B bylo popsáno v předchozí kapitole, je nutno pouze poznamenat, že pro ovladače jsou tyto signály aktivní ve stavu 0. Vstupní informace jsou dostupné pomocí vstupních portů F0H a F1H a to v obou režimech funkce počítače (MZ-700/MZ-800). Přes port F0H je přístupný ovladač 1, přes port F1H ovladač 2. Data čtená z obou těchto portů mají následující formát:

Bit	7	6	5	4	3	2	1	0	Port F0H/F1H
	I	I	TRIG2	TRIG1	RIGHT	LEFT	BACK	FWD	

7. Programovatelný zvukový generátor.

Počítač MZ-800 má bohatou zásobu příkazů v jazyce BASIC, přičemž se nezapomělo ani na ovládání programovatelného zvukového generátoru, který je v počítači zabudován. Pomocí těchto příkazů lze v určitých mezích tento generátor snadno programovat. Musíme však upozornit na chybu v uživatelské příručce, kde je na straně 6-72 uvedeno, že pomocí příkazu NOISE lze generovat dvě části šumu, jejichž popis je oddělen středníkem. Použitý generátor však může generovat pouze jeden šum, takže se nelze divit tomu, že příkazem NOISE nedosáhneme požadovaného efektu. Tímto příkazem je generován takzvaný bílý šum. Později se dozvíme, že lze generovat také jiný typ šumu ale není k tomu možné použít příkaz NOISE.

Programovatelný zvukový generátor (PSG) je přístupný na výstupní adrese F2H (242). Jedná se o obvod typu SN 76489AN a tento obvod obsahuje osm vnitřních registrů. Každá dvojice registrů je určena k řízení jednoho zvukového kanálu. Tři tyto kanály jsou identické a je pomocí nich možno generovat tříténový akord v rozsahu šesti oktáv. Čtvrtý kanál je určen ke generování šumu a jeho ovládání se poněkud liší.

První z dvojice registrů každého kanálu určuje kmitočet tónu nebo šumu. Nastavení kmitočtu pro tónové generátory vyžaduje desetibitovou hodnotu a proto se provádí pomocí dvou bajtů. Šumový generátor má odlišný způsob řízení kmitočtu a stačí proto jediný bajt k jeho nastavení.

Druhý registr každého kanálu slouží k řízení útlumu signálu na výstupu příslušného kanálu. Toto řízení je pro všechny kanály shodné.

Pro rozlišení zda se jedná o první nebo druhý datový bajt se používá bit D7 v datovém slově. Je-li nastaven, jedná se o první (nebo jediný) datový bajt, je-li vynulován, jde o druhý datový bajt a je určen k řízení posledně adresovaného registru. Adresování registrů se provádí pomocí bitů D4 až D6 v prvním datovém bajtu. Toto adresování se provádí následujícím způsobem:

D6	D5	D4	Funkce
0	0	0	frekvence tónu 0
0	0	1	útlum tónu 0
0	1	0	frekvence tónu 1
0	1	1	útlum tónu 1
1	0	0	frekvence tónu 2
1	0	1	útlum tónu 2
1	1	0	řízení šumu
1	1	1	útlum šumu

Tab. 7.1 - Adresování registrů PSG.

7.1 Nastavení frekvence tónu.

Frekvence tónu je dána vztahem:

$$f_{výst} = \frac{f_{vst}}{32 * N} \quad \text{kde } f_{vst} = 3.55 * 10^3 \text{ kHz}$$

6.2 Obvod 8253.

Jedná se o programovatelný obvod, který obsahuje 3 samostatné čítače/časovače. Funkce jednotlivých čítačů jsou závislé na volbě režimu, který se nahraje do řídícího registru.

Obvod je přístupný na adresách:

režim		registr
MZ-700	MZ-800	
E004H	D4H	čítač 0
E005H	D5H	čítač 1
E006H	D6H	čítač 2
E007H	D7H	stavový a řídící registr

Tabulka 6.4 Adresy obvodu 8253

Čítač 0 tvoří dělič základní frekvence CKMS, která je generována zákaznickým obvodem a vytváří vstupní frekvenci pro programovatelný zvukový generátor. Hradlovací vstup tohoto čítače je ovládán rovněž zákaznickým obvodem. Výstup čítače 0 je navíc dostupný i prostřednictvím obvodu PIO, který bude popsán dále.

Čítač 1 pracuje poze jako dělič frekvence HSYN (15625 kHz) a jeho výstup není externě dostupný. Signál z tohoto výstupu je přiveden na vstup čítače 2.

Čítač 2 se používá jako hodinový čítač. Jeho výstup je zapojen tak, že může vyvolat přerušení. Vznik přerušovacího signálu, který řídí přímo vstup INT procesoru Z80, je podmíněn stavem výstupu PC2 obvodu 8255 (viz Tabulka 6.2).

6.3 Obvod Z80 PIO.

Tento programovatelný obvod obsahuje dvě obousměrné osmibitové brány, doplněné obvody pro řízení spolupráce v přerušovacím režimu.

Obvod je dostupný na adresách:

FCH	řídící a stavový registr brány A
FDH	řídící a stavový registr brány B
FEH	datový registr brány A
FFH	datový registr brány B

Počítač MZ-800 používá tento obvod k obsluze tiskárny. Pouze 2 bity brány A se používají k jiným účelům.

Brána A pracuje v obousměrném bitovém režimu a jednotlivé bity mají následující význam:

- PA0 - vstup - signál používaný k řízení komunikace mezi počítačem a tiskárnou (BUSY)
- PA1 - vstup - stavový signál tiskárny (Konec papíru atd.)
- PA2, PA3 - nevyužívá se
- PA4 - vstup - stavový signál systému sledující výstup časovače 0 na 8253
- PA5 - vstup - stavový signál systému sledující signál VBLK (vertikální zatemnění orazovky)
- PA6 - výstup - signál IRT používaný k inicializaci tiskárny (reset). Polaritu tohoto signálu lze ovlivnit přepínačem SW2
- PA7 - výstup - signál RDP používaný k řízení komunikace mezi tiskárnou a počítačem (STROBE). Jeho polaritu lze ovlivnit přepínačem SW3

Tak jako u klávesnice i zde aktivní stav signálu (stisknuté tlačítko) odpovídá úrovni 0.

Příklad: Napíšeme program, který bude tisknout znaky v závislosti na směru zvoleném na ovladači. Výpis znaků se ukončí tlačítkem FIRE, které bude připojeno na signál TRIG1.

```
        ORG 5000H
        XOR A
        LD (E000H),A            ;aktivace COM A/COM B
LBL1:   IN A,(FCH)              ;čtení tlačítek ovladače 1
        LD B,A                  ;kopírování obsahu
        BIT 4,B                 ;FIRE ?
        JP Z,EA5EH              ;teplý start MONITORU
        LD A,55H                ;U - nahoru
        BIT 0,B
        JR Z,LBL2
        LD A,44H                ;D - dolů
        BIT 1,B
        JR Z,LBL2
        LD A,4CH                ;L - vlevo
        BIT 2,B
        JR Z,LBL2
        LD A,52H                ;R - vpravo
        BIT 3,B
        JR Z,LBL2
        XOR A
LBL2:   CALL 0012H              ;výpis znaku z A registru
        JP LBL1
```

6.1.2 Ukládání dat na magnetofon.

V monitoru MZ-800 jsou obsaženy podprogramy, které potřebujeme k tomu, aby bylo možno uložit na magnetickou pásku obsah určité paměťové oblasti a naopak nahrát z magnetické pásky datový soubor do určitého paměťového místa. Je důležité si uvědomit následující skutečnosti:

- podprogamy pracují pouze v režimu MZ-700 (paměťové mapování I/O)
- dají se použít pouze tehdy, je-li rozdělení paměti shodné s MZ-80K (0000H - 0FFFH ROM, 1000H - CFFFH RAM, D000H - FFFFH VideoRAM/ROM).

Data se na magnetofon ukládají ve formátu SHARP-PWM (Puls Width Modulation - modulace šířky pulzu). To znamená, že bit s hodnotou 1 je zakódován do dlouhého pulzu a bit s hodnotou 0 do krátkého pulzu.

Data se na magnetofon ukládají ve dvou blocích:

- První blok se nazývá HEADER - hlavička, která obsahuje informace jako například počáteční adresa bloku, startovací adresa bloku, délka bloku, jméno a typ souboru a poznámku.
- Druhý blok souboru obsahuje vlastní data.

Hlavička je dlouhá vždy 128 bajtů a pro její zpracování existují zvláštní podprogramy. Rozdělení souborů do dvou částí má tu výhodu, že čtené informace mohou být nahrávacím programem změněny - například program adresovaný od 1200H může být nahrán od adresy 4000H.

Pro hlavičku programu mají význam následující adresy monitoru:

- 10F0H - označení typu souboru
  - 01 - OBJ - strojový program
  - 02 - BTX - programové soubory v BASICu
  - 03 - BSD - datové soubory v BASICu
- 10F1H - název souboru (maximálně 16 znaků doplněných znakem CR)
- 1102H - délka souboru (vypočítá se podle vztahu koncová adresa - počáteční adresa + 1)
- 1104H - počáteční adresa programu
- 1106H - startovací adresa programu (adresa na kterou monitor po nahrání programu předá řízení)

Následující podprogramy slouží k nahrání a uložení programu:

0021H - uložení hlavičky (je-li to třeba vypíše na obrazovku pobídku RECORD. PLAY a jakmile začne ukládat data vypíše WRITING.

0024H - uložení programu (program se zapíše na magnetofonový pásek a to od počáteční adresy uložené na 1104H s délkou uloženou na 1102H)

0027H - nahrání hlavičky (tento podprogram lze vyvolat kdykoli, neboť nejprve vyhledá nejbližší informační blok na pásce a data v tomto bloku obsažená nahraje na adresy z nichž byla uložena)

002AH - čtení datového bloku (podprogram se řídí informacemi, které byly přečteny z předchozího informačního bloku - data se uloží na adresu, která je uložená na 1104H a přečte se tolik dat, kolik udává hodnota na adrese 1102H; nesouhlasí-li délka souboru s předpokládanou délkou v hlavičce, hlásí se chyba)

Všechny výše uvedené podprogramy navrátí řízení monitoru, budou-li stisknuta tlačítka SHIFT+BREAK, nebo vyskytne-li se chyba. Dojde-li k přerušení činnosti podprogramu z některé z uvedených příčin, dojde při návratu z podprogramu k nastavení příznaku CARRY. Navíc obsah akumulátoru umožní rozlišení chybového přerušení (A=2), nebo přerušení pomocí tlačítek SHIFT+BREAK (A=1).

Příklad: Následující program uloží obsah paměti 1200H - 2000H pod označením BTX na pásku.

```
        ORG 5000H
        LD A,2                  ;BTX
        LD (10F0H),A
        LD HL,1200              ;počátek
        LD (1104H),HL
        LD HL,0E14H             ;délka
        LD (1102H),HL
        LD HL,0                 ;start
        LD (1106),HL
        LD A,0DH                ;CR
        LD (10F1H),A            ;jméno - prázdný řetězec
        CALL 0021H              ;zápis hlavičky (WRINF)
        RET C                   ;chyba
        CALL 0024H              ;zápis dat (WRD)
```

Příklad: Program, který musí skopírovat libovolný jiný program, nezávisle na jeho startovací adrese a to i tehdy, když se kopírovací a kopírovaný program překrývají. Takový program musí vyřešit následující problémy:

- Kopírovaný program musí být nahrán do paměti tak, aby nepřepsal kopírovací program.
- Hlavička programu musí být obnovena tak, aby kopírovaný program mohl být při normálním nahrávání uložen na správné místo.

```
        ORG 1200H
        LD A,16H                ;Kód pro výmaz obrazovky
        CALL 0012H              ;CLS
        CALL 0027H              ;čtení hlavičky
        LD DE,MSG1              ;hlášení LOADING
        CALL 0015H
        LD DE,10F1H             ;název souboru
        CALL 0015H
        CALL 0006H              ;nový řádek
        LD HL,(1104)            ;počáteční adresa do HL
        LD (STADD),HL           ;uložení nahrávací adresy
        LD HL,NADD
        LD (1104H),HL           ;nová nahrávací adresa
        CALL 002AH              ;nahrání bloku dat
        LD DE,MSG2              ;hlášení vyměň kazety
        CALL 0015H
LBL1:   CALL 001BH              ;získání kódu tlačítka
        OR A
        JR Z,LBL1               ;čekání na tlačítko
```



```
        LD HL,(STADD)
        LD (1104H),HL           ;obnovení počáteční adresy
        CALL 0006H
        CALL 0006H
        CALL 0021H              ;zápis hlavičky
        LD HL,NADD              ;adresa na které jsou data
        LD (1104H),HL
        CALL 0024H              ;zápis dat
MSG1:   DEFM "LOADING"
        DEFB 0DH
MSG2:   DEFM "CHANGE CASS.,PRESS ANY KEY"
        DEFB 0DH
STADD:  DEFW 0
NADD:   DEFW 3000H
```

Jak je vidět, předpokládá program ideální stav, to znamená, že při čtení i zápisu nenastane žádná chyba (netestuje se příznak C ani obsah akumulátoru při návratu z podprogramů pro spolupráci s magnetofonem).

Jedna poznámka k nahrávacímu systému IPL. Programy, které nahrává IPL se bez ohledu na svou počáteční adresu ukládají od 1200H a po nahrání se přesunou na správné adresy. Má to tu výhodu, že programy, které se ukládají od adresy 0000H mohou být na tuto adresu přímo sestaveny, protože IPL před přesunem programu otevře v rozsahu 0000H až 0FFFH paměť RAM. Naproti tomu nevýhodou je, že oblast od 1200H je při nahrávání vždy přepsána.