Poznámky k programování ve strojovém kódu Z 80 na mikropočítači ZX Spectrum a ZX 81

Vydalo Centrum pro mládež, vědu a techniku ÚV SSM
a oddělení mládeže, tělesné výchovy a sportu ÚRO

Určeno pro vnitřní potřebu SSM a ROH

Obsah:

1. ÚVOD 5

2. ZÁKLAONÍ MATEMATICKÉ POJMY 6
2.2. ARITMETICKÉ OPERACE S BINÁRNÍMI ČÍSLY 10
2.3. LOGICKÉ OPERACE 13
2.4. HEXAOECIMÁLNÍ SOUSTAVA 14
2.5. OSTATNÍ ČÍSELNÉ SOUSTAVY 16
2.6. OSTATNÍ POJMY 17

3. HAROWARE POČÍTACŮ SINCLAIR 17
3.1. MIKROPROCESOR Z 80 18

4. SOFTWARE POČÍTAČŮ SINCLAIR 21
4.1. ASSEMBLER Z80 21
4.2. STROJOVÉ INSTRUKCE Z 80 23
4.3. ŘÍOÍCÍ INSTRUKCE 27
4.4. INSTRUKCE VSTUPU A VÝSTUPU 28
4.5. INSTRUKCE PRENOSU DAT 33
4.6. INSTRUKCE BLOKOVÉHO PŘENOSU OAT 41
4.7. INSTRUKCE ARITMETICKÝCH A LOGICKÝCH OPERACÍ 43
4.8. INSTRUKCE BLOKOVÉHO PROHLEOÁVÁNÍ 52
4.9. INSTRUKCE INKREMENTACE A OEKREMENTACE 54
4.10. INSTRUKCE SKOKŮ 55
4.11. INSTRUKCE ROTACÍ A POSUNŮ 63
4.12. INSTRUKCE PRO PRÁCI S JEDNOTLIVÝMI BITY 72

5. PSEUDOINSTRUKCE ASSEMBLERU 75

6. SESTAVENÍ PROGRAMU VE STROJOVÉM KÓOU 79

7. PŘÍLOHA A 85

8. PŘÍLOHA B 86

9. PŘÍLOHA C 87

1. ÚVOD

Předpokládám, že většina z vás, kteří začínáte číst tuto příručku, vlastníte buď ZX 81 nebo Spectrum a suverénně ovládáte programování v Basicu, ale při zmínce o programu ve strojovém kódu vám běhá mráz po zádech a připadáte si poněkud méněcenní.

Verse Basicu používaná u počítačů Sinclair se sice poněkud odchyluje od (neexistujícího) standardu, ale nepracuje se s ní špatně. Hlavní nevýhodou všech verzí Basicu, i těch nejdokonalejších, je jejich pomalost a větší náročnost na paměť. Při provádění basicového příkazu musí počítač vždy použít překladače z Basicu do strojového kódu procesoru (který je zakódován v ROM). Opakuje-li se stejný příkaz stokrát za sebou, počítač musí daný příkaz stokrát přeložit do reči procesoru. Tato pomalost je sice určitou nevýhodou při provádění obsáhlejších matematických výpočtů, ale hlavní překážku tvoří při programování většiny her, u kterých se většinou požaduje rychlost. Program sestavený v Basicu je příliš pomalý a nestačí na rychle se měnící herní situace. V tomto případě zjedná nápravu pouze program napsaný ve strojovém kódu.

Možnosti programování ve strojovém kódu jsou prakticky neomezené a teprve programováním ve strojovém kódu můžeme plně využít všech schopností mikroprocesoru Z80, který je základem Spectra. Stačí si uvědomit, že tentýž Z80 pracuje v řadě jiných počítačů, mnohonásobně výkonnějších (a bohužel také dražších) než je naše Sprectrum. Různá omezení Sinclairu nejsou dána použitým mikroprocesorem, ale tím, co ho obklopuje (a toho je velice malo). Existují dva krajní názory na mikroprocesor. Jeden názor tvrdí, že mikroprocesor je universální genius, druhý říká, že mikroprocesor je geniální blbec. V obou názorech je část pravdy. Celou pravdou je to, že mikroprocesor umí jen to, co mu formou

programu předloží programátor a je tak dobrý, jak je dobrý jeho programátor.

Nic nestojí v cestě tomu, aby i vy jste se stali experty ve strojovém kódu. Není to nic světoborného. Chce to jen trochu logického uvažování a hodně trpělivosti. Musíte vyjít od těch nejjednodušších programů a postupně experimentovat se stále složitějšími aplikacemi. Postupně zjistíte, že původní strašidelnost se vytrácí, a že jste schopni ve strojovém kódu řešit i komplikované programy, resp. jejich kombinace s programy basicovými.

Ale pozor! Tento stručný návod není samospasitelný a jeho přečtením se z vás nestanou rázem experti. Je to pouze začátek a předpokládá další studium.

2. ZÁKLAONÍ MATEMATICKÉ POJMY

Poněvadž z vlastní zkušenosti vím, jak je někdy obtížné získat odpovídající literaturu, resp. potřebné informace bez konzumace spousty dalších nepotřebných údajů, začnu od Adama a v první kapitole se pokusím pro ty, kteří se doposud podobnými problémy nezabývali, stručně shrnout základní matematické pojmy, bez nichž není možno úspěšně zvládnout programování ve strojovém kódu.

Mikroprocesor je elektrické zařízení, pracující (zjednodušeně řečeno) na bázi vypnuto-zapnuto". Základními pracovními prvky všech mikroprocesorů jsou tzv. registry, což jsou soustavy elektrických prvků, které jsou schopny uchovávat, resp. různě kombinovat a převádět právě tyto dva stavy a nic víc. Matematicky toto "vypnuto-zpnuto" vyjadřujeme hodnotami "0" a "1". Musíme tedy pro všechny další úvahy vystačit pouze s nulou a jedničkou.

2.1. BINÁRNÍ SOUSTAVA

Matematický systém používající pouze hodnot 0 a 1 je nazýván binárním nebo dvojkovým systémem. V tomto systému jsou vyjádřeny jako mocniny čísla 2: $2\uparrow 0=1$, $2\uparrow 2=4$, $2\uparrow 3=8$, $2\uparrow 4=16$ atd.

Kombinacemi těchto mocnin pak můžeme vyjádřit jakékoliv číslo. Forma tohoto vyjádření je soustava nul a jedniček. např. 1010. Toto číslo však nevyjadřuje hodnotu 1 010, jak jsme zvyklí z běžné desítkové soustavy, ale hodnotu 10. Aby nemohlo dojít k omylu, označují se čísla ve dvojkové (binární) soustavě písmenem B a čísla v desítkové (decimální) soustavě písmenem D (případně b a d). Můžeme pak napsat 1010b = 10d.

Aby nedocházelo k záměně písmene O a číslice nula (0), používá se pro nulu přeškrtnutá nula. Poněvadž v tomto textu nebude k záměně docházet, budu pro číslici nula používat normálního označení 0.

Číselná řada ve dvojkové soustavě vypadá následovně:

```
    0b = 0d
    1b = 1d
   10b = 2d
   11b = 10b + 1b = 2d + 1d = 3d
  100b = 4d
  101b = 100b + 1b = 4d + 1d = 5d
  110b = 100b + 10b = 4d + 2d = 6d
  111b = 100b + 10b + 1b = 4d + 2d + 1d = 7d
 1000b = 8d
 1001b = 1000b + 1b = 8d + 1d = 9d
    :
    :
```

I když to vypadá složitě, systém je jednoduchý: 0 znaená nula, jedna znamená, že na daném místě je odpovídající mocnina dvou. Mocniny dvou se postupně zvyšují zprava doleva. Zcela vpravo je vždy 2^0, každý další posun doleva znamená násobení dvěma, t.j. o řád vyšší mocnina 2:

.....	2^6	2^5	2^4	2^3	2^2	2^1	2^0
.....	64	32	16	8	4	2	1

Náš uvedený příklad, t.j. 1010b vyjadřuje:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 0 + 2 + 0 = 10d.$$

Jedničky a nuly v binárním čísle nazýváme bity. 8 bitů tvoří Bajt. 8 bitů proto, že většina procesoru pracuje právě s osmimístnými registry. Maximální hodnota jednoho bajtu může být 1111 1111b = 255d.

Při práci se strojovým kódem budeme potřebovat nejen kladná, ale i záporná čísla. V binárním tvaru se používá k vyjádření záporného čísla tzv doplňkového kódu. Je to další bit přidaný k binárnímu číslu. Je-li jeho hodnota 0, jedná se o kladné číslo, je-li 1, jedná se o záporné číslo. Při běžné používaných osmibitových slovech budeme nejvyšší bit považovat za znaménkový bit a k vyjádření hodnoty pak zbýva jen 7 bitu. To znamená, že v tomto případě můžeme 8 bity vyjádřit maximální hodnotu od -128 do +127. 8 bitu někdy nazýváme slovem a 4 bity slabikou.

Zápornou hodnotu čísla získáme tak, že toto číslo odečteme od čísla 2^n, kde n je počet významných bitů. V případě 8-bitového čísla je první bit znaménkový a zbývajících 7 bitu významových. Hodnota od které odečítáme je potom 2^7 = 128. Pak -1 = 128 -1 = 127, což vyjádřeno binárně je 1111 1111b = -1d (1. jednička je znaménkový bit, ostatní jsou významové).

-2 je 128 - 2 = 126, binarne 1111 1110,
-3 je 128 - 3 = 125, binarne 1111 1101,

a td. Nejlépe je to vidět na číselné řade:

```
127d = 0111 1111b
          :
          :
  5d = 0000 0101b
  4d = 0000 0100b
  3d = 0000 0011b
  2d = 0000 0010b
  1d = 0000 0000b

  0d = 0000 0000b

-1d (127d) = 1111 1111b
-2d (126d) = 1111 1110b
-3d (125d) = 1111 1101b
-4d (124d) = 1111 1100b
-5d (123d) = 1111 1011b
          :
          :
-127d (1d) = 1000 0001b
-128d (0d) = 1000 0000b
```

Převod kladných čísel na záporná se provádí tak, že se invertují všechny bity včetně znaménkového (t.j. místo 1 se napíše 0 a naopak) a k takto vzniklému číslu se přičte 1:

```
+85d = 0101 0101b
       1010 1010b   invertováno
     +        1b
       ----------
-85d = 1010 1011b
```

Při převodu záporných čísel na kladná se postupuje tak, že od daného čísla se odečte 1 a takto vzniklé číslo se invertuje:

```
-103d = 1001 1001b
      -        1b
        ----------
        1001 1000b
-103d = 0110 0111b    invertováno
```

Můžeme postupovat i následujícím způsobem, který platí jak pro převod kladných čísel na zaporná, tak i pro opak. Postupujeme zprava doleva. Nuly ponecháváme až narazíme na první jedničku. Tu také ještě ponecháme, všechny ostatní bity invertujeme:

```
 +84d = 0101 0100b
        |----||-|  ponecháme
      invertujeme
  -84 = 1010 1100b

-103d = 1001 1001b
        |------|| ponecháme
      invertujeme
+103d = 0110 0111b
```

2.2. ARITMETICKÉ OPERACE S BINÁRNÍMI ČÍSLY

Binární čísla můžeme sečítat, odečítat, násobit i dělit. Platí následující pravidla:

- sčítání binárních čísel:

0B + 0B = 0B
0B + 1B = 1B
1B + 1B = 10B

což můžeme také vyjádřit 1B + 1B = 0B + přenos
Tyto zásady platí pro jakoukoliv pozici bitu v binárním čísle. Nejlépe je to zřejmé na příkladech:

```
 1D      01B          3D      11B ┐
+1D     +01B         +1D     +01B ├─ 1. krok
 ──      ───                  ───┤
 2D      10B                  10-│
          ↑                     │ přenos ┐
        přenos                  +1┘        ├─ 2. krok
                                 ──        │
                               4d   100b   ┘

 5d     101b ┐
+7d    +111b ├─ 1. krok
        ───  │
        010  ┘
      +1 1    přenos ┐
 ──   ────           ├─ 2. krok
12d   1100b          ┘
        ↑
      přenos
```

- odečítání binárních čísel:

Odečítáme tak, ze sčítáme dvojkové doplňky (záporná čísla) podle pravidla A - B = A + (-B) , nebo -A -B = +(-A) + (-B).

```
  12d      0000 1100b
+ -7d      1111 1001b
  ───      ──────────
   5d      0000 0101b
```

Vystačíme tak vlastně pro sečítání i odečítání s jedním početním úkonem.

- **násobení a dělení binárních čísel:**

Každý posun bitu o jedno místo doleva znamená násobení dvěma, a naopak posun o jedno místo doprava znamená dělení dvěma:

0001b = 1d	1000b = 8d
0010b = 1 * 2 = 2d	0100b = 8 / 2 = 4d
0100b = 2 * 2 = 4d	0010b = 4 / 2 = 2d
1000b = 4 * 2 = 8d	0001b = 2 / 2 = 1d

Můžeme pokračovat doprava až za desetinnou tečku, takže:

0.1b = 1 / 2 = 0.5d	= 2↑-1
0.01b = 0.5 / 2 = 0.25d	= 2↑-2
0.001b = 0.25 / 2 = 0.125d	= 2↑-3

Tímto způsobem můžeme vyjádřit i desetinná čísla. Chceme-li násobit dvě binární čísla, musíme násobitele rozdělit na násobky 2. Napr. 10d * 10d = 10d * (2d + 8d).

```
10d          =      1010b
10d * 2d     =     10100b
```

1x posun doleva. t.j. přidá se jedna nula

```
+10d * 8d    =  +1010000b 3x posun doleva. t.j. přidají
---------       --------  se tři nuly
 10d * 10d   =   1100100b
```

Jiný příklad 10d * 5d = 10d * (1d + 4d) :

```
 10d         =      1010b
 10d * 1d    =      1010b  nic se nemění
+10d * 4d    =   +101000b  2 posuny doleva - 2 nuly navíc
 ---------      ---------
 10d * 5d    =    110010b
```

Poznámka - místo desetinné čárky se zásadně používá tečky a pro násobení hvězdičky a dělení zlomkové čáry.

Dělení dvou binárních čísel je opakem násobení. Dělíme tak, že od dělence postupně odečítáme jeho násobky dvou, podle toho, je-li příslušný násobek obsazen v děliteli formou jedničky. Je-li na daném místě dělitele nula, neodečítáme nic. Násobky dvou dostaneme posunem dělence o jedno místo doprava.

Při všech aritmetických operacích musíme dbát na to, aby nedošlo k tzv. přetečení. Přetečení nastane tehdy, dojde-li k přenosu neexistujícího bitu, nebo do znaménkového bitu a dojde tak ke zkreslení výsledku. Ve čtyřbitovém registru bude např. při součtu 1111b + 0001b = 1 0000b. Výsledek v registru je 0000b, což je nepravdivé. ↑ neexistující bit.

2.3. LOGICKÉ OPERACE

Vedle běžné algebry, tak jak ji znáte ze školy, existuje ještě Booleova algebra, která nepracuje s čísly, ale s pojmy pravda, nepravda. Pravda je vyjádřena jedničkou, nepravda nulou. V Booleově algebře existují tři operace logický součin, označovaný AND nebo symbolem /|, logicky součet, označovaný OR nebo symbolem |/ a binární nebo exkluzivní součet, označovaný NOR nebo XOR nebo symbolem plus v kruhu.Pro jednotlivé funkce platí:

- **logický součin:** 1 AND 0 = 0
  0 AND 0 = 0
  1 AND 1 = 1
  0 AND 1 = 0

- **Logický součet:** 1 OR 0 = 1
  0 OR 0 = 0
  1 OR 1 = 1
  0 OR 1 = 1

- **exkluzivní (binární) součet:** 1 NOR 0 = 1
  0 NOR 0 = 0
  1 NOR 1 = 0
  0 NOR 1 = 1

Příklady jednotlivých operací:

```
   AND        OR         NOR (XOR)
-----------------------------------
  0011       0011         0011
 /|0101     |/0101      (+)0101
  ----       ----         ----
  0001       0111         0110
```

2.4. HEXADECIMÁLNÍ SOUSTAVA

Práce s kolonami jedniček a nul je značně nepřehledná a vede ke vzniku chyb. Pro ulehčení této práce byly vytvořeny další soustavy, z nichž je nejpoužívanější soustava hexadecimální či šestnáctková.

Tato soustava vznikne tak, že se sečtou hodnoty vždy čtyř bitů a takto vzniklé číslo je číslicí hexadecimální soustavy. Různými kombinacemi 4 bitů dostaneme nejvýše 16 hodnot, odtud tedy název soustavy-šestnáctková. Poněvadž číslice základního modulu soustavy musí být jednociferné, čehož nemohlo být dosaženo normálními číslicemi 0 až 9, bylo přibráno prvních 6 písmen abecedy tak, že po číslici 9 nenásleduje číslice 10, ale A a pokračuje se postupně až po

F. Hodnoty vyjádřené v hexadecimální soustavě označujeme písmenem H (nebo h) připojeným za číslice této soustavy.

Číselná řada v šestnáctkové soustavě vypadá následovně:

0d	=	0000b	=	0h
1d	=	0001b	=	1h
2d	=	0010b	=	2h
3d	=	0011b	=	3h
4d	=	0100b	=	4h
5d	=	0101b	=	5h
6d	=	0110b	=	6h
7d	=	0111b	=	7h
8d	=	1000b	=	8h
9d	=	1001b	=	9h
10d	=	1010b	=	Ah
11d	=	1011b	=	Bh
12d	=	1100b	=	Ch
13d	=	1101b	=	Dh
14d	=	1110b	=	Eh
15d	=	1111b	=	Fh

Chceme-li vyjádřit v hexadecimální (HEX) soustavě hodnoty vyšší než 15, musíme použít dvou a více HEX-číslic:

16d	=	0001 0000b	=	10h
17d	=	0001 0001b	=	11h
		:		
		:		
27d	=	0001 1011	=	1Bh
		:		
		:		

Každá čtveřice bitů má svoji odpovídající Hex-číslici. Můžeme tak vytvořit libovolnou hodnotu. Prakticky nejvyšší Hex-číslo, které budeme používat, je FFFFh, což odpovídá 16-ti bitovému číslu 1111 1111 1111 1111b nebo 65 535d. Nejvyšší proto, že procesor Z80 může pracovat maximálně se šestnáctibitovými čísly (v binární soustavě).

Počítání v HEX-soustavě je značně jednodušší, než v soustavě binární. Problém je při přechodu z HEX na desetinnou soustavu nebo při aritmetických operacích. Při

převodech decimálních čísel na hexadecimální se používá tabulek nebo různých programů pro převody (příloha A). Při aritmetických nebo logických operacích musíme obvykle HEX číslo převést do binárního tvaru (někdy také postačí do desetinného tvaru), provést operaci a výsledek znovu převést do HEX-formy.

Bylo-li v binární soustavě 1111 1111b rovno -1d, platí to obdobně i v HEX tvaru: FFh je rovno 255d, ale také -1d, FEh je jednak 254d, ale i -2d. Nejlépe je to zase zřejmě z číselné řady:

127d	=	0111 1111b	=	7Fh
:				
:				
3d	=	0000 0011b	=	03h
2d	=	0000 0010b	=	02h
1d	=	0000 0001b	=	01h
0d	=	0000 0000b	=	00h
-1d	=	1111 1111b	=	FFh
-2d	=	1111 1110b	=	FEh
-3d	=	1111 1101b	=	FDh
-4d	=	1111 1100b	=	FCh
:				
:				
-127d	=	1000 0001b	=	81h
-128d	=	1000 0000b	=	80h

Tohoto způsobu se používá při zadání valikosti odskoku u relativních skoků, proto je třeba si systém záporných čísel v HEX soustavě dobře osvojit. V počítačové literature se používá pro binární čísla znaku % a pro HEX cisla znaku $ a píší se před číslo.

2.5. OSTATNÍ ČÍSELNÉ SOUSTAVY

Vedle binární a hexadecimální soustavy se v počítačové praxi užívají ještě dvě další dvě soustavy. Je to soustava osmičková či oktalova soustava "BCD". Při další práci je nebudeme potřebovat, proto jen stručně o obou:

OKTALOVÁ SOUSTAVA

Je to obdoba šestnáctkové soustavy, ale vychází jen ze tří bitů (což je dáno typem procesoru, který pracuje se šestnáctibitovými registry). Maximální počet kombinací tří bitů je 8, proto osmičková. Pracuje pouze s číslicemi 0 až 7, číslice 8 a 9 nezná.

BCD SOUSTAVA

Soustava BCD převádí jednotlivé číslice desítkového čísla do 4 bitů binární soustavy a naopak: 327d = 0011 0010 0111 BCD. Procesor Z80 má speciální příkazy pro počítání v BCD soustavě.

2.6. OSTATNÍ POJMY

PARITA

Paritou rozumíme součet jedniček v binárním čísle. Je-li tento součet sudý, je parita sudá, je-li tento součet lichý, je parita lichá. Používá se jako kontrolní hodnota při práci s binárními hodnotami.

3. HARDWARE POČÍTAČŮ SINCLAIR

Jak ZX 81 tak ZX Spectrum se vyznačují tím, že obsahují minimální počet integrovaných obvodů. U ZX 81 jsou to pouze 4 integrované obvody (případně 5, podle typu použitých pamětí), u Spectra je to 13 integrovaných obvodů, z čehož 10 obvodů je 16k bajt pamět a její pomocné obvody. Je to pravděpodobně světový rekord a je tím podmíněna nízka cena obou počítačů. Na druhé straně to představuje určitá omezení. Poněvadž tato práce se nezabývá hardwarovou stránkou Sinclairu, t.j. možnostmi připojení dalších obvodů a zařízení (návod na stavbu univerzálního interfejsu MIREK a návod na připojení jednotky pružných disků ZX-FLOPPY je

možné získat u 666.ZO Svazarmu PS64 Praha 6 16900), uvedu jen to nejnutnější.

3.1. MIKROPROCESOR Z 80

Hlavní součástí každého počítače je centrální procesorová jednotka (CPU), která jednak vykonává veškeré početní operace, jednak řídí činnost celého počítače. V obou Sinclairech je použito CPU firmy Zilog, typ Z80A, což je tychlejší verze původního typu Z80. Je to jeden z nejrozšířenějších a nejvýkonnějších osmibitových procesorů. Umožňuje provádět 158 strojových instrukcí. Z80 může adresovat paměť do rozsahu 64k bajtů, čehož plně využívá pouze ZX Spectrum, zatímci ZX 81 je díky neúplnému dekódování adresového výstupu A15 rozsah adresování omezen na 32 kBajtů.

Nejdůležitější součástí každého procesoru jsou jeho registry. Z80 má celkem 22 registru :

- **registr** A (akumulátor). Osmibitový registr, v němž se provádí většina aritmetických a logických operací.

- **registr** F (stavový registr). Jednotlivé bity tohoto registru informují systém o výsledcích jednotlivých operací. 8 bitů tohoto registru, tzv. příznakové bity nebo stavové bity, mají následující označení a význam:

  -- **bit 7** : znaménkový bit. Je nastaven, t.j. má hodnotu 1, je-li při matematických operacích nastaven nejvyšší bit registru (t.j. je-li v nejvyšším bitu 1, což **znamená zápornou hodnotu**).

  -- **bit 6** : indikátor nuly. Bit je nastaven, je-li výsledkem aritmetických nebo logických operací v akumulátoru roven nule.

-- bit 5 : neobsazen, nepoužívá se

-- bit 4 : indikátor polovičního přenosu. Používá se při instrukce DAA a indikuje přenos mezi 3 a 4 bitem.

-- bit 3 : neobsazen, nepoužívá se.

-- bit 2 : indikátor parity a přetečení. Bit je nastaven, dojde-li při aritmetických operacích k přetečení, t.j. je-li výsledkem číslo větší než kapacita registrů. Při logiských operacích a rotacích je nastaven, je-li parita sudá.

-- bit 1 : indikátor sčítání a odčítání. Používá se při DAA a je nastaven, byla-li předchozí operace odčítání.

-- bit 0 : indikátor přenosu. Bit je nastaven, dojde-li při aritmetických operacích k přenosu ze znaménkového bitu.

Podrobněji budou jednotlivé funkce stavových bitů tohoto registru popsány při popsány při popisu strojových instrukcí. Jejich stav podmiňuje provedení některých strojových instrukcí.

- **registry B, C, D,** E, H, L. Osmibitové všeobecné registry pro manipulaci s daty. Tyto registry je možno spojovat do šestnáctibitových registrových párů BC, DE, HL. Registrový pár BC je pro nás zajímavý tím,že po návratu z programu ve strojovém kódu do Basicu bude jeho obsah vypsán na obrazovku, bylo-li ke spuštění programu použito instrukce PRINT USR.

- **registry IX, IY.** Šestnáctibitové indexové registry pro tzv. indexové adresování.

- **registr** PC. Programový čítač. Šestnáctibitový registr obsahující adresu prováděné instrukce. Při každém zapojení počítače je vynulován a jeho obsah se postupně zvyšuje o jedničku.

- **registr** SP. Ukazatel zásobníku. Šestnáctibitový registr obsahující adresu uživatelského zásobníku (stack).

- **registr** I. Osmibitový registr vektoru přerušení.

- **registr** R. Osmibitový registr, jehož prostřednictvím provádí Z80 automatické osvěžování dynamických pamětí.

Vedle registrů A, F, B, C, D, E, H a L má Z80 ještě registry A´, F´, B´, C´, D´, E´, H´ a L´. Jsou to tzv. zdvojené registry. Jsou shodné se základními registry a mohou si s nimi vyměňovat obsah. Programově přímo přístupné nejsou.

pro lepší názornost ještě tabulka všech registrů Z80:

A	F	A´	F´
B	C	B´	C´
D	E	D´	E´
H	L	H´	L´
IX			
IY			
PC			
SP			
I	R		

4. SOFTWARE POČÍTAČŮ SINCLAIR

4.1. ASSEMBLER Z80

Každý mikroprocesor rozumí pouze pevně stanovenému souboru strojových instrukcí, které jsou vyjádřeny kombinacemi jednoho až čtyř bajtů. Tak pro Z 80 znamená bajt 76 strojovou instrukci HALT. Program ve strojovém kódu, kterému jedině mikroprocesor rozumí, je pak posloupnost strojových instrukcí, vyjádřená jako posloupnost bajtů. Takový jednoduchý program může vypadat např. následovně: 01 D6 02 0C 40 ED 5B 04 40 00 ED B0 C9 /tento program přenese obsah registru obrazovky za RAMTOP/. Pamatovat si, že instrukce pro "HALT" je 76 je dost nepohodlné a při práci s jednotlivými bajty se dopustíme určité řady chyb. Proto byl vyvinut jazyk symbolických adres, nazývaný též assemblerem, který značně usnadňuje práci se strojovými instrukcemi mikroprocesoru. Původně je assembler označení programu, který překládá program napsaný v jazyku symbolických adres do strojového kódu procesoru, ale pro jednoduchost se pro jazyk symbolických adres používá označení assembler. Pokud bude někde potřeba rozlišení těchto dvou pojmů, budu pro jazyk symbolických adres používat výrazu assembler a pro pomocný program výrazu Asembler.

Pro sestavování programu v assembleru platí určitá obecná pravidla. Každá část : návěští, instrukcí, operandy a poznámky je od ostatních oddělena mezerami. Návěští musí být ukončeno dvojtečkou, před poznámkou musí být středník. Program uvedený v předchozím odstavci vypadá v assembleru takto:

návěští	instrukce	operandy	poznámky
začátek:	ld BC	C2 D6	; do reg. BC hodnota C2D6
	ld HL	(4C0C)	; do reg. HL D-FILE
	ld DE	(4004)	; do reg. DE RAMTOP
	nop		; žádná operace
	ldir		; blokový přenos
konec :	ret		; zpět do BASICu

Nyní musíme provést to, co normálně provádí Assembler, t.j. jednotlivým instrukcím assembleru musíme přiřadit odpovídající hexadecimální kódy strojových instrukcí Z 80. Výsledkem bude relativní strojový program, t.j. program, kterému už bude Z 80 rozumět, který ale ještě nefunguje, poněvadž chybí počáteční adresa programu.

relat. stroj program	návěští	instrukce	operandy	pozámky
C1 D6 C2	zacatek:	ld BC	C2 D6	; do reg.BC....
EA 0C 40		ld HL	(400C)	; do reg.HL....
ED 5B C4 40		ld DE	(4CC4)	; do reg.DE....
00		nop		; žádná oper.
ED 80		ldir		; blok.přenos
C9	konec	ret		; zpět do.....

Zvolením počáteční adresy, např. 4082, dostaneme absolutní strojový program, který můžeme vložit do počítače a který již počítač provede. Poněvadž se v programech obvykle používá i absolutních skoků, píšeme pak takový program tak, že na první místo ještě napíšeme jednotlivé adresy. Definitivní výsledek naší práce pak bude vypadat následovně :

adresa	kód	náv.	instr.	operand	poznámka
4C82	C1 D6 C2	zač:	ld BC	C2 D6	; do reg.BC C2 D6
4C85	2A 0C 40		ld HL	(40 0C)	; do reg.HL adr.
4C88	ED 5B C4 4C		ld DE	(40 C4)	; do reg.DE RAMT.
4CBC	00		nop		; žádná operace
4C8D	ED B0		ldir		; blok.přenos
4C8F	C9	kon:	ret		; návrat

Tento program je napsán pro ZX 81 a přenese 726 d bajtů obrazového registru, t.j. obsah obrazovky do oblasti za RAMTOP, který předtím snížil.

To bylo na úvod. K assembleru se ještě na závěr kapitoly vrátíme, ale nyní se musíme věnovat jednotlivým

strojovým instrukcím Z 80, poněvadž bez jejich perfektní znalosti nám ani nejlepší Asembler nepomůže.

4.2. STROJOVÉ INSTRUKCE Z 80

Mikroprocesor Z 80 může provádět 158 strojových instrukcí. Tyto instrukce můžeme rozdělit do následujících skupin :

- **řídící instrukce** nop,halt,di,ei,im,ccf,scf
- **vstup a výstup** in, ini, inir, ind, indr, out, outi, outir, outd, outdr
- **přenos dat** ld,push,pop,ex,exx
- **blokový přenos** ldi,ldir,ldd,lddr
- **aritmetické a logické operace** add, adc, sub, sbc, daa, cpl, neg, and, or, xor, cp
- **blokové prohledávání** cpi,cpir,cpd,cpdr
- **inkrementace a dekrementace** inc,dec
- **skoky** jp,jr,jr DIS,djnz
- **volání a návraty** call, ret, reti, retn, rst
- **rotace a posuny** rlca, rla, rrla, rra, rlc, rl, rrc, rld, rr, sra, srl
- **bitové operace** bit,set,res

Jednotlivé instrukce se označují mnemotechnickými zkratkami, např. ld pro load (= přenes, vlož), inc pro increment (= zvyš) jp pro jump (= skoč) a pod. Ten kdo zná alespoň základy angličtiny to má jednodušší, ale ani ten kdo o angličtinu v životě nezavadil si nemusí zoufat, poněvadž toho zase není tak moc.

V dalším textu bude použito následujícího způsobu značení jednotlivých pojmů:

- mnemotechnické zkratky budou psány malými písmeny
- označení registrů a čísel bude psáno velkými písmeny (někde v literatuře jsou malými písmeny i registry)
- registr obecně bude označen R
- jednobajtové číslo bude obecně značeno N
- hodnota odskoku DIS u relativních skoků bude vyjadřována také jednobajtovým číslem N
- dvoubajtové číslo bude obecně NN , případně MN
- index bude obecně označován d (jednobajtové číslo)
- číslo v závorce znamená obsah paměťového místa v závorce
- registr v závorce znamená obsah paměťového místa, jehož adresa je v daném registru
- A - H znamená, že obsah registru H se přenese do registru A
- jpl.,2. znamená, že skok na operand 2 se provede tehdy, je-li splněna podmínka daná operandem 1
- pro rozlišení registru C a přenosového bitu C v registru F bude pro přenosový bit použito označení CY
-CY znamená opačnou hodnotu než má CY=C (je-li CY=0, je -CY=1)

Než se pustíme do jednotlivých strojových instrukcí Z 80, musíme se vrátit ke stavovým indikátorům, což jsou některé bity registru F. Tyto indikátory poskytují informace o stavu CPU a především o způsobu dokončení poslední aritmetické operace. Jejich poloha v ragistru F je na následujícím schématu :

registr F

č. bitu	7	6	5	4	3	2	1	0
indikátor	S	Z	-	H	-	P/V	N	C

C indikátor přenosu (CY)

Udává vznik přenosu ze znaménkového bitu při sčítání nebo odčítání. Dojde-li k přenosu, je indikátor nastaven, t.j. jeho hodnota je 1, opačně 0. Dále je nastaven instrukcí daa jsou-li splněny podmínky pro převod do BCD kódu. Při provádění logických operací and,or,xor je indikátor nulován. Z indikátorem přenosu můžeme manipulovat instrukcemi scf a ccf.

N indikátor sčítání a odčítání

Používá se pouze při instrukci daa a udává, byla-li předchozí instrukce sčítání (N=0) nebo odčítání (N=1).

P/V indikátor parity a přetečení

Při aritmetických operacích bude indikátor nastaven, je-li výsledkem aritmetické operace větší hodnota než je kapacita registru ve kterém se operace provádí.
Při logických operacích a rotacích udává indikátor paritu výsledku. Je nastaven (P/V=1), je-li parita sidá a nulován (P/V=0), je-li parita lichá.

Při instrukcích blokového prohledávání a blokového přenosu udává indikátor stav registru BC (čítače). Je-li hodnota čítače odlišná od nuly je nastaven, t.j. P/V=1, dosáhne-li hodnota čítače nuly, je indikátor nulován, t.j. P/V=0.

Při instrukcích ld A,I a ld A,R bude P/V nastaven obsahem klopného obvodu uvolnění přerušení IFF2.
Při čtení bajtu ze vstupního nebo výstupního zařízení instrukcí in R,(C) bude P/V indikovat paritu přenášeného bajtu.

H indikátor polovičního přenosu

Používá se při provádění instrukce daa a indikuje přenos mezi 3. a 4. bitem při provádění aritmetických operaci.

Z indikátor nuly

Při osmibitových aritmetických a logických operacích dojde k nastavení indikátoru (Z=1), je-li výsledkem operace v registru A nula. Je-li výsledek nenulový, je Z=0. (pozor plete se to)

Při instrukcích srovnávání bude indikátor nastaven (Z=1) při shodě porovnávaných hodnot. Při testování bitu bude se indikátor rovnat komplementu (t.j. opačné hodnotě) testovaného bitu. Při instrukcích ini,ina a outd bude indikátor nastaven, bude-li obsah registru B roven 1.
Při instrukci in R,(C) bude indikátor nastaven v případě nulového vstupu.

S indikátor znaménka

Tento indikátor udává stav nejvýznamnějšího bitu registru ve kterém je uložen výsledek aritmetické operace. Při záporném výsledku je S=1.

Dále bude stav jednotlivých indikátorů označován následovně (u podmíněných instrukcí) :

indikátor	S	Z	P/V	C
nenastaven (0)	P	NZ	PO	NC
nastaven (1)	M	Z	PE	C

Nyní se musíme detailně seznámit s jednotlivými strojovými instrukcemi.

4.3. ŘÍDÍCÍ INSTRUKCE

Slouží k nastavení pracovního režimu procesoru, povolení a znemožnění maskovatelného přerušení, k nastavení CPU a k manipulaci se stavovým indikátorem CY.

nop žádná operace (prazdná operace) op.kód 00

Nenásleduje žádná činnost, procesor čeká na další instrukci. Nastavení indikátoru zůstává nezmeneno.

halt zastavení op.kód 76

Zastaví CPU a provádění operace nop až do přijetí přerušení nebo vynulování CPU.
Nastavení indikátorů zůstává nezměněno.

di zamaskování přerušení op.kód F3

Instrukce znemožňuje přijetí zamaskovatelného přerušení až do jeho uvolnění instrukcí ei.
Nastavení indikátorů zůstává nezměněno.

ei uvolnění přerušení op.kód FB

Instrukce uvolní přerušení zakázané instrukcí di.
Nastavení indikátoru zůstává nezměněno.

im nastavení režimu přerušení

Instrukcí im je možno nastavit režim přerušení 0,1 nebo 2, který se uvede do argumentu instrukce.
Nastavení indikátorů se nezmění.

instrukce	op.kód
im 0	ED 46
im 1	ED 56
im 2	ED 5E

ccf inverze indikátoru C (CY) op.kód 3F

Instrukce provede inverzi bitu C v registru F.
Indikátory S,Z a P/V nezměněny, N nulován, C invertován, H převezme předchozí nastavení C.

scf nastavení indikátoru C (CY) op.kód 37

Instrukce nastaví indikátor C na hodnotu 1.
Indikátory S,Z a P/V nezměněny, N=0, H=0, C=1.

4.4. INSTRUKCE VSTUPU A VÝSTUPU

in instrukce vstupu

Instrukce provádí přenos jednoho bajtu dat z periferního zařízení do registru Z 80. Jsou možné následující kombinace:

instrukce	op.kód
in A,(C)	ED 78
in B,(C)	ED 40
in C,(C)	ED 48
in D,(C)	ED 50
in E,(C)	ED 58
in H,(C)	ED 60

```
in L,(C)        ED 6B
in A,(N)        DB N
```

kde (C) znamená adresu periferního zařízení, která je uložena v registru C a (N) znamená adresu periferního zařízení.
Nastavení indikátorů: v případě in A,(N) nezměněno, v případech in I.,(C) bude H=0, N=0, C nezměněn, S, Z a P/V nastaveny podle hodnot vstupujících dat.

ini instrukce blokového vstupu s inkrementací bez opakování (in+inc) op.kód ED A2

Instrukce provede přenos jednoho bajtu z periferního zařízení, jehož adresa je v registru C, na adresové místo, jehož adresa je v registru HL, Po skončení přenosu dekrementuje registr B a inkrementuje registr HL. Po skončení operace je stav registrů následující:

(HL) - (C) ; B - B-1 ; HL - HL+1

Nastavení indikátoru: S, H a P/V nedefinovaný, C nezměněn, N nastaven a Z nastaven v případě, je-li B-1=0.

inir instrukce blokového vstupu s inkrementací a opakováním (in+inc+repeat) op.kód ED B2

Instrukce provádí totéž jako ini opakovaně tak dlouho, pokud je B 0. Po dosažení stavu B=0 se pokračuje další instrukci. U této instrukce se používá registru B jako počítadla: před instrukcí inir musíme do registru B uložit počet bajtů, které chceme přenést.
Nastavení indikátorů je jako u ini.

ind instrukce blokového vstupu s dekrementací bez opakování (in+dec) op.kód ED AA

Instrukce provádí přenos jednoho bajtu z periferního zařízení podobně jako instrukce ini s tím rozdílem, že po provedení přenosu inkrementuje jak registr B tak registr HL. Po skončení operace je tudíž stav registru následující:

(HL) - (C) ; B - B-1 ; HL - HL-1

Nastavení indikátoru jako u instrukce ini.

indr instrukce blokového vstupu s dekrementací a opakováním (in+dec+repeat) op.kód ED 2A

Instrukce provádí přenos jednoho bajtu z periferního zařízení jako instrukce ind ale opakovaně tak dlouho, pokud je B 0. Po dosažení stavu B=0 se pokračuje další instrukcí. Registru B se opět používá jako počítadla.
Nastavení indikátorů jako u instrukce ini.

out instrukce výstupu

Instrukce provádí přenos jednoho bajtu z registru Z 80, daného 2. operandem, do periferního zařízení daného prvním operandem. Jsou možné následující kombinace:

instrukce	**op.kód**
out (C),A	ED 79
out (C),B	ED 41
out (C),C	ED 49
out (C),D	ED 51
out (C),E	ED 59
out (C),H	ED 61
out (C),L	ED 69
out (N),A	D3 N

kde (C) znamená adresu periferního zařízení uloženou v registru C a (N) znamená adresu periferijního zařízení.
Nastavení indikátorů nezměněno.

outi instrukce blokového vystupu s inkrementací bez opakování (out+inc) op.kód ED A3

Instrukce provede přenos jednoho bajtu z paměťového místa, jehož adresa je v registru HL, do periferijního zařízení, jehož adresa je v registru C. Po skončení přenosu je dekrementován registr HL , t.j. stav registru po skončení operace je následující:

(C) - (HL) ; B - B-1 ; HL - HL+1

Nastavení indikátoru jako u instrukce ini.

otir instrukce blokového výstupu s inkrementací a opakováním (out+inc+repeat) op.kód ED B3

Provádí totéž jako instrukce outi opakovaně tak dlouho, pokud je B 0. Po dosažení stavu B=0 pokračuje další instrukcí. Registr B je opět použit jako počítadlo.

Nastavení indikátoru je jako u instrukce ini.

outd instrukce blokového vystupu s dekrementací bez opakování (uot+dec) op.kód ED AB

Instrukce provádí totéž co instrukce outi s tím rozdílem, že po provedení přenosu dekrementuje jak registr HL,tak registr B, takže po skončení operace je stav registrů:

(C) - (HL) ; B - B-1 ; HL - HL-1

Nastavení indikátorů jako u instrukce ini.

otdr instrukce blokového výstupu s dekremantací a opakováním (out+dec+repeat) op.kód ED EB

Instrukce provádí totéž jako instrukce outd opakovaně tak dlouho, pokud je B 0. Po dosažení stavu B=0 pokračuje další instrukci. Registru B je opět použito jako počítadla. Nastavení indikátoru jako u instrukce ini.

Příklad na použití instrukcí vstupu a výstupu:

Přenesme blok 20 bajtů počínaje adresou 4C B2 do periferijního zařízení o adrese 00 10.

```
ld B, 20      ; nastavení počítadla
ld C, 10      ; nastavení adresy periferijního zařízení do C
ld HL, 40 B2  ; nastavení adresy paměti od níž začne přenos
otir          ; provedení blokového přenosu
```

Stejný případ můžeme řešit použitím instrukce outi:

```
     ld B, 20      ; nastavení počítadla
     ld C, 10      ; adresa periferního zařízení do C
     ld HL, 40 B2  ; adr. paměti odkud začne přenos
op:  outi          ; přenos jednoho bajtu, inc HL, dec B
     jr nz,op      ; test indikátoru Z, je-li Z=0, t.j. B 0
                     skok na návěští op, je-li Z=1, t.j. B=0,
                     pokračování další instrukcí
```

Můžeme také použít pouhé instrukce out:

```
     ld B, 20      ; nastavení počítadla
     ld C, 10      ; adresa perif. zař. do registru C
     ld HL, 40 B2  ; adresa paměti od níž začne přenos
op:  ld A,(HL)     ; přenos obsahu paměť. místa 40 B2 do A
     out (C), A    ; přenos bajtu z reg. A do perif. zař.
     inc HL        ; adresa dalšího paměťového místa
     djnz op       ; dekrementuje počítadlo v B, testuje,
                     je-li B=0. Je-li B 0 skok na návěští
                     op, je-li B=0 pokračuje další instrukcí
```

Vidíme, že stejnou úlohu je možno řešit nejméně třemi i když různě složitými způsoby.

4.5. INSTRUKCE PRENOSU DAT

ld 1., 2. instrukce přenosu dat

Instrukce provádí přenos jednoho nebo dvou bajtů z registru nebo paměťového místa daného operandem 2 do registru nebo paměťového místa daného operandem 1, případně čísla daného operandem do registru nebo paměťového místa daného operandem 1.

Instrukce ld můžeme rozdělit do několika skupin:

- instrukce, které uloží do osmibitového registru jednobajtové číslo N. Jsou možné tyto kombinace:

instukce	op.kód
ld A, N	3E N
ld B, N	06 N
ld C, N	0E N
ld D, N	16 N
ld E, N	1E N
ld H, N	26 N
ld L, N	2E N

- instrukce, které uloží do dvojice registru dvoubajtové číslo NN. Jsou možné tyto kombinace:

instrukce	op.kód
ld BC, NN	01 NN
ld DE, NN	11 NN
ld HL, NN	21 NN

U této instrukce a u všech dalších instrukcí používajících dvoubajtových čísel je třeba do op-kódu uvádět bajty v opačném pořadí než jak je píšeme v assembleru.Chceme-li do registru BC uložit číslo 40 00, napíšeme instrukci v asembleru ve tvaru ld BC, 40 00. Při převedení do op.kódu musíme na prvním místě uvést kód instrukce, t.j. 01 a pak

oba bajty v obráceném pořadí, instrukce v op.kódu bude mít tvar: 01 00 40 !

- instrukce, které přenesou obsah jednoho registru do druhého registru (aniž tím vymaže obsah registru, ze kterého se informace přenáší). V následující tabulce jsou sestaveny všechny možnosti a jejich op.kódy:

R =	A	B	C	D	E	H	L
ld A, R	7F	78	79	7A	78	7C	7D
ld B, R	47	40	41	42	43	44	45
ld C, R	4F	48	49	4A	4B	4C	4D
ld D, R	57	50	51	52	53	54	55
ld E, R	5F	58	59	5A	5B	5C	5D
ld H, R	67	60	61	62	63	64	65
ld L, R	6F	68	69	6A	6B	6C	6D

- Instrukce pro nepřímé adresování. Jsou to instrukce typu ld A, (NN) nebo ld A, (BC) a jejich opak, t.j. instrukce typu ld (NN), A nebo ls (BC), A. Instrukce typu ld A, (NN) přenese do registru A obsah paměťového místa na adrese NN, instrukce ld A, (BC) přenese do registru A obsah paměťového místa, jehož adresa je v registru BC, instrukce ld (NN), A naopak přenese obsah registru do paměťového místa daného adresou NN a instrukce ld (BC), A přenese obsah registru A do paměťového místa, jehož adresa je v registru BC.

V následující tabulce jsou všechny možné kombinace a jejich op.kódy:

instrukce	op.kód
ld (NN), A	32 NN
ld (BC), A	02
ld (DE), A	12
ld (HL), A	77
ld (HL), B	70
ld (HL), C	71
ld (HL), D	72
ld (HL), E	73
ld (HL), H	74

instrukce	op.kód
ld A, (NN)	3A NN
ld A, (BC)	0A
ld A, (DE)	1A
ld A, (HL)	7E
ld B, (HL)	46
ld C, (HL)	4E
ld D, (HL)	56
ld E, (HL)	5E
ld H, (HL)	66

ld (HL), L 75 ld L, (HL) 6E

Opět platí, že v instrukci ld (NN), A a ld A, (NN) je třeba do op.kódu psát bajty v opačném pořadí.

- instrukce pro nepřímé adresování, pracují se čtyřmi bajty. Jsou to instrukce typu ld BC, (NN), které uloží do registru C obsah paměťového místa na adrese NN a do registru B obsah paměťového místa na adrese NN+1 (opět se ukládání děje v opačném pořadí) a instrukce tupu ld (NN), BC ,které uloží obsah registru C do paměťového místa na adrese NN a obsah registru B do paměťového místa na adrese NN+1. Jsou možné tyto kombinace:

instrukce	**op.kód**	**instrukce**	**op.kód**
ld BC, (NN)	ED 4B NN	ld (NN), BC	ED 43 NN
ld DE, (NN)	ED 5B NN	ld (NN), DE	ED 53 NN
ld HL, (NN)	2A NN	ld (NN), HL	22 NN

Do op.kódu se udává pouze adresa prvního paměťového místa (jako obvykle v opačném pořadí), instrukce sama provede přechod na následující stranu.

- instrukce pro přenos dat z a do registru SP, které dovolují manipulaci s uživatelským zásobníkem (ale pozor,tyto instrukce si smí dovolit jen ten, kdo skutečně ovládá perfektně mechaniku uživatelského zásobníku, poněvadž jejich navhodné použití znamená zhroucení systému).

Existují následující možnosti:

instrukce	op.kód	instrukce	op.kód
ld SP, NN	31 NN	ld SP, (IX+d)	BD F9 d
ld SP, (NN)	EB 7B NN	ld SP, (IY+d)	FD F9 d
ld SP, HL	F9	ld (NN), SP	ED 73 NN

Instrukce ld SP, NN uloží do registru SP dvoubajtové číslo NN. Instrukce ld SP, (NN) přenese obsah paměťového místa na adrese NN do nižšího bajtu registru SP a obsah adresového místa na adrese NN+1 do vyššího bajtu registru SP (SP je sestnáctibitový registr). Instrukce ld (NN), SP přenese obsah nižšího bajtu registru SP do paměťového místa na adrese NN a obsah vyššího bajtu registru SP do paměťového místa na adrese NN+1. Instrukce ld SP, HL přenese obsah dvojice registrů do registru SP. Do op.kódu se udává pouze adresa prvního adresového místa (v opačném pořadí).

- Instrukce pro práci s registry IX, IY a pro indexové adresování. Instrukce pro registr IX mají v op.kódu první bajt DD, instrukce pro registr IY mají v první části op.kódu bajt FD, další část op.kódu je totožná, např. instrukce ld SP, IX má op.kód DD F9, zatímco instrukce ld SP,IY má op.kód FD F9. Pro zkrácení uvedu pouze instrukce pro registr IX, z nichž obdržíme instrukce pro registr IY nahrazením DD v op.kódu FD. Existují následující kombinace:

instrukce	op.kód	instrukce	op.kód
ld A, (IX+d)	DD 7E d	ld (IX+d), A	DD 77 d
ld B, (IX+d)	DD 46 d	ld (IX+d), B	DD 70 d
ld C, (IX+d)	DD 4B d	ld (IX+d), C	DD 71 d
ld D, (IX+d)	DD 56 d	ld (IX+d), D	DD 72 d
ld E, (IX+d)	DD 5E d	ld (IX+d), E	DD 73 d
ld H, (IX+d)	DD 66 d	ld (IX+d), H	DD 74 d
ld L, (IX+d)	DD 6E d	ld (IX+d), L	DD 75 d
ld (IX+d), N	DD 36 d N		
ld IX, (NN)	DD 2A NN	ld (NN), IX	DD 22 NN
ld IX, NN	DD 21 NN	ld SP, IX	DD F9

Tyto instrukce se používají pro tzv. indexové adresování. Instrukce ld A, (IX+d) přenese do registru A obsah paměťového místa, jehož adresou je hodnota v registru IX, zvětšená o hodnotu indexu d. Podobně instrukce ld (IX+d), A přenese obsah registru A do paměťového místa, jehož adresa je obsah registru IX zvětšený o hodnotu indexu d. Index d může nabývat krajních hodnot od 80h (=-128d) do 7Fh (=-127d), t.j. používá se dvojkového doplňku. Je-li v registru IX např. adresa 40 00, pak instrukce ld A, (IX+20) přenese do registru A obsah paměťového s adresou 40 20. Instrukce ld A, (IX+FA) odečte od adresy 40 00, uložené v registru IX, dvojkový doplňek FA, t.j. 6 a přenese do registru A obsah paměťového místa na adrese 3F EA (40 00 - 00 06 = 3F FA).

Pozor při převodu do op.kódu! Kód indexu d musí být vždy na třetím místě, i v případě instrukce ld (IX+d), N, kde jednobajtové číslo N je až na posledním místě.
V příloze je přilozena přehledná tabulka všech možných kombinací instrukce ld.
Nastavení indikátoru zůstává u všech instrukcí ld nezměněno.

push instrukce vkládání do uživatelského zásobníku

Instrukce push přenese obsah dvojice registrů do uživatelského zásobníku. Děje se to tím způsobem, že nejprve se dekrementuje registr SP, na adresu jim určenou se na vrchol zásobníku přenese obsah vyššího registru z dané dvojice (t.j. regisru A, B, D, H nebo vyšších 8 bitů registru IX nebo IY), opět se dekrementuje registr SP a na dekrementovaný vrchol zásobníku se přenese obsah nižšího z obou registru (t.j. registru F, C, E, L nebo nižších 8 bitů registru IX nebo IY - registry IX a IY jsou šestnáctibitové.
Možné kombinace instrukce push:

instrukce	op.kód
push	AF F5
push	BC C5
push	DE D5
push	HL E5
push IX	DD E5
push IY	FD E5

Nastavení indikátoru zůstává nezměněno.

pop instrukce vybírání ze zásobníku

Instrukce pop přenese obsah dvou paměťových míst z vrcholu zásobníku do dvojice registrů. Nejprve přenese obsah vrcholu zásobníku do nižšího z dvojic registrů nebo do nižších 8 bitů registru IX nebo IY, pak inkrementuje registr SP a přenese obsah inkrementovaného vrcholu zásobníku do vyššího z dané dvojice nebo do vyšších 8 bitů registru IX nebo IY. Tato instrukce je opakem instrukce push, t.j. informace vložené do zásobníku instrukcí push získáme zpět instrukcí pop.

Možné kombinace instrukce pop:

instrukce	op.kód
pop	AF F1
pop	BC C1
pop	DE D1
pop	HL E1
pop IX	DD E1
pop IY	FD E1

Nastavení indikátoru zůstává nezměněno.

ex instrukce výměny obsahu registrů

Instrukce ex provádí výměnu obsahu dvou registrových párů. Jsou možné tyto kombinace:

instrukce	op.kód
ex AF, AF´	C8
ex DE, HL	EB
ex (SP), HL	E3
ex (SP), IX	DD E3
ex (SP), IY	FD E3

Instrukce ex (SP), HL, ex (SP), IX a ex (SP), IY provádí výměnu obsahu uvedených registrových párů a obsahu uživatelského zásobníku (adresa vrcholu zásobníku je v SP) způsobem popsaným u instrukcí push a pop.
Nastavení indikátoru zůstává nezměněno.

exx instrukce výměny obsahu hlavních a čárkovaných registrů op.kód D9

Instrukce exx provede současnou výměnu obsahu dvojic registrů BC - BC´, DE - DE´, HL - HL´.
Nastavení indikátorů zůstává nezměněno.

Příklady na použití instrukcí přenosu dat.

Příklady na použití instrukcí ld byly již vlastně v předchozím odstavci, kdy pro vložení hodnot do registru bylo použito instrukcí ld, např. ld B, 2C pro vložení hodnoty 2C do registru B, nebo ld HL, 4C 82 pro vložení adresy začátku přenosu dat do registru HL. To byly ty nejjednodušší případy. Chceme-li např. u ZX-81 uložit do registru BC adresu začátku registru obrazovky, která je uložena v systémové proměnné D-FILE (registr obrazovky nemá pevnou adresu jako u Spectra), musíme použít instrukci pro nepřímé adresování ld BC, (40 00). V op.kódu to bude ED 4B 00 40. ED 4B je op.kód instrukce ld BC, (NN) a 00 40 je v opačném pořadí napsaná adresa 40 00 (t.j. to NN v instrukci).
Při práci se strojovými programy potřebujeme často snížit RAMTOP. Děje se to vložením nové adresy do systémové proměnné RAMTPOP. Toto se však nedá přímo, poněvadž

neexistuje instrukce ld (NN), NN, ale musíme použít okliku přes některý registrový pár. Mohli bychom použít kterýkoliv ze tří registrových párů BC, DE, nebo HL, poněvadž všechny tři páry mají instrukci ld RR, NN a současně ld (NN), RR. Pro registrové páry BC a DE mají obě instrukce po čtyřech bajtech, zatímco pro registrový pár HL mají obě instrukce po třech bajtech. Je tudíž vhodnější použít registrového páru HL, čímž se ušetří celkem 2 bajty. Bude pak:

```
ld HL, 7F F0    ; nová adresa RAMTOP do reg. HL
ld (40 04), HL  ; nová adresa do systémové proměnné RAMTOP
```

Máme-li snížen RAMTOP, je naším příztím úkolem přenést strojový program vložený do instrukce REM na začátku basicové oblasti, za RAMTOP. Vezmeme opět příklad pro ZX-81. Máme strojový program o délce 10 bajtů, který chceme přenést od počáteční adresy 40 82 (=16514d) na novou počáteční adresu 7F F0 (=7FFF-10). Jelikož neexistuje instrukce ld (NN), (NN), musíme použít okliku přes některý registr, v tomto případě registr a, který jako jediný umožňuje přenosy. Postupujeme tak, ze přeneseme obsah prvního paměťového místa do registru A, odtud ho další instrukcí přeneseme na novou adresu, zvýšíme o jedničku jak počáteční tak novou adresu a celý postup opakujeme šestnáctkrát (pozor! pohybujeme se stále v hex. soustavě). K odpočítání potřebného počtu opakovaných přenosů použijeme opět registru B jako počítadla spolu s instrukcí djnz, která bude vysvětlena později.

```
        ld DE, 402B    ; nastavení počátku přenosu
        ld HL, 7FF0    ; nastavení nové počáteční adresy
        ld B, 10       ; nastavení počítadla
znovu:  ld A, (DE)     ; přenos obsahu adresy do reg. A
        ld (HL),A      ; přenos obsahu reg. A do první
                         nové adresy
        inc DE         ; další adresa
        inc HL         ; další nová adresa
        djnz znovu     ; test, je-li B 0, není provedeno
                         10 přenosu, skok na návěští
                         znovu, t.j. opakování přenosu.
                         Je-li B=0, je přenos ukončen a
                         pokračuje se další instrukcí
```

Tento příklad lze daleko jednodušeji řešit pomocí instrukcí blokového přenosu, které budou uvedeny v následující kapitole.

Před spuštěním strojovýho programu si musíme vždy uvědomit, že v registrech procesoru jsou informace, které procesor potřebuje pro svojí další práci po ukončení strojovýho programu. Přepíšeme-li tyto informace nově vloženými hodnotami, může se stát, že se program po návratu zhroutí. Aby k tomu nedošlo, musíme uchovat pro další použití obsah registrů. Používáme obvykle instrukci push a pop. Chceme-li uchovat a později do registru HL vrátit jeho obsah, použijeme push HL, pak může pokračovat strojový kód a po jeho ukončení musí následovat pop HL. Obou instrukcí můžeme také použít pro přenos obsahu registrů. Chceme-li přenést obsah registrového páru CD do registrového páru HL, můžeme použít:

```
push CD      ; přenese obsah CD na vrchol zásobníku
pop HL       ; přenese obsah vrcholu zásobníku, t.j.
               původní obsah CD, do HL
```

4.6. INSTRUKCE BLOKOVÉHO PŘENOSU DAT

<u>ldi instrukce blokového přenosu s inkrementací bez opakování (ld+inc) op.kód ED A0</u>

Instrukce provádí přenos jednoho bajtu z adresy bajtu z adresy která se nachází v registrovém páru HL do adresy která se nachází v registrovém páru DE. Instrukce současně inkrementuje HL a DE a dekrementuje BC, kterého se používá jako počítadla.

Stav po provedení instrukce je následující :
(DE) - (HL) : DE - DE+1 : HL - HL+1 : BC - BC-1
Nastavení indikátoru : S,Z a C nezměněn. H a N nulovan.

P/V nastaven, je-li BC-1 =/= 0

ldir instrukce blokového přenosu s inkrementací a opakováním (ld+inc+repeat) op-kód ED B0

Instrukce provádí činnost jako instrukce ldi, ale opakovaně pokud je BC-1 =/= 0. Po dosažení stavu BC-1=0 se pokračuje v provádění následující instrukce.
Nastavení indikátorů jako u instrukce ldi (po skončení je P/V nulován, poněvadž je BC-1=0).

ldd instrukce blokového přenosu s dekrementací bez opakování (ld + dec) op-kód ED A8

Instrukce provádí obdobnou činnost jako instrukce ldi, ale místo inkrementování dekrementuje oba registrové páry DE a HL a současně dekrementuje registový pár BC.
Stav po provedení instrukce:
(DE) - (HL) : DE - DE-1 : HL - HL-1 : BC - BC-1
Nastavení indikátoru jako u ldir.

lddr instrukce blokového přenosu s dekrementací s opakováním (ld+dec+repeat) op-kód ED B8

Instrukce provádí činnost jako instrukce ldd, ale opakovaně, pokud je BC-1 =/= 0. Po dosažení stavu BC-1=0 se pokračuje další instrukcí.
Nastavení indikátoru jako u ldir.

Příklady použití blokového přenosu:

Úlohu z předchozí kapitoly, t.j. přenos 10 bajtů, počínajících adresou 40 2B od nové adresy 7F F0 vyřešíme pomocí blokového přenosu velmi jednoduše:

```
ld BC, 10          ; nastavení počítadla
ld HL, 40 28       ; nastavení první adresy přenosu
ld DE, 7F F0       ; nastavení první adresy nové lokace
ldir               ; blokový přenos až do stavu BC-1=0
```

4.7. INSTRUKCE ARITMETICKÝCH A LOGICKÝCH OPERACÍ

add instrukce aritmetického sčítání

Tato instrukce provádí součet dvou operandů a výsledek uloží do prvního operandu. Může pracovat s jednobajtovými i dvoubajtovými čísly.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
add A,(HL)	86	add A,A	87
add A,(IX+d)	DD 86 d	add A,B	80
add A,(IY+d)	FD 86 d	add A,C	81
add HL, BC	09	add A,D	82
add HL, DE	19	add A,E	83
add HL, HL	29	add A,H	84
add HL, SP	39	add A,L	85
		add A,N	C6 N
add IX, BC	DD 09	add IY, BC	FD 09
add IX, DE	DD 19	add IY, DE	FD 19
add IX, HL	DD 29	add IY, HL	FD 29
add IX, SP	DD 39	add IY, SP	FD 39

Nastavení indikátorů:

	8-bitová operace	16-bitová operace
S	nastaven, je-li výsledek záporný	nezměněn
Z	nastaven, je-li výsledek nulový	nezměněn
H	nastaven, je-li přenos ze 3. bitu	nastaven, je-li přenos z 11.bitu
P/V	nastaven, dojde-li k přetečení	nezměněn
N	nulován	nezměněn
C	nastaven, dojde-li k přenosu ze znaménkového bitu	

adc instrukce aritmetického sčítání s přenosem

Instrukce provádí sčítání dvou operandu jako instrukce add a navíc přičte obsah indikátoru CY. Výsledek uloží do prvního operandu.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
adc A,(HL)	8E	adc A, A	87
adc A,(IX+d)	DD 8E d	adc A, B	88
adc A,(IY+d)	FD 8E d	adc A, C	89
adc HL, BC	ED 4A	adc A, D	8A
adc HL, DE	ED 5A	adc A, E	8B
adc HL, HL	ED 6A	adc A, H	8C
adc HL, SP	ED 7A	adc A, L	8D
		adc A, N	CE N

Nastavení indikátorů:

Stejně jako u instrukce add u osmibitové operace. H je nastaven, dojde-li u šestnáctibitové operace k přenosu mezi 11 a 12 bitem.

sub instrukce aritmetického sčítání

Instrukce odečte operand od obsahu registru A a výsledek uloží do registru A.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
sub A, A	97	sub A, H	94
sub A, B	9C	sub A, L	95
sub A, C	91	sub A, N	D6 N
sub A, D	92	sub A,(HL)	96
sub A, E	93	sub A,(IX+d)	DD 96 d
		sub A,(IY+d)	FD 96 d

Nastavení indikátoru:

S nastaven, je-li výsledek záporný

Z nastaven, je-li výsledek nula

H nastaven při přenosu mezi 3. a 4. bitem

P/V nastaven dojde-li k přečtení
N nastaven
C nastaven, dojde-li k přenosu ze znaménkového bitu.

adc instrukce aritmetického odčítání s přenosem

Instrukce odečte obsah druhého operandu a obsahu indikátoru C od prvního operandu a výsledek uloží do prvního operandu.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
abc A,(HL)	9E	abc A, A	9F
abc A,(IX+d)	DD 9E d	abc A, B	98
abc A,(IY+d)	FD 9E d	abc A, C	99
abc HL, BC	ED 42	abc A, D	9A
abc HL, DE	ED 52	abc A, E	9B
abc HL, HL	ED 62	abc A, H	9C
abc HL, SP	ED 72	abc A, L	9D
		abc A, N	DE N

Nastavení indikátorů:
Stejné jako u instrukce sub, navíc u šestnácibitových operací bude H nastaven, dojde-li k přenosu z 11. do 12. bitu.

daa instrukce opravy pro BCD soustavu op-kód 27

Instrukce opravuje výsledek aritmetických operací add, adc, sub, sbc a dále výsledky instrukcí inc, dec a neg, týkající se registru A tak, aby odpovídaly výsledkům daných operací prováděných v BCD soustavě.

Z BD pracuje v binární soustavě a k jejímu vyjádření používáme HEX soustavu. BCD soustava pracuje pouze s číslicemi 0 až 9, HEX číslice A až F nezná. Pracujeme-li s čísly v soustave BCD, musíme použít instrukci daa, aby nám

hodnoty A až F převedla na BCD vyjádření . Nejlépe je to zřejmé z příkladu. Máme v BCD (t.j. vlastně v desítkové soustavě) provést tři součty :

28+11=39 , 28+13=41 a 28+19=47.

BCD soustava převádí každou desítkovou číslici na 4 bity. 28D převede na 0010 1000, 11D na 0001 0001, 13D na 0001 0011 a 12D na 0001 1001.

Registr A provede uvedené součty následovně:

```
28+11:    0010 1000   28D = 28H
     ---+0001 0001---11D = 11H
          0011 1001 = 39H = 39D. Oprava není třeba

28+13:    0010 1000   28D = 28H
     ---+0001 0011---13D = 13H
          0011 1011 = 3BH
```

nelze vyjádřit v BCD, instrukce daa musí provést opravu, aby výsledkem bylo

```
          0100 0001   41D
```

Provede to tak, že k výsledku binárního součtu přičte 6.

```
Bude pak  0011 1011   3BH
     ---+0000 0110---06H
          0100 0001   41H = 41D.

28+19:    0010 1000   28D = 28H
     ---+0001 1001---19D = 19H
          0100 0001   41H
```

Výsledek je sice vyjádřitelný v BCD soustavě, ale nesprávný, poněvadž došlo k přenosu mezi 3. a 4. bitem, což v BCD neplatí. Instrukce daa opět zjedná nápravu přičtením 6:

```
          0100 0001   41H
     ---+0000 0110---06H
          0100 0111 = 47H = 47D
```

U odečítání se 6 od výsledku operace odečítá. Důležité je to, že mezi aritmetickou operací a instrukcí daa nesmí být instrukce, jejíž provedení ovlivní nastavení indikátorů N a H.

cpl instrukce komplementu op-kód: 2F

Instrukce v registru A nahradí jedničky nulami a naopak.
Nastavení indikátorů zůstává nezměněno.

neg instrukce negace op-kód: ED 44

Instrukce změní znaménko čísla v registru A (odečte obsah registru A od nuly.
Nastavení indikátorů:

S nastaven, je-li výsledek záporný
Z nastaven, je-li výsledek nula
H nastaven, došlo-li k přenosu mezi 3. a 4. bitem
P/V nastaven, byl-li obsah registru A před operací 8CH
N nastaven
C nastaven, nebyl-li před operací obsah reg.A roven nule.

Příklady použití aritmetických operací

U ZX-81 potřebujeme najít adresu pozice tisku, která vypíše určitý symbol do středu obrazovky. Víme, že tuto adresu dostaneme přičtením hodnoty 38CD = 01 70H k adrese uložené v systémové proměnné D-FILE. Použijeme :

```
ld HL ,(40 0C)     ; adresa začátku obrazové informace v
                     paměti do HL
ld BC , 01 70      ; přičítaná hodnota
add HL, BC         ; sečtení obou hodnot a uložení do HL
```

Potřebujeme sečíst dvě 32-bitová čísla. K dispozici jsou však pouze 16-bitové registry. Musíme tudíž nejprve sečíst nižších 16 bitů obou čísel a pak vyšších 16 bitů obou čísel.

Při sečtení nižších 16 bitů může ale dojít k přetečení (t.j. výsledek je větší než je možno vyjádřit 16-ti bitovým registrem). V tom případě dojde k nastavení indikátoru C. Použijeme k tomu instrukce adc. Zvolíme-li dvě konkrétní čísla například 23 45 67 89H a 01 28 A2 38, bude pak:

```
ld HL, 67 89      ; nižších 16 bitů 1.operandu do HL
ld BC, A2 38      ; nižších 16 bitů 1.operandu do BC
add HL, BC        ; součet 16-ti nižších bitů
push HL           ; obsah HL do zásobníku
ld HL, 23 45      ; vyšších 16 bitů 1.operandu do HL
ld BC, C1 28      ; vyšších 16 bitů 2.operandu do BC
adc HL, BC        ; součet 16-ti vyšších bitů s
                    přičtením pře nosového bitu z
                    předchozího součtu
ld DE, HL         ; výsledek druhého součtu do DE
pop BC            ; výsledek prvního součtu ze
                    zásobníku do BC
```

Nižších 16 bitů výsledku bude v registrovém páru BC a vyšších 16 bitů výsledku v DE.

Instrukce daa byla již dostatečně vysvětlena v textu, jen zápis v assembleru (28+13):

```
ld A, 28          ; první operand do reg. A
add A, 13         ; přičtení druhého operandu
daa               ; převede výsledek do BCD soustavy
```

Máme-li v registru A hodnotu 0101 0101, bude v něm po instrukci cpl 1010 1010.
Je-li v registru A hodnota 0000 0001 /=1D/, bude v něm po provedení instrukce neg 1111 1111 /=-1D/.

and instrukce logického součinu

Instrukce provede logický součin odpovídajících si bitů registru A a operandu. Výsledek uloží do registru A.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
and A	A7	and H	A4
and 8	AC	and L	A5
and C	A1	and N	E6 N
and D	A2	and (HL)	A6
and E	A3	and (IX+d)	DD A6 d
		and (IY+d)	FD A6 d

Nastavení indikátorů:

S nastaven, je-li výsledek záporný
Z nastaven, je-li výsledkem nula
H nastaven
P/V nastaven v případě sůde parity
N nulován
C nulován

or instrukce logického součtu

Instrukce provede logický součet odpovídajících si bitů registru A a operandu. Výsledek uloží do registru A.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
or A	B7	or H	B4
or 8	BC	or L	B5
or C	B1	or N	F6 N
or D	B2	or (HL)	B6
or E	B3	or (IX+d)	DD B6 d
		or (IY+d)	FD B6 d

Nastavení indikátorů jako u instrukce and s tím rozdílem, že indikátor H je or nulován /u and nastaven/.

xor instrukce binárního /exkluzivního/ součtu

Instrukce provede binární součet odpovídajících si bitů registru A a operandu. Výsledek uloží do registru A.

Možné kombinace:

instruce	op-kód	instrukce	op-kód
xor A	AF	xor H	AC
xor B	A8	xor L	AD
xor C	A9	xor N	EE N
xor D	AA	xor (HL)	AE
xor E	AB	xor (IX+d)	DD AE d
		xor (IY+d)	FD AE d

Nastavení indikátorů:

S nastaven při záporném výsledku
Z nastaven, je-li výsledek nulový
H nulován
P/V nastaven, je-li parita sudá
N nulován
C nulován

cp instrukce srovnávání

Instrukce provede odečtení operandu od obsahu registru A. Výsledek nikam neukládá, pouze nastavuje indikátory.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
cp A	BF	cp H	BC
cp B	B8	cp L	BD
cp C	B9	cp N	FE N
cp D	BA	cp (HL)	DE

cp E	BB	cp (IX+d)	DD BE d
		cp (IY+d)	FD BE d

Nastavení indikátorů:

S nastaven při záporném výsledku

Z nastaven, je-li výsledek nulový, je-li shoda A a operandu

H nastaven při přenosu ze 3. do 4. bitu

P/V nastaven došlo-li k přečtení

N nastaven

C nastaven při přenosu ze znaménkového bitu

Příklady použití logických operací:

Chceme-li vynulovat vyšší čtyři bity registru A a nižší čtyři bity ponechat, provedeme

```
   and CF
```

anuluje 4 vyšší bity reg. A, nižší 4 bity ponechá

Pro lepší představu jak to funguje:

```
obsah reg.A      0101 0101
and CF     ----/|0000 1111---
výsledek         0000 0101
```

Chceme-li vynulovat registr A, je xor A lepší než ld A,00 /xor má op-kód 1 bajt, ld A,00 op-kód 2bajty/:

```
obsah reg.A      0101 0101
xor A      ---(+)0101 0101---
výsledek         0000 0000
```

Instrukci cp používáme při větvení programu, příp. při tvoření smyček. Máme provést skok na návěští "vpravo",bude-li v registru A honota 20, nebo skok na návěští "vlevo",bude-li v registru A hodnota 40:

```
   cp 20          ; porovnání obsahu A s hodnotou 20
   jr Z, vpravo   ; je-li A=20, t.j. Z=1, skoč na "vpravo"
```

```
cp 40          ; porovnání A s hodnotou 40
jr Z, vlevo    ; je-li A=40, t.j. Z=1, skoč na "vlevo"
```

4.8. INSTRUKCE BLOKOVÉHO PROHLEDÁVÁNÍ

cpi instrukce blokového prohledávání s inkrementací bez opakování /cp+inc/ op-kód: ED A1

Instrukce provede odečtení obsahu paměťového místa, jehož adresa je v registrovém páru HL od obsahu registru A. Současně inkrementuje registrový pár HL a dekrementuje registrový pár BC /kterého se používá jako počítadla/.
Stav po provedení instrukce:

A - (HL) ; HL - HL + 1 ; BC - BC - 1

Výsledek operace se neukládá do žádného z registrů, pouze dochází k nastavení indikátorů, které je následující:

S nastaven, je-li výsledek záporný
Z nastaven, je-li výsledek nula, t.j. je-li A = (HL)
H nastaven, došlo-li k přenosu mezi 3. a 4. bitem
P/V nastaven, je-li BC - 1 =/= 0
N nastaven
C nezměněn

cpir instrukce blokového prohledávání s inkrementací a opakováním /cp+inc+repeat/ op-kód ED B1

Instrukce provádí stejnou činnost jako instrukce cpi, ale opakovaně pokud je BC-1=/=0 nebo pokud je A-(HL)=/=0. Pokud nastane některý z případů BC-1=0 nebo A-(HL)=0, pokračuje ne prováděním další instrukce.
Nastavení indikátorů stejně jako u instrukce cpi.

cpd instrukce blokového prohledávání a dekrementaci bez opakování /cp+dec/ op-kód: ED A9

Instrukce provádí obdobnou činnost jako instrukce cpi, jenom s tím rozdílem, že inkrementuje jak registrový pár HL, tak registrový pár BC. Stav po provedení instrukce:
A - (HL) ; HL - HL - 1 ; BC - BC -1
Nastavení indikátorů stejně jako u instrukce cpi.

cpdr instrukce blokového prohledávání s dekrementací a opakováním /cp+dec+repeat/ op-kód: ED B9

Instrukce provádí stejnou činnost jako instrukce cpd, ale opakovaně pokud je BC-1=/=0 nebo pokud je A-(HL)=/=0. Pokud nastane jeden z případů BC-1=0 nebo A-(HL)=0, opakování se ukončí a pokračuje se prováděním následující instrukce.
Nastavení indikátoru je stejně jako u instrukce cpi.

Příklady použití instrukcí blokového prohledávání:
Máme v paměťovém bloku 100 bajtů, končícím adresou 7F FF, najít první nulový bajt odshora. Použijeme-li instrukce cpdr, bude:

```
ld HL, 7F FF   ; nastavení počátku prohledávání
ld BC, 01 00   ; nastavení počítadla
ld A, 0        ; nastavení hledané /porovnávané/ hodnoty
cpdr           ; blokové prohledávání až do stavu (HL)=A nebo
                 BC-1=0
```

4.9. INSTRUKCE INKREMENTACE A DEKREMENTACE

inc instrukce inkrementace

Instrukce provede zvýšení hodnoty operandu o jedničku.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
inc A	3C	inc BC	03
inc B	04	inc DE	13
inc C	0C	inc HL	23
inc D	14	inc IX	DD 23
inc E	1C	inc IY	FD 23
inc H	24	inc SP	33
inc L	2C	inc (HL)	34
		inc (IX+d)	DD 34 d
		inc (IY+d)	FD 34 d

Nastavení indikátorů:

	8-bitová operace	16-bitová operace
S	nastaven, je-li výsledek záporný	nezměněn
Z	nastaven, je-li výsledek nulový	nezměněn
H	nastaven při přenosu mezi 3.a 4.bitem	nezměněn
P/V	nastaven při přečtení	nezměněn
N	nulován	nezměněn
C	nezměněn	nezměněn

dec instrukce dekrementace

Instrukce provede snížení hodnoty operandu o jedničku.

Možné kombinace:

instrukce	op-kód	instrukce	op-kód
dec A	3D	dec BC	0B
dec B	05	dec DE	1B
dec C	0D	dec HL	2B
dec D	15	dec IX	DD 2B
dec E	1D	dec IY	FD 2B
dec H	25	dec SP	3B
dec L	2D	dec (HL)	35
		dec (IX+d)	DD 35 d
		dec (IY+d)	FD 35 d

Nastavení indikátoru stejně jako u instrukce inc, pouze s jedním rozdílem , N je nastaven /u inc nulován/.

Příklad:

Máme zvýšit adresu uloženou v registrovém páru HL o 2:

```
inc HL     ; zvýšení o 1
inc HL     ; zvýšení o 1
```

4.10. INSTRUKCE SKOKŮ

<u>jp instrukce nepodmíněného skoku</u>

Instrukce provede skok na paměťové místo, jehož adresa je daná operandem /tím, že uloží hodnotu danou operandem do registru PC/.

Možné kombinace:

instrukce	op-kód
jp NN	C3 NN
jp (HL)	E9
jp (IX)	DD E9
jp (IY)	FD E9

POZOR ! změna dohodnutého způsobu označování:

V instrukci jp neznamená (HL) obsah paměťového místa, jehož adresa je v registrovém páru HL, ale přímo adresu skoku, která je uložena v registrovém páru HL /podobně to platí u jp(IX) a jp(IY)/. Nastavení indikátorů zůstává nezměněno.

jp 1.,2. instrukce podmíněného skoku

Instrukce provede skok na adresu danou druhým operandem pouze tehdy, je-li splněna podmínka daná prvním operandem. Touto podmínkou v prvním operandu je stav indikátorů S, Z, P/V a C. Neni-li splněna, provádí se další instrukce.

instrukce	op-kód	stav indikátorů
jp M, NN	FA NN	S nastaven /výsledek 0/
jp P, NN	F2 NN	S nulován /výsledek =0/
jp Z, NN	CA NN	Z nastaven /výsledek =0/
jp NZ, NN	C2 NN	Z nulován /výsledek =/=0/
jp PE, NN	EA NN	P/V nastaven /sudá priorita/
jp PO, NN	E2 NN	P/V nulován /lichá parita/
jp C, NN	DA NN	C nastaven /přenos/
jp NC, NN	D2 NN	C nulován /bez přenosu/

Nastavení indikátorů zůstává nezměněno.

jr DIS instrukce relativního skoku op-kód 18 N

Istrukce provede relativní skok, jehož velikost je určena operandem. Adresu skoku získáme přičtením hodnoty operandu k adrese instrukce relativního skoku zvětšené o 2. Operand může nabývat hodnot od 80H /=-128D/ do 7FH /=127/.

U dokonalejších překladačů /Assembleru/ je možno zadat jako operand návěští a překladač si velikost odskoku vypočte sám /např. jr DIS, ZNOVU/.
Nastavení indikátorů zůstává nezměněno.

jr 1.,DIS instrukce relativního podmíněného skoku

Instrukce provede relativní skok na adresu danou operandem DIS jen tehdy, je-li splněna podmínka daná prvním operandem. Není-li tato podmínka splněna, pokračuje se další instrukcí. Pro adresu skoku a velikost operandu DIS platí totéž jako u instrukce jr DIS. Podmínkou v prvním operandu je nastavení indikátoru Z a C.

Možné kombinace:

instrukce	**op-kód**	**stav indikátorů**
jr Z, DIS	28 N	Z nastaven /výsledek nulový/
jr NZ,DIS	20 N	Z nulován /výsledek nenulový/
jr C, DIS	38 N	C nastaven /přenos/
jr NC,DIS	30 N	C nulován / bez přenosu/

Nastavení indikátorú zůstává nezměněno.
Některé překladače si velikost operandu DIS vypočítávájí samy /podobně jako u instrukce jr DIS/

djnz DIS instrukce relativního skoku podmíněného čítačem

Instrukce provádí opakovaně dekrementaci registru B /kterého se používá jako počítadla/ a v případě, že B-1=/=0 provede relativní skok na adresu danou operandem DIS. Dosáhne-li se stavu B-1=0, pokračuje se prováděním následující instrukce. Pro adresu odskoku a velikost operandu DIS platí totéž jako u jr. U dokonalejších překladačů /Assembleru/ je možno zadat jako operand návěští a překladač si velikost odskoku vypočte sám /např. jr DIS, ZNOVU/.
Nastavení indikátorů zůstává nezměněno.

jr 1.,DIS instrukce relativního podmíněného skoku

Instrukce provede relativní skok na adresu danou operandem DIS jen tehdy, je-li splněna podmínka daná prvním operandem. Není-li tato podmínka splněna, pokračuje se další instrukcí. Pro adresu skoku a velikost operandu DIS platí totéž jako u instrukce jr DIS. Podmínkou v prvním operandu je nastavení indikátoru Z a C.

Možné kombinace:

instrukce	op-kód	stav indikátorů
jr Z, DIS	28 N	Z nastaven /výsledek nulový/
jr NZ,DIS	20 N	Z nulován /výsledek nenulový/
jr C, DIS	38 N	C nastaven /přenos/
jr NC,DIS	30 N	C nulován / bez přenosu/

Nastavení indikátorů zůstává nezměněno.
Některé překladače si velikost operandu DIS vypočítávají samy /podobně jako u instrukce jr DIS/.

djnz DIS instrukce relativního skoku podmíněného čítačem

Instrukce provádí opakovaně dekrementaci registru B /kterého se používá jako počítadla/ a v případě, že B-1=/=0 provede relativní skok na adresu danou operandem DIS. Dosáhne-li se stavu B-1=0, pokračuje se prováděním následující instrukce.

Pro adresu odskoku a velikost operandu DIS platí totéž jako u insrukce jr DIS. Některé překladače si velikost odskoku vypočtou samy.

Nastavení indikátorů zůstává nezměněno.

Příklady použití instrukcí skoku:

Máme ze strojového programu skočit na podprogram v ROM, který je na adrese 00 18. Postačí

```
jp 00 18        ; skok na dresu 00 18
```

Chceme-li upravit program uvedený v kapitole blokového prohledávání tak, aby při nalezení bajtu s obsahem FF skočil do ROM na adresu 00 20:

```
ld HL, 7F FF    ; pocatecni adresa doHL
ld BC, 01 00    ; nastaveni pocitadla
ld A, FF        ; nastaveni hledane hodnoty
cpdr            ; blokove prohledavani
jp Z, 00 20     ; je-li Z=1 /coz nastane pri A=FF/,
                  skok na 00 20
```

Adresy absolutních skoků jsou pevné a nemění se při posunu strojového programu v paměti. Používáme je většinou při skocích do ROM, t.j. na neměnné adresy, poněvadž při použití ke skokům na adresy musíme při posunech progamu v paměti provést přeadresování.

Naproti tomu relativní skoky nemají pevnou adresu, ale mají určen pouze interval /počet bajtů/ od adresy instrukce skoku zvětšené o 2. Při posunech progamu v paměti se posunují i adresy relativních skoků, takže není třeba relativní skoky přeadresovávat. Navíc zaberou v programu o bajt méně než skoky absolutní.
Velikost odskoku je omezena hodnotou -128D=80H a 127D=7FH bajtů /nikoliv instrukcí!/ a obvykle ho musíme stanovit až po převedení assembleru do relativního strojového programu podle skutečného počtu přeskakovaných bajtů. Postupujeme podle schématu uvedeného na následující straně. Pokud potřebujeme realizovat větší odskok než -128D nebo než +127D, musíme zkombinovat dva nebo i více relativních skoků tak, že do adresy na kterou se provede skok umístíme další relativní skok.

op-kód	navěští	mnemotech. zkratka	honota DIS	
21		ld HL, NN	F4	
FF		FF	F5	↑
7F		7F	F6	
01		ld BC, NN	F7	
01		01	F8	
01		01	F9	
3E	ZNOVU	ld A, N	FA	
FF		FF	FB	
ED		cpdr	FC	
B9			FD	
18		jr DIS	FE	
?		(?)	FF	skok dozadu
3E		ld A, N	00	
28		28	01	skok dopředu
C6		add A, N	02	
13		13	03	
27		daa	04	
C9	KONEC	ret	05	v

Pokud chceme v tomto případě skočit na návěští ZNOVU, bude hodnota DIS rovna FA, pokud chceme skočit na návěští KONEC, bude to 05. Tato hodnota se objeví místo otazníku. Pracujeme-li s překladačem, který dovoluje skoky na návěští, je to jednodušší. Nemusíme nic počítat a stačí

jr ZNOVU nebo jr KONEC

Instrukce djnz odpovídá basicové instrukci FOR-NEXT. Chceme 3x provést podprogram uložený na adrese 7F E0

06		ld B, N	; nastavení počítadla
03		03	; na hodnotu 03
CD	ZNOVU	call NN	; volání podprogramu
E0		E0	; nižší bajt adresy
7F		7F	; vyšší bajt adresy
10		djnz DIS	; skok na ZNOVU
FB		(FB)	; hodnota DI

Instrukce volání a návratu

call NN instrukce volání podprogramu op-kód: CD NN

Instrukce provádí vyvolání podprogramu uloženého na adrese NN (t.j.skok na adresu NN) a po instrukci návratu, kterou musí být každý program ukončen,pokračuje program instrukcí následující po instrukci call.Jeden podprogram je mozno vyvolat z více míst programu a z jednoho podprogramu lze volat další podprogram. Poněvadž instrukce call ukládá adresu instrukce následující po instrukci call na vrchol zásobníku obdobnym zpusobem jako instrukce push,musíme bdát na to,aby při návratu z podprogramu byla takto ulozena adresa opět na vrcholu zásobníku.Platí to pro případ,že v podprogramu manipulujeme instrukcemi push a pop se zásobníkem.

Nastavení indikátorů zůstává nezměněno.

call 1.,NN instrukce podmíněného volání podprogramu

Instrukce provádí vyvolání podprogramu jako call NN, ale jen tehdy,je-li splněna podmínka daná 1.operandem , kterou je nastavení indikátorů. Není-li tato podmínka splněna,pokračuje program následující instrukcí. Možné kombinace:

instrukce	**op-kód**	**stav indikátorů**
call M,NN	FC NN	S nastaven /výsledek 0/
call P,NN	F4 NN	S nulován /výsledek =0/
call Z,NN	CC NN	Z nastven /výsledek =0/
call NZ,NN	C4 NN	Z nulován /vysldek 0/
call PE,NN	EC NN	P/v nastaven /sudá parita/
call PO,NN	E4 NN	P/v nulován /lichá parita/
call C,NN	DC NN	C nastaven /přenos/
call NC,NN	D4 NN	C nulován /bez přenosu/

Nastavení **indikátorů zůstává** nezměněno.

ret návratu z **podprogramu** op-kód C9

Instruce provede návrat z podprogramu na instrukci následující po instrukci call,která podprogram vyvolala. Nastavení indikátorů zůstává nezměněno.

ret l. instrukce podmíněného návratu z podprogramu

Instrukce provede návrat z podprogramu podobně jako instrukce ret,ale pouze tehdy,je-li splněna podmínka daná v operandu.Touto podmínkou je nastavení indikátorů.

instrukce	op-kód	stav indikátorů
ret M	F8	S nastaven /výsledek 0/
ret P	FC	S nulován /výsledek =0/
ret Z	C8	Z nastaven /výsledek 0/
ret NZ	C0	Z nulován /výsledek 0/
ret PE	E8	P/V nastaven /sudá parita/
ret PO	E0	P/V nulován /lichá parita/
ret C	D8	C nastaven /přenos/
ret NC	D0	C nulován /bez přenosu/

Není-li splněna podmínka v operandu,pokračuje program následující instrukcí.Nastavení indikátorů zůstává nezměněno.

reti instrukce návratu z mask. přerušení op.kód ED 4D

Instrukce provádí návrat z obslužného programu přerušení obdobně jako instrukce ret a současně oznamuje periferním zařízení, že končí jeho obsluhu. Instrukce neprovádí uvolnění přerušení,proto je před ní třeba vložit instrukci ei.Nastavení indikátorů zůstává nezměněno.

retn instrukce návratu z nemask. op.kód ED 45

Instrukce provádí návrat z nemaskovaného přerušení obdobně jako instrukce ret.Nastavení indikátorů zůstává nezměněno.

rst instrukce restartu

Instrukce provede skok na adresu danou operandem.

Možné adresy instrukce rst:

instrukce	op-kód
rst 00	C7
rst 08	CF
rst 10	D7
rst 18	DF
rst 20	E7
rst 28	EF
rst 30	F7
rst 38	FF

Nastavení indikátorů zůstává nezměněno.

Instrukce v této kapitole jsou tak jednoduché,že na ně není třeba uvádět příklady.

4.11. INSTRUKCE ROTACÍ A POSUNŮ

sla instrukce aritmetického posunu vlevo
/shift left arithmetic/

Instrukce posune bity v daném registru,který je uveden v operandu,tak,že bit 7 se přesune do CY a bit 0 se naplní 0.

```
                                CY   7  6  5  4  3  2  1  0
Obsah registru:před operaci ( ) (X)(X)(X)(X)(X)(X)(X)(X)
                                  /  /  /  /  /  /  /  /
                po operaci     (X) (X)(X)(X)(X)(X)(X)(X)(0) 0
```

Možné kombinace operandu:

instrukce	op-kód	instrukce	op-kód
sla A	CB 27	sla H	CB 24
sla B	CB 20	sla L	CB 25
sla C	CB 21	sla (HL)	CB 26
sla D	CB 22	sla (IX+d)	DD CB d 2B
sla E	CB 23	sla (IY+d)	FD CB d 2B

POZOR! u instrukcí sla (IX+d) a sla (IY+d) musí být v op. kódu hodnota indexu d vždy na třetím místě.

Nastavení indikátorů:

S nastaven, je-li výsledek záporný

Z nastaven, je-li výsledek nulový

H nulován

P/V nastaven, je-li parita sudá

N nulován

C obsahuje 7.bit operandu

sra instrukce aritmetického posunu vpravo

/shift right arithmetic/

Instrukce posune bity v registru daném operandem doprava tak,že bit 7 zůstane zachován a bit 0 se přesune do CY.

```
                                7  6  5  4  3  2  1  0   CY
Obsah registru:před operaci (X)(X)(X)(X)(X)(X)(X)(X) ( )
                             |  |  |  |  |  |  |  |  |
                po operaci  (X)(X)(X)(X)(X)(X)(X)(X) (X)
```

Možné operandy:

instrukce	op-kód	instrukce	op-kód
sra A	CB 2F	sra H	CB 2C
sra B	CB 28	sra L	CB 2D
sra C	CB 29	sra (HL)	CB 2E
sra D	CB 2A	sra (IX+d)	DD CB d 2E
sra E	CB 2B	sra (IY+d)	FD CB d 2E

Pozor na postaveni indexu d!

Nastavení indikátorů:

Jako u instrukce sla,jenom s tím rozdílem,že CY obsahuje 0.bit operandu.

srl instrukce logického posunu vpravo

/shift right logic/

Instrukce posune bity v registru daném operandem doprava tak,že bit 7 se naplní nulou a bit 0 se přesune do CY.

		7	6	5	4	3	2	1	0	CY
Obsah registru:	před operaci	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	( )
	po operaci	0	(0)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

Možné kombinace operandu:

instrukce	op-kód	instrukce	op-kód
srl A	CB 3F	srl H	CB 3C
srl B	CB 3B	srl L	CB 3D
srl C	CB 39	srl (HL)	CB 3E
srl D	CB 3A	srl (IX+d)	DD CB d 3E
srl E	CB 3B	srl (IY+d)	FD CB d 3E

Nastavení indikátorů jako u instrukce sra

rl rotace operandu vlevo přes CY

/rotate left/

Instrukce provádí rotaci bitu v registru daném operandem vlevo přes indikátor C.Obsah indikátoru C se přesune do bitu 0 operandu.

```
                    CY   7   6   5   4   3   2   1   0
Obsah reg.: před operaci (X) (X)(X)(X)(X)(X)(X)(X)(X)
                          /   /   /   /   /   /   /   /
            po operaci   ( ) ( )( )( )( )( )( )( )( ) ┐
```

Možné operandy:

instrukce	op-kód	instrukce	op-kód
rl A	CB 17	rl H	CB 14
rl B	CB 10	rl L	CB 15
rl C	CB 11	rl (HL)	CB 16
rl D	CB 12	rl (IX+d)	DD CB d 16
rl E	CB 13	rl (IY+d)	FD CB d 16

Pozor na postaveni indexu d!

Nastavení indikátorů jako u instrukce sla.

rla rotace registru A vlevo přes CY op.kód 17

/rotate left A/

Instrukce provádí totéž jako instrukce rl,ale platí pouze pro registr A.

Nastavení indikátorů:

S,Z a P/V nezměněny

R a N nulovány

C obsahuje bit 7 registru A

rlc instrukce rotace operandu vlevo do CY

/rotate left carry/

Instrukce provede rotaci bitu operandu doleva tak,že bit 7 přesune do bitu 0 a současně do CY.

```
                                CY   7  6  5  4  3  2  1  0
Obsah registrů:před operací    (X) (X)(X)(X)(X)(X)(X)(X)(X)
                                /   /  /  /  /  /  /  /
                po operaci    ( )|( )( )( )( )( )( )( )( ) ┐
                                 └──────────────────────────┘
```

Možné operandy:

instrukce	op-kód	instrukce	op-kód
rlc A	CB 07	rlc H	CB 04
rlc B	CB 00	rlc L	CB 05
rlc C	CB 01	rlc (HL)	CB 06
rlc D	CB 02	rlc (IX+d)	DD CB d 06
rlc E	CB 03	rlc (IY+d)	FD CB d 06

Pozor na postavení indexu d !

Nastavení indikátorů jako u instrukce sla.

rlca instrukce rotace registru A vlevo do CY op.kód: 07

/rotate left carry A/

Instrukce provádí totéž jako instrukce rlc,ale platí pouze pro registr A.

Nastavení indikátorů stejné jako u instrukce rla.

rr rotace operandu vpravo přes CY

/rotate right/

Instrukce provádí rotaci bitu operandu vpravo přes CY tak,že bit 0 operandu se přesune do CY a obsah CY do bitu 7 operandu.

7 6 5 4 3 2 1 0 CY

```
Obsah registru:před operaci (X)(X)(X)(X)(X)(X)(X)(X) (X)
                                                      |
                po operaci  ( )( )( )( )( )( )( )( ) ( )
```

Možné operandy:

instrukce	op-kód	instrukce	op-kód
rr A	CB 1F	rr H	CB 1C
rr B	CB 1B	rr L	CB 1D
rr C	CB 19	rr (HL)	CB 1E
rr D	CB 1A	rr (IX+d)	DD CB d 1E
rr E	CB 1B	rr (IY+d)	FD CB d 1E

Pozor na postavení indexu d !
Nastavení indikátorů jako u instrukce sra.

rra instrukce rotace registru A vpravo přes CY op.kód 1F
/rotate right A/

Instrukce provádí totéž jako instrukce rr,ale platí pouze pro registr A.
Nastavení indikátorů:

S,Z a P/V nezměněny
R a N nulovány
C obsahuje bit 0 registru A.

rrc instrukce rotace operandu vpravo do CY
/rotate right carry/

Instrukce provádí rotaci bitu operandu doprava tak,že bit 0 přenese do bitu 7 a současně do CY.

```
                                 7  6  5  4  3  2  1  0   CY
Obsah registru:před operaci (X)(X)(X)(X)(X)(X)(X)(X) (X)
                                                      |
                po operaci  ( )( )( )( )( )( )( )( ) ( )
```

Možné operandy:

instrukce	op-kód	instrukce	op-kód
rrc A	CB 0F	rrc H	CB 0C
rrc B	CB 08	rrc L	CB 0D
rrc C	CB 09	rrc HL	CB 0E
rrc D	CB 0A	rrc IX+d	DD CB d 0E
rrc E	CB 0B	rrc IY+d	FD CB d 0E

Nastavení indikátorů jako u instrukce sra.
Pozor na postavení indexu d !

rrca instrukce rotace registru A vpravo do CY op.kód 0F

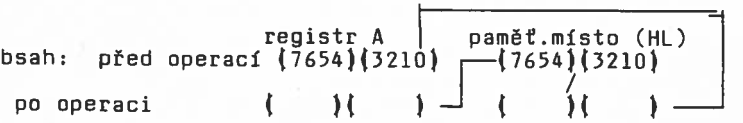
/rotate right carry A/

Instrukce provádí totéž jako instrukce rra,ale platí pouze pro registr A.
Nastavení indikátorů stejné jako u instrukce rra.

rld instrukce BCD rotace vlevo op-kód ED 6F

/rotate left BCD/

Instrukce provádí BCD rotaci mezi nižšími čtyřmi bity registru A a bity paměťového místa,jehož adresa je v registru HL.Způsob přenosu je následující:nižší čtyři bity paměťového místa se přenesou do vyšších čtyř bitů tohoto paměťového místa,vyšší čtyři bity paměťového místa se přenesou do nižších čtyř bitu registru A a nižší čtyři bity registru A se přenesou do nižších čtyř bitů paměťového místa.Vypadá to složitě a názorněji je to vidět na schématu:



Nastavení indikátorů:

S nastaven,je-li obsah registru A záporný

Z nastaven,je-li obsah registru A nula

H nulován

P/V nastaven,má-li registr A sudou paritu

N nulován

C nezměněn

rrd instrukce BCD rotace vpravo op-kód ED 67

/rotate right BCD/

Instrukce provádí rotaci nižších čtyř bitů regisru A a bitu paměťového místa,jehož adresa je v registru HL. Způsob přenosu je následující:nižší čtyři bity registru A se přenesou do vyšších čtyř bitů paměťového místa.vyšší čtyři bity paměťového místa se přenesou do jeho nižších čtyř bitů a nižší čtyři bity paměťového místa se přenesou do nižších čtyr bitů registru A.

```
                          registr A     paměť.místo (HL)
Obsah:před operací       (7654)(3210)  (7654)(3210)
                                       |
      po operaci         (    )(    )  (    )(    )
                                       ^
```

Nastavení indikátorů jako u instrukce rld.

U všech instrukcí rotací a posunů platí,že (HL) znamená na obsah paměťového místa,jehož adresa je v registru HL, a že (IX+d) je obsah paměťového místa jehož adresu dostaneme přičtením indexu d k adrese v registru IX.

Příklady na použití instrukci rotací a posunů:

Chceme na obrazovku vypsat v HEX vyjádření obsah registru. Počítač hodnotu obsazenou v registru interpoluje jako kód znaku a tento znak,jehož kód je v registru, vypíše

na obrazovku.Chceme-li na obrazovku dostat hexadecimální obsah registru,musíme jeho obsah rozdělit na dvě poloviny.Horní čtyři bity reprezentují první HEX číslici,dolní čtyři bity druhou HEX číslici.Tyto čtyři bity musíme samostatně vyjádřit jako kód znaku a přičíst k tomuto kódu kód nuly.U ZX 81 je to 28D=1CH, u Spectra je to 4BD=30H.Postupujeme tak,že po uložení obsahu registru do zásobníku přesuneme čtyřnásobnou instrukcí srl horní čtyři bity do dolních čtyř bitů. V horních čtyřech bitech jsou nuly.Nyní k takto získanému kódu můžeme přičíst kód nuly a máme první HEX číslici ,kterou můžeme přenést do části paměti vyhražené pro obsah obrazovky.V assembleru to bude /pro ZX 81/:

```
ld A, 9C          ;HEX hodnota 9C do registru A
push AF           ;uchování registrů A pro pozdější použití
srl A             ;posun reg.A o jeden bit vpravo
srl A             ;dtto
srl A             ;dtto
srl A             ;dtto
add A, 1C         ;přičtení kódu nuly ke kódu znaku v A
ld DE, (40 0C)    ;syst.proměnná D-FILE do registru DE
inc DE            ;zvýšení o 1,pozice tisku prvního znaku
ld (DE) , A       ;   uložení kódu první HEX číslice na adresu
                      D-FILE + 1,t.j. znak v levém horním
                      rohu obrazovky.
```

Získání druhé HEX číslice bude o něco jednodušší.Po obnovení původního obsahu registru A instrukci pop v něm máme opět obě HEX číslice,t.j. 9C.První číslici "9" již máme.Horní čtyři bity tudíž nepotřebujeme a můžeme je vynulovat instrukcí and CF.Pak už jen zbývá přičíst 1C a přenést takto získaný kód do další pozice tisku na obrazovku.Assembler bude pokračovat:

```
pop AF            ;obnovení původního obsahu registru A
and 0F            ;vynulování horních čtyř bitů registru A
add A, 1C         ;přičtení kódu nuly ke kódu znaku v A
inc DE            ;adresa další pozice tisku
ld (DE), A        ;přenesení druhého znaku
```

Jinak se instrukci z této kapitoly používá převážně při provádění aritmetických operací násobení a dělení.Máme

vynásobit dvě 8-bitová kladná čísla,např.02 a 06.Bude to vypadat následovně:

```
     ld E, 02          ;první číslo do E
     ld C, 03          ;druhé číslo do C
     ld HL, 00         ;vynulování HL pro ukládání výsledků
     ld D, 0           ;vynulování D
     ld B, 8           ;nastavení počítadla
ZNOVU: srl C             ;další bit do CY
     jr NC, DAL        ;je-li CY=0, nesčítat
     add HL, DE        ;jejli CY=1, přičíst k HL
DAL: sla E             ;násobení DE
     rl D              ;dvěma
     djnz ZNOVU        ;test na 8=0,t.j. provedení osmi cyklů
```

Komentář snad není třeba,nechť se každý procvičí přemýšlením jak to funguje.

4.12. INSTRUKCE PRO PRÁCI S JEDNOTLIVÝMI BITY

Následující instrukce umožňují testování,nastavování nebo nulování libovolného bitu v registrech a paměťových místech.

bit b instrukce pro test bitu

Instrukce přenese komplement bitu "b" operandu do indikátoru Z.

Možné kombinace:

instrukce	op-kód								
R=	A	B	C	D	E	H	L	(HL)	(IX+d)
bit 0,R	CB47	CB40	CB41	CB42	CB43	CB44	CB45	CB46	DDCBd46
bit 1,R	CB4F	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	DDCBd4E
bit 2,R	CB57	CB50	CB51	CB52	CB53	CB54	CB55	CB56	DDCBd56
bit 3,R	CB5F	CB5B	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	DDCBd5E
bit 4,R	CB67	CB60	CB61	CB62	CB63	CB64	CB65	CB66	DDCBd66
bit 5,R	CB6F	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	DDCBd6E
bit 6,R	CB77	CB70	CB71	CB72	CB73	CB74	CB75	CB76	DDCBd76
bit 7,R	CB7F	CB7B	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	DDCBd7E

Op-kód pro instrukce bit b, (IY+d) mají první bajt FD místo DD,jinak se shodují s instrukcemi bit b,(IX+d).

Nastavení indikátorů:

S a P/V nedefinovaný

H nastaven

N nulován

C nezměněn

Z Obsahuje komplement specifikovaného bitu.

set b instrukce nastavení bitu

Instrukce provede nastavení bitu "b" operandu,t.j.uloží do něj hodnotu 1.

Možné kombinace:

instrukce	op-kód								
R=	A	B	C	D	E	H	L	(HL)	(IX+d)
set 0,R	CBC7	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	DDCBdC6
set 1,R	CBCF	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	DDCBdCE
set 2,R	CBD7	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	DDCBdD6
set 3,R	CBDF	CBDB	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	DDCBdDE
set 4,R	CBE7	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	DDCBdE6
set 5,R	CBEF	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	DDCBdEE
set 6,R	CBF7	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	DDCBdF6
set 7,R	CBFF	CBFB	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	DDCBdFE

Op-kódy instrukcí set b (IY+d) mají první bajt FD místo DD, jinak se shodují s instrukcemi set b, (IX+d).

Nastavení indikátorů zůstává nezměněno.

res b instrukce nulování bitu

Instrukce provede nulování bitu "b" operandu,t.j.uloží do nej hodnotu 0.

Možné kombinace:

instrukce	op-kód								
R=	A	B	C	D	E	H	L	(HL)	(IX+d)
res 0,R	CB87	CB80	CB81	CB82	CB83	CB84	CB85	CB86	DDCBdB6
res 1,R	CB8F	CB8B	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	DDCBdBE
res 2,R	CB97	CB90	CB91	CB92	CB93	CB94	CB95	CB96	DDCBd96
res 3,R	CB9F	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	DDCBd9E
res 4,R	CBA7	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	DDCBdA6
res 5,R	CBAF	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	DDCBdAE
res 6,R	CBB7	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	DDCBdB6
res 7,R	CBBF	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	DDCBdBE

Op-kódy instrukcí res b,(IY+d) mají první bajt FD místo

DD,jinak se shodují s instrukcemi res b,(IX+D). Nastavení indikátorů zůstává nezměněno.

5. PSEUDOINSTRUKCE ASSEMBLERU

V předchozí kapitole jsme probrali všechny instrukce mikroprocesoru Z 80. Nyní se ale ještě musíme vrátit k assembleru. Každý "dospělý" Assembler (t.j. překladač) může vedle strojových instrukcí pracovat i s pseudoinstrukcemi. I když při ručním vkládání programu nejsou pseudoinstrukce nic platné, je třeba je znát. Jednak pro případ, že budete pracovat s takovým Assemblerem, který dovoluje použití pseudoinstrukcí (i když ne třeba všech), jednak pro případ, že budete aplikovat programy z různých pramenů, ve kterých také může být použito pseudounstrukcí.

Pseudoinstrukce nemají přímý vliv na činnost procesoru, jako strojové instrukce, ale pouze řídí překládací program, t.j. assembler. Jejich je stejný, jako formát strojových instrukcí, t. j. píšeme je do stejného místa a stejným způsobem jako strojové instrukce. Podobně jako u strojových instrukcí i u pseudoinstrukcí můžeme používt návěští, operandy a komentáře. Nyní k jednotlivým pseudoinstrukcím.

defb pseudoinstrukce definice bajtů

Pseudoinstrukce generuje jeden bajt s hodnotou danou operandem v tom místě, kde je pseudoinstrukce v programu uvedena. Poněvadž se jedná pouze o 1 bajt, může být generována hodnota (t.j. hodnota daná operandem) v mezích -128d=80h az 255d=FFh.

Příklad:

adresa	návěští	instrukce	komentář
4520	DVE:	defb 02	; uloží na adresu 4520 hodnotu 02

defw pseudoinstrukce definice slova

Pseudoinstrukce generuje dva bajty s hodnotou danou operandem v tom místě, kde je pseudoinstrukce v programu uvedena. Nejprve se ukládá nižší bajt a pak vyšší bajt.

Příklad:

4520	ČÍSLO:	defw 02 5D	; uloží na adresu 4520 hodnotu 5D a na adresu 4521 hodnotu 02

U obou pseudoinstrukcí defb a defw může být více operandů. Pseudoinstrukce vykonává svoji funkci opakovaně pro každý operand.

Příklad:

4520	defb 01, 00, 00;	uloží postupně na adresy 4520 hodnotu 01, 4521 00, 4522 00
4520	defw 01 00, 02 00;	uloží postupně na adresy: 4520 00°, 4521 01, 4522 00, 4523 02

defm pseudoinstrukce definice hlášení

Pseudoinstrukce generuje posloupnost bajtů obsahujících ASCII kód znaku operandu. Operandu musí být řetězec, t.j. výraz musí být uzavřen v uvozovkách.

Příklad:

4520 defm "TEXT" ; generuje na adresách: 4520 54 (ASCII kód T), 4521 45 E, 4522 58 X, 4523 54 T

deft pseudoinstrukce definice textu

Pseudoinstrukce vykonává stejnou činnost jako defm, ale navíc do prvního bajtu ukládá délku řetězce (operandu).

Příklad:

4520 deft "TEXT" ; generuje na adresách: 4520 hodnota 04, 4521 54 atd. jako v předchozím příkladu

defs pseudoinstrukce definice paměti

Pseudoinstrukce rezervuje v paměti počet bajtů daný operandem, počínaje místem ve kterém je pseudoinstrukce v programu uvedena.

Příklad:

4520 VYSL : defs 4 ; rezervuje 4 bajty na adresach 4520 až 4523 pro uložení výsledku

equ pseudoinstrukce definice symbolů

Pseudoinstrukce přiřadí hodnotu operandu k návěští. V dalším textu již nemusíme vypisovat vždy hodnotu operandu, ale stačí uvést návěští.

Příklad:

```
4520   RAMTOP : equ 4004 ;     přiřadí RAMTOP hodnotu 4004:
5200            ld HL,(RAMTOP) ; provede ld HL,(4004)
```

defl pseudoinstrukce definice návěští

Pseudoinstrukce má stejnou funkci jako equ s tím rozdílem, že operand defl může mít na různých místech programu různé hodnoty. Hodnota operandu platí vždy až do dalšího předefinování.

Příklad:

```
4520   ZNOVU: defl 05     ; přiřadí DEFL hodnotu 05
4550          djnz ZNOVU ; provede djnz 05
4580   ZNOVU: defl FD     ; přiřadí DEFL hodnotu FD
459F           djnz ZNOVU ; provede djnz FD
```

org pseudoinstrukce nastavení referenčního čítače

Pseudoinstrukce nastaví referenční čítac na adresu danou operandem. Znamená to, že ta část programu, která následuje za pseudoinstrukcí org se překládá až od adresy udané operandem.

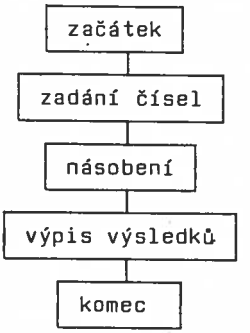
Příklad:

```
26EB             org 26EB ; program se začne překládat od
26EB                      ; začátek programu na adrese 26EB
26EB                      ; vypsání obsahu RAMTOPU na obr.
26EB                      ; PRINT USR 9963
26EB
26EB  ED4BC440  ld BC,(RAMTOP) ; RAMTOP do BC
26EF  C9        ret            ; konec programu
```

6. SESTAVENÍ PROGRAMU VE STROJOVÉM KÓDU

Když teď ovládáte jak instrukce Z80 tak pseudoinstrukce assembleru, můžete klidně sestavovat programy ve strojovém kódu. Není to ovšem tak jednoduché. Prvním a základním předpokladem správné funkce programu je důkladný rozbor dané úlohy a sestavení podrobného vývojového diagramu. V Basicu se u jednodušších programů bez vývojového diagramu klidně obejdeme, ale při sestavování strojového programu je téměř nezbytný, poněvadž na něm musíme "vyháčkovat" funkci jednotlivých registrů a indikátorů.

Vezměme si například program pro násobení dvou čísel. V Basicu vystačíme s jedním příkazem LET C = A * B. Vývojový diagram je velice jednoduchý a může vypadat následovně:



Odpovídající Basicový program bude vypadat následovně:

```
10 INPUT A
20 INPUT B
30 LET C = A * B
40 PRINT C
50 STOP
```

Velice jednoduchá záležitost. Horší to bude ve strojovém kódu. Pokusme se sestavit vývojový diagram a celý program pro vynásobení dvou celých kladných čísel.

Nejprve se musíme vžít do role procesoru a představit si, jak bude danou úlohu řešit. Mušíme zpět do druhé kapitoly k násobení binárních čísel, poněvadž procesor nezná nic jiného než binární soustavu. Pro lepší názornost zvolme konkrétní případ násobení dvou čísel 7 * 5. Víme, že binárně násobíme tak, že sečítáme násobky dvou. V našem případě bude 7 * 5 = 7 * (1 + 4). Přeneseno do binární soustavy bude:

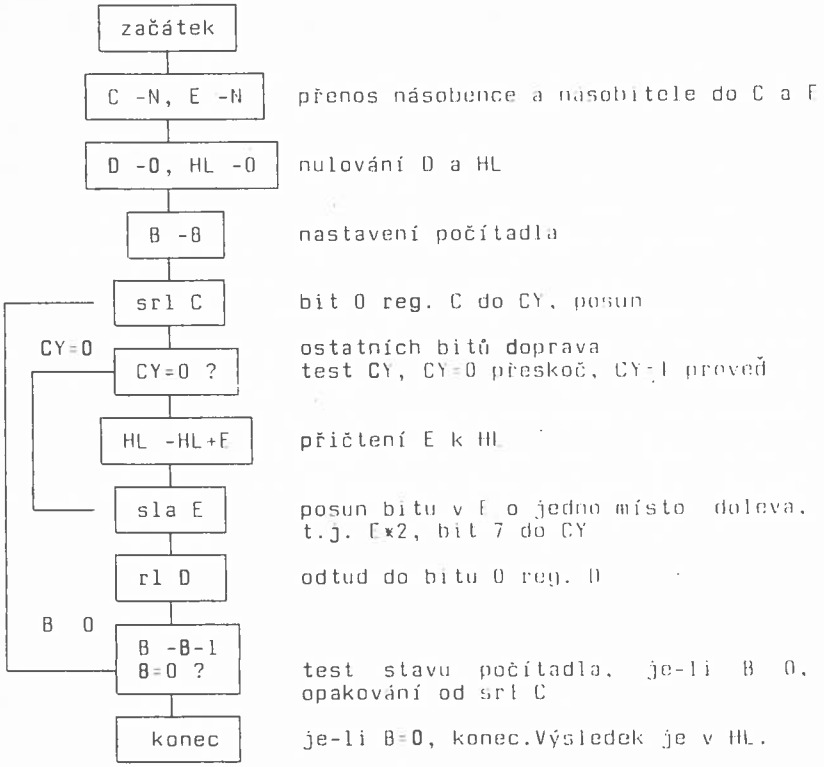
počáteční stav registru	0000 0000	
násobitel 05 = 0000 0101		
násobenec		
05*1=0000 0111	0000 0111	je v násobiteli, přičteme
07*2=0000 1110		není v násobiteli, nepřičítáme
07*4=0001 1100	0001 1100	je v násobiteli, přičteme
07*8=0011 1000		není v násobiteli, nepřičteme

dál už jsou v násobiteli jen nuly, nic nepřičítáme

Výsledek 0010 0011 = 23h = 35d

Teď se jedná o to realizovat tento postup v registrech Z80. Nejprve musíme uvážit, které registry použijeme. Víme, že postup přičítání se bude opakovat. Potřebujeme počítadlo, tudíž registr B. Násobenec budeme násobit tak, že ho budeme v registru posunovat doleva. Jeden registr nám tudíž nebude stačit, mohlo by dojít k jeho přetečení, tudíž to nemůže být C, ale registr E, přičemž D je rezervován pro případné přetečení. Nemůžeme použít H a L, poněvadž ty musíme použít pro uložení výsledku, t.j. pro postupné přičítání jednotlivých dvojkových násobků násobence, poněvadž pouze u HL existuje instrukce add HL,DE. (šlo by to provézt i naopak a pak ld D,H a ld E,L, ale je to delší.) Pro násobitel nám zbývá buď A nebo C. Je v podstatě jedno, který registr použijeme, ale je lepší ponechat A volný pro případ dalšího použití (např. přenos dat z paměti, má jako jediný možnost ld A,(NN)). Zbývá nám tudíž registr C.

Když máme obsazeny registry, můžeme uvažovat dále. Rozhodující jsou jednotlivé bity násobitele. Musíme je tudíž postupně odprava otestovat, mají-li hodnotu 0 nebo 1. Při hodnotě 1 budeme hodnotu v registru E přičítat k HL, bude-li hodnota bitu 0, nepřičteme nic. Při každém dalším bitu násobitele musíme násobenec vynásobit dvěma, t.j. posunout v registru o jedno místo doleva. Násobitel má 8 bitu, t.j. musíme tento postup 8x opakovat. K testování bitů násobitele bychom mohli použít např. test 0,C, test 1,C atd. až test 7,C ale bylo by to adlouhavé. Lepší je pomocí instrukce posunu přesunout bit 0 do CY. Nesmíme zapomenout na ošetření přetečení registru E, t.j. přesun přetečeného bitu do registru D. Můžeme to provést tak, ze bit 7, který se při posunu deleva dostane do CY přeneseme z CY do bitu 0 registru D pomocí instrukce posunu přez CY. Vývojový diagram pak může vypadat následovně:



Máme-li vývojový diagram, o kterém se domníváme, ze je správně sestaven, můžeme se pustit do assembleru. Obvykle postupujeme ve čtyřech etapách. V první etapě sestavíme vlastní assembler. Místa skoku a volání podprogramů musíme označit návěštími. Ve druhé etapě dopíšeme kódy jednotlivých instrukcí a operandů. Zatím neznámé hodnoty relativních skoků DIS a adresy volání call a absolutních skoků. Pro ně vynecháme volná místa. Je nejlepší tato místa označit čtverečky. Ve třetí etapě spočítáme počet přeskokových bajtů a určíme hodnoty DIS, které doplníme do čtverečků. V poslední čtvrté etapě doplníme absolutní adresy a podle nich i adresy absolutních skoků a volání (dopíšeme je do volných

čtverečků). Takto sestavený assembler pro ZX 81 16k RAM s počáteční adresou programu 16514d=40B2h bude vypadat takto:

```
-----------------------------------------------------------
4.et.  (3.) a 2.et.          1. etapa
-----------------------------------------------------------
4082 0E 05                   ld C,05     ; násobitel do C
4084 1E 07                   ld E,07     ; násobenec do E
40B6 16 00                   ld D,00     ; nulování D
4088 21 00 00                ld HL,00    ; nulování HL
408B 06 08                   ld B,08     ; nastavení počítadla
408D CB 39        ZNOVU :    srl C       ; bit 0 reg.C do CY
408F 30(01)                  jr NC DAL   ; test CY=0 skok na DAL
4091 19                      add HL,DE   ; přičti DE k HL
4092 CB 23        DAL   :    sla E       ; násobení E dvěma
4094 CB 12                   rl D        ; přenos bitu do D
4096 10(F5)                  djnz ZNOVU  ; test počítadla, B=0
                                                 skok na ZNOVU
4098 44                      ld B,H      ; přenos výsledku
4099 4D                      ld C,L      ; z HL do BC
409A C9                      ret         ; návrat do Basicu
-----------------------------------------------------------
```

Ještě k poslední instrukci ret. Zatím v předchozích ukázkách nebyla uváděna, poněvadž se jednalo o výseky programů ve strojovém bez dalších vazeb. Tento program je vlastně prvním fungujícím programem a proto musí být ukončen instrukcí ret. Platí jednoznačná zásada : KAŽDÝ STROJNÍ PROGRAM MUSÍ BÝT UKONČEN INSTRUKCÍ RET. Je to proto, že naše programy ve strojovém kódu jsou vlastně podprogramy Basicového systému počítače a musí se poslední instrukcí vrátit do Basicu. I pouhé PRINT USR 16514, které vyvolá na ZX81 náš vzorový program, je vlastně basicovým podprogramem a způsobí, že se na obrazovku vypíše obsah registrového páru BC (nebuďte překvapení, když se na obrazovce objeví číslo 7, poněvadž ZX81 interpretuje obsah BC jako kód znaku, a poněvadž kód 23h=35d odpovídá znaku "7").

Ve druhém sloupci assembleru máme zdrojový program, který pak obvykle píšeme jako serii bajtů, t.j. 0E 05 1E 07 .... 44 4D C9, nebo ve formě tzv. HEX-DUMP. HEX-DUMP je absolutním zdrojovým programem, poněvadž obsahuje i adresy. Je uváděna vždy adresa prvního bajtu v každém řádku, každý

další bajt má adresu o jedničku vyšší. Někdy bývá na konci každého řádku uváděno ještě kontrolní číslo, které vyjadřuje buď paritu nebo hodnotu soyčtu bajtu v řádku. Náš program bude v HEX-DUMP vypadat následovně:

```
4082 = 16514  -  0E 05 1E 07 16 00 21 00  =  111
408A = 16522  -  00 06 0B CB 39 30 C1 01  =  348
4092 = 16530  -  CB 23 CB 12 10 F5 44 4D  =  865
409A = 16538  -  C9                       =  201
```

Aby program fungoval i na Spectru, museli bychom změnit počáteční adresu na 23296.

Timto by vlastně mělo skončit vlastní sestavení programu ve strojovém kódu. Ted zbývá pouze vložit program do počítače, spustit a čekat, bude-li fungovat. Funguje-li, je vyhráno, jestliže nefunguje, čeká nás obtížné hledání chyb. Bohužel málokterý program funguje hned napoprvé a většinou se musí pracně odlaďovat, t.j. hledat chyby.

7. PŘÍLOHA A

Decimální hodnoty HEX číslic:

4.číslice HEX		3. DEC	2.	1.
0	0	0	0	0
1	4 096	256	16	1
2	8 192	512	32	2
3	12 288	768	48	3
4	16 384	1 024	64	4
5	20 480	1 280	80	5
6	24 576	1 536	96	6
7	28 672	1 792	112	7
8	32 768	2 048	128	8
9	36 864	2 304	144	9
A	40 960	2 560	160	10
B	45 056	2 816	176	11
C	49 152	3 072	192	12
D	53 248	3 328	208	13
E	57 344	3 584	224	14
F	61 440	3 840	240	15

Převod HEX-DEC hodnot jednobajtových čísel:

TABULKA 00h = 0d až FFh = 255 nebo -1 dec.

8. PŘÍLOHA B

Možné kombinace instrukce ld 1.,2.

2.|1.	(BC	DE	HL	IX+d	IY+d	NN)	A	B	C	D	E	H	L	I	R	IX	IY	SP	BC	DE	HL
(BC)	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(DE)	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(HL)	.	.	.	.	.	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
(IX+d)	.	.	.	.	.	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
(IY+d)	.	.	.	.	.	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
(NN)	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	x	x	x	x	x	x
A	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	.	.	.	.	.	.
B	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
C	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
D	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
E	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
H	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
L	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
F	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.
R	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.
N	.	.	x	x	x	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.
IX	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.
IY	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.
SP	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
BC	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DE	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
HL	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.
NN	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	x	x	x	x	x

9. PŘÍLOHA C

Doby provádění jednotlivých strojových instrukcí:

Instrukce	Počet cyklů	poznámka
ld A,r	4	r = registr
ld r,N	7	
ld r,HL	7	
ld A,NN	13	
ld rr,NN	10	rr = dvojice registru
ld HL,NN	16	
push rr	11	
pop rr	10	
add A,r	4	
inc r	4	
inc HL	11	
rla	4	
jp NN	10	
jr DIS	12	
jr l.	12	podmínka nenalezena
	7	podmínka nalezena
call NN	17	
ret	10	

Při kmitočtu oscilátoru 3,5 MHz je délka jednoho cyklu 0,2857us.