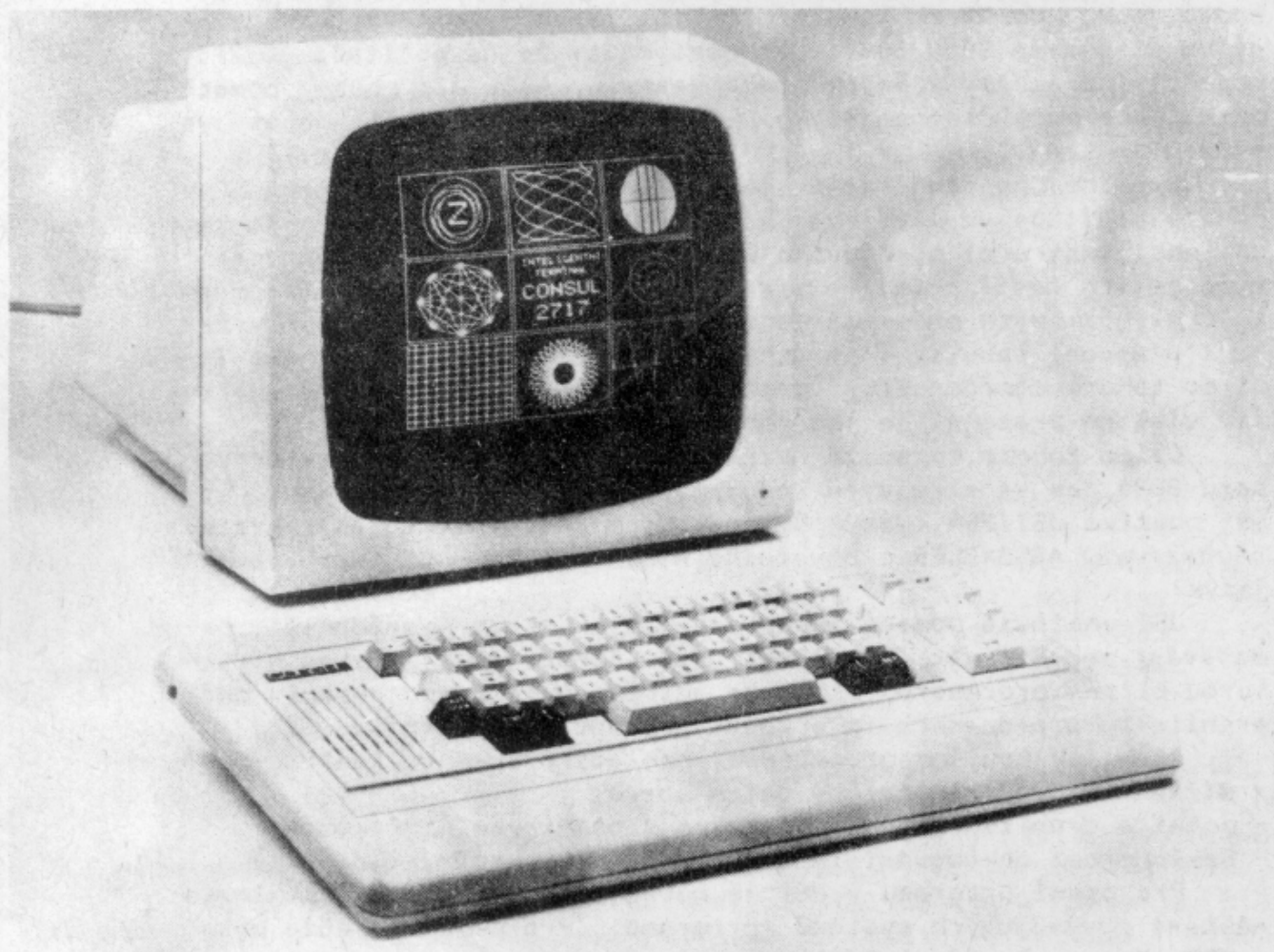


 **CONSUL 2717** 



PRÁCE S POČÍTAČEM
KURS INSTRUKCÍ 8080

KURS INSTRUKCI MIKROPROCESORU MHB-8080:

=====

UVOD:

Základním obvodem libovolného počítače je procesor, který je schopen vykonávat požadované funkce pouze prostřednictvím programu zapsaného v paměti počítače ve strojovém kódu tohoto procesoru. Srdcem počítače CONSUL 2717 je mikroprocesor MHB 8080A, proto jsou v pamětech tohoto počítače všechny programy zakódovány ve strojovém kódu 8080. Lze namítnout, že na počítači C 2717 pracují i programy v Basicu nebo Pascalu. Ano, ale pouze pomocí překladatele příslušného jazyka, který je v paměti uložen opět ve strojovém kódu.

Studium programů (algoritmů) dříve napsaných a uložených v paměti počítače ve strojovém kódu (např. základní monitor) je bez znalosti instrukcí strojového kódu nemožné. Vytváření programů nebo jejich částí přímo ve strojovém kódu je velmi pracné, opravy a úpravy takových programů jsou značně komplikované, neboť vyžadují převodní tabulky instrukcí, kódů a pod. Přesto se mnoho lidí do tohoto dobrodružství pouští - cesta za poznáním, jak počítač vlastně pracuje, je jim dostatečnou motivací.

Cílem tohoto kursu je seznámení se všemi instrukcemi procesoru 8080 jak ve strojovém kódu, tak i v symbolickém tvaru, jak jej používá JSI/JSA (Jazyk Symbolických Instrukcí/Adres), zkráceně nazývaný ASSEMBLER z původního ASSEMBLY LANGUAGE (sestavovací jazyk).

JSI umožňuje pomocí mnemotechnických zkratk snadnější zapamatování jednotlivých typů instrukcí. Tento jazyk je svou strukturou blízký procesoru, proto je nutné pro jeho pochopení znát architekturu procesoru (pro různé procesory jsou různé JSI).

Architekturu (programátorský pohled) procesoru tvoří:

- síťka sběrnic pro přesuny dat a adres;
- počet a druh registrů pro dočasné zapamatování informací;
- posloupnost dekodování instrukcí (jejich uložení v paměti).

Pro psaní programu v JSI je nutno mít v počítači instalován některý z vývojových systémů (programů). Pro C2717 to může být:

- MRS -Memory Resident System;
- DAM -Debugger Assembler Monitor;
- VMON-VideORAM MONitor.

Tento kurs nebude na těchto vývojových systémech závislý, neboť uváděné příklady bude možno zapsat přímo do paměti počítače a na obrazovce si ověřit jejich funkčnost.

TEMA: ARCHITEKTURA MIKROPROCESORU MHB 8080A

Lekce: 1

NOVE POJMY: Bit, Byte(slovo), registry, sběrnice
 ----- 8-bitová data, 16-bitová adresa
 Paměti ROM (EPROM) a RWM (RAM)
 Čítač programu-PC, dekodér instrukcí
 NOVE INSTRUKCE: 00 = NOP (No Operation=žádná činnost)

Jednotkou informace je dvojková číslice BIT (Binary digit), nabývající hodnot 1 (PRAVDA=TRUE) a 0 (NEPRAVDA=FALSE). Skupina 8 bitů tvoří BYTE (bajt), v němž jsou bity očíslovány zprava doleva, od nejméně významného d0=LSB (Least Significant Bit) až po nejvíce významný bit d7=MSB (Most Significant Bit) takto:

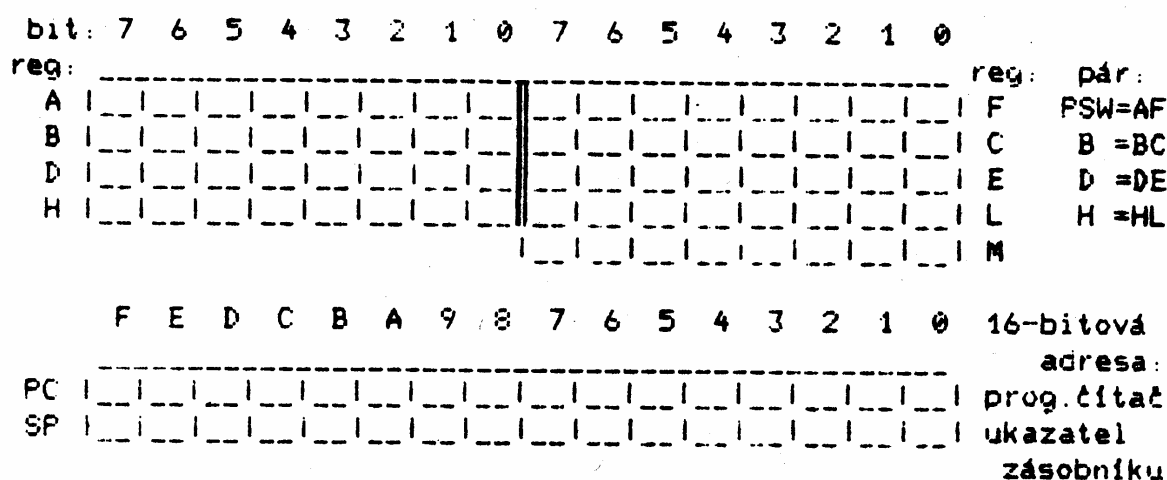
bity	d7	d6	d5	d4	d3	d2	d1	d0
BYTE								
	-----				-----			
	MSB				LSB			
váha	128	64	32	16	8	4	2	1

Obsah: šestnáctkové	dvojkové								desítkové
80H	1	0	0	0	0	0	0	0	=128
91H	1	0	0	1	0	0	0	1	=145
A2H	1	0	1	0	0	0	1	0	=162
B3H	1	0	1	1	0	0	1	1	=179
C4H	1	1	0	0	0	1	0	0	=196
D5H	1	1	0	1	0	1	0	1	=213
E6H	1	1	1	0	0	1	1	0	=230
F7H	1	1	1	1	0	1	1	1	=247

Čtveřice bitů umožňuje zakódovat šestnáct číslic (0-15), označených 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. Obsah jednoho BYTU je možno vyjádřit dvěma šestnáctkovými číslicemi, za nimiž je písmeno H (Hexadecimal), pro odlišení od desítkových čísel. Převod mezi některými čísly soustav je uveden výše v tabulce.

Procesor 8080 patří mezi tzv. 8-bitové vzhledem k tomu, že jeho datová sběrnice má 8 bitů, většina registrů jeho zápisníkové paměti je také 8-bitová a především jeho aritmeticko-logická jednotka pracuje se stejným počtem bitů.

Vnitřní struktura mikroprocesoru je velmi složitá (tisíce tranzistorů), z hlediska uživatele (programátora) se zmenší jen na přístupné registry: těch je 8 osmibitových (A,F,B,C,D,E,H,L) a dva 16-bitové (PC a SP); přitom lze pracovat i s páry registrů AF,BC,DE,HL. Dvojice AF je označována PSW (Programm Status Word).



Zvláštní postavení má paměťové místo M (registr v paměti), které je určeno adresou ve dvojici registrů HL.

Registr A (Accumulator-střádač) je nejuniverzálnější a používá se v největším počtu instrukcí. Registr F (Flags) obsahuje 5 příznakových bitů, které obsahují bližší informace o výsledku operací ve střádači A:

S Z 0 AC 0 PE 1 CY

FLAGS: |___|___|___|___|___|___|___|___|=příznaky

S (Sign) -znaménko, bit je shodný s MSB (7) bitem střádače;

Z (Zero) -nulový výsledek ve střádači: Z=1; nenulový: Z=0;

AC (Auxiliary Carry)-pomocný přenos ze 3.bitu do 4.bitu střádače

PE (Parity Even) -sudá parita: PE=1-sudý počet jedniček střádače

CY (Carry)-přenos z nejvyššího řádu (MSB) střádače (přetečení).

Zbývající dva bity jsou nulové a jeden trvale jednotkový.

Pomocí uvedených příznaků lze vytvářet programy, které využívají podmíněné skoky, podmíněná volání podprogramů a podmíněné návraty z nich (podmíněná větvení programů podle příznaků).

Procesor MHB8080A je základním prvkem počítače C2717. Dalšími prvky jsou pevná paměť, umožňující pouze čtení informací (ROM-Read Only Memory), tvořená obvody EPROM (Eraseble Programmable ROM-vymazatelná a programovatelná). V EPROM je uložen základní monitor počítače (jednoduchý operační systém) ve strojovém kódu od adresy 8000H (H=hexadecimal-šestnáctkově), a od adresy 9C00H je uložen interpret jazyka Basic-G (ve strojovém kódu). Ve starší verzi C2717, která se hlásí velkými písmeny BASIC-G, je interpret uložen od adresy 900CH. Tento interpret je po zapnutí počítače přesunut na adresy 0000-23FFH paměti RWM (Read Write Memory-paměť pro čtení/zápis), někdy nepřesně nazývané jako RAM (Random Access Memory - paměť s náhodným přístupem; takovou paměť je samozřejmě i ROM). Programy psané v jazyku Basic-G se nahrávají od adresy 2400H, pro pole proměnných je vymezena paměť od adresy 6000H, pro pomocné programy ve strojovém kódu je určena RWM od 7000H do 7F00H, videopaměť tvoří oblast adres C000H-FFFFH.

Celý paměťový prostor, adresovatelný procesorem, tvoří 65536 adres od 0000H do FFFFH. Na každou adresu lze zapsat jeden BYTE o obsahu 00H-FFH (0-255). Například obsah paměti od počátku je:

Adresa:	Obsah bitů adresy:								Hexa:	Dek
0000H	0	0	1	0	0	0	0	1	21H	33
0001H	1	0	0	0	0	0	1	0	82H	130
0002H	0	1	1	1	1	1	1	1	7FH	127

1. cvičení na C2717:

Přepněte do MONITORU pomocí kláves SHIFT (↑) a RCL. Zobrazí se ++0s ready++ v dialogovém řádku a můžeme zadávat příkazy monitoru (MEM, SUB, JUMP, DUMP).

Zadejte: DUMP 0000 (a EOL), výpis zastavte klávesou STOP.

Zobrazí se: 0000 21 82 7F 3E 0F C3 47 00
0008 7E E3 BE 23 E3 C2 FD 00

Zadejte: MEM 0000 (a EOL);

Zobrazí se: SUB 0000 21 82 7F 3E 0F C3 47 00 7E E3 BE 23 E3 ...

Funkce SUB (SUBstitute=nahradiť) umožňuje měnit obsah buněk paměti. Posuňte kurzor pomocí → na číslici 2, změňte ji na 0 klávesou 0, ponechte 1, změňte 82 na 23, 7F na 45. Stlaťte EOL, tím zapíšete změněné a zbývající obsahy do paměti. Současně se do pracovní části obrazovky opiše celý dialogový řádek, a zobrazí se: SUB 0010-pro modifikaci dalších adres.

Stlaťte: CLEAR (smaže se dialogový řádek); CLEAR=čistit

Zadejte: MEM 0000 (zjistíme co se zapsalo do paměti):

Zobrazí se: SUB 0000 01 23 45 3E 0F C3 47 00 7E E3 BE 23 E3 ...

Stlaťte: CLEAR

Zadejte: DUMP 0000 (a EOL, STOP): DUMPfile = výpis souboru

Vypíše se: 0000 01 23 45 3E 0F C3 47 00
0008 7E E3 BE 23 E3 C2 FD 00

Tak jste se naučili měnit obsah určené oblasti paměti.

Podobným způsobem si zkuste změnit obsah paměti od 8000H:

Zadejte: MEM 8000

Zobrazí se: SUB 8000 31 00 80 C3 06 80 3E 82

Změňte na: AB CD EF (a EOL; 'změna' je na stínítku)

Stlaťte: CLEAR

Zadejte: MEM 8000

Zobrazí se: SUB 8000 31 00 80 C3 06 80 3E 82 (tj. původní obsah)

Obsah adres není změněn proto, že od 8000H jsou adresovány obvody EPROM, které lze pouze číst a nelze do nich zapisovat nový obsah (je zde strojový kód MONITORu a BASIC-G).

Nyní si zkuste vypsát obsah paměti od C000H; najdete tam:

MEM C000

Zobrazí se: SUB C000 00 00 00 00 00 00 00 00 ...

Změňte na: 33 33 33 33 (a EOL)

V nejvyšší části obrazovky se objeví několik pomlček, protože jste zapsali nový obsah do videopaměti začínající od zadané adresy C000H.

Zkuste dále: SUB C010 77 77 77 77 (a EOL)

zobrazí se jiné pomlčky polojasem. Zadejte dále:

SUB C014 88 88 88 88 (a EOL)

zobrazí se blikající tečky. Pokud zadáte:

SUB C018 FF FF FF FF (a EOL)

zobrazí blikající úsečku polojasem. Konec videopaměti si vyzkoušejte pomocí: SUB FFC0 88 88 88 88

Instrukce, které tvoří program, jsou uloženy v paměti v určitém sledu. Čítač instrukcí programu PC (Programm Counter) tvoří 16-bitový registr, který obsahuje adresu paměti, z níž si bude procesor číst instrukci nebo její operandy (to závisí na formátu instrukce). Procesor považuje obsah PC vždy za adresu paměti, kterou vysílá na adresovou sběrnici a po ní do paměti. Obsah adresované buňky se po datové sběrnici (8bit) přečte do registru instrukce, kde se dekóduje. Současně je zvýšen obsah PC o +1, aby se mohla přečíst další instrukce, nebo operandy (jeden nebo dva) předcházející instrukce. Tato činnost tvoří instrukční cyklus. Čtení bytu z paměti je paměťový cyklus (Mi), tvořený několika takty procesoru (obvykle T1-T4), odvozenými od hodinových pulsů.

Po zapnutí počítače je PC=0000, což zajišťuje tzv. řadič uvnitř mikroprocesoru. Proto by měl program začínat od této adresy. Zapneme-li počítač, jsou adresy od 0000 prázdné, jejich obsah je 00, proto se čtou do procesoru instrukce s kódem 00, jejichž význam je "nic nedělej" (mnemotechnická zkratka NOP z anglického No Operation=žádná činnost). Procesor postupně zvyšuje obsah PC, až přečte na adrese 8000H první instrukci: 31 (LXI SP); dekóduje ji, přečte její operandy a tak začne provádět program uložený v paměti EPROM. Zde začíná tzv. studený start programu, který zajišťuje nastavení základních obvodů a funkcí počítače (např. snímání kódu stlačené klávesy, zobrazení znaku, výpis připravenosti interpretu jazyka Basic-G a pod.).

2. cvičení:

Zkusíme si odstartovat základní program v pamětech EPROM.

Zadejte: ↑+RCL (přepnutí do MONITORU)

Zobrazí se: ++0s ready++

Zadejte: JUMP 8000 (JUMP = skok)

Problikne: ++Executive++ (informace, že se provádí program)

Zobrazí se: BASIC-G (nebo Basic-G u novějších C2717)

OK

Podobně bychom mohli odstartovat program načtení BASIC-G z

EPROM: JUMP 9000 (nebo JUMP 9C00 u novějších C2717).

Tzv. teplý start interpretu BASIC zajišťuje: JUMP 0000

Nyní můžete totéž zkusit pomocí příkazu Basic-G, například:

Zadejte: A=USR('8000) /nebo A=USR('9000), nebo A=USR(0000)/

Nezapomeňte na apostrof, musí předcházet šestnáctková čísla.

TEMA: INSTRUKCE PRESUNU MEZI REGISTRY A PAMETI

Lekce: 2

NOVE POJMY:	Přesuny dat 8bitů a 16bitů v registrech a paměti
-----	Výměna obsahu registrů
NOVE INSTRUKCE:	MOV r,r-MOVE register/memory to register/memory
-----	MVI r,d-Move Immediate data 8bit register/memory
	LXI p,w -Load eXtended Immediate
	LDA,STA,LDAX,STAX-Load/Store Accumulator
	LHLD,SHLD -Load/Store pair HL
	XCHG -eXCHanGe pairs HL<=>DE
	HLT -HaLT/zastavení procesoru

Nejčastěji používanými instrukcemi jsou přesuny mezi registry a pamětí. Jsou převážně 1 bytové a proto velmi rychlé, neboť mají zakódovány operandy převážně uvnitř kódu. Základní 8-bitové přesuny MOV (MOVE=přesun) mezi registry trvají 5 strojových cyklů, mezi registry a pamětí trvají 7 cyklů. Při zápisu v jazyku symbolických instrukcí je nutno pamatovat na to, že se přesouvá obsah z registru uvedeného vpravo za čárkou, do registru vlevo. Tyto instrukce nemění příznaky v registru F. M je tzv. paměťový registr: adresa příslušné paměťové buňky musí být předem zapsána do páru registrů HL. Instrukce MOV mají následující formáty:

Kódy:	Symbolika:	Přesun: do<-z	Cykly:	Příznaky
40H...7FH	MOV x,y	; x <- y	5T	nemění F
70H-75H,77H	MOV M,y	;[HL] <- y	7T	
46H,4EH,...7EH	MOV x,M	; x <- [HL]	7T	
76H	HLT	;zastavení procesoru (MOV M,M)		

Instrukce přesunu z paměti do stejného místa v paměti nemá smysl, proto byl její kód využit k zastavení činnosti procesoru. Hranaté závorky znamenají obsah registrů uvnitř závorek, např. [HL] značí buňku paměti, jejíž adresa je v páru registrů HL. Pro zjednodušení není závorkování používáno u samotných registrů.

Při dalším výkladu budeme využívat tabulku kódů instrukcí, která je uvedena v příloze, nebo byla vydána jako pomůcka SWP4. Dříve než uvedeme příklad na instrukce MOV, musíme naplnit registry definovaným obsahem. K tomu jsou určeny přesunové instrukce typu MVI r,d -kde d je přímý 8-bitový operand s hodnotou 00-FFH, a r je zkratka registru, do kterého se data d zapisují. Instrukce obsazuje 2 buňky paměti (kód instrukce=opkód, přímý operand). Instrukce nemění příznaky F, trvají 7 taktů; MVI M,d má 10 taktů:

Kódy:	Symbolika:	Funkce instrukcí:
0600	MVI B,0	;vynulování registru B
0E08	MVI C,8	;zápis konstanty 8 do registru C
1650	MVI D,50H	;zápis šestnáctkového čísla do reg. D
1E50	MVI E,80	;zápis desítkového čísla do reg. E
26FF	MVI H,0FFH	;zápis čísla začínajícího číslicí do H
2EC0	MVI L,192	;zápis nižšího bytu adresy do reg. L
3600	MVI M,0	;vynulování paměťové buňky M=[HL]=FFC0H
3E53	MVI A,'S'	;zápis kódu znaku S do střádače A

1. cvičení: vynulujte všechny registry procesoru

----- 1. způsob:	0600 MVI B,0	2. způsob:	0600 MVI B,0
	0E00 MVI C,0		48 MOV C,B
	1600 MVI D,0		51 MOV D,C
	1E00 MVI E,0		5A MOV E,D
	2600 MVI H,0		63 MOV H,E
	2E00 MVI L,0		6C MOV L,H
	3E00 MVI A,0		7D MOV A,L
	C9 RET		C9 RET

Oba způsoby jsou rovnocenné, ale první zabere v paměti 14 bytů, zatímco úspornější druhý jen 8 bytů. Zkus-
te si oba programy zapsat do paměti a spustit. Např.:

Zadejte: MEM 7000 (a EOL)
SUB 7000 06 00 48 51 5A 63 6C 7D C9 (a EOL)
CLEAR (stisk klávesy, nikoli příkaz)
JUMP 7000 spuštěním programu se ovšem "nic" nestane.

Poznámka: Pro ukončení programu jsme použili instrukci návratu z podprogramu C9=RET, kterou si osvětlíme v 5. lekci.

Zadejte: JUMP 0000 návrat do Basic-G
K=USR('7000'):PRINT K ...proměnná K uchovává střadač A
Vypiše se: 0 výsledek programu: vypíše se obsah střadače

Zadejte: ↑+RCL MEM 7000
Místo 0 zadáme do registru B konstantu FFH=255 takto:

Změňte: SUB 7000 06 FF 48 51 5A 63 6C 7D C9

Stiskněte: CLEAR

Zadejte: JUMP 0000
K=USR('7000'):PRINT K

Vypiše se: 255 výsledek programu, desítkový obsah střadače

Pro naplnění páru registrů 16-bitovými daty w (Word=slovo) slouží instrukce typu: LXI rp,w. Jsou 3-bytové, trvají 10 taktů a nemění příznaky F:

Kódy: Symbolika: Funkce instrukcí:

01CCBB LXI B,0BBCCH naplnění páru B(=BC) konstantou

115533 LXI D,3355H naplnění páru D(=DE) konstantou

210000 LXI H,0 vynulování páru H(=HL)

*! 31FE7F LXI SP,7FFE7F naplnění ukazatele zásobníku adresou

Podívejte se pozorně, jak se konstanty zapisují do paměti. Za kódem instrukce je nejprve hodnota nižšího bytu konstanty: CC, 55 nebo FE, a na druhém místě je hodnota vyššího bytu: BB, 33, 7F. To je dáno konstrukcí mikroprocesoru a je nutno to respektovat. Některé překladače/assemblery vypisují operandy kódu obráceně, např. M-80 vypíše: 01BBC0, 113355 atd. Proto je nutno ve výpisech rozlišovat a včas si uvědomit, jak výpis chápat.

Páry registrů lze naplnit též ze zásobníku. O tom později.

Obdobou instrukce MOV M,A je instrukce: STA a16 (Store Accumulator direct); na zadanou adresu uloží obsah střadače A. Naplnění střadače ze zadané adresy zajišťuje instrukce: LDA a16, (Load Accumulator Direct). Obě instrukce mají 3 byty, trvají 13 taktů a nemění příznaky.

Kódy:	Symbolika:	Funkce instrukcí:
326745	STA 4567H	uchování obsahu střadače na adrese
3A6745	LDA 4567H	naplnění střadače z adresy 4567H

2. cvičení: naplňte střadač daty a uložte je do videopaměti -VRAM

```

-----7000 3E33 MVI A,51 ;konstanta 51=33H do střadače
          7002 3200C0 STA 0C000H ;uchování na 0.adrese VRAM
          7005 322FC0 STA 0C02FH ;uchování na 47.adrese VRAM
          7008 C9 RET ;konec programu (kód C9=návrat)
Zadejte: SUB 7000 3E 33 32 00 C0 32 2F C0 C9 (a EOL,CLEAR)
          JUMP 7000 (po EOL se v levém i pravém horním rohu
                    stínítka objeví dvě čárky ze dvou bodů)

```

Z Basicu: R=USR('7000)-se stejným výsledkem
 nebo: POKE'7001,15:R=USR('7000)
 nebo: FOR K=1TO255:POKE'7001,K:R=USR('7000):PAUSE1:NEXT
 a pozorujte pravý horní roh stínítka.

Další instrukce umožňují uchovat/naplnit střadač pomocí nepřímých (indirect) adres uložených v registrových párech B a D. Jsou to instrukce jen jednobytové, trvají 7 taktů a nemění F.

Kódy:	Symbolika:	Funkce instrukcí:
02	STAX B	uchovej střadač na adrese uložené v páru BC
0A	LDAX B	naplň střadač z adresy uložené v páru BC
12	STAX D	Store Accumulator indirect on [DE]
1A	LDAX D	Load Accumulator indirect from [DE]

Další dvojice instrukcí SHLD a16 (Store HL Direct on a16) a LHLD a16 (Load HL Direct) umožňují uchovat(naplň) pár registrů HL na(z) uvedenou přímou adresu a16. Mají 3 byty, 16 taktů a nemění F.

Kódy:	Symbolika:	Funkce instrukcí:
220070	SHLD 7000H	uchování obsahu HL na adresách: [L] -> 7000H a [H] -> 7001H
2A0270	LHLD 7002H	naplnění HL: L<-[7002H],H<-[7003H]

3. cvičení: naplňte HL daty z paměti a uchovejte jinde v paměti

```

-----7002 2A0270 LHLD 7002H ;obsah 7002 a 7003 do HL
          7005 220070 SHLD 7000H ;ulož L(7000) a H(7001)
          7008 C9 RET ;instrukce návratu
Zadejte: SUB 7000 00 00 2A 02 70 22 00 70 C9 (a EOL,CLEAR)
          JUMP 7002 (a EOL)
          MEM 7000 (nebo DUMP 7000 a STOP)
Zobrazí: SUB 7000 2A 02 2A 02 70 22 00 70 C9 ....

```

Obsah adres 7002 a 7003 se přes registry HL přenesl jinam. Zkuste si změnit adresu v LHLD např. na 7007H a adresu v SHLD na C015H a pozorujte, co se vám uloží do VRAM (nahore na stínítku).

Pro výměnu obsahů mezi páry registrů DE a HL slouží instrukce XCHG (eXCHanGe HL with DE), kód EB, 1 byte, 4 takty, nemění F. Tato instrukce se často používá k rychlé úschově a vložení nového obsahu do páru HL (označovaného M) pro instrukce s nepřímou adresací:

EB XCHG ;výměna obsahů reg.párů DE<=>HL

Pokud potřebujeme v paměti vyhradit určité místo pro data, používá se k tomu v jazyku symbolických instrukcí několika tzv. pseudoinstrukcí, které nemají ve strojovém kódu ekvivalent. Jsou to:

- DB -Define Byte (definuj místo pro byte)
- DW -Define Word (definuj místo pro slovo = 2 byty)
- DS n -Define Storage (definuj oblast paměti n bytů)

Překladač/assembler je využívá ke své práci, tj. vymezení určeného počtu bytů v paměti.

4. cvičení: změňte pořadí bytů ve čtyřmístné oblasti paměti

-----	7000	1122	DW	1122H	;definováno slovo v paměti
	7002	3344	DW	3344H	;Define Word in memory
	7004	2A0070	LHLD	7000H	;počátek oblasti v paměti
	7007	EB	XCHG		;uchovej v registrech DE
	7008	2A0270	LHLD	7002H	;druhou polovinu oblasti
	700B	7D	MOV	A,L	;uchovej L dočasně v A
	700C	6C	MOV	L,H	;přesuh z H do L
	700D	67	MOV	H,A	;přesuh bývalé L z A do H
	700E	220070	SHLD	7000H	;a zapiš do nového místa
	7011	6A	MOV	L,D	;přesuh křížem druhý byte
	7012	63	MOV	H,E	;a také první byte
	7013	220270	SHLD	7002H	;a ulož je na nové místo
	7014	C9	RET		;návrat z podprogramu

Zadejte: SUB 7000 11 22 33 44 2A0070 EB 2A0270 7D 6C 67 (EOL)
SUB 700E 220070 6A 63 220270 C9 (EOL,CLEAR)
JUMP 7004 (EOL)
MEM 7000 (nebo DUMP 7000)

Zobrazí: SUB 7000 44 33 22 11 2A 00 70 EB ...

Z výpisu je vidět zrcadlová výměna obsahů adres 7000-7003. Při zápisu dat pomocí SUB.... není nutno zapisovat mezery.

TEMA: JEDNODUCHÉ ARITMETICKÉ INSTRUKCE

Lekce: 3

NOVE POJMY:

Inkrementace (zvětšování) a dekrementace (zmenšování) obsahu registrů, vliv na příznakové bity, 16-bitové sečítání, jednoduché násobení, výpočet funkce pomocí tabulky, dekadická korekce.

NOVE INSTRUKCE:

INR r INcrement Register-zvětšení obsahu r o 1
 INX rp INcrement eXtended-zvětšení páru rp o 1
 DCR r DeCrement Register-zmenšení obsahu r o 1
 DCX rp DeCrement eXtended-zmenšení páru rp o 1
 DAD rp Double ADdition-součet reg. páru rp s HL
 DAA Decimal Adjust Accumulator-desítkové nastavení obsahu střadače

Jednoduché aritmetické instrukce umožňují zvětšování nebo zmenšování obsahu registru nebo páru registrů o hodnotu 1. Nejprve zvětšování (increment) obsahu registru nebo buňky paměti:

04 INR B (5T) 14 INR D (5T) 24 INR H (5T) 34 INR M (10T)
 0C INR C 1C INR E 2C INR L 3C INR A (5T)

Instrukce trvají 5 taktů, pouze instrukce s pamětí má 10 taktů, jak je uvedeno v závorkách.

Tyto instrukce neovlivňují přenosový bit CY příznaků, ostatní bity se mění v souladu se změnou obsahu registru, na který se instrukce vztahuje. Proto se instrukce tohoto typu často využívají při počítání přechodů cyklem ke zvyšování hodnoty registru ve funkci čítače. Po dosažení maximální hodnoty čítače 0FFH pokračuje čítání od nuly.

Změny příznakových bitů si vysvětlíme na několika ukázkách. Vlevo uvedeme bitový obsah registru před a po instrukci INR, vedle bude uveden výchozí a změněný obsah příznaků v 'registru' F:

1.příklad: bit: 7 6 5 4 3 2 1 0 S Z 0 AC 0 PE 1 CY
 ----- B: 0 0 0 0 0 0 0 0 =00 F: 0 1 0 0 0 1 1 0
 po INR B: 0 0 0 0 0 0 0 1 =01 F: 0 0 0 0 0 0 1 0

Před instrukcí byl obsah registru B nulový, čemuž odpovídá:
 S=0 - příznak znaménka se předává z bitu 7 výsledku operace, je nulový v obou případech, což odpovídá kladnému znaménku;
 Z=1 - předcházející operace vynulovala registr B, proto je nastavena platnost Z=1 (je pravda, že výsledek je nula: ZERO=1)
 - po instrukci INR B tato podmínka splněna není, proto: Z=0;
 AC=0- nedošlo k přenosu z pravé čtveřice (tetrády) bitů do levé,
 PE=1- je pravda, že počet jedniček v registru je sudý (0 je sudé číslo), proto je PE=1;
 - po instrukci INR není tato podmínka splněna, proto: PE=0;
 CY - příznak přenosu není instrukcemi typu INR ovlivňován.

2.příklad: bit: 7 6 5 4 3 2 1 0 S Z 0 AC 0 PE 1 CY
 ----- C: 0 0 0 0 1 1 1 1=0FH F: 0 0 0 0 0 1 1 0
 po INR C: 0 0 0 1 0 0 0 0=10H F: 0 0 0 1 0 0 1 0

V tomto příkladu se změnil bit AC, protože došlo k přenosu z pravé tetrády (0-3) do levé (4-7). Současně přestal platit sudý počet jednotek, a proto se i PE vynulovalo (není pravda...).

3.příklad: bit: 7 6 5 4 3 2 1 0 S Z 0 AC 0 PE 1 CY
 ----- D: 1 1 1 1 1 1 1 1=FFH F: 1 0 0 0 0 1 1 0
 po INR D: 0 0 0 0 0 0 0 0=00 F: 0 1 0 1 0 1 1 0

Bit znaménka byl nastaven předchozí operací na S=1, ale po INR D byl registr D vynulován a proto je S=0 a také Z=1. Současně došlo i k vnitřnímu přenosu, proto je AC=1 a sudý počet jedniček se nezměnil na lichý a proto je v obou případech PE=1.

Opačný význam mají instrukce pro zmenšení (decrement) hodnoty obsahu registru nebo buňky paměti. Trvají 5 taktů.

05 DCR B 15 DCR D 25 DCR H 35 DCR M (10taktů)

0D DCR C 1D DCR E 2D DCR L 3D DCR A

Příznaky ovlivňují stejně jako INR, nemění CY bit.

Zvětšování nebo zmenšování obsahu páru registrů o 1 umožňují instrukce, jejich trvání je shodně 5 taktů:

03 INX B 13 INX D 23 INX H 33 INX SP

0B DCX B 1B DCX D 2B DCX H 3B DCX SP

Tyto instrukce nemění žádné příznaky, používají se pro zvyšování nebo snižování hodnot adres v programových cyklech, hledání v tabulkách, využívají se i v cyklech 16-bitových čítačů, kdy lze zadat maximální počet čítání 0FFFFH=65535. Příklady budou uvedeny v dalších lekcích.

Instrukce typu DAD rp (Double ADdition) umožňují přičíst obsah registrového páru rp k registrovému páru HL. Výsledek operace zůstane v HL. Tyto instrukce ovlivňují pouze příznak přenosu CY a trvají 10 taktů:

09 DAD B 19 DAD D 29 DAD H 39 DAD SP

4.příklad: bit: F E D C B A 9 8 7 6 5 4 3 2 1 0
 ----- DE: 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 0 = 2268H
 HL: 0 1 1 1 0 1 1 1 0 1 0 1 0 0 1 1 = 7753H

výsledek DAD D: 1 0 0 1 1 0 0 1 1 0 1 1 1 0 1 1 = 99BBH CY=0

1.úloha: sečtete dvě šestnáctková čísla: $2268 + 7753 = 998B$

```
----- 7000 6822 DW 2268H ;první sčítanec
        7002 5377 DW 7753H ;druhý sčítanec
        7004 0000 DW 0 ;místo pro výsledek
        7006 2A0070 LHLD 7000H ;načtení prvního sčítance
        7009 EB XCHG ;jeho uchování v DE
        700A 2A0270 LHLD 7002H ;načtení druhého sčítance
        700D 19 DAD D ;a sečtení s prvním
        700E 220470 SHLD 7004 ;uchování výsledku
        7011 C9 RET ;navrát z podprogramu
```

Zadejte: SUB 7000 6822 5377 0000 2A0070 EB 2A0270 19 220470 C9
(EOL,CLEAR)
JUMP 7006 (EOL)
MEM 7004 BB 99 ... (výsledek součtu)

Instrukce DAD H může být využita například pro jednoduché násobení nebo posouvání dat v 16-bitovém registru:

5.příklad: HL: 0000 0000 0000 0011 = 0003
DAD H: 0000 0000 0000 0110 = 0006 (x 2) =6
DAD H: 0000 0000 0000 1100 = 000CH (x 4) =12
DAD H: 0000 0000 0001 1000 = 0018H (x 8) =24
DAD H: 0000 0000 0011 0000 = 0030H (x 16) =48
DAD H: 0000 0000 0110 0000 = 0060H (x 32) =96

2.úloha: vypočtete druhou mocninu čísla pomocí tabulky

```
----- 701E 03 DB 3 ;mocněnec (vstupní číslo)
        701F 00 DB 0 ;místo pro výsledek
        7020 3A1E70 LDA 701EH ;čti data (vstup)
        7023 6F MOV L,A ;pomocí dat vytvoř vstupní
        7024 2600 MVI H,0 ; adresu tabulky: HL=0003
        7026 112F70 LXI D,702FH ;počáteční adresa tabulky
        7029 19 DAD D ;vypočti adresu mocniny
        702A 7E MOV A,M ;přečti mocninu čísla
        702B 321F70 STA 701FH ;ulož výsledek
        702E C9 RET ;navrát z podprogramu
        702F 00 DB 0 ;0^2
        7030 01 DB 1 ;1^2
        7031 04 DB 4 ;2^2
        7032 09 DB 9 ;3^2
        7033 10 DB 16 ;4^2 =10H
        7034 19 DB 25 ;5^2 =19H
        7035 24 DB 36 ;6^2 =24H
        7036 31 DB 49 ;7^2 =31H
```

Zadejte: SUB 701E 0300 3A1E70 6F 2600 112F70 197E 321F70...
JUMP 7020
MEM 701F 09 ... (výsledek)

Aritmetické výpočty, které nelze realizovat pomocí několika instrukcí, se často řeší pomocí tabulek, jejichž použití urychlí výpočet. Tabulky však obsazují část paměti, proto je nutno rozhodovat o jejich použití kompromisem mezi rychlostí výpočtu, přesností a nároky na paměť.

Instrukce DAA (Decimal Adjust Accumulator) upraví 8-bitový výsledek předcházející operace ve střadači tak, že v něm vytvoří dvě čtyřbitové dvojkové kódované desítkové číslice (kód BCD: Binary Coded Decimal). DAA se používá po sčítání desítkových čísel a jako jediná využívá a vyžaduje příznak pomocného přenosu AC.

Instrukce pracuje takto:

- a/je-li hodnota pravé čtveřice bitů střadače >9, nebo
je-li nastaven pomocný přenos AC v příznakovém registru,
připočte se k obsahu střadače hodnota 06;

6. příklad:	0000 0101 + 0000 0111 ----- A: 0000 1100 = 0CH (DAA=> +6: 0000 0110 po DAA: 0001 0010 = 12	0000 1001 + 0000 0111 ----- A: 0001 0000 = 10H +AC 0000 0110 0001 0110 = 16
-------------	---	--

výsledek ve střadači je desítkové a nikoli šestnáctkové číslo!

- b/je-li hodnota levé poloviny střadače >9, nebo
je-li předchozí instrukcí nastaven přenos CY,
připočte DAA k obsahu střadače hodnotu 60H;
pokud je splněna i podmínka a, připočte se hodnota 66H.

7. příklad:	0101 0101 = 55 + 0101 0101 = 55 ----- A: 1010 1010 DAA=> +66: 0110 0110 po DAA: CY=1 0001 0000 = 110	1001 0000 = 90 + 1001 0000 = 90 ----- CY=1 0010 0000 +60 0110 0000 CY=1 1000 0000 = 180
-------------	---	--

číslice odpovídající stovce je v CY bitu jako přenos do dalšího řádu. Praktickou aplikaci DAA uvedeme po seznámení se s instrukcemi sečítání a odečítání. Kód DAA je 27H, trvá 4 takty a mění příznakové bity.

TEMA: INSTRUKCE PRO VĚTVENÍ PROGRAMU

Lekce: 4

=====

=====

NOVE POJMY: Nepodmíněné a podmíněné skoky, cykly,
 ----- čekací smyčky a jejich přesnost.

NOVE INSTRUKCE: JMP adr JUMP unconditional -skok na danou adresu
 ----- JC adr Jump on Carry-skok při nastavení CY=1
 JNC adr Jump on No Carry-skok při nastavení CY=0
 JZ adr Jump on Zero-skok při nastavení Z=1
 JNZ adr Jump on No Zero-skok při nastavení Z=0
 JPE adr Jump on Parity Even- skok při PE=1
 JPO adr Jump on Parity Odd-skok při PE=0
 JM adr Jump on Minus-skok při nastavení S=1
 JP adr Jump on Positive-skok při nastavení S=0
 PCHL HL to Program Counter-skok na adr. v HL

Instrukce pro větvení programu lze rozdělit do čtyř skupin:
 -skupina instrukcí JMP: nepodmíněný a 8 podmíněných skoků dle F;
 -skupina instrukcí volání podprogramů typu CALL a Cxx;
 -skupina instrukcí návratu z podprogramů typu RET a Rxx;
 -speciální instrukce PCHL a restartové typu RSTi.

Instrukce nepodmíněného skoku pracuje tak, že přečtený operand (adresu) přepíše do programového čítače PC, čímž způsobí pokračování programu na nové adrese (skok na tuto adresu). Je obdobou instrukce GOTO v Basicu. Používá se obvykle k přeskoku pole dat v programu, často se používá k realizaci tabulky vnějších vstupů do souboru podprogramů nebo programových modulů, např. jsou takto volány jednotlivé služby modulu BDOS v systému CP/M a také služby nového MONITORU C2717 počínaje adresou 9000H. Přesvědčte se o tom: DUMP 9000, a uvidíte: C3, C3, C3 atd.

Podmíněné skokové instrukce provedou totéž co JMP za předpokladu, že je splněna podmínka, vyjádřená pojmenováním (kódem) instrukce, jak je stručně uvedeno v záhlaví lekce.

Skokové instrukce nemění příznakové bity, trvají 10 taktů a jejich délka je 3 byty (kód a adresa):

C3 JMP A C2 JNZ A D2 JNC A E2 JPO A F2 JP A
 CA JZ A DA JC A EA JPE A FA JM A

Nejlépe si činnost instrukcí objasníme na těchto úlohách:

1. úloha: naplňte definovanou zónu videopaměti konstantou 36H.

-----	7000	2100E0	LXI HL, 0E000H	; počáteční adresa zóny VRAM
	7003	06FF	MVI B, 255	; počet adres zóny VRAM
	7005	3636	MVI M, 36H	; zapíše konstantu na adresu
	7007	23	INX H	; další adresa
	7008	05	DCR B	; zmenšení čítače bytů zóny
	7009	C20570	JNZ 7005H	; čítač nulový? ne, skok
	700C	C9	RET	; ano, návrat z podprogramu

Zadejte: SUB 7000 2100E0 06FF 3636 2305 C20570 C9 (EOL,CLEAR)
JUMP 7000 - a uprostřed obrazovky se vytvoří silná pře-
rušovaná čára, tvořená 4 mikrořádky bytů 36H

Co se stane, bude-li mít adresa 7006 nulový obsah (tj. MVI M,0)?
Zkuste na 7006 zadat postupně šestnáctková data: FF,BF,7F a 3F.

Část informace, uložená v bytu zapisovaném do VRAM, svými
nejvyššími bity tvoří tzv. atributy zobrazení: bit 7=0 značí zo-
brazení stále, zatímco bit 7=1 značí blikání; bit 6=0 je pro plný
jas a poslední atribut bit 6=1 je pro zobrazení polojasem:

00xxxxxxx -zobrazení obsahu 'xxxxxx' plným jasem;
01xxxxxxx -zobrazení obsahu 'xxxxxx' polojasem;
10xxxxxxx -zobrazení obsahu 'xxxxxx' blikáním plným jasem;
11xxxxxxx -zobrazení obsahu 'xxxxxx' blikáním polojasem.

2.úloha: modifikujte předchozí program na zápis obsahu B do VRAM
----- 7005 70 MOV M,B ;ulož obsah čítače do VRAM
7006 00 NOP ;prázdná instrukce

Zadejte: MEM 7005

Změňte: SUB 7005 70 00... (EOL,CLEAR)

Start: JUMP 7000 - na původním místě se zobrazí:

1. mikrořádek blikající polojasem (byty FFH-C0H);
2. mikrořádek blikající plným jasem (BFH-80H);
3. mikrořádek polojasem (7FH-40H) a
4. mikrořádek plným jasem (3FH-00).

Na stínítku se zobrazuje vedle sebe maximálně 48 znaků, tj.
30H (00-2FH), dalších 16=10H bytů není zobrazováno, a je používáno
monitorem počítače nebo BASICem jako zápisníková paměť (pouze
některé adresy od C030H do C1F0H). První mikrořádek paměti tvoří
adresy C000-C02FH, v polovině obrazovky jsou to např. adresy E000
-E02FH. Další mikrořádek má adresy o 64=40H větší. S touto sku-
tečností počítá následující úloha.

3.úloha: naplňte vybrané adresy videopaměti konstantou 36H.

----- 7000 2100E0 LXI HL,0E000H ;počáteční adresa zóny VRAM
7003 0650 MVI B,80 ;počet mikrořádků VRAM
7005 114000 LXI D,40H ;přírůstek adres mikrořádku
7008 3636 MVI M,36H ;zapiše konstantu na adresu
700A 19 DAD D ;další mikrořádek
700B 05 DCR B ;zmenšení čítače mikrořádků
700C C20870 JNZ 7008H ;čítač nulový? ne, skok
700F C9 RET ;ano, návrat z podprogramu

Funkci programu si ověřte výše uvedeným způsobem. Potom si pro-
gram modifikujte tak, aby nekreslil pod sebe, ale vpravo šikmo
dolů. Toho lze dosáhnout tím, že zvolíte přírůstek 41H místo 40H
v instrukci LXI D,41H. Zkuste si také přírůstek 80H nebo C0H.

Aby se výsledná zobrazení nepřekrývala, smažte si stínítko pomocí kláves $\uparrow + \leftarrow$ (SHIFT a dvojšipka vlevo). Poprvé se změní stínítko na inverzní zobrazení a podruhé se vymaže.

Často je nutné vytvořit programem určité časové zpoždění. V takovém případě je čas odměřován provedením určitého počtu zvolených instrukcí, tvořících cyklus. Každá instrukce se v počítači provádí během přesně definovaného počtu taktů. Známe-li hodinový kmitočet počítače, snadno stanovíme dobu nutnou na provedení kterékoli instrukce. Skokové instrukce trvají 10 taktů. Je-li hodinový kmitočet 2717 2 MHz, je jeho takt 0,5 mikrosekundy. Potom na provedení skokové instrukce spotřebujeme $10 \times 0,5 = 5$ mikrosekund (miliontin sekundy: 10^{-6}). Základní zpoždění, vypočtené z dob trvání jednotlivých instrukcí, je nutno vynásobit počtem opakování těchto instrukcí v cyklu. Uvedeme si následující úlohu:

4. úloha: vytvoření programového modulu pro zpoždění 0,1 sec

```
----- 7000 110034 LXI D,3400H ;parametry zpoždění; 34H=52
          7003 1D      DCR E      ;zmenšení vnitřního čítače
          7004 C20370 JNZ 7003H   ;[E] >0 - opakuj
          7007 15      DCR D      ;zmenšení vnějšího čítače
          7008 C20370 JNZ 7003H   ;[D] >0 - opakuj znovu
          700B C9      RET        ;[DE]=0 - konec podprogramu
```

Obsah registru E je postupně zmenšován (00, FF, FE, ..., 01, 0). Pokud je nenulový, vrací se program podmíněným skokem na 7003H. Až dosáhne nuly, zmenší se obsah registru D. Je-li obsah D nenulový, opakuje se zmenšování E, program končí při nulovém obsahu dvojice registrů DE.

Nyní si vypočítáme trvání jednotlivých smytek a celého programového modulu:

trvání: DCR r - 5 taktů = 2,5 mksec	$256 \times 7,5 = 1920$ mksec
JNZ a - 10 taktů = 5,0 mksec	= 1,9200 milisek
-----	DCR D, JNZ = 0,0075 milisek
celkem 15 taktů = 7,5 mksec	celkem = 1,9275 milisek
	$100 / 1,9275 = 51,88 \Rightarrow \text{reg. D}$

Protože jsme požadovali zpoždění 0,1 sec = 100 milisekund, musíme smyčku zmenšování E opakovat asi 52 krát, abychom dosáhli zadané zpoždění. Do reg. D můžeme zadat pouze celé číslo, nikoli desetinné, proto se bude výsledné zpoždění mírně lišit od zadaného: $52 \times 1,9275 = 100,23$ milisekund (rozdíl jen 0,23 %). Připočteme-li čas instrukcí LXI a RET (20 taktů = 10 mksec), zvětší se celkový čas provedení programu na 100,24 milisekund. Takovou přesnost nemusíme obvykle po prvním sestavení programu získat, a je nutno např. doplnit nějakou instrukci navíc. Obvykle je to instrukce NOP=00 (4 takty=2mksec) nebo MOV r,r (5 taktů), které nemění obsahy žádného registru a tím neovlivní výsledek programu.

Zkuste doplnit smyčku zmenšování reg. E o NOP nebo MOV E,E a získáte: NOP+DCR+JNZ=>19 taktů => 9,5 mksec =>D=41 =>zp=100,02ms
MOV+DCR+JNZ=>20 taktů => 10 mksec 39 100,14ms
V obou případech je přesnost lepší, ovšem program má 1byte navíc. Pokud vás zajímá, jaké největší zpoždění můžete uvedeným původním programem získat, ověřte si výpočtem: $256 \times 1,9275 = 0,493$ s. Prakticky si uvedené řešení ukážeme v následující úloze:

5. úloha: zobrazte mikrořádek bytů se zpožděním 0,5 s mezi byty
-----7000 2100F0 LXI H,0F000H ;adresa začátku mikrořádk. VRAM
7003 062F MVI B,47 ;počet bytů na mikrořádku
-> 7005 70 MOV M,B ;zobrazení obsahu B na stínit.
7006 110000 LXI D,0 ;parametry zpoždění: 256+256
-> 7009 1D DCR E ;zmenšení vnitřního čítače
700A C20970 JNZ 7009H ;[E] >0 - opakuj
700D 15 DCR D ;zmenšení vnějšího čítače
700E C20970 JNZ 7009H ;[D] >0 - opakuj znovu
7011 23 INX H ;další adresa na mikrořádku
7012 05 DCR B ;je už poslední? (B=0 ?)
7013 C20570 JNZ 7005H ;ne, pokračuj se zpožděním
7016 C9 RET ;ano, návrat z podprogramu

Zadejte: SUB 7000 2100F0 062F70 110000 1D C20970 15 C20970 (EOL)
SUB 7011 2305 C20670 C9 (EOL,CLEAR)
JUMP 7000 -program se začne provádět; trvá asi 28 sekund,
prodloužení způsobuje obnovování informace na
obrazovce tzv. kradením cyklů procesoru.

Seznámili jsme se s jedním typem instrukce podmíněného skoku, ostatní použijeme později. Poněkud netypickou instrukcí skoku je PCHL, která vloží obsah páru HL do programového čítače PC, jak tomu napovídá i mnemotechnická zkratka instrukce. PCHL nemá žádný operand, má délku 1byte a trvá 5 taktů. Je jedinou skokovou instrukcí, která převádí řízení na proměnnou hodnotu adresy. Může být tedy výhodně využívána ke skoku na vypočtenou adresu, na adresu určenou z tabulky adres, na adresu nastavenou jiným programem nebo jiným příchodem téhož programu. Ukažme si funkci instrukce PCHL na následujícím příkladu:

6. úloha: popište adresu obsahem registru, využijte PCHL v cyklu.
-----7000 1100D0 LXI D,0D000H ;adresa začátku části VRAM
7003 3E00 MVI A,0 ;1. byte pro VRAM přes střadač
-> 7005 12 STAX D ;zapsán do paměti
7006 13 INX D ;další adresa na mikrořádku
7007 3D DCR A ;je už poslední? (A=0 ?)
7008 CA0F70 JZ 700FH ;ano, skok na konec podprogramu
700B 210570 LXI H,7005H ;ne, příprava adresy skoku
700E F9 PCHL ;skok na začátek cyklu
-> 700F C9 RET

Zapište úlohu známým způsobem do paměti a spusťte: JUMP 7000.

TEMA: INSTRUKCE PRO PRACI S PODPROGRAMY

Lekce: 5

NOVE POJMY: Volání podprogramu podmíněné a nepodmíněné,
----- návrat z podprogramu podmíněný a nepodmíněný,
uchování návratové adresy v zásobníku.

NOVE INSTRUKCE: CALL a-nepodmíněné volání podprogramu na adrese
----- CC a,CNC a,CZ a,CNZ a,CPE a,CPO a,CM a,CP a
-podmíněná volání podprogramů na adrese a
RET -nepodmíněný návrat z podprogramu
RC, RNC, RZ, RNZ, RPE, RPO, RM, RP
-podmíněné návraty z podprogramů

Podprogramem rozumíme úsek programu, který může být vícenásobně využit tak, že je volán (CALL) z různých míst hlavního programu. Po jeho provedení je řízení vráceno zpět za instrukci volání na tzv. návratovou adresu. Tato návratová adresa byla při volání umístěna v programovém čítači PC, a proto musí být během provádění podprogramu někde uchována, neboť podprogram používá ke své činnosti také PC. Pro uchování PC slouží zásobník (STACK) umístěný někde v paměti počítače. Adresa+1 jeho první volné buňky je uložena v registru SP (Stack Pointer - ukazovátka zásobníku) procesoru, aby byla kdykoli rychle k dispozici. Zápis této adresy pomocí LXI SP,adr se provádí obvykle po zapnutí počítače. Jakou má skutečnou hodnotu se můžeme přesvědčit takto:

1.úloha: zjistíte současnou adresu v ukazateli zásobníku SP
----- 7000 210000 LXI H,0 ;vynulování páru HL
7003 39 DAD SP ;přičtení hodnoty-adresy v SP
7004 220870 SHLD 7008H ;uchování aktuální SP v RAM
7007 C9 RET ;návrat z podprogramu
7008 0000 DW '0' ;vynulování buněk pro SP
Zadejte: SUB 7000 210000 39 220800 C90000 (EOL,CLEAR)
JUMP 7000 (EOL)
MEM 7008 xx 7F ... => 7Fxx je hodnota uchovaného SP.

Instrukce CALL adr. se vykonává ve dvou fázích:

- zmenší hodnotu ukazovátka v SP o 2, a na takto určenou adresu uloží návratovou adresu (aktuální obsah PC); ve skutečnosti se na adresu z SP uloží vyšší byte PC, zmenší se SP o 1, na tuto adresu se uloží nižší byte PC, zmenší se SP o 1, aby byl připraven na uložení dalších návratových adres nebo jiných dat;
- do PC vloží adresu začátku podprogramu (2. a 3. byte instrukce CALL), čímž předá 'procesor' podprogramu.

Volání podprogramu si můžeme představit jako programovou vsuvku, zařazenou v místě volání; je to odskok programu na určenou adresu (začátek podprogramu), a po vykonání tohoto úseku návrat zpět do původního programu.

Pro návrat z podprogramu zpět do hlavního programu je určen soubor instrukcí RETURN -nepodmíněná RET a podmíněné ostatní:

C9 RET	C0 RNZ	D0 RNC	E0 RPO	F0 RP
	C8 RZ	D8 RC	E8 RPE	F8 RM

Návratové instrukce převedou uchovanou návratovou adresu ze zásobníku zpět do PC a zvětší adresu ukazovátka v SP o +2. Program pokračuje tou instrukcí, která následovala po instrukci volání - nepodmíněného CALL, nebo podmíněného nastavením příznaků.

CD.... CALL a					
C4.... CNZ a	D4.... CNC a	E4.... CPO a	F4.... CP a		
CC.... CZ a	DC.... CC a	EC.... CPE a	FC.... CM a		

Jako příklad si uvedeme volání podprogramu zobrazení znaku, jehož ASCII kód je předáván ve střadači:

2. úloha: zobrazte na stínítku pod posledním výpisem daný text

----- 7000 110F70	LXI D,700FH	adresa začátku textu
7003 0605	MVI B,5	počítadlo znaků textu
7005 1A	LDAX D	znak je přes střadač předáván
7006 CD0085	CALL 8500H	podprogramu zobrazení znaku
7009 13	INX D	adresa dalšího znaku textu
700A 05	DCR B	byl to poslední znak?
700B C20570	JNZ 7005H	ne, pokračuj
700E C9	RET	ano, konec podprogramu
700F 535441	DB 'STA'	zóna textu
7012 5254	DB 'RT'	

Program zapište do paměti obvyklým způsobem a spusťte jej. Pod posledním opsaným příkazem (JUMP 7000) se vypíše text START.

Pro vysvětlení funkce zásobníku si znázorníme vývoj PC a SP při provádění programu. Budeme předpokládat, že [SP]=7FFE. Vedle zásobníku jsou uvedeny i obsahy párů registrů, pokud je program ovlivňuje (původní nebo jinde měněné hodnoty jsou označeny ...).

PC	obsah	SP	obsah	B	C	D	E	H	L	A	F
7000	11....	7FFE	700F
7003	06...	7FFE	..	05..	700F
7005	1A...	7FFE	..	05..	700F	53..
7006	CD0085	7FFE	..	05..	700F	53..
7009	->	7FFD	-> 70								
		7FFC	-> 09								
8500	C5	7FFC	..	05..	700F	53..
...											
8578	C9	7FFC	..	05..	700F
7009	<-	7FFC	<- 09								
		7FFD	<- 70								
7009	13	7FFE	..	05..	7010
700A	05	7FFE	..	04..	7010	Z=0
700B	C20570	7FFE	..	04..	7010
7005	1A...	7FFE	..	04..	7010	54..
7006	CD0085	7FFE	..	04..	7010	54..

Při využívání podprogramů je nutno mít na zřeteli nejen parametry nebo data, která podprogramu předáváme (v našem případě kód znaku ve střadači), ale také to, které registry podprogram používá a tím mění jejich původní obsah. V našem příkladu bychom asi těžko získali správný výpis na obrazovce, kdyby podprogram na adrese 8500H měnil obsahy registrů B, D a E. Potom by nám nepracoval správně hlavní program po návratu z podprogramu. Ve skutečnosti volaný podprogram tyto registry používá, ale před jejich použitím uchová jejich obsah a před návratem jej obnoví. Jak se to dělá se dozvíme později.

Uvedenou skutečnost bychom si měli dobře zapamatovat a každý podprogram doplnit informacemi tohoto typu:

- které registry podprogram používá (mění jejich obsah);
- jak podprogram přebírá parametry pro činnost při jeho volání;
- kde podprogram předává výsledky své činnosti.

Pokud jsou podprogramy psány v jazyku symbolických instrukcí (assembleru), uvádí se doplňkové informace uvedené středníkem (jako poznámka). Pro naši úlohu by to mohlo být takto:

podprogram výpisu textu na stínítko

používá registry A, B, D, E

vstupy: text v těle podprogramu

výstupy: text na stínítku C2717

Zkusme si poslední úlohu upravit na několikanásobné použití určené obsahem nevyužitého registru C. Abychom nemuseli celý podprogram přepisovat, zařadíme jeho volání až za něj od adr. 7014H.

3. úloha: zopakujte úlohu 2 podle hodnoty registru C

-----7014 0E40	MVI C, 64	zpět opakování podprogramu
7016 CD0070	CALL 7000H	jeho volání
7019 000000	DS 3	vyhrazeny 3 byty pro CALL
701C 0D	DCR C	je vyčerpán počet opakování?
701D C21670	JNZ 7016H	ne, pokračuj
7020 C9	RET	ano, konec podprogramu

Zadejte program do paměti a spusťte jej známým způsobem.

Pokud by se vám to zdálo moc jednoduché, zkuste si zavolat ještě jeden podprogram MONITORU C2717 'cyklická změna atributů' na adrese 8C6BH a zařadte jej do 3. úlohy místo úmyslně prázdných bytů 000000 (3x instrukce NOP):

7019 CD6B8C	CALL 8C6BH	volání podprogramu cyklické změny atributů zobrazení
-------------	------------	--

Parametry mohou být předávány různými způsoby, například:

- v registrech, obvykle v opačném pořadí univerzálnosti: B, C, D, E, H, L, A, pro výstup z podprogramu naopak univerzální A, HL a pod.

- ve stavových-příznakových bitech jsou nejčastěji předávány výstupní parametry, např. CY=1 (příznak chyby), Z=1 (př.srovnání);
- v určeném paměťovém místě nebo paměťové zóně -dohodnuté adresy;
- v paměťových místech bezprostředně za instrukci CALL; potom je nutné zvětšit návratovou adresu o počet využitých buněk paměti;
- nedoporučuje se k tomu používat zásobník, komplikuje to řešení.

Možná se vám při některé z předcházejících úloh stalo, že se program zacyklil vlivem nesprávně zadanych instrukcí. V tom případě se v dialogovém řádku objevilo ++Executive++ a počítač je neovladatelný. Zkusme si do takového programu zařadit testování stisku klávesy STOP, které je v MONITORU na adrese 8C74H. Podmínky použití podprogramu jsou:

```

;test klávesy STOP
;registry: používá A
;návrat:   Z, A=3   => STOP stisknuto
           NZ, A=40H => STOP nestisknuto
8C74 DBF5 IN 0F5H ;čtení stavu klávesnice na portu SYS55B=F5H
8C76 E640 AND 40H ;STOP je v bitu 6, je aktivní?
8C78 C0 RNZ ;STOP nestisknuto
8C79 3E03 MVI A,3 ;kód stisku STOP
8C7B C9 RET ;STOP je stisknuto

```

Pro použití podprogramu nám postačuje zařadit jej tam, kde nebude mít vliv na obsah střadače, za CALL 8C74H zařadit podmíněný návrat z podprogramu RZ (po stisku STOP se končí podprogram).

4.úloha: zařaďte do podprogramu 3.úlohy test stisku klávesy STOP

```

-----7014 0E40 MVI C,64 ;počet opakování podprogramu
       7016 CD0070 CALL 7000H ;jeho volání
       7019 CD6B8C CALL 8C6BH ;cyklická změna atributů obrazu
       701C CD748C CALL 8C74H ;test STOP: Z=>stisknuto
       701F C8 RZ ;ukončení podprogramu po STOP
       7020 0D DCR C ;je vyčerpán počet opakování?
       7021 C21670 JNZ 7016H ;ne, pokračuj
       7024 C9 RET ;ano, konec podprogramu

```

Zadejte program do paměti a po jeho spuštění jej zkuste zastavit pomocí klávesy STOP. Pokud to nestihnete, po ↑+RCL vyvoláte znovu JUMP 7014 a na druhý nebo třetí pokus to snad vyjde. A pokud ne, zadejte větší počet opakování na adrese 7015H (>40H).

Snad jste si všimli, že jsou v podprogramu dva návraty RZ a RET, ale to není nic divného: jeden je podmíněný, a pokud není podmínka splněna, uplatní se ten poslední -nepodmíněný.

TEMA: INSTRUKCE PRO SEČITÁNÍ

Lekce: 6

=====

=====

NOVE POJMY: Sčítání 8 bitových dat bez přenosu i s přenosem.
----- programové násobení, převod dvojkově-dekadický,
simulované provádění programu tužkou na papíře.

NOVE INSTRUKCE: ADD r ADD register/memory-součet registrů/pam.
----- ADI d8 ADD Immediate-přičtení přímé konstanty
ADC r ADD reg. with Carry-součet s přenosem
ACI d8 Add Imm. with Carry-součet s konst. a přen

Všechny následující aritmetické instrukce používají sčítače pro uložení jednoho ze vstupních operandů a potom k uložení výsledku operace.

Instrukce ADD r připočte k obsahu sčítače obsah registru r a výsledek uloží do sčítače, čímž jeho původní obsah přepíše. Instrukce ADD ovlivňuje výsledkem všechny příznaky, trvá 4 takty nebo 7 taktů, je-li druhý sčítanec uložen v paměti (ADD M):

80 ADD B 82 ADD D 84 ADD H 86 ADD M
81 ADD C 83 ADD E 85 ADD L 87 ADD A

Příklad:

----- [A] = 2CH = 0010 1100 Příznakové bity:
[B] = 6EH = 0110 1110 S Z AC PE CY
po ADD B: [A] = 9AH = 1001 1010 [F] = 1 0 0 1 0 1 1 0

Instrukce ADI d8 = C6xx připočte ke sčítači přímou konstantu xx. Je to 2 bytová instrukce, trvá 7 taktů a mění příznaky F.

Instrukce ADC r připočte ke sčítači jak obsah registru, tak i obsah přenosového bitu CY. Trvá 4 takty nebo 7 taktů (ADC M) a mění příznakové bity:

88 ADC B 8A ADC D 8C ADC H 8E ADC M
89 ADC C 8B ADC E 8D ADC L 8F ADC A

Podobným způsobem připočte konstantu a přenos CY k obsahu sčítače instrukce s přímým operandem ACI d8 = CExx, která trvá 7 taktů, má 2 byty a mění příznaky.

1. úloha: sečtete celá čísla na adresách 7000H a 7001H a výsledek
----- uložte na adresu 7002H resp. 7003H

7000 A0	DB 160	1. sčítanec	160	A0H
7001 61	DB 97	2. sčítanec	+ 97	61H
7002 0000	DW 0	uložení výsledku:	257 = 0101H	
=> 7004 210070	LXI H, 7000	adresa 1. sčítance		
7007 7E	MOV A, M	načten do sčítače		
7008 23	INX H	adresa 2. sčítance		
7009 86	ADD M	sečten s prvním		
700A 23	INX H	adresa uložení výsledku		
700B 77	MOV M, A	uložen		
700C D0	RNC	nebyl-li přenos, tak návrat		
700D 23	INX H	adresa uložení přenosu součtu		
700E 3E00	MVI A, 0	vynulování sčítače		
7010 8F	ADC A	přičtení CY do sčítače		
7011 77	MOV M, A	a uchování v paměti		
7012 C9	RET	konec programu		

Výsledek práce programu si ověřte známým způsobem. Instrukci ADC nemůžeme použít jako první při sčítání (adr. 7009H), neboť neznáme stav CY bitu. Podmíněný návrat RNC zkracuje trvání programu, nedojde-li k přenosu při sčítání. Dojde-li k přenosu, musíme CY přičíst k vynulovanému sčítadlu, jinak bychom se dopustili chyby.

Instrukci ADD A zdvojnásobíme obsah sčítadla. To lze využít při násobení čísel, např. v následující úloze:

2. úloha: převedte číslo 29 z BCD kódu do binárního (=1DH).

```

-----7000 02      DB 2          ;desítky a
       7001 09      DB 9          ;jednotky převáděného čísla
       7002 00      DB 0          ;adresa uložení výsledku
=> 7003 210070     LXI H,7000H     ;ukazatel na adresu desítek
       7006 7E      MOV A,M        ;desítky do sčítadla
       7007 87      ADD A          ;vynásobeny 2x: [A]=4
       7008 47      MOV B,A        ;a uložen: [B]=4
       7009 87      ADD A          ;desítky x4: [A]=8
       700A 87      ADD A          ;desítky x8: [A]=10H
       700B 80      ADD B          ;desítky x10: [B]+[A]=14H=20
       700C 23      INX H          ;ukazatel na jednotky čísla
       700D 86      ADD M          ;přičteny jednotky
       700E 23      INX H          ;adresa uložení výsledku
       700F 77      MOV M,A
       7010 C9      RET

```

Ověřte si funkčnost programu nejen v MONITORU (JUMP 7003 a MEM 7002), ale i pomocí Basicu: PRINT(USR('7003)) - vytiskne obsah sčítadla, nikoli buňky 7002H.

Dále si zkusíme jednoduché 8-bitové binární násobení, v němž násobíme nulou nebo jednotkou v osmi bitech bytu (proto 8 cyklů) -bude-li bit násobitele 1, přičteme k mezisoučinu hodnotu násobence, bude-li 0 nic se nestane, pokračuje se v cyklu.

3. úloha: vynásobte čísla na adresách 7000H a 7001H; výsledek uložte na adresy 7002H a 7003H.

```

-----7000 02      DB 2          ;násobenec: 0000 0010
       7001 09      DB 9          ;násobitel 0000 1001
       7002 0000     DW 0          ;adresy uložení výsledku
=> 7004 210070     LXI H,7000H     ;adresa násobence
       7007 5E      MOV E,M        ;násobenec načten do
       7008 1600     MVI D,0        ; páru DE: [DE]=0002
       700A 23      INX H          ;adresa násobitele
       700B 7E      MOV A,M        ;načten násobitel
       700C 210000   LXI H,0        ;vynulován pár pro součin
       700F 0608     MVI B,8        ;počítadlo cyklů=bitů registru
-> 7011 29          DAD H          ;mezisoučin: 0,0,0,0,0,2,4,8,16
       7012 87      ADD A          ;posun násobit. do CY: 0001 0010

```



```

7013 D21770 JNC 7017      ;je násobeno jedničkou?
7016 19      DAD D        ;ano: [HL]=2,12H
-> 7017 05     DCR B        ;ne; všechny bity násobitele?
7018 C21170 JNZ 7011H     ;ne,pokračuj
701B 220270 SHLD 7002H    ;ulož výsledek
701E C9      RET

```

Celý proces binárního násobení spočívá v následujícím:
a-inicializace součinu a nastavení počítadla bitů=cyklů (8)
b-posun součinu o 1 bit vlevo (7011H)
c-posun násobitele o 1 bit vlevo (7012H)
d-při CY=1 seříst mezisoučin a násobence (7016H)
e-zmenšit počítadlo a test jeho nulovosti, je-li >0 jde se na b.
Protože postup je netradiční, zkusme jej znázornit simulací
práce procesoru na papíře. Uvidíme, co se děje v registrech:

	BC	DE	HL	CY	A
7007 MOV E,M:MVI D,0		0002			
700B MOV A,M					00001001
700C LXI H,0			0000		
700F MVI B,8	0800				
7011 DAD H			0000		
7012 ADD A:JNC 7017				0	00010010
7017 DCR B:JNZ 07					
7011 DAD H			0000		
7012 ADD A:JNC 7017				0	00100100
DCR B:JNZ 06					
7011 DAD H			0000		
7012 ADD A:JNC 7017				0	01001000
DCR B:JNZ 05					
7011 DAD H			0000		
7012 ADD A:JNC 7017				0	10010000
DCR B:JNZ 04					
7011 DAD H			0000		
7012 ADD A				1	00100000
7016 DAD D			0002		
7017 DCR B:JNZ 03					
7011 DAD H			0004		
7012 ADD A:JNC 7017				0	01000000
DCR B:JNZ 02					
7011 DAD H			0008		
7012 ADD A:JNC 7017				0	10000000
DCR B:JNZ 01					
7011 DAD H			0010		
7012 ADD A				1	00000000
7016 DAD D			0012		
7017 DCR B	00				
701B SHLD 7002H			0012		----->[7002]=12

[7003]=00

Zkuste si vypočítat, jak dlouho takové násobení trvá. Volná místo vpravo od simulace programu k tomu přímo vybízí. Nezapomínejte na časy neuvedených instrukcí.

V páru HL jsme získali dvojkový (binární) výsledek. Pokusme se jej převést na desítkový změnou konce předchozího programu:

4. úloha: převedte dvojkový obsah HL na desítkový

```
-----701E 110000 LXI D,0      ;navázání na 3.úlohu -místo RET
       7021 010010 LXI B,1000H ;B-čítač cyklů (16 bitů má HL)
       7024 79     MOV A,C      ;A=00
-> 7025 29     DAD H      ;další bit: 0024
       7026 7B     MOV A,E
       7027 8F     ADC A      ;přičítá se přenos z DAD H
       7028 27     DAA
       7029 5F     MOV E,A      ;nižší 2 číslice
       702A 7A     MOV A,D
       702B 8F     ADC A      ;přičítá se přenos z DAA
       702C 27     DAA
       702D 57     MOV D,A      ;vyšší 2 číslice
       702E 79     MOV A,C
       702F 8F     ADC A      ;přičítá se přenos z DAA
       7030 4F     MOV C,A      ;nejvyšší číslice
       7031 05     DCR B      ;B=0F,0E,0D,0C
       7032 022570 JNZ 7027H    ;ještě
       7035 EB     XCHG      ;už ne, výsledek je v A a HL
       7036 224070 SHLD 7040H   ;uloží HL do paměti
       7039 324270 STA 7042H    ;a za ně i nejvyšší číslici z A
       703C C9     RET
```

Zadejte i tuto úlohu do paměti a spusťte úlohu 3: JUMP 7004
o výsledku se převeďte na adresách 7040-7042 pomocí DUMP 7040.

Zkuste zadat i jiné vstupní proměnné, tj. násobence a násobitele na adresách 7000 a 7001.

Samostatné spuštění 4. úlohy je sice možné, ale výsledek bude jiný, neboť mezi úlohami se změnil obsah HL použitými funkcemi MONITORu C2717.

TEMA: INSTRUKCE PRO ODEČITÁNÍ

Lekce: 7

NOVE POJMY: Odčítání 8 bitových dat s výpůjčkou i bez ní,
programové dělení.

NOVE INSTRUKCE: SUB r SUBtract register/memory-odečti reg./pamět
SUI d8 SUBtract Immediate-odečti přímou konstantu
SBB b SuBtract register/memory with Borrow-
odečti registr/pamět s výpůjčkou
SBI d8 Subtract Immediate with Borrow
odečti přímou konstantu s výpůjčkou

Všechny instrukce odečítání používají střadat, od jehož obsahu (menšence) je odečítán menšitel, uložený v registru, paměti nebo uvedený jako přímá konstanta v instrukci. Výsledek je vždy ukládán ve střadači.

Instrukce SUB r odečte od střadate obsah registru r; výsledek přepíše původní obsah střadate, ovlivní všechny příznakové bity. Trvá 4 takty, je-li menšence v paměti (M), tak 7 taktů.

90 SUB B 92 SUB D 94 SUB H 96 SUB M
91 SUB C 93 SUB E 95 SUB L 97 SUB A

Mikroprocesor odečítat přímo neumí, ale používají pro odčítání metodu přičtení dvojkového doplňku a výsledný přenos CY po tomto přičtení negují. Dvojkový doplněk je vypočten tak, že se k jednotkovému doplňku binárního vyjádření čísla (inverzní číslo s opačnými hodnotami jednotek a nul) přičte jednotka.

1. příklad: stanovte dvojkový doplněk čísla 3:

----- dvojkové vyjádření čísla: 3 = 00000011

jednotkový doplněk čísla: 3 11111100
připočtení jednotky: + 00000001

dvojkový doplněk čísla: 3 = 11111101 přenos: 0 → 1

2. příklad: odečtete od střadate [A] = 9 registr [E] = 3

----- [A] = 9 = 00001001
SUB E = -3 = 11111101 -dvojkový doplněk 3

'součet' = 6 [CY] = 1 00000110 = [A] - výsledek: 6 → CY = 0

Význam inverze [CY] pochopíme později při odečítání větších čísel od menších, kdy je nutno počítat s výpůjčkou z vyššího řádu čísla, které je uloženo ve dvou a více bytech.

1. úloha: Odečtete dvě čísla uložená v různých registrech

```
-----7000 0685      MVI B,85H      ,menšenec
          7002 1E42      MVI E,42H      ,menšitel
          7004 78        MOV A,B        ,menšenec do střadače      85H
          7005 93        SUB E          ,odečten menšitel          -42H
          7006 320A70     STA 700AH      ,uložení výsledku          43H
          7009 C9        RET
          700A 00        DB 0           ,buňka pro uložení výsledku 43H
```

Zadejte program do paměti a spusťte jej. Ověřte jeho funkčnost i z Basicu tak, že pomocí POKE '7001,xx zapíšete menšence, pomocí POKE '7003,yy zapíšete menšitele. Program spustíte pomocí příkazu A=USR('7000) a výsledek ověříte pomocí PRINT A. Zkuste zadat větší menšitel a zdůvodnit, proč je výsledek takový, jaký je.

Vynulovat střadač můžeme jednoduše pomocí instrukce SUB A, což se často využívá. Zkuste si vynulovat střadač podobně 2. příkladu a zjistíte, že nastal přenos a proto je CY=0, Z=1, S=0, AC=1, PE=1

Instrukce SUI d8=D6yy odette od střadače přímou konstantu yy; je to 2 bytová instrukce, trvá 7 taktů a mění příznaky.

Odečítáme-li ve vicemístném čísle větší číslici od menší, např. 432-246, počítáme, jak známo, zprava: šest a kolik je dvánáct nikoli: šest a kolik je dvě. Pro toto odečtení si půjčujeme 1 z vyššího řádu čísla (desítek, stovek a pod.) Aby byl výsledek správný, musíme tutu výpůjčku připočítat k menšiteli takto: 4+1=5 a kolik je 13 (opět 1 vypůjčena), a nakonec 2+1=3 a kolik je 4. Zde už jsme si nepůjčili. Autoři algoritmu odečítání v procesoru 8080 se inspirovali také touto starou lidskou zkušeností, a proto do souboru instrukcí zavedli instrukce s výpůjčkou:

DEyy SBI yy - odečtení přímé konstanty s uvažováním výpůjčky; trvá 7 taktů a nastavuje příznaky F.

```
98 SBB B      9A SBB D      9C SBB H      9E SBB M
99 SBB C      9B SBB E      9D SBB L      9F SBB A
```

Tyto instrukce trvají 4 takty, SBB M 7 taktů, mění příznaky a v paměti zabírají jediný byte.

Následující příklad a úloha budou řešit shodný problém: odečíst 6F5CH - 13C5H, menšenec bude v páru DE a menšitel v BC.

3. příklad: ukázka odečítání 'sečítáním' s výpůjčkou

```
-----BC:13C5      ->      B: 00010011 13H      C: 11000101 C5H
-----
2-doplňák          11101101 EDH          00111011 3BH
DE:6F5C      ->      D: 01101111 6FH      E: 01011100 5CH
-----
mezisoučet:        1 01011100 5CH
výpůjčka (2-d)      11111111 <----1 <- 0 10010111 97H
-----
'součet'           0<- 1 01011011
výsledek: CY=0      5BH          97H
```

2. úloha: uvedený příklad naprogramujte a uložte výsledek do DE

```

-----7000 115C6F LXI D,6F5CH ;menšenec
          7003 01C513 LXI B,13C5H ;menšitel
          7006 7B     MOV A,E     ;od nižšího bytu bude odčítán
          7007 91     SUB C       ;nižší byte menšitele
          7008 5F     MOV E,A     ;a výsledek uložen v E
          7009 7A     MOV A,D     ;od vyššího bytu bude odčítán
          700A 98     SBB B       ;vyšší menšitel s výpůjčkou
          700B 57     MOV D,A     ;a výsledek uložen v D
          700C EB     XCHG        ;přepsán do HL
          700D 221170 SHLD 7011H ;a pro kontrolu do paměti
          7010 C9     RET
          7011 0000     DW 0       ;2 byty v paměti pro výsledek
  
```

Vložte si program do paměti a spusťte. Ověřte si jeho funkci na různých hodnotách 16-bitových čísel.

Dělení se provádí také obdobně ručnicku způsobu pokusného odečítání. Každý krok binárního dělení spočívá v následujícím:

-je-li možné odečíst dělitele od zbývajících osmi vyšších bitů dělenice bez výpůjčky z vyššího řádu, bude v odpovídajícím řádu podílu jednička;

-v opačném případě nula;

-při každém kroku je nutno zajistit správné srovnání řádů dělenice, dělitele i podílu; lze to udělat posunutím dělenice i podílu při každém kroku o jeden bit vlevo;

-dělenec i podíl mohou být v jediném páru registrů, protože se v době určení dalšího bitu podílu uvolní jeden bit dělenice.

Pro dělení 16-bitového čísla 8-bitovým číslem takovým, že je podíl 8-bitový (dělitel > vyšší byte dělenice) a současně jsou znaménkové bity (MSB) obou čísel nulové, lze v literatuře najít program, použitý ve 3. úloze:

3. úloha: podělte v paměti 0040H/08H a výsledek uložte do paměti

```

-----7000 4000     DW 0040H ;dělenec
          7002 08     DB 8     ;dělitel
          7003 0000     DW 0000 ;zóna výsledku
=> 7005 2A0070     LHLD 7000H ;dělenec do HL
          7008 3A0270     LDA 7002H ;dělitel do střadače
          700B 4F     MOV C,A     ;a registru C
          700C 0608     MVI B,8   ;počítadlo cyklu/bitů dělitele
-> 700E 29     DAD H     ;dělenec i podíl o 1 bit vlevo
          700F 7C     MOV A,H     ;vyšší byte dělenice do střadače
          7010 91     SUB C       ;je > než dělitel?
          7011 DA1670     JC 7016H ;ne, pokračuje další cyklus
          7014 67     MOV H,A     ;ano, zmenšen dělenec o dělitele
          7015 2C     INR L       ;přičtena 1 k podílu
-> 7016 05     DCR B       ;byly už všechny bity dělitele?
          7017 C20E70     JNZ 700EH ;ne, pokračuje dělení
          701A 220370     SHLD 7003H ;ano, uchován výsledek
          701D C9     RET
  
```

Výsledek dělení je z registru L uchován na adrese 7003H, a zbytek (< dělitel) je uchován z H registru na 7004H. Podle zadání úlohy by měl být celočíselný výsledek 8. Zkuste si nejen toto zadání, ale i jiný dělenec pro podíl se zbytkem. Zkuste i větší čísla taková, aby byla splněna podmínka nulovosti znamének -MSB.

Instrukce SUB r se často užívá k testování 16-bitových dat s jinými daty, např. zda bylo dosaženo určité adresy nebo naopak zda byl vynulován čítač v páru registrů. Např. pro čítač v BC by to mohlo být: DCR C:JNZ...DCRB:MOV A,B:SUB C:JNZ...skok při B<>C. Jako praktickou úlohu si uvedeme přenos informací (obrazu) na obrazovce na jiné místo:

4.úloha: přeneste levý horní kvadrant stínítka do pravého dolního

```

Px.  7F00 2118FF LXI H,0FF18H ;koncová adr.cíle-vlevo dole
      7F03 22327F SHLD 7F32H   ;uchována pro testování
      7F06 2100C0 LXI H,0C0000H ;počáteční adresa zdroje
      7F09 1118E0 LXI D,0E018H ;počáteční adresa cíle přenosu
-> 7F0C 012818 LXI B,1828H   ;počet bytů zdroje a přírůstek
Gx. -> 7F0F 7E     MOV A,M     ;informace ze zdroje
      7F10 12     STAX D      ;uložena na cílovou adresu
      7F11 23     INX H       ;další adresa zdroje
      7F12 13     INX D       ;další adresa cíle
      7F13 05     DCR B       ;je poslední na mikrotádku?
      7F14 C20F7F JNZ 7F0FH   ;ne, pokračuje přenos
      7F17 09     DAD B       ;ano, přičti přírůstek adr.zdroje
      7F18 EB     XCHG        ;cílovou adresu do páru HL
      7F19 09     DAD B       ;přičti přírůstek k adr. cíle
      7F1A EB     XCHG        ;a vrať zpět do původních reg.
      7F1B 22307F SHLD 7F30H   ;dočasné uchovej adresu zdroje
      7F1E 2A327F LHLD 7F32H   ;načti zadanou konc. adr. cíle
      7F21 7C     MOV A,H     ;odpovídá její vyšší byte
      7F22 92     SUB D       ;vyššímu bytu konc.adr.zdroje?
      7F23 C2297F JNZ 7F29H   ;ne, pokračuje přenos
      7F26 7D     MOV A,L     ;ano, nižší byte cílové adresy
      7F27 93     SUB E       ;odpovídá bytu koncové adresy?
      7F28 C8     RZ          ;ano, konec podprogramu
-> 7F29 2A307F LHLD 7F30H   ;ne, obnovení zdrojové adresy
      7F2C C30C7F JMP 7F0CH   ;pokračuje další přenos
      7F2F 00     DB 0        ;nic
      7F30 0000    DW 0        ;pro uchování adresy zdroje
      7F32 0000    DW 0        ;uchovává koncovou adresu cíle

```

Pokud vás překvapily jiné adresy, je to proto, že můžeme podprogram uplatnit v Basicu takto:

- do proměnné Px запиšeme: Px="2118FF22327F2100C01118E0012818"
- do proměnné Gx запиšeme: Gx="7E12231305C20F... 2A307FC30C7F"
- potom sloučíme obě řetězcové proměnné do jediné: Zx=Px+Gx
- příkazem CODE Zx se strojový kód v proměnných přeneseme do paměti na adresy od 7F00H a spustí se od adresy 7F00H;
- po provedení podprogramu se řízení vrátí do Basicu

Do uvedené oblasti paměti nelze program v ++0s ready++ zapísat, neboť je zde zásobník MONITORU a došlo by ke kolizi.

TEMA: LOGICKE INSTRUKCE A POSUVY

LEKCE: 8

=====

=====

NOVE POJMY: Logický součin (AND), logický součet (OR),

 exkluzivní součet neboli nonekvivalence (XOR),
 logické srovnání (CMP), logické nastavení,
 logický doplněk, posuv střádate vlevo a vpravo,
 NOVE INSTRUKCE: ANA r -AND register/memory=log.součin reg./paměti

 ANI d8-AND Immediate-log.součin s přímou konst.
 ORA r -OR register/memory=log.součet reg./paměti
 ORI d8-OR Immediate-log.součet s přímou konstant.
 XRA r -eXclusive oR reg./mem.-neekvivalence r./p.
 XRI d8-eX.oR Immediate=neekvivalence s konstantou
 CMP r -CoMPare reg./mem.-srovnání reg./paměti
 CPI d8-ComPare Immediate-srovnání s konstantou
 CMA -CoMplement Accumulat.-log.doplněk střádate
 CMC -CoMplement Carry-log.doplněk carry bitu
 STC -SeT Carry-nastavení carry bitu do log. 1
 RRC,RLC-Rotate Right/Left-rotace vpravo či vlevo
 RAR,RAL-Rotate Accumul. Right/Left through carry-
 -rotace střad. vpravo/vlevo přes bit carry

Logické instrukce jsou definovány jako operace mezi daty ve střadati a uvedeným registrem/paměti (M) nebo přímou konstantou. Logická funkce se postupně provede mezi odpovídajícími si bity a podle výsledku ve střadati se nastaví příznakové bity.

Výsledkem logického součinu je log. 1 pouze tehdy, jsou-li oba bity jednotkové. Ukážeme si to na příkladu:

1.příklad:	[A]=00111001=39H	[A]=01100011=33H
-----	0FH=00001111	[L]=00100010=22H
	-----	-----

po ANI 0FH:[A]=00001001=09 po ANA L:[A]=00100010=22H

Instrukce logických součinů trvají 4 nebo 7 taktů, nastavují příznakové bity mimo AC a CY, mají tyto strojové kódy:

E6xx-ANI xx (7T)	A0-ANA B	A2-ANA D	A4-ANA H	A6-ANA M (7T)
	A1-ANA C	A3-ANA E	A5-ANA L	A7-ANA A

Pomocí těchto instrukcí lze provádět tzv.maskování. Jak je vidět z příkladu, vynulují se bity střádate tam,kde má maska nuly, a tam kde má maska jedničky, zůstane původní obsah. Takto se můžeme zbavit nežádoucích bitů, nebo si vybrat pouze potřebné. Levá část příkladu ukazuje, jak lze číslici 9 v kódu ASCII změnit na číslo 09 v kódu BCD pomocí masky 0FH a instrukce ANI 0FH.

Logický součet OR má nulový výsledek pouze v těch bitech, kde jsou obě nuly, v ostatních případech je výsledkem log.1

2. príklad: [A]=00001001=09 [A]=01010101=55H
----- 30H=00110000 [B]=10100110=A6H

 po ORI 30H: [A]=00111001=39H po ORA B: [A]=11110111=F7H

Instrukce logických součtů trvají 4 nebo 7 taktů, nastavují příznakové bity mimo AC a CY, mají tyto strojové kódy:

F6xx-ORI xx (7T), B0-ORA B B2-ORA D B4-ORA H B6-ORA M (7T)
B1-ORA C B3-ORA E B5-ORA L B7-ORA A

Jak ukazuje příklad, lze logickým součtem přidat log.1 tam, kde je to zapotřebí, např. z číslice v kódu BCD udělat kód ASCII stejné číslice: z 09 je 39H.

Neekvivalence je srozumitelnější slovní náhrada pro exkluzivní součet proto, že jednoznačně určuje, které bity jsou nulové (totožné=ekvivalentní) a které jednotkové (neekvivalentní):

3. pŕiklad: [A]=00001001=09 [A]=01010101=55H
 ----- 09H=00001001 [B]=10100110=A6H

 po XRI 09H: [A]=00000000=00H po XRA B: [A]=11110011=FBH

Neekvivalence obsahu střadače se stejnou konstantou vynuluje střadat (XRA A → [A]=00) a nastaví příznakové bity mimo AC a CY, které se vynulují. Instrukce neekvivalence trvají 4 nebo 7 taktů a mají tyto kódy:

EExx-XRI xx (7T) A8-XRA B AA-XRA D AC-XRA H AE-XRA M (7T)
A9-XRA C AB-XRA E AD-XRA L AF-XRA A

Porovnání obsahu střadače s přímou konstantou nebo obsahem registru či paměťové buňky zajišťují následující instrukce:

FExx-CPI xx (7T)	B8-CMP B	BA-CMP D	BC-CMP H	BE-CMP M (7T)
	B9-CMP C	BB-CMP E	BD-CMP L	BF-CMP A

Mimo CPI jsou jednobytové a trvají 4 nebo 7 taktů a příznaky nastavují všechny. Porovnání se uskuteční tak, že se přímý operand nebo daný registr odečte od střadače, aniž se obsah střadače mění, ale podle 'výsledku' se nastaví všechny příznaky, především: Z=1 -výsledkem odečtení je nula, porovnávané obsahy jsou shodné; Z=0 -obsahy se navzájem liší;

CY=1-obsah střadače je < než obsah druhého operandu (registru);
CY=0-obsah střadače je > nebo = obsahu druhého operandu.

Ostatní příznaky se nastaví podle 'výsledku' nikde neuchovaného.

```

4. příklad:      [A]=00000110=06
-----
+(-[B])=11111100=-4
-----
po CMP B: 1 00000110=02
          CY=0  Z=0

```


1. úloha: převedte šestnáctkový kód čísla na ASCII znak čísla

```
-----7000 0C      DB 12      ;převáděné šestnáctkové číslo
       7001 00      DB 0       ;buňka pro výsledek převodu
       7002 3A0070  LDA 7000H   ;načtení hodnoty čísla
       7005 FE0A     CPI 10     ;je 10 nebo více ?
       7007 DA0C70  JC 700CH   ;ne, skok na řešení
       700A C607     ADI 7      ;ano, přičti posun pro písmena
-> 700C C630     ADI '0'      ;přičti základní posun pro nulu
       700E 320170  STA 7001H   ;ulož ASCII výsledek
       7011 C9      RET
```

Zadejte program do paměti, spusťte pomocí JUMP 7002 a ověřte pro různá čísla 00-0F, jimž odpovídají ASCII kódy 30H - 39H a 41H (A) - 46H (F). Posun pro písmena je dán rozdílem kódů pro A=41H a 'kódu' následujícího za 9 (39H), tj. 3AH; proto 41H-3AH=7.

2. úloha: určete délku ASCII řetězu znaků, končícího znakem CR=0DH

```
-----7000 00      DB 0       ;buňka pro uložení délky řetězu
       7001 5052     DW 'P' 'R'
       7003 494B     DW 'I' 'K'
       7005 4C41     DW 'L' 'A'
       7007 440D     DW 'D' '0D' ;řetěz znaků
-> 7009 210170  LXI H,7001H   ;ukazatel počátku řetězu
       700C 0600     MVI B,0   ;počítadlo délky řetězu
       700E 3E0D     MVI A,0DH ;je koncový znak řetězu
-> 7010 BE        CMP M       ;shodný se srovnávaným znakem?
       7011 CA1970  JZ 7019H   ;ano, nalezen konec
       7014 04       INR B     ;ne, přičti 1 k počítadlu délky
       7015 23       INX H     ;adresa dalšího znaku
       7016 C31070  JMP 7010H  ;pokračuj v testování
-> 7019 78         MOV A,B     ;počítadlo délky
       701A 320070  STA 7000H  ;ulož na uvedenou adresu
       701D C9      RET
```

Ověřte funkčnost programu, před druhým spuštěním vložte 0DH na adresu 7004H.

Instrukce pro jednotkový doplněk střadate je označována jako CMA (Complement Accumulator), má kód 2FH, nemění bity příznaků a trvá 4 takty.

5. příklad: [A]=01010011=53H
----- po CMA: 10101100=ACH

Instrukce pro nastavení přenosového bitu CY (carry) má mnemotechnickou zkratku STC (Set Carry), kód 37H, trvá 4 takty, mění pouze CY bit na log. 1.

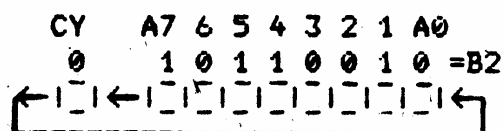
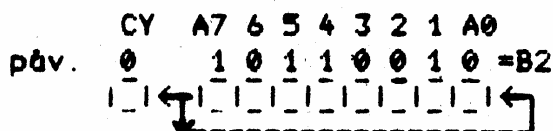
K nastavení CY=0 můžeme po předcházející instrukci použít CMC (Complement Carry), s kódem 3FH a trváním 4 takty, která mění pouze hodnotu CY příznakového bitu na opačnou: 0->1 nebo 1->0.

Pro posuvy dat ve střadači jsou využívány dva typy instrukcí podle toho, jedná-li se o 8-bitový nebo 9-bitový kruhový posuv, tj. posouvá-li se informace pouze uvnitř střadače nebo přes bit CY (proto 9-bitový posuv). Při posuvech vlevo je bit A7 přesunut do A0 a také do CY-instrukce RLC, nebo při RAL je posunut do CY a předchozí obsah CY je posunut do A0; ostatní bity vlevo.

6 příklad:

----- rotace vlevo: RLC

rotace vlevo přes CY: RAL



1. RLC 1 0 1 1 0 0 1 0 1 =65
 2. RLC 0 1 1 0 0 1 0 1 0 =CA
 3. RLC 1 1 0 0 1 0 1 0 1 =95
 4. RLC 1 0 0 1 0 1 0 1 1 =2B

1. RAL 1 0 1 1 0 0 1 0 0 =64
 2. RAL 0 1 1 0 0 1 0 0 1 =C9
 3. RAL 1 1 0 0 1 0 0 1 0 =92
 4. RAL 1 0 0 1 0 0 1 0 1 =25

Z příkladu je zřejmé, že po 4-násobném provedení RLC se prohodí obsahy levé a pravé poloviny střadače, zatímco u RAL se zadá mezi posouvající se bity ještě předem nastavený bit CY.

Instrukce trvají 4 takty a mění pouze příznakový bit CY, jejich kódy a kódy obdobných posuvů vpravo jsou následující:

07-RLC

17-RAL

0F-RRC

1F-RAR

2. úloha: rozdělte byte s dvouciferným číslem na samostatná čísla

-----7000 39 DB 39 ; číslo, které má být rozděleno
 7001 00 DB 0 ; uložení 'desítek'-levé poloviny
 7002 00 DB 0 ; buňka pro 'jednotky'
 => 7003 210070 LXI H,7000H ; adresa vstupních dat
 7006 7E MOV A,M ; čti data ([A]=39)
 7007 47 MOV B,A ; a ulož k pozdějšímu zpracování
 7008 0F RRC
 7009 0F RRC
 700A 0F RRC
 700B 0F RRC ; posuň o 4 bity vpravo ([A]=93)
 700C E60F ANI 0FH ; ponechej 4 nižší bity ([A]=03)
 700E 23 INX H ; adresa uložení 'desítek' čísla
 700F 77 MOV M,A ; ulož výsledek
 7010 78 MOV A,B ; obnovu původního čísla ([A]=39)
 7011 E60F ANI 0FH ; maskuj 4 nižší bity ([A]=09)
 7013 23 INX H ; adresa uložení 'jednotek' čísla
 7014 77 MOV M,A ; ulož výsledek
 7015 C9 RET

TEMA: INSTRUKCE VSTUPU/VÝSTUPU A PRERUŠENÍ.

LEKCE: 9

=====

=====

NOVE POJMY: Vstupní a výstupní operace, adresování

přidavných (periferních) zařízení,
režim přerušení práce procesoru, jeho po-
volení a zakázání, instrukce restartu RST.

NOVE INSTRUKCE: IN a8-Input from port a8 -vstup dat z portu a8

OUT a8-Output to port a8 -výstup dat na port a8
EI -Enable Interrupts -povolení přerušení
DI -Disable Interrupts -zakáz přerušení
RST n -Restart -znovuspuštění na adrese n*8

Soubor instrukcí mikroprocesoru 8080 obsahuje jednu vstupní a jednu výstupní instrukci. Obě jsou dvoubytové, jejich operand urtuje 8-bitovou adresu portu (brány) -vstupního nebo výstupního obvodu (zařízení). Adresy portů jsou určeny připojením v/v obvodů v počítači. V následující tabulce je uveden přehled adres v/v obvodů počítače C2717:

adresa:	brána:	obvod:	funkce:
244=F4H	PA	8255	řízení klávesnice
245=F5H	PB		čtení stavu klávesnice
246=F6H	PC		konfigurace počítače, akust. výstup
247=F7H	CWR		řídící registr (Control Word Reg.)
30 =1EH	DATA	8251	v/v seriového přenosu a magnetofon
31 =1FH	CWR, ST		řídící a stavový (Status) registr
72 =48H	PA	8255*)	čtení EPROM inteligentního kabelu
73 =49H	PB		adresa terminálu v intel. kabelu
76 =4CH	PA		univerzální paralelní v/v
77 =4DH	PB		paralelní tiskárna
78 =4EH	PC		bitové vstupy a výstupy
79 =4FH	CWR		řídící registr paralelního v/v
92 =5CH	CT0	8253	časovač pro magnetofon -kanál 0
93 =5DH	CT1		univerzální čítač/časovač -kanál 1
94 =5EH	CT2		univerzální čítač/časovač -kanál 2
95 =5FH	CWR		řídící registr čítačů a časovačů

*)není v C2717 standardně zapojen, je umístěn v krabičce tzv. in-
teligentního kabelu pro tiskárnu, V.24 nebo programátor EPROM.

V/v instrukce neobsahují data, předpokládá se, že čtená da-
ta ze vstupního obvodu se přenesou do střadate, kde se uchovají
pro další zpracování. Obdobně je nutné před použitím výstupní
instrukce zapsat výstupní data do střadate, odkud se v okamžiku
provedení OUT přenesou datovou sběrnici do adresovaného obvodu.

Obě instrukce trvají 10 taktů a nemění příznakové registry.
Jejich strojové kódy jsou: D3 xx = OUT xx, DB xx = IN xx.

1.příklad: změna bitu zvuku (akustického výstupu) monitoru C2717

80D8 DBF6 IN 0F6H ;čtení stavu systémové brány PC
80DA EE02 XRI 2 ;změna bitu PC1 akustic. výstupu
80DC D3F6 OUT 0F6H ;nastavení nové hodnoty PC1
80DE C9 RET

Podobným způsobem se bitem PC2 zapíná a vypíná pípnutí při stisku klávesy, jak je uvedeno v monitoru na adresách 8167-816D, nebo zrušení akustické signalizace (81EE-81F4) pomocí ANI 0FCH.

2.příklad: test klávesy STOP v monitoru C2717

```
----- 8C74 DBF5 IN 0F5H ;čtení stavu brány klávesnice
          8C76 E640 ANI 40H ;test bitu klávesy STOP
          8C78 C0 RNZ ;návrát při nestisknutém STOP
          8C79 3E03 MVI A,3 ;příznak stisku klávesy:Z=1,A=03
          8C7B C9 RET
```

3.příklad: zápis bytu pro magnetofon do 8251 v monitoru C2717

```
----- 8D7E D31E OUT 1EH ;zápis bytu ze střadače do 8251
          8D80 DB1F IN 1FH ;čtení stavu 8251 (USART STATUS)
          8D82 0F RRC ;posuv bitu A0=TxRDY do CY
          8D83 D2808D JNC 8D80H;čekání na TxRDY (uvolnění 8251)
          8D84 C9 RET ;TxRDY=1-8251 čeká byte
```

4.příklad: inicializace 8251 a 8253 v monitoru C2717

```
----- 8B41 210020 LXI H,20H ;konstanta kmitočtu čítače CT0

          8B4E 3E40 MVI A,40H ;příkaz pro vnitřní nulování
          8B50 D31F OUT 1FH ;vyslání do systémové 8251
          8B52 3EED MVI A,0EDH;režim:ASYNC+8b+2STOPbity+1*
          8B54 0E21 MVI C,21H ;povolení vysílání a žádost vys

          8B5C D31F OUT 1FH ;nastavení režimu 8251
          8B5E 79 MOV A,C ;povolení a žádost do střadače
          8B5F D31F OUT 1FH ;předáno 8251,zahajuje činnost
          8B61 3E36 MVI A,36H ;povel pro CT0, nastavit MOD 3
          8B63 D35F OUT 5FH ;zapsán do 8253
          8B65 7D MOV A,L ;nižší byte konstanty kmitočtu
          8B66 D35C OUT 5CH ;zapsán do CT0
          8B68 7C MOV A,H ;vyšší byte konstanty kmitočtu
          8B69 D35C OUT 5CH ;zapsán do CT0
          8B6B C9 RET
```

Instrukce zákazu přerušení DI (Disable Interrupts) s kódem F3H trvá 4 takty a jejím účelem je zablokovat příjem požadavků na přerušení práce procesoru signálem INT. Instrukce povolující přerušení EI (Enable Interrupts) s kódem FBH trvá také 4 takty a odblokuje vstup INT 8080 po následující instrukci, kterou obvykle bývá návrat z podprogramu. Činnost přerušovacího systému se zakazuje jen tehdy, když procesor již zpracovává nějaké přerušení, aby toto zpracování bez komplikací dokončil, nebo když posloupnost instrukcí nesmí být přerušena. V monitoru C2717 je přerušení zablokováno při rolování obrazovky (808E a následující).

Požadavek přerušení práce procesoru může přijít od libovolného obvodu v sestavě počítače nebo od vnějšího zařízení signálem INT. Procesor dokončí prováděnou instrukci a na datové sběrnici očekává instrukci od přerušujícího obvodu. Nejvhodnější je instrukce jednobyťová typu RST n (ReStart-znovuspuštění), ale může to být i instrukce podmíněného volání, skoku a podobné. Aby

se obsluha přerušení od různých obvodů lišila, byl soubor instrukcí 8080 vybaven 8 instrukcemi typu RST n, které pracují jako instrukce volání podprogramu na pevných adresách s hodnotami:
 0000 -RST0=C7H 0010H-RST2=D7H 0020H-RST4=E7H 0030H-RST6=F7H
 0008 -RST1=CFH 0018H-RST3=DFH 0028H-RST5=EFH 0038H-RST7=FFH

Tyto instrukce trvají 11 taktů a pracují tak, že nejprve uloží obsah čítače instrukcí PC do zásobníku a do PC nastaví výše uvedené adresy, např. 0038H pro RST7. Na této adrese musí začít obslužný podprogram pro ošetření tohoto přerušení, který musí končit některou návratovou instrukcí (RET aj.). Tak je zajištěno pokračování přerušeno programu. Pokud podprogram ošetření přerušení bude používat nějaké registry, musí samozřejmě uchovat jejich obsah v paměti a před návratem pak jejich obsah obnovit.

Následující úloha bude kombinací toho, co bylo výše uvedeno. Se zakázaným přerušením nastavíme časovač na 153,6 kHz a potom seriový port 8251A (USART) na asynchronní přenos 16* (tj. rychlostí: 153,6/16=9600 baud). Změníme původní obsah adresy pro RST 7 (0038H) na skok do podprogramu přenosu části videopaměti jinam a instrukcí RST 7 uměle vyvoláme 'přerušení'. Ověříme si, jak program pracuje různými způsoby jeho spuštění.

1. úloha: nastavte časovač a seriový port pro asynchronní přenos
 ----- s vyvoláním přenosu přes instrukci RST 7:

7000 F3	DI	zákaz přerušení při nastavování
7001 3E3E	MVI A,3EH	řídící slovo CT0-R/W LB+HB-mod3
7003 D35F	OUT 5FH	odesláno do 8253
7005 3E40	MVI A,64	příkaz pro vnitřní nulování 8251A
7007 D31F	OUT 1FH	odeslán do 8251A
7009 3E13	MVI A,13H	nižší byte (LB) konst. 153,6 kHz
700B D35C	OUT 5CH	odeslán do CT0 8253
700D 3E4E	MVI A,4EH	režim 8251A: async+8bitů+16*
700F D31F	OUT 1FH	odeslán do 8251A
7011 3E00	MVI A,0	vyšší byte (HB) konst. 153,6 kHz
7013 D35C	OUT 5CH	odeslán do CT0 8253
7015 3E37	MVI A,37H	start 8251: RTS+ER+RxEN+DTR+TxEN
7017 D31F	OUT 1FH	zapsán do 8251A
7019 DB1F	IN 1FHL	čtení stavu 8251A-nuluje chyby
701B DB1E	IN 1EH	čtení dat 8251A - shodí RxRDY
701D DB1E	IN 1EH	zopakováno (pro jistotu)
701F FB	EI	povoleno přerušení po instr. MVI
7020 3EC3	MVI A,0C3H	kód instrukce skoku JMP
7022 323800	STA 0038H	uložen na adresu pro instr. RST 7
7025 213070	LXI H,7030H	adresní část instrukce skoku JMP
7028 223900	SHLD 0039H	uložena za kód JMP jako operand
702B FF	RST 7	restart na adrese 0038H
702C C9	RET	

Zadejte program do paměti pomocí SUB 7000 (po FB) a SUB7020 (zbytek). Program nespouštějte!

2.úloha: zobrazení na stínítku změňte na inverzní

```

-----7030 2100C0 LXI H,0C000H;počáteční adresa VRAM
7033 1140FF LXI D,0FF40H;koncová adresa pro inverzi VRAM
-> 7036 011030 LXI B,3010H ;znaků na řádku+přírutek adres
7039 7C MOV A,H ;odpovídá vyšší byte adresy
703A 92 SUB D ;vyššímu bytu koncové adresy?
703B C24170 JNZ 7041H ;ne, skok na úpravu obsahu adres
703E 7D MOV A,L ;ano, odpovídá nižší byte adresy
703F 93 SUB E ;nižšímu bytu koncové adresy?
7040 C8 RZ ;ano,konec podprogramu.
-> 7041 7E MOV A,M ;ne, načten byte z VRAM
7042 2F CMA ;dvoujkový doplněk (inverze)
7043 E63F ANI 3FH ;zrušení atributů zobrazení
7045 77 MOV M,A ;zápis na původní místo ve VRAM
7046 23 INX H ;další adresa VRAM
7047 05 DCR B ;je poslední na mikrořádku ?
7048 C24170 JNZ 7041H ;ne, pokračuj
704B 09 DAD B ;ano,vypočti adr.dalšího mikroř.
704C C33670 JMP 7036H ;invertuj další mikrořádek VRAM

```

Tento podprogram zadejte do paměti (SUB 7030) a spusťte pomocí JUMP 7030. Tím jste ověřili jeho funkčnost. Nyní odstartujte program z 1.úlohy pomocí JUMP 7000. Inverzi zobrazení neprovedl tento program, ale podprogram odstartovaný přes RST7. Pokud tomu nevěříte, zkuste si JUMP 0038 a opět dojde k inverzi VRAM. A naposled si totéž vyzkoušejte pomocí JUMP 702B - zde odstartujete pouze dvoubytový program: FFC9 (RST7 + RET)

Pokud máte připojen počítač v síti,nebo do objímky magnetofonu zasunete 7-kolíkový konektor s odpory 47 ohmů mezi špičkami 5 a 2 (zem) a 7 a 2 (přizpůsobení linky seriového přenosu), můžete si vyzkoušet program testující 8251A a seriový vysílač a přijímač následující úlohou. Přenos spusťte přes 1.úlohu: JUMP 7000

3.úloha: autonomní test seriových přenosových obvodů

```

-----7030 2100C0 LXI H,0C000H;počáteční adresa VRAM
7033 1100D0 LXI D,0D000H;cílová adresa zóny VRAM
7036 01002C LXI B,2C00H ;délka přenášené zprávy
-> 7039 DB1F IN 1FH ;čtení stavu 8251A
703B E601 ANI 1 ;test přítomnosti signálu TxRDY
703D CA3970 JZ 7039H ;ne, testuj dále
7040 7E MOV A,M ;ano, další byte VRAM
7041 F680 ORI 80H ;změněny atributy zobrazení
7043 D31E OUT 1EH ;výstup bytu do 8251A
7045 00 NOP ;uklidnění sběrnice
-> 7046 DB1F IN 1FH ;čtení stavu 8251A
7048 E602 ANI 2 ;test RxRDY-přijata data?
704A CA4670 JZ 7046H ;ne, testuj
704D DB1E IN 1EH ;ano,data do střadače
704F 12 STAX D ;za odtud na cílovou adresu
7050 13 INX D ;zvýšení adres
7051 23 INX H
7052 0B DCX B ;zmenšení počítadla délky
7053 78 MOV A,B ;testuj ve střadači
7054 FEFF CPI 0FFH ;zda je počítadlo přenosu <0000
7056 C23970 JNZ 7039H ;ne, pokračuje přenos
7059 C9 RET ;ano, konec.

```

NOVE POJMY: Zásobník a jeho používání, uchování
----- obsahu registrů procesoru při přerušení,
předávání parametrů přes zásobník,
podprogram pro kopii obrazovky na tiskárně.

NOVE INSTRUKCE: PUSH rp -uložit obsah páru registrů do zásobníku
----- POP rp -naplnit pár registrů ze zásobníku
LXI SP, a -naplnit ukazatel zásobníku adresou
SPHL -naplnit ukazatel zásobníku z páru HL
XTHL -výměna obsahu zásobníku s HL: [SP] <=> HL

Zásobník je tvořen vymezeným úsekem paměti RWM, který slouží k přechodnému uchování dat nebo adres, především při zpracování přerušení, vyvolávání a návratů z podprogramů. Monitor počítače C2717 nastaví běžnou adresu zásobníku na adrese 8021 takto:
8021 310080 LXI SP, 8000H

Naplnit ukazatel by také bylo možno pomocí dvou instrukcí LXI HL, 8000H a navazující SPHL (kód F9H, 5 taktů, nemění příznaky).

SP (Stack Pointer) je 16-bitový registr procesoru 8080, ve kterém se uchovává platná adresa zásobníku. Vlastní zásobník je tvořen od adresy 8000H (tzv. dno zásobníku) směrem 'dole', proto jsou data naposledy zapsaná na nejnižší adrese (tzv. vrchol zásobníku). Zásobník pracuje systémem LIFO (Last In First Out) - poslední dovnitř a první ven. Příklad po zápisu dvou bytů:

původní ukazatel: SP = 8000 | _____ | <- dno zásobníku
7FFF | _____ | 1. byte

nový ukazatel: SP = 7FFE | _____ | 12. byte <- vrchol zásobníku

Snižování adresy v SP je automatické u instrukcí plnění zásobníku (PUSH nebo CALL a RST n), a zvyšování je opět automatické při odebírání ze zásobníku (POP a RET). Při práci se zásobníkem je nutno dbát, aby všechna zapsaná data byla opět vyzvednuta a nedošlo k přeplnění zásobníku a tím narušení programu, který se v paměti nachází pod zásobníkem.

Instrukce PUSH rp trvají 11 taktů a nemění příznaky F, instrukce POP rp trvají 10 taktů a také nemění F mimo POP PSW, která obnoví původní stavové slovo procesoru (Programm Status Word) a tak přepíše všechny příznaky. Z páru registrů se ten, který je uveden na prvním místě (A, B, D, H), zapisuje jako první, tj. na vyšší adresu, při čtení se naopak jako první naplňují z nižších adres 'druhé' registry (F, C, E, L).

Pořadí uchování registrů musí být opačné než jejich plnění:

1. příklad: uchování a obnovení obsahu registrů a stav SP:

----- C5 PUSH B , uchování: [B] -> 7FFF, [C] -> 7FFE, [SP] = 7FFE
D5 PUSH D [D] -> 7FFD, [E] -> 7FFC, 7FFC
E5 PUSH H [H] -> 7FFB, [L] -> 7FFA, 7FFA
F5 PUSH PSW [A] -> 7FF9, [F] -> 7FF8, 7FF8
.....; vlastní podprogram
F1 POP PSW , obnovení: [7FF8] -> F, [7FF9] -> A, 7FFA
E1 POP H [7FFA] -> L, [7FFB] -> H, 7FFC
D1 POP D [7FFC] -> E, [7FFD] -> D, 7FFE
C1 POP B [7FFE] -> C, [7FFF] -> B, 8000

Pokud uchováme příznaky F pomocí PUSH PSW a přečteme je pomocí POP B, dostanou se do registru C a zde je můžeme v případě potřeby analyzovat.

Zásobník je využíván pro předávání parametrů podprogramům, např. při načtení bloku dat z EPROM modulu inteligentního kabelu.

2.příklad: načtení bloku dat z EPROM modulu C2717:

-----6000	CD008C	CALL 8C00H	/volání podprogramu 'ROMIN'	
6003	0000	DW 0	/adresa začátku v ROM modulu	
6005	FF07	DW 07FFH	/počet čtených bytů - 1	
6007	0070	DW 7000H	/adresa cíle-záčátku v RAM	
8C00	E1	POP H	/převzetí adresy: 6003->HL	
8C01	5E	MOV E,M	/nižší byte adr. v ROM: 00->E	
8C02	23	INX H	/další adresa: HL=6004	
8C03	56	MOV D,M	/vyšší byte adr. v ROM: 00->D	
8C04	23	INX X	/adresa počtu bytů: HL=6005	
8C05	D5	PUSH D	/uchování adr. v ROM v zásobníku	
8C06	4E	MOV C,M	/nižší byte počtu: FF->C	
8C07	23	INX H	/adresa vyššího bytu: HL=6006	
8C08	46	MOV B,M	/vyšší byte počtu: 07->B	
8C09	23	INX H	/adresa nižší bytu cíle: HL=6007	
8C0A	5E	MOV E,M	/nižší byte adr. cíle: 00->E	
8C0B	23	INX H	/další adresa: HL=6008	
8C0C	56	MOV D,M	/vyšší byte adr. cíle: 70->D	
8C0D	23	INX X	/adresa HL=6009 návratu po CALL	
8C0E	E3	XTHL	/uchována v zásobníku, HL=0000	
8C0F	03	INX B	/počet čtených bytů přesně: 0800	
8C10	7C	MOV A,H	/vyšší byte adresy začát. v ROM	
8C11	FE40	CPI 40H	/je 1.ROM (00) nebo 2.ROM (40)	
8C13	3E89	MVI A,89H	/říd. slovo 8255 pro čtení ROM	
8C15	D22D8C	JNC 8C2D	/skok na čtení 2.modulu ROM	
8C18	D34F	OUT 4FH	/zápis říd.slova 8255 1.modulu	
-> 8C1A	7D	MOV A,L	/nižší byte adresy odkud:00->A	
8C1B	D34C	OUT 4CH	/vyslán do PA 8255 1.modulu	
8C1D	7C	MOV A,H	/vyšší byte adresy odkud:00->A	
8C1E	D34D	OUT 4DH	/vyslán do PB 8255 1.modulu	
8C20	DB48	IN 48H	/přečtený byte ROM do A	
8C22	CD7788	CALL 8877H	/volá podprogram uložení bytu	
8C25	C21A8C	JNZ 8C1AH	/pokračování do vyčerpání počtu	
8C28	3E9B	MVI A,9BH	/nová inicializace 8255 modulu	
8C2A	D34F	OUT 4FH	/zapsána do 8255 v kabelu	
8C2C	C9	RET		
Pprogr.->	8877	12	STAX D	/ulož do paměti
	8878	13	INX D	/další cílová adresa v RAM
	8879	23	INX H	/další zdrojová adresa pro ROM
	887A	0B	DCX B	/zmenšení obsahu čítače bytů
	887B	78	MOV A,B	/je vyšší byte v reg.B (=0?)
	887C	B1	ORA C	/roven nižšímu v reg.C (=0?)
	887D	C9	RET	/návrat, nastaven příznak Z!

V programu je použita instrukce XTHL (eXchange Top of stack with H and L-zaměň obsahy zásobníku a registrů HL), která trvá 18 taktů a nemá vliv na příznaky. Její smysl a význam je zřejmý z uvedeného příkladu.

Na závěr tohoto kursu strojového kódu si uvedeme příklad či úlohu, ukazující jednu možnost programu grafické kopie obrazovky na tiskárně typu CONSUL 201 (EPSON FX80) s interface Centronix. Pro jiné typy tiskáren jsou programy na kazetě SWK4 - "Strojový kód a aplikace".

1. úloha: program HARDCOPY obrazovky na tiskárně CONSUL 201

7400 0D	DB 'CR'	; návrat vozu (pole konstant programu)
7401 0A	DB 'LF'	; Line Feed- nový řádek
7402 1B	DB 'ESC'	; ESCape-návěští řídících znaků tiskár.
7403 33	DB 33H	; řádkování N/216 palce: 33/216=grafika
7404 18	DB 24	; konstanta řádkování
7405 1B4B	DW 'ESC' 'K'	; grafický mod 480 bodů/řádek tisku
7407 20	DB 32	; nižší byte délky: 288mod256=32=2bytek
7408 01	DB 1	; vyšší byte délky: INT(288/256)=1
-> 7409 110074	LXI D, 7400H	; adresa počátku pole konstant programu
740C 0E09	MVI C, 9	; počet bytů pole konstant
-> 740E 1A	LDAX D	; byte z pole konstant
740F CD1874	CALL 7418H	; vyslán do tiskárny
7412 13	INX D	; další adresa v poli konstant
7413 0D	DCR C	; byla poslední?
7414 C20E74	JNZ 740EH	; ne, pokračuje vysílání řídících bytů
7417 C9	RET	; ano, konec nastavení tiskárny
-> 7418 F5	PUSH PSW	; uchování grafického bytu pro tiskárnu
-> 7419 DB4E	IN 4EH	; čtení portu PC s bitem INTRb (--ACK)
741B E601	ANI 1	; test PC1
741D CA1974	JZ 7419H	; nepřišel, čekej dále na PC1
7420 F1	POP PSW	; přišel, návrat grafický byte do A
7421 D34D	OUT 4DH	; a vyšli jej do tiskárny přes port PB
7423 C9	RET	; konec podprogramu tisku graf. bytu
=> 7424 3E84	MVI A, 132	; řídící slovo 8255: PB-výstup/mod1
7426 D34F	OUT 4FH	; vysláno do inteligentního kabelu-CWR
7428 3E05	MVI A, 5	; nastavení PC2=1 (podmínka pro INTRb)
742A D34F	OUT 4FH	; předáno do CWR 8255 v kabelu
742C 2100C0	LXI H, 0C000H	; počáteční adresa videopaměti
-> 742F CD0974	CALL 7409H	; vyslání příkazů nastavení graf. režimu
7432 0630	MVI B, 48	; počet 'znaků'/bytů na (mikro)řádku
-> 7434 22A074	SHLD 74A0H	; uchování aktuální adresy videopaměti
7437 0E08	MVI C, 8	; počítadlo mikrořádků=počet jehel tisk.
7439 11A274	LXI D, 74A2H	; začátek zóny 8 bytů převáděného znaku
-> 743C 7E	MOV A, M	; byte z VRAM
743D 12	STAX D	; uchován pro převod
743E 13	INX D	; další adresa v zóně převodu
743F D5	PUSH D	; dočasně uchována v zásobníku
7440 114000	LXI D, 40H	; přírůstek adres mikrořádku ve VRAM
7443 19	DAD D	; připočten k aktuální adrese VRAM
7444 D1	POP D	; vrácena

7445	0D	DCR C	poslední (osmý) mikrořádek ?
7446	C23C74	JNZ 743CH	ne, pokračuje čtení bytů pro převod
7449	0E06	MVI C,6	ano, nastav počítadlo graf. bytů
744B	21AA74	LXI H,74AAH	začátek zóny tisku grafických bytů
->	744E	11A274	LXI D,74A2H; začátek zóny převáděného 'znaku'
7451	3600	MVI M,0	počáteční prázdný znak grafiky
->	7453	7E	MOV A,M přesunut do střadače
7454	07	RLC	posunut
7455	77	MOV M,A	a vrácen
7456	1A	LDAX D	převáděný byte z mikrořádku 'znaku'
7457	0F	RRC	posunut, bit pro tisk je v CY(1=tisk)
7458	12	STAX D	a vrácen zpět
7459	D25D74	JNC 745DH	CY=0-bit nemá být zobrazen
745C	34	INR M	CY=1-bit bude vytisknut jehličkou
745D	13	INX D	další adresa v zóně převodu
745E	3EAA	MVI A,0AAH	je poslední ? (tj. 74AAH)
7460	BB	CMP E	
7461	C25374	JNZ 7453H	ne, pokračuje převod
7464	23	INX H	ano, další adresa v zóně pro tisk
7465	0D	DCR C	všechny byty převedeny na tisk ?
7466	C24E74	JNZ 744EH	ne, pokračuje převod
7469	21AA74	LXI H,74AAH	ano, nastav začátek zóny graf. tisku
746C	0E06	MVI C,6	počítadlo bytů pro tisk
->	746E	7E	MOV A,M načten grafický byte pro tisk
746F	CD1874	CALL 7418H	a vyslán do tiskárny
7472	000000	DS 3	místo pro zopakování CD1874
7475	000000	DS 3	místo pro 3. zopakování CD1874
7478	23	INX H	adresa dalšího bytu pro tisk
7479	0D	DCR C	je poslední?
747A	C26E74	JNZ 746EH	ne
747D	2AA074	LHLD 74A0	ano, načti adresu aktuál. znaku VRAM
7480	23	INX H	adresa dalšího znaku
7481	05	DCR B	je to poslední znak řádku?
7482	C23474	JNZ 7434H	ne
7485	11D001	LXI D,1D0H	přirustek adres řádku VRAM =8 mikroř.
7488	19	DAD D	přičten k aktuální adrese VRAM
7489	7C	MOV A,H	vyšší byte této adresy
748A	D6FC	SUB 0FCH	odpovídá konci videostránky ?
748C	C8	RZ	ano, konec překladu a tisku
748D	C32F74	JMP 742FH	ne, pokračuj
7490		DS 16	neobsazeno
74A0	0000	DW 0	místo uchování aktuální adresy VRAM
74A2		DS 8	zóna 8 bytů převáděného znaku
74AA		DS 6	zóna 6 bytů pro grafický tisk

Pokud máte uvedenou tiskárnu, zadejte program do paměti (SUB 7400) po 16 bytech (SUB 7410 atd.) a nahrejte na kazetu:

MGSV 00,7400-74AF,C201 ASM (a EOL)

Potom už jenom zapnete tiskárnu a program odstartujete pomocí

JUMP 7424 (a EOL)

nebo z Basic-G pomocí příkazu: T=USR('7424)

Instrukce strojového kódu MHB 8080 -vzestupně řazené podle kódu:

=====

...instrukce nedefinována M=paměťové místo adresované [HL]
d=data 8bitů w=data 16bitů a=adresa 16bitů a8=adresa 8bitů

Kód Mnemo taktů

00 NOP	4T	10 ...	20 ...	30 ...
01 LXI B,w	10T	11 LXI D,w	10T	21 LXI H,w
02 STAX B	7T	12 STAX D	7T	22 SHLD a
03 INX B	5T	13 INX D	5T	23 INX H
04 INR B	5T	14 INR D	5T	24 INR H
05 DCR B	5T	15 DCR D	5T	25 DCR H
06 MVI B,d	7T	16 MVI D,d	7T	26 MVI H,d
07 RLC	4T	17 RAL	4T	27 DAA
				37 STC

08 ...	18 ...	28 ...	38 ...
09 DAD B	10T	19 DAD D	10T
0A LDAX B	7T	1A LDAX D	7T
0B DCX B	5T	1B DCX D	5T
0C INR C	5T	1C INR E	5T
0D DCR C	5T	1D DCR E	5T
0E MVI C,d	7T	1E MVI E,d	7T
0F RRC	4T	1F RAR	4T
		2F CMA	4T
		3F CMC	4T

40 MOV B,B	5T	50 MOV D,B	5T	60 MOV H,B	5T	70 MOV M,B	7T
41 MOV B,C	5T	51 MOV D,C	5T	61 MOV H,C	5T	71 MOV M,C	7T
42 MOV B,D	5T	52 MOV D,D	5T	62 MOV H,D	5T	72 MOV M,D	7T
43 MOV B,E	5T	53 MOV D,E	5T	63 MOV H,E	5T	73 MOV M,E	7T
44 MOV B,H	5T	54 MOV D,H	5T	64 MOV H,H	5T	74 MOV M,H	7T
45 MOV B,L	5T	55 MOV D,L	5T	65 MOV H,L	5T	75 MOV M,L	7T
46 MOV B,M	7T	56 MOV D,M	7T	66 MOV H,M	7T	76 HLT	
47 MOV B,A	5T	57 MOV D,A	5T	67 MOV H,A	5T	77 MOV M,A	7T

48 MOV C,B	5T	58 MOV E,B	5T	68 MOV L,B	5T	78 MOV A,B	5T
49 MOV C,C	5T	59 MOV E,C	5T	69 MOV L,C	5T	79 MOV A,C	5T
4A MOV C,D	5T	5A MOV E,D	5T	6A MOV L,D	5T	7A MOV A,D	5T
4B MOV C,E	5T	5B MOV E,E	5T	6B MOV L,E	5T	7B MOV A,E	5T
4C MOV C,H	5T	5C MOV E,H	5T	6C MOV L,H	5T	7C MOV A,H	5T
4D MOV C,L	5T	5D MOV E,L	5T	6D MOV L,L	5T	7D MOV A,L	5T
4E MOV C,M	7T	5E MOV E,M	7T	6E MOV L,M	7T	7E MOV A,M	7T
4F MOV C,A	5T	5F MOV E,A	5T	6F MOV L,A	5T	7F MOV A,A	5T

80 ADD B	4T	90 SUB B	4T	A0 ANA B	4T	B0 ORA B	4T
81 ADD C	4T	91 SUB C	4T	A1 ANA C	4T	B1 ORA C	4T
82 ADD D	4T	92 SUB D	4T	A2 ANA D	4T	B2 ORA D	4T
83 ADD E	4T	93 SUB E	4T	A3 ANA E	4T	B3 ORA E	4T
84 ADD H	4T	94 SUB H	4T	A4 ANA H	4T	B4 ORA H	4T
85 ADD L	4T	95 SUB L	4T	A5 ANA L	4T	B5 ORA L	4T
86 ADD M	7T	96 SUB M	7T	A6 ANA M	7T	B6 ORA M	7T
87 ADD A	4T	97 SUB A	4T	A7 ANA A	4T	B7 ORA A	4T

88 ADC B	4T	98 SBB B	4T	A8 XRA B	4T	B8 CMP B	4T
89 ADC C	4T	99 SBB C	4T	A9 XRA C	4T	B9 CMP C	4T
8A ADC D	4T	9A SBB D	4T	AA XRA D	4T	BA CMP D	4T
8B ADC E	4T	9B SBB E	4T	AB XRA E	4T	BB CMP E	4T
8C ADC H	4T	9C SBB H	4T	AC XRA H	4T	BC CMP H	4T
8D ADC L	4T	9D SBB L	4T	AD XRA L	4T	BD CMP L	4T
8E ADC M	7T	9E SBB M	7T	AE XRA M	7T	BE CMP M	7T
8F ADC A	4T	9F SBB A	4T	AF XRA A	4T	BF CMP A	4T

C0 RNZ	5/11T	D0 RNC	5/11T	E0 RPO	5/11T	F0 RP	5/11T
C1 POP B	10T	D1 POP D	10T	E1 POP H	10T	F1 POP PSW	10T
C2 JNZ a	10T	D2 JNC a	10T	E2 JPO a	10T	F2 JP a	10T
C3 JMP a	10T	D3 OUT a8	10T	E3 XTHL	10T	F3 DI	4T
C4 CNZ a11/17T		D4 CNC a11/17T		E4 CPD a11/17T		F4 CP a	11/17T
C5 PUSH B	11T	D5 PUSH D	11T	E5 PUSH H	11T	F5 PUSH PSW	11T
C6 ADI d	7T	D6 SUI d	7T	E6 ANI d	7T	F6 ORI d	7T
C7 RST 0	11T	D7 RST 2	11T	E7 RST 4	11T	F7 RST 6	11T

C8 RZ	5/11T	D8 RC	5/11T	E8 RPE	5/11T	F8 RM	5/11T
C9 RET	10T	D9 ...		E9 PCHL	5T	F9 SPHL	5T
CA JZ a	10T	DA JC a	10T	EA JPE a	10T	FA JM a	10T
CB ...		DB IN a8	10T	EB XCHG	4T	FB EI	4T
CC CZ a 11/17T		DC CC a 11/17T		EC CPE a11/17T		FC CM a	11/17T
CD CALL a	17T	DD ...		ED ...		FD ...	
CE ACI d	7T	DE SBI d	7T	EE XRI d	7T	FE CPI d	7T
CF RST 1	11T	DF RST 3	11T	EF RST 5	11T	FF RST 7	11T

Přehled instrukcí ovlivňujících bity příznakového registru F:

Aritmetické:		ACI d, ADC r, ADD r, ADI d	S Z AC P CY
		DAA	S Z AC P CY
		DAD	CY
		DCR	S Z AC P
		INR	S Z AC P
		SBB r, SBI d, SUB r, SUI d	S Z AC P CY
Logické:		ANA r, ANI r	S Z AC P 0
		CMC	CY
		CMP r, CPI d	S Z AC P CY
		ORA r, ORI d	S Z 0 P 0
		RAL, RAR, RLC, RRC	CY
		STC	1
		XRA r, XRI d	S Z 0 P 0
Ostatní:		POP PSW	S Z AC P CY

Přehled skupin instrukcí podle typů:

Skoky:	Volání:	Návraty:	Restarty:	Zásobník:
JMP a -C3....	CALL a-CD....	RET -C9	RST 0 -C7	PUSH B -C5
JNZ a -C2....	CNZ a -C4....	RNZ -C0	RST 1 -CF	PUSH D -D5
JZ a -CA....	CZ a -CC....	RZ -C8	RST 2 -D7	PUSH H -E5
JNC a -D2....	CNC a -D4....	RNC -D0	RST 3 -DF	PUSH PSW-F5
JC a -DA....	CC a -DC....	RC -D8	RST 4 -E7	
JPO a -E2....	CPO a -E4....	RPO -E0	RST 5 -EF	POP PSW -F1
JPE a -EA....	CPE a -EC....	RPE -E8	RST 6 -F7	POP H -E1
JP a -F2....	CP a -F4....	RP -F0	RST 7 -FF	POP D -D1
JM a -FA....	CM a -FC....	RM -F8		POP B -C1
PCHL -E9				

Zvětšení:	Zmenšení:	Naplnění:	Přímé oper.	Různé:
INR B -04	DCR B -05	MVI B,d-06..	ADI d -C6..	DAD B -09
INR C -0C	DCR C -0D	MVI C,d-0E..	ACI d -CE..	DAD D -19
INR D -14	DCR D -15	MVI D,d-16..	SUI d -D6..	DAD H -29
INR E -1C	DCR E -1D	MVI E,d-1E..	SBI d -DE..	DAD SP -39
INR H -24	DCR H -25	MVI H,d-26..	ANI d -E6..	RLC -07
INR L -2C	DCR L -2D	MVI L,d-2E..	XRI d -EE..	RRC -0F
INR M -34	DCR M -35	MVI M,d-36..	ORI d -F6..	RAL -17
INR A -3C	DCR A -3D	MVI A,d-3E..	CPI d -FE..	RAR -1F
INX B -03	DCX B -0B	LDAX B -0A	LXI B,w -01....	DI -F3
INX D -13	DCX D -1B	LDAX D -1A	LXI D,w -11....	EI -FB
INX H -23	DCX H -2B	LHLD a -2A....	LXI H,w -21....	OUT a8-D3
INX SP-33	DCX SP-3B	LDA a -3A....	LXI SP,w-31....	IN a8 -DB
		STAX B -02		DAA -27
XTHL -E3		STAX D -12		CMA -2F
XCHG -EB		SHLD a -22....		STC -37
SPHL -F9		STA a -32....		CMC -3F

Adresování bytů ve videopaměti:

=====

0000000000000000000011111111111111112222222222222222

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF mikrotádek

C000		1.
C040		2.
C080		3.
C0C0		4.
C100		5.
C140		6.
C180		7.
C1C0		8.
C200		9.

0000000000000000000011111111111111112222222222222222

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF Řádek:

C000		0.
C240		1.
C480		2.
C6C0		3.
C900		4.
CB40		5.
CD80		6.
CFC0		7.
D200		8.
D440		9.
D680		10.
D8C0		11.
DB00		12.
DD40		13.
DF80		14.
E1C0		15.
E400		16.
E640		17.
E880		18.
EAC0		19.
ED00		20.
EF40		21.
F180		22.
F3C0		23.
F600		24.
F840		25.
FAB0		26.
FCC0		
FF00		
FFC0		

dialogový

0000000000000000000011111111111111112222222222222222

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF

LITERATURA:

=====

1. Petr Partyk, Ivo Machačka:
Základní instrukce mikroprocesoru 8080, Tesla ELTOS,
edice: Mikroelektronické systémy TESLA, Praha 1987
2. Jiří Zdeněk:
Technika mikropočítačů,
knihnice CSVTS - mikroprocesorová technika, sv. 11, Praha 1983
3. Ladislav Smejkal:
Kurs programování systémů s mikroprocesorem 8080
příloha časopisu Automatizace v letech 1980-1983
4. 8080 Assembly language, programming manual,
INTEL Corp. Santa Clara, USA, 1976
5. Igor Vavro, Josef Hrdlička, Zdeněk Opršal:
CONSUL 2717: Práce s počítačem - základní monitor (8000-8FFF)
INCOTEX Brno, 1990
6. Igor Vavro, Josef Hrdlička, Zdeněk Opršal:
CONSUL 2717: Práce s počítačem - rozšířený monitor (9000-9BFF)
INCOTEX Brno, 1990
7. Pavel Hlaváček:
Instrukce strojového kódu a jejich zkratky,
pomůcka uživatele C2717, INCOTEX Brno, 1989
8. Pavel Hlaváček:
Kazeta lekcí strojového kódu a aplikací pro G2717 (SWK 4)
INCOTEX Brno, 1989
9. Pavel Hlaváček:
Práce s obrazovkou PMD-85 (včetně HARDCOPY na D100)
Elektronika č. 7, 1988

Obsah:

strana:

=====

Lekce: Téma: Nové pojmy: Nové instrukce

Úvod	2
1. Architektura mikroprocesoru MHB 8080A	3
Bit, byte, registry, sběrnice, data 8bit, adresa 16bit, paměti ROM a RWM, čítač programu PC, dekodér instrukcí 00 = NOP	
2. Instrukce přesunu mezi registry a paměti	7
Přesuny dat 8bit a 16bit v registrech a paměti, výměna obsahu registrů MOV, MVI, LXI, LDA, STA, LDAX, STAX, LHLD, SHLD, XCHG, HLT	
3. Jednoduché aritmetické instrukce	11
Inkrementace a dekrementace obsahu registrů, vliv na příznakové bity, 16bitové sečítání, jednoduché násobení INR, INX, DCR, DCX, DAD, DAA	
4. Instrukce pro větvení programu	15
Nepodmíněné a podmíněné skoky, cykly, čekatí smyčky JMP, JC, JNC, JZ, JNZ, JPE, JPO, JM, JP, PCHL	
5. Instrukce pro práci s podprogramy	19
Volání podprogramu podmíněné a nepodmíněné, návraty z podprogramů, uchování návratové adresy v zásobníku CALL, CC, CNC, CZ, CNZ, ..., RET, RC, RNC, RZ, RNZ, ...	
6. Instrukce pro sečítání	23
Sečítání 8bitových dat bez přenosu a s přenosem, programové násobení, převod dvojkové desítkový, simulace programu ADD, ADI, ADC, ACI	
7. Instrukce pro odečítání	27
Odečítání 8bitových dat bez výpůjčky a s výpůjčkou, programové dělení, přenos dat videopaměti na jiné místo SUB, SUI, SBB, SBI	
8. Logické instrukce a posuvy	31
Logický součin a součet, exklusivní součet, logické srovnání, nastavení a doplněk, posuvy vlevo a vpravo ANA, ANI, ORA, ORI, XRA, XRI, CMP, CPI, CMA, CMC, RRC, RLC, RAR, RAL	
9. Instrukce vstupu/výstupu a přerušení	35
Vstupní a výstupní operace, adresování přídavných zařízení, režim přerušení, jeho povolení a zákaz, instrukce restartu. IN, OUT, EI, DI, RST n.	
10. Instrukce pracující se zásobníkem	39
Zásobník a jeho používání, uchování obsahu registrů, předávání parametrů přes zásobník, hardcopy obrazovky PUSH, POP, LXI SP, SPHL, XTHL	
Přílohy: Přehled instrukcí podle kódů, typů a příznaků	43
Adresování bytů ve videopaměti	46
Literatura	47

Název: CONSUL 2717: Práce s počítačem - Kurs instrukcí 8080
Sestavil: Ing. Pavel Hlaváček
Vydal: Incotex, s.p., Hybešova 42, 65664 Brno
Cena smluvní: 10,- Kčs podle vyhlášky FCU č. 35/1990