RNDr. Miloslav Feil, CSc.

STROJOVÝ KÓD IQ 151

Komenium, n. p., Praha

Úvod

Tato příručka je určena všem, kteří již zvládli základy programování školního mikropočítače IQ 151 v jazyce BASIC a mají základní znalosti z práce s uvedeným mikropočítačem v režimu MONITOR v rozsahu těchto publikací:

Jedlička Z. - Feil M.: " Basic pro začátečníky "

Feil M.: " Monitor IQ 151 "

Obě jmenované publikace vydalo Komenium a může je doplňně studovat i žák střední školy samostatně bez pomoci učitele. Rovněž i tato příručka není určena pouze vyučujícím, ale je psána tak, aby ji mohl samostatně studovat žák střední školy. Příručka je určena všem zájemcům o základy programování mikropočítače IQ 151 ve strojovém kódu. Jednoduchou a přístupnou formou seznamuje čtenáře s ukládáním programů ve strojovém kódu do paměti mikropočítače, způsobem jejich vyvolávání v režimu BASIC nebo MONITOR a ve své nejrozsáhlejší části ukazuje význam a funkci všech instrukcí, které se při strojovém programování používají. Není zde uveden pouze výklad a popis činnosti instrukcí, text je doplněn řadou velmi jednoduchých aplikací, většinou krátkými programy ve strojovém

...kódu, které ukazují princip funkce určitých instrukcí, způsoby vložení programu do počítače apod. Příklady jsou záměrně voleny velmi jednoduché, aby se v nich čtenář snadno a rychle orientoval.

Příručka je doplněna nejdůležitějšími tabulkami a přehledy.

Protože samotná znalost jednotlivých instrukcí a jejich funkce nestačí k tomu, aby se čtenář naučil samostatně programovat mikropočítač IQ 151 ve strojovém kódu, doporučuji, aby se snažil průběžně při studiu příručky tvořit samostatné jednoduché příklady programů ve strojovém kódu, které jsou analogické k příkladům v textu a tím postupně získával určitou rutinu při strojovém programování.

Autor

1.2 Programování příkazů ve strojovém kódu

Programování počítače ve strojovém kódu spočívá v tom, že se na vhodné místo v paměti RAM počítače vloží posloupnost kódů instrukcí. Přehled instrukcí pro programování ve strojovém kódu a jejich příslušných hexadecimálních kódů je v posledních dvou tabulkách přílohy této příručky. Zatím pochopitelně takové instrukce čtenáři nic nerikají, teprve během studia příručky se čtenář dozví jejich význam a funkci. Proti programování v jazyce BASIC nebo jiném vyšším programovacím jazyce je programování ve strojovém kódu podstatně složitější. Má však tu výhodu, že chod programu ve strojovém kódu je podstatně rychlejší, protože počítač nemusí slova vyššího programovacího jazyka interpretovat pomocí překladače.

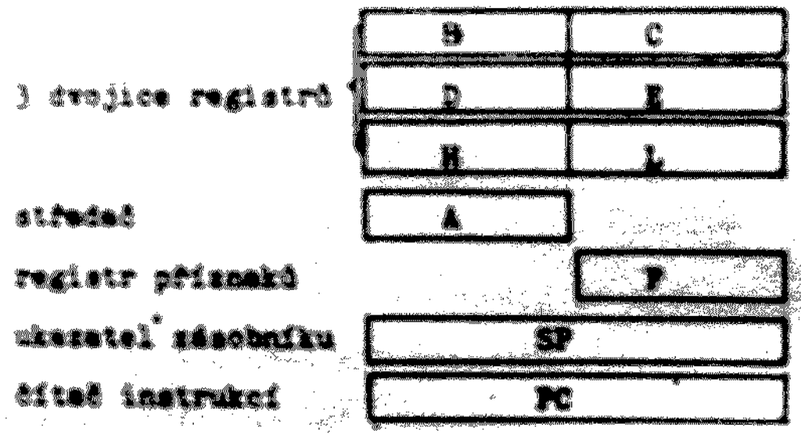
Program ve strojovém kódu, vložený do paměti počítače, není možno spustit příkazem RUN nebo analogickým příkazem jiného vyššího programovacího jazyka. Start programu ve strojovém kódu /dále strojového programu/ provádíme buď příkazy C nebo G v režimu MONITOR, nebo speciálním příkazem CALL v režimu BASIC. Uvedené případy budou ještě podrobně v dalším textu vysvětleny. Mnohé uživatelské programy obsahují jednak části, které jsou programovány v jazyku BASIC nebo jiném vyšším programovacím jazyku a jednak části, které jsou ve strojovém kódu.

V úvodních kapitolách se musíme seznámit, jak pracuje strojový program a čísly, do jakých míst je ukládá, dále musíme znát oblast v paměti počítače, do níž lze strojové programy ukládat. Dále musíme vědět, jak zajistit přechod od programu v jazyce BASIC ke strojovému a naopak.

1.1 Registry počítače

Pod pojmem registr je myšleno určité paměťové místo mikroprocesoru, které je k dispozici pro uchovávání číselných údajů. Obsah registrů můžeme měnit pomocí instrukcí strojového kódu. Tzv. jednoduché registry mají 8 bitů. Je možno pracovat i s dvojicemi jednoduchých registrů /v příručce nazývané: dvojregistry/, které potom zaujímají 16 bitů.

Schema registrů mikroprocesoru je následující:



V dvodních partiích příručky budeme potřebovat zvláště registry A, B, C, D, E, H, L a registr příznaků F. S prvními sedmi registry můžeme pracovat samostatně - tj. dosazovat a měnit vloženou číselnou hodnotu v každém jmenovaném registru zvlášť a nezávisle na hodnotách v ostatních registrech - ale pomocí určitých instrukcí lze s registry B, C, D, E, H, L pracovat i jako s dvojregistry - tedy jedinou instrukcí lze změnit obsah dvojregistru BC nebo DE nebo HL. Má to tu výhodu že v dvojregistru lze zobrazit větší obor čísel, než v jednoduchém registru. Je vám již známo, že v jednoduchém registru,

analogicky jako na paměťovém místě - může být pouze celé číslo z intervalu $\langle 0; 255 \rangle$ decimálně, což je $\langle 0; FF \rangle$ hexadecimálně. Ve dvojregistru může být celé číslo z intervalu $\langle 0; 65535 \rangle$ decimálně, což je $\langle 0; FFFF \rangle$ hexadecimálně.

Registr příznaků F není možno obsadit přímo nějakým vhodným číslem, jeho funkce je naprosto rozdílná. Uchovává totiž určité informace o výsledku různých operací, které probíhají v registru A - tzv. střadači, nebo v některých jiných registrech. Registr F se používá pro testování výsledku ve střadači nebo jiném registru. Obsahu registru A a F se dohromady říká stavové slovo a označuje se často PSW.

Dvojregistry SP a PC se nedají rozdělit na dva jednoduché registry. V počítači IQ 151 se po řadě znalí zkrácení pouze písmeny S a P. S jejich funkcí se podrobněji seznámíme až ke konci příručky, uvedeme zatím pouze, že tyto registry řídí chod strojového programu, zajišťují správné návraty z podprogramů, mohou program větvit apod.

1.2 Aritmetika čísel v registrech

Strojové programy mohou provádět s číslem, které je ve střadači nebo jiném jednoduchém registru, pouze následující aritmetické operace - mohou k číslu v registru nějaké jiné číslo přičíst, nebo odečíst. Neexistují tedy instrukce strojového kódu pro násobení, dělení apod.

Navíc ve většině skutečností, že při sečítání dvou čísel v registru bychom v některých případech mohli dostat výsledek, který je větší než dekadické číslo 255 a takové číslo se již do registru "nevejde".

Například tedy je-li v určitém jednoduchém registru číslo 1111 1111 /dvojkově/, pak po přičtení jedničky k uvedenému obsahu registru dostaneme výsledek ØØØØ ØØØØ - v registru se neprojeví skutečnost, že na devátém místě zprava by měla být jednotka, protože každý jednoduchý registr - tedy paměťové místo počítače - má pouze 8 bitů. Tato jednotka se však objeví v určitém bitu registru příznaků F, lze s ní v následujících operacích pracovat.

Snadno si nyní ukážete sami na jednoduchých příkladech, jak probíhá sečítání čísel ve dvojkovém kódu v registru počítače. Pro rychlé nalezení dvojkového kódu dekadických čísel a naopak vám poslouží tabulky uvedené v příloze.

1.2.1 Pojem přenosu

Sečítáte-li v desítkové soustavě 5Ø+5Ø , dostanete jako výsledek číslo 1ØØ. Všimněte si, že první řád výsledku je určen součtem dvou pětek, které tvoří jednotku ve vyšším řádu výsledku, tedy ve druhém. Tomuto procesu říkáme přenos z prvního řádu do druhého. Vrátíme-li se k jednoduchému příkladu z minulého článku 1.2, kde se přičítala jednotka ke dvojkovému číslu 1111 1111, vidíme, že při sečtení došlo k přenosu ve všech řádech počínaje nultým a konče sedmým. Přenos ze sedmého řádu nebyl zachycen v registru, ale ve speciálním registru příznaků F.

1.2.2 Odečítání čísel v registru

Mezi instrukcemi strojového kódu existují i takové, které umožňují odečíst nějaké číslo od čísla v daném registru. Počítač však realizuje odečtení čísla v registru tak, že k menšenci přičítá tzv. dvojkový doplněk menšitele. Přičítá vlastně opačné číslo. Ve dvojkovém kódu však nelze vyjádřit záporné celé číslo, proto počítač využívá pro ně dvojkového doplňkového kódu, jehož tabulka je rovněž v příloze.

Příklad:

Máme spočítat, kolik je 3 - 2 . Počítač vyjádří nejdříve dekadické číslo 3 ve dvojkovém kódu, což je ØØØØ ØØ11. Pak vyjádří menšitele, což je dekadické číslo 2,ve dvojkovém kódu a dostane tím ØØØØ ØØ1Ø. Nyní vytvoří tzv. jedničkový doplněk menšitele, který vznikne tak, že se zamění v menšiteli všechny nuly za jedničky a naopak. Tím dostane číslo 1111 11Ø1. K tomuto číslu přičte na posledním řádu jednotku a dostane tedy 1111 111Ø. Dostane tak vyjádření dekadického čísla -2 ve dvojkovém doplňkovém kódu - srovnejte s tabulkou převodů mezi dekadickými čísly a jejich dvojkovým doplňkovým kódem. Nyní provede součet

```
   ØØØØ ØØ11
+  1111 111Ø
  1 ØØØØ ØØØ1
```

V příslušném registru, kde výpočet probíhal, však bude pouze ØØØØ ØØØ1, přenos ze 7. řádu je registrován v příznakovém registru F. Převedeme-li výsledek do dekadické soustavy, vidíme, že je 1, což je ve shodě s výpočtem prováděným v decimální soustavě, který si jistě provádíte souběžně zpaměti.

Provedete-li obdobným způsobem postup pro výpočet rozdílu dekadických čísel 2 - 3, dostanete jako výsledek v registru číslo 1111 1111. Převedete-li ho z dvoj-

kového doplňkového kódu do decimálních čísel, zjistíte, že je to číslo -1.

Příklad:

Ukažte, jak probíhá výpočet - 2 - 3.

Řešení:

Dekadické číslo -2 má v dvojkovém doplňkovém kódu tvar 1111 111Ø, dekadické číslo -3 tvar 1111 11Ø1. Po jejich sečtení dostaneme 1 1111 1Ø11, přenos ze 7. řádu se v registru neuplatní. Výsledek v registru je tedy 1111 1Ø11, po převedení z dvojkového doplňkového kódu je to číslo -5 /dekadicky/.

1.3 Zobrazení registrů v paměti počítače, jejich obsazení v režimu MONITOR

Obsahy všech registrů uvedených v 1.1 lze například měnit v režimu MONITOR. Zapneme počítač a pomocí tlačítka BR přejdeme do režimu MONITOR. Nyní zadáme příkaz X a odešleme ho tlačítkem CR. Tento příkaz je jedním ze tří příkazů režimu MONITOR, o nichž jsme se dříve záměrně nezmiňovali. Po odeslání příkazu se na obrazovce objeví výpis registrů a jejich obsahů v tomto tvaru:

A F B C D E H L S P

8A 86 Ø1 EC FF ØØ ØØ FF 7F62 1ØØØ .

To znamená, že v registru A je číslo, jehož hexadecimální vyjádření je 8A, v registru příznaků F je číslo, jehož hexadecimální vyjádření je 86 atd.

Nyní se podíváme, na jaká místa v paměti počítače se obsahy registrů zobrazují. V režimu MONITOR zadáme

příkaz D a po jeho odeslání tlačítkem CR zjistíme snadno z výpisu obsazení jednotlivých paměťových míst, že registry se zobrazují na pam. místa s adresami od 22 do 2D /hexadecimálně/, tj. od 34 do 45 /decimálně/ Platí následující tabulka:

registr	adresa hexadecimální	adresa decimální
A	23	35
F	22	34
B	25	37
C	24	36
D	27	39
E	26	38
H	29	41
L	28	4Ø
S	2B 2A	43 42
P	2D 2C	45 44

Pomocí příkazu X v režimu MONITOR lze rovněž měnit obsah libovolného registru. Chceme-li tedy změnit například obsah registru E, dáme v režimu MONITOR příkaz XE a ihned se objeví v řádce původní obsah tohoto registru a za ním vodorovná čárka. Za čárku můžeme napsat libovolné dvouciferné hexadecimální číslo, kupříkladu DE. Pak stiskneme CR a pomocí výpisu obsahu registrů v režimu MONITOR pomocí příkazu X zjistíme, že se obsah registru E změnil požadovaným způsobem.

Pokud chceme měnit i obsahy dalších registrů v pořadí

za registrem E, nestiskneme po první změně obsahu tlačítko CR, ale jen SP a na obrazovce se nabízí změna obsahu registru H, po dalším stisknutí SP změna obsahu registru L atd.

Upozornění:

Vrátíme-li se z režimu MONITOR do režimu BASIC pomocí příkazu R, nastaví se původní obsahy všech registrů kromě registru P. Po stisku RES se nastaví původní hodnoty ve všech registrech.

1.4. Význam registru příznaků F

Pomocí registru F počítač vyhodnocuje, jaké číselné procesy se odehrávají ve střadači /tj. registru A/, nebo v některých jiných registrech. Registr F má rovněž 8 bitů očíslovaných celými dekadickými čísly od Ø do 7. Číslování probíhá vždy od pravého konce, souhlasí tak s řádem příslušné cifry, která je na daném bitu. Procesy ve střadači nebo jiných registrech ovlivňují pouze bity Ø, 2, 4, 6 a 7 registru příznaků.

Význam jednotlivých bitů registru F

7. bit - příznak znaménka - značí se S

Je-li ve střadači výsledek nějaké aritmetické operace, pak se cifra ze sedmého /nejvyššího/ řádu střadače překopíruje do sedmého bitu registru příznaků. Je-li tedy výsledek ve střadači Ø11Ø ØØ1Ø, dosadí se do 7. bitu registru příznaků Ø, je-li výsledek ve střadači 1ØØØ Ø1Ø1, dosadí se do tohoto bitu registru příznaků cifra 1. Říkáme zkráceně, že S = 1. Podle tabulek dvojkového doplňkového kódu snadno zjistíte, že S = Ø

v případě, že výsledkem operace ve střadači je číslo kladné nebo nula, S = 1 v případě, že výsledkem ve střadači je číslo záporné.

6. bit - příznak nuly - značí se Z

Vznikne-li ve střadači jako výsledek nějaké aritmetické operace nula - tedy ØØØØ ØØØØ - dosadí se automaticky do tohoto bitu registru příznaků cifra 1. Je-li výsledek ve střadači nenulový, dosadí se automaticky do tohoto bitu cifra Ø. Říkáme zkráceně, že Z = Ø.

5. bit - stabilně obsazen cifrou Ø

4. bit - příznak pomocného přenosu - značí se AC

Jestliže při aritmetické operaci ve střadači dojde k přenosu ze třetího řádu do čvrtého řádu, dosadí se automaticky do tohoto bitu registru příznaků cifra 1. Pokud k přenosu ze třetího do čtvrtého řádu nedojde, dosadí se do tohoto bitu cifra Ø. V takovém případě říkáme zkráceně, že AC = Ø.

Provede-li se ve střadači součet ØØØØ 1ØØØ + ØØØØ 1ØØØ, je výsledek ØØØ1 ØØØØ a AC = 1.

3. bit - stabilně obsazen cifrou Ø

2. bit - příznak parity - značí se P

Má-li výsledek operace ve střadači ve svém dvojkovém vyjádření sudý počet jedniček, dosadí se do tohoto bitu registru příznaků cifra 1, je-li počet jedniček výsledku lichý, dosadí se do tohoto bitu cifra Ø. Zkráceně píšeme v takovém případě opět P = Ø. Vznikne-li například jako výsledek aritmetické operace ve střadači ØØØØ ØØØØ, je počet jedniček ve výsledku

nulový, tedy sudý a je tedy P = 1.

1. bit - stabilně obsazen cifrou 1

Ø. bit - příznak přenosu - značí se CY

Tento bit ovlivňují různé numerické operace ve střadači. Bit se rozdílně obsazuje při sečítacích operacích a při odečítacích operacích.

Při sečítání ve střadači - vznikl-li přenos ze 7. řádu střadače A, dosadí se do tohoto bitu registru příznaků cifra 1. Pokud přenos ze 7. řádu nevznikl, dosadí se tam cifra Ø.

Při odečítání ve střadači - vznikl-li přenos ze 7. řádu střadače A, dosadí se do tohoto bitu registru příznaků cifra Ø. Pokud přenos ze 7. řádu nevznikl, dosadí se tam cifra 1.

1.5 Vložení programu ve strojovém kódu do paměti počítače

Program ve strojovém kódu není možno umístit do libovolné oblasti paměti RAM počítače, protože většinou spolu se strojovým programem jsou v paměti počítače ještě jiné programy třeba v jazyce BASIC apod., na určitá místa v paměti si počítač zaznamenává sám zavedené proměnné. Při neopatrném umístění strojového programu do paměti počítače by mohlo dojít k jeho poškození nebo přepsání z důvodů výše uvedených efektů.

Oblast paměti, do níž lze bez nebezpečí umístit strojový program, je tzv. oblast USR. Oblast USR vhodného rozsahu si ovšem musíme v paměti počítače vybudovat sami. K tomu se využívá příkazu CLEAR s parametry, který lze ovšem použít jen v režimu BASIC.

Víme, že samotný příkaz CLEAR v režimu BASIC maže z paměti číselné i řetězcové proměnné. Pro úplnost si zapamatujme, že příkaz

CLEAR, parametr

definuje rozsah části paměti, do níž je možno ukládat řetězcové proměnné. Po zapnutí počítače má zmíněná oblast 48 paměťových míst, po zadání příkazu

CLEAR 8Ø

a jeho odeslání tlačítkem CR v režimu BASIC se počet míst uvedené oblasti změní na 8Ø. Je samozřejmé, že parametr za příkazem CLEAR je decimální, neboť je to příkaz používaný v režimu BASIC.

Nyní si řekneme o třetí a pro nás nejdůležitější funkci příkazu CLEAR. Jestliže kromě prvního parametru přidáme ještě druhý, například

CLEAR 8Ø, 2ØØ

a odešleme celý příkaz tlačítkem CR v režimu BASIC, pak si počítač kromě určitého počtu míst v paměti pro řetězcové proměnné vyhradí celkem 2ØØ paměťových míst pro programy ve strojovém kódu. Říkáme, že jsme připravili oblast USR paměti s rozsahem 2ØØ paměťových míst.

Nestačí nám ale znát, jak velká je oblast paměti, do níž můžeme programy ve strojovém kódu ukládat, nutno vědět přesně, od které adresy do které se oblast USR v paměti rozkládá. Zapamatujeme si, že počáteční adresa oblasti USR je uložena na adresách A4 a A5 hexadecimálně, tj. 164 a 165 decimálně. Nižší dvě cifry jsou vždy na nižší adrese A4 této dvojice adres.

Koncová adresa oblasti USR je uložena na adresách

166 a 167 decimálně, což je

A6 a A7 hexadecimálně.

Po zapnutí počítače se můžeme v režimu MONITOR podívat, že počáteční a koncová adresa oblasti USR je 7FAØ/hexadecimálně/, což je 32 672 decimálně. V tomto stavu je rozsah oblasti USR Ø paměťových míst. Pokud po zapnutí počítače odešleme tlačítkem CR příkaz

CLEAR 5Ø, 16Ø ,

zjistíte snadno pomocí příkazu D v režimu MONITOR, že počáteční adresa oblasti USR je hexadecimálně 7FØØ, tedy 32512 decimálně a koncová adresa oblasti USR je 7FAØ hexadecimálně, tedy 32672 decimálně. Jak se snadno přesvědčíte, mezi těmito dvěma adresami je skutečně oblast paměti o obsahu 16Ø paměťových míst.

1.6 Přehrávání programů mezi pamětí počítače a magn. páskem

K této tématice je nutno se vrátit pouze stručnými poznámkami. Nyní víme, že programy mohou být jednak v jazyce BASIC, jednak ve strojovém kódu. Jak se přehrávají programy, které jsou v jazyce BASIC, dobře známe - používají se k tomu příkazy MSAVE a MLOAD.

Avšak programy ve strojovém kódu vložené nejčastěji do oblasti USR se nenahrávají současně s programem v jazyce BASIC, musíme je přehrávat zvlášť, a to v režimu MONITOR pomocí příkazů W a L postupem, který byl popsán v předchozí příručce, vztahující se k práci s počítačem v režimu MONITOR. K tomu potřebujeme vždy znát počáteční a koncovou adresu oblasti, v níž se náš program ve strojovém kódu nachází.

Nemusí to být vždy celá USR oblast, stačí jen ta její část, v níž se náš strojový program nachází.

1.7 Přechod mezi programem v jazyce BASIC a programem ve strojovém kódu

Protože mnoho programů je koncipováno tak, že část je v jazyce BASIC a část ve strojovém kódu, nyní se naučíme, jak zajistit přechod mezi oběma druhy programů, tedy přechod od programu v jazyce BASIC do programu ve strojovém kódu a naopak.

1/ V místě programu v jazyce BASIC, v němž má dojít k přechodu na strojový program, je jedno z těchto slov:

CALL USR WORD .

První slovo je příkaz, může stát tedy ihned za číslem programového řádku nebo za dvojtečkou. Druhá dvě slova jsou funkce, tedy před nimi musí být příkaz jazyka BASIC, jako například PRINT, LET A = apod.

Za každým ze slov CALL, USR a WORD musí být adresa, na níž začíná program ve strojovém kódu. Tato adresa musí být decimální, pro USR a WORD musí být navíc v závorkách - jako všechny argumenty funkcí. Podrobněji ještě viz 12.1.

Abychom nemuseli složitě přepočítávat hexadecimální adresy na decimální, můžeme použít funkce HEX, jejíž argument je hexadecimální číslo a funkční hodnota je příslušné číslo decimální.

Jestliže tedy program ve strojovém kódu bude začínat na hexadecimální adrese 7F3C v paměti počítače, pak v příslušném místě v programu v jazyce BASIC, kde se na program ve strojovém kódu přechází, musí být jeden z těchto pří-

kazů:

```
CALL HEX ( 7F3C )
PRINT USR ( HEX ( 7F3C ) )
PRINT WORD ( HEX ( 7F3C ) )
LET A = USR ( HEX ( 7F3C ) )
LET B = WORD ( HEX ( 7F3C ))
```

apod.

2/ V místě, na němž má dojít k návratu ze strojového programu do programu v jazyce BASIC a do místa, z něhož se realizoval odskok do strojového programu, musí být instrukce typu RETURN /nejčastěji RET s hexadecimálním kódem C9/, kterou většina strojových programů končí.

Nyní si ukážeme jednoduchý příklad.

Vypneme a zapneme počítač, uvolníme si 1ØØ paměťových míst oblasti USR pomocí příkazu CLEAR 5Ø, 1ØØ. Nyní přejdeme do režimu MONITOR a na adresách A4, A5, A6 a A7 /hexadecimálně/ zjistíme, že oblast USR se rozkládá od hexadecimální adresy 7F3C do hexadecimální adresy 7FAØ.

Nyní si některé adresy v oblasti USR obsadíme pomocí příkazu S v režimu MONITOR následujícími hexadecimálními kódy určitých instrukcí podle následující tabulky.

adresa	kód instrukce
7F3C	26
7F3D	ED
7F3E	2E
7F3F	3F
7F4Ø	36
7F41	4Ø

adresa	kód instrukce
7F42	C9

Je samozřejmé, že dosud nevíte, co přesně vyjadřují dvojciferná hexadecimální čísla. Pamatujte si zatím pouze, že každá instrukce pro strojové programování má tzv. kód. Tímto kódem je celé číslo od Ø do 255 /decimálně/, což je od Ø do FF /hexadecimálně/. Kódy daných instrukcí pro strojové programování se vkládají do řady na určitá paměťová místa počítače a tvoří tak strojový program. Pokud vkládáte strojový program v režimu MONITOR, musíte vkládat kódy hexadecimální, s jinými čísly v režimu MONITOR nepracujete. Tabulky instrukcí strojového programování a jejich hexadecimální kódy jsou v příloze - jejich význam je však dosud pro vás nepochopitelný.

Máte-li náš strojový program již vložen do paměti počítače, pomocí příkazu R se vrátíte do režimu BASIC a vložíte tento jednoduchý program:

```
1Ø CLS
2Ø PRINT "PRECHAZIM NA STROJOVY PROGRAM"
3Ø CALL HEX ( 7F3C)
4Ø PRINT "VRATIL JSEM SE DO BASICU"
5Ø END
```

Spustíte-li tento program příkazem RUN v režimu BASIC, objeví se na obrazovce obě hlášení způsobená programovými řádky 2Ø a 4Ø a navíc na pravém okraji obrazovky se objeví grafický znak. Prohlédneme-li si program v jazyce BASIC, vidíme, že zde není žádný příkaz, kterým by se grafický znak na obrazov-

ce nakreslil. Jeho objevení na obrazovce je zřejmě výsledkem našeho strojového programu. Všimneme si, že náš program v jazyce BASIC obsahuje na řádku 3Ø příkaz pro přechod na strojový program začínající na hexadecimální adrese 7F3C a náš strojový program obsahuje jako poslední instrukci RET, která se projevila svým kódem C9 na posledním místě řady instrukcí a navrací chod programu do režimu BASIC, v našem případě na řádek 4Ø.

Nyní změníme řádek 3Ø na tvar

3Ø PRINT USR(HEX(7F3C)) .

Po startu programu v režimu BASIC se na obrazovce objeví nejen dva známé texty podle řádků 2Ø a 4Ø a grafický symbol, ale i číslo 254. Protože je způsobeno příkazem PRINT a tedy bylo napsáno v důsledku realizace příkazu v režimu BASIC, je decimální. Pamatujme si, že je to dekadické vyjádření čísla, které bylo ve střadači po ukončení chodu programu ve strojovém kódu. Tímto způsobem je možno přenášet hodnoty z registru do programu v jazyce BASIC a dále s nimi pracovat.

Nyní změníme řádek 3Ø na tvar

3Ø PRINT WORD(HEX(7F3C)) .

Po startu takto upraveného programu se kromě známých věcí objeví na obrazovce číslo 6Ø735, o němž si budeme pamatovat, že je decimální a je to číslo uložené v dvojregistru HL po ukončení chodu strojového programu.

Tedy pomocí slov USR a WORD jazyka BASIC lze některé parametry získané při chodu strojového programu přenášet do programu v jazyce BASIC a dále s nimi pracovat v tomto vyšším programovacím jazyce.

Obě zmíněná slova samozřejmě vyvolávají automaticky chod strojového programu od zadané adresy tak, jako příkaz CALL.

Vrátíme se nyní do režimu MONITOR a náš strojový program doplníme následujícím způsobem:

adresa	kód instrukce
7F42	ØØ
7F43	ØØ
7F44	ØØ
7F45	ØØ
7F46	ØØ
7F47	C9

Doplnili jsme vlastně do našeho strojového programu několik instrukcí, jejichž kód je ØØ. Spustíme-li takto doplněný program, zjistíme, že dělá to samé, jako program nedoplněný. Instrukce, jejíž kód je ØØ, je zřejmě instrukcí, která nemá žádný účinek. Nazývá se NOP.

Shrnutí

Známe nyní již dvě instrukce strojového kódu. Jsou to:

RET , její hexadecimální kód je C9 , způsobí návrat do programu v jazyce BASIC

NOP , její hexadecimální kód je ØØ , je to instrukce bez účinku.
S výhodou se používá pro rezervaci určitých paměťových míst pro jejich pozdější jiné obsazení.

1.8 Volání strojových programů v režimu MONITOR

Zatím umíme startovat programy ve strojovém kódu jen pomocí slov CALL, USR nebo WORD, tedy z režimu BASIC. Strojové programy můžeme však startovat i v režimu MONITOR - používají se k tomu příkazy C a G. Jistě máte ještě v paměti počítače strojový program z minulého článku 1.7. Vyčistěte pomocí příkazu CLS v režimu BASIC obrazovku. Přejdeme pomocí tlačítka BR do režimu MONITOR a tlačítkem CR odešleme příkaz

C 7F3C

Všimněte si, že 7F3C je opět počáteční adresa našeho strojového programu. Na obrazovce se objeví grafický symbol, což znamená, že strojový program byl spuštěn. Jakmile došlo na poslední instrukci RET s kódem C9, chod strojového programu byl ukončen a počítač se vrátil do režimu, z něhož vyšel, tedy do režimu MONITOR.

Strojový program v režimu MONITOR je možno startovat také příkazem G místo příkazu C. Za G musí být rovněž první adresa strojového programu tak, jako za příkazem C. Pokud ale používáme příkaz G, není zaručen návrat do režimu, z něhož byl strojový program vyvolán. To se projeví nejčastěji tím, že se neobjeví na obrazovce blikající kurzor, i když chod strojového programu skončil.

Tímto jsme rovněž ukončili úplný přehled všech příkazů režimu MONITOR a jejich funkce. Příkazy F, R, S, L, M, W a D známe z příručky o práci v režimu MONITOR, funkci příkazu X z kapitoly 1.3, C a G jsme poznali nyní.

Poznámka: Víme, že aritmetické operace využívají dvojkový a dvojkový doplňkový kód. Lze však pracovat ještě v tzv.

dvojkovém přímém kódu, pokud programátor vhodně upravuje určité bity registru příznaků. Dvojkového přímého kódu se nepoužívá často, přesto jej pro úplnost uvádíme v převodních tabulkách v příloze. Je tam označen D-DP.

Nyní máte všechny potřebné znalosti k tomu, abychom si mohli ukázat význam a funkci jednotlivých instrukcí pro strojové programování, jejich hexadecimální kódy apod. Při studiu následujících kapitol vám již nesmí dělat obtíže vkládání strojového programu do paměti počítače, jeho vyvolávání v režimu BASIC nebo MONITOR, pojmy jednoduchých registrů a dvojregistrů. Doporučujeme tyto základní dovednosti a poznatky zopakovat a pak teprve přistoupit k následujícím řádkům.

Poznámka:

V některých publikacích se dodržuje konvence v psaní adres v tom, že se důsledně uvádějí čtyřciferné. Například hexadecimální adresy 8, 2A, 8Ø1 se podle této konvence píšou ØØØ8, ØØ2A, Ø8Ø1. V této příručce budeme ve většině případů používat kratší zápis, který je výhodnější zvláště v tabulkách.

2. Základní instrukce strojového kódu

2.1 Instrukce pro přímé obsazení jednoduchých registrů

Pomocí těchto instrukcí lze do libovolného jednoduchého registru A, B, C, D, E, H, L vložit dané číslo. Označují se MVI, za tímto znakem je vždy ještě uveden registr, kterého se instrukce týká. Pak následuje čárka a za ní je hexadecimální dvojciferné číslo, které instrukce do příslušného registru umístí. Například

MVI B, 1C

znamená, že hexadecimální číslo 1C bude umístěno v registru B touto instrukcí. Převod čísla 1C na dvojkový kód zajišťuje počítač při vkládání čísla do registru automaticky. Číslo za čárkou je z intervalu $\langle ØØ; FF \rangle$.

Všechny instrukce jmenovaného typu jsou následující:

kód instrukce	instrukce
3E	MVI A, d
Ø6	MVI B, d
ØE	MVI C, d
16	MVI D, d
1E	MVI E, d
26	MVI H, d
2E	MVI L, d

Kódy instrukcí v tabulce jsou samozřejmě hexadecimální, d je značka pro hexadecimální číslo, které bude do registru vloženo, $d \in \langle ØØ; FF \rangle$.

Příklad:

Pomocí strojového programu vložte do registru C číslo 1A.

Řešení:

Vypněte a zapněte počítač. Pomocí BR přejděte do režimu MONITOR a pomocí příkazu X si nechte vypsat obsahy všech registrů. Vidíte, že v registru C je hexadecimální číslo EC. Nyní pomocí příkazu S v režimu MONITOR vložíme tento krátký strojový program provádějící instrukci MVI C, 1A od hexadecimální adresy 5ØØ do 5Ø2:

adresa	instrukce	kód
5ØØ	MVI C, d	ØE
5Ø1	1A	1A
5Ø2	RET	C9

Všimněte si, že celá instrukce MVI zaujímá dvě paměťová místa - na prvním je kód instrukce, na druhém je hexadecimální číslo, s nímž bude tato instrukce pracovat. Program končí instrukcí RET.

Po vložení programu do paměti jej spustíme pomocí příkazu C5ØØ v režimu MONITOR. Pomocí příkazu X v tomtéž režimu se můžete přesvědčit, že se obsah registru C předepsaným způsobem změnil.

2.2 Instrukce pro přímé obsazení dvojregistrů BC, DE, HL, SP

Dvojregistr bychom mohli obsadit tak, že bychom pomocí instrukcí MVI obsadili obě části zvoleného dvojregistru zvlášť. Můžeme však také obsadit dvojregistr naráz pomocí instrukcí typu LXI. Za touto instrukcí se píše vždy první registr uvažovaného dvojregistru, pak následuje čárka a za

čárkou je uvedeno hexadecimální číslo, které při provádění instrukce přejde do uvažovaného dvojregistru. Například tedy

LXI B, 1A1B

znamená, že dvojregistr BC bude obsazen číslem 1A1B tak, že v registru B bude číslo 1A a v registru C bude číslo 1B. Číslo za čárkou v zápise instrukce je celé hexadecimální číslo z intervalu ⟨ØØØØ; FFFF⟩ .

Všechny instrukce tohoto typu jsou následující:

kód instrukce	instrukce
Ø1	LXI B, w
11	LXI D, w
21	LXI H, w
31	LXI SP, w

w je hexadecimální celé číslo z intervalu ⟨ØØØØ; FFFF⟩ .

Vložte pomocí strojového programu do dvojregistru HL číslo 7F1C.

Postup:

V režimu MONITOR zjistíme pomocí příkazu X stávající obsazení registrů a tedy i dvojregistru HL. Nyní pomocí příkazu S ve stejném režimu vložíme tento strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód
5ØØ	LXI H, w	21
5Ø1		1C
5Ø2		7F
5Ø3	RET	C9

Poznámka: Všimněte si, že číslo musíme ve strojovém programu vkládat "obráceně", tj. nejdříve dva nižší řády a pak teprve dva vyšší řády.

Spustíme tento program příkazem C5ØØ v režimu MONITOR a pomocí příkazu X v tomtéž režimu si můžeme ověřit, že došlo k obsazení dvojregistru HL požadovaným způsobem.

2.3 Přímé obsazení registru příznaků F a dvojregistru PC

A/ Přímé obsazení dvojregistru PC zvoleným hexadecimálním číslem pomocí jediné instrukce není možné.

B/ Přímé obsazení registru příznaků F zvoleným hexadecimálním číslem z intervalu ⟨ØØ; FF⟩ není možné, registr příznaků F se obsazuje automaticky podle výsledků operací v jiném registru. Je pouze možno dosazovat nebo změnit cifru na nultém bitu tohoto registru - tedy svévolně měnit příznak přenosu CY.

Změna hodnoty nultého bitu registru F se provádí pomocí těchto instrukcí:

kód	instrukce	význam
37	STC	dosadí jedničku na nultý bit reg. F

kód	instrukce	význam
3F	CMC	změní cifru na nultém bitu reg. F z nuly na jedničku nebo opačně

Příklad:

Změňte nultý bit registru příznaků F pomocí strojového programu.

Postup:

Po zapnutí počítače je registr F obsazen hexadecimálním číslem 86, což je číslo sudé, tedy na nultém bitu v registru F je cifra Ø. V režimu MONITOR vložíme pomocí příkazu S tento strojový program od adresy 5ØØ /hexadecimálně/.

adresa	instrukce	kód
5ØØ	STC	37
5Ø1	RET	C9

V režimu MONITOR spustíme tento program příkazem C5ØØ a pak pomocí příkazu X ve stejném režimu necháme vypsat obsahy všech registrů. Vidíme, že v registru příznaků F je nyní hexadecimální číslo 87, což je číslo liché, takže na nultém bitu registru F je nyní cifra 1.

Poznámka:

Naučte se již nyní používat tabulky instrukcí strojového kódu, které jsou uvedeny v příloze. Najděte si instrukce, které již znáte, jejich kód apod.

2.4 Instrukce pro přímý přesun čísla z jednoduchého registru do jiného jednoduchého registru

Jsou to instrukce, které se označují MOV. Za instrukcí následuje písmeno jmenující registr, do něhož bude číslo přesunuto, následuje čárka a potom písmeno jmenující registr, z něhož bude číslo přesunuto.

Například

MOV H, A

znamená, že při provádění této instrukce se obsah střadače A přesune do registru H, přičemž obsah registru A zůstane při provedení instrukce zachován. Hexadecimální kód této instrukce je 67.

Všechny přesunové instrukce uvedeného typu mají následující hexadecimální kódy - srovnejte s tabulkou kódů instrukcí v příloze.(O instrukcích obsahujících M si řekneme později.)

4Ø; ... ; 45; 47; ... ; 4D; 4F; ... ; 55; 57; ... ; 5D;
5F; ... ; 65; 67; ... ; 6D; 6F; 78; ... ; 7D; 7F .

Příklad:

Přesuňte obsah registru B do registru H.

Řešení:

Po zapnutí počítače je v registru B hexadecimální číslo Ø1 a v registru H hexadecimální číslo ØØ, o čemž se lze přesvědčit pomocí příkazu X v režimu MONITOR. Pomocí příkazu S v režimu MONITOR vložíme tento krátký strojový program od adresy 5ØØ:

adresa	instrukce	kód
5ØØ	MOV H, B	6Ø
5Ø1	RET	C9

Program spustíme v režimu MONITOR příkazem C5ØØ a pak pomocí příkazu X v tomtéž režimu zjistíme, že se číslo Ø1 z registru B přesunulo do registru H, přičemž však Ø1 v registru B zůstalo zachováno.

Poznámka:

Instrukce tvaru MOV A, A nebo MOV B, B apod., kde probíhá vlastně přesun do téhož registru, se chovají jako instrukce bez účinku /NOP/, trvají však delší dobu.

2.5 Instrukce pro přímý přesun čísla z dvojregistru do jiného dvojregistru

Pomocí jediné instrukce jde přesunout obsah dvojregistru HL jedině do dvojregistru SP nebo PC. První instrukce je

SPHL s kódem F9 ,

která přesune obsah dvojregistru HL do dvojregistru SP a druhá je

PCHL s kódem E9 ,

která přesune obsah dvojregistru HL do dvojregistru PC. Obsah dvojregistru HL při provádění obou instrukcí zůstane opět zachován.

Jednoduchý příklad uvádět nebudeme, protože nevhodným obsazením dvojregistru SP nebo PC bychom se nemuseli úspěšně navrátit po ukončení ukázkového programu do režimu, z něhož

byl program volán. Víme už, že dvojregistry SP a PC řídí chod programu, proto zásah do jejich obsahu může způsobit, že počítač ve své paměti "zabloudí".

Poznámka:

Kdybychom chtěli provádět přesun obsahu dvojregistru BC do dvojregistru DE, nestačí nám k tomu pouze jediná instrukce - taková totiž neexistuje. Museli bychom použít dvou instrukcí typu MOV - první by přesunula obsah jednoduchého registru B do jednoduchého registru D a druhá analogicky by obsah C přesunula do E. Jednalo by se o instrukce MOV D, B a MOV E, C prováděné po sobě.

2.6 Instrukce pro vzájemnou výměnu obsahů dvojregistrů HL a DE

Vzájemnou výměnu /nikoliv jen jednostranný přesun!/ obsahů uvedených dvojregistrů zajišťuje instrukce:

XCHG s kódem EB .

Činnost instrukce nám ukáže následující jednoduchý příklad.

Příklad:

Proveďte vzájemnou výměnu obsahů dvojregistrů HL a DE.

Řešení:

Po zapnutí počítače je dvojregistr HL obsazen hexadecimálním číslem ØØFF - tj. v jednom jednoduchém registru je ØØ a ve druhém je FF. Ve dvojregistru DE je po zapnutí počítače hexadecimální číslo FFØØ. Přesvědčíme se o tom pomocí příkazu X v režimu MONITOR. Pomocí příkazu S v tomtéž režimu vložíme od hexadecimální adresy 5ØØ tento program:

adresa	instrukce	kód
5ØØ	XCHG	EB
5Ø1	RET	C9

Po startu tohoto programu příkazem C5ØØ v režimu MONITOR se pomocí příkazu X přesvědčíme, že došlo ke vzájemné výměně obsahů obou jmenovaných dvojregistrů.

Poznámka:

Vzájemnou výměnu obsahů jiných dvojregistrů nelze zajistit pomocí jediné instrukce - taková opět neexistuje. Lze to však provést pomocí vhodné posloupnosti přesunových instrukcí typu MOV, které působí na jednoduché registry. Nutno si uvědomit, že MOV zachovává původní obsazení registru, z něhož je číslo "kopírováno" do jiného registru.

2.7 Naplnění registrů pomocí obsahu paměťových míst o dané adrese - tzv. přímé adresování

Zatím jsme se naučili pouze naplnit registry daným hexadecimálním číslem, které bylo uvedeno vždy za příslušnou instrukcí - vzpomeňte si na instrukce typu MVI a LXI. Nyní si ukážeme, jak je možno do určitých registrů umístit hexadecimální číslo, které je uloženo v určitém paměťovém místě počítače. Toto paměťové místo má pochopitelně svou hexadecimální čtyřcifernou adresu - např. ØØØA, Ø16C, ECØØ apod. Jde vlastně o překopírování obsahu zvoleného paměťového místa do určitého registru, přičemž původní obsazení paměťového místa zůstane při tomto procesu nezměněno.

Uvedeným způsobem lze pomocí jediné instrukce obsadit

pouze střadač A a dvojregistr HL.

Instrukce, která naplní střadač A obsahem paměťového místa, určeného hexadecimální adresou a, je:

LDA, a s kódem 3A ,

kde a je z intervalu ⟨ØØØØ; FFFF⟩ .

Tedy konkrétně

LDA, 1A5Ø

okopíruje do střadače A hexadecimální číslo, které je umístěno na paměťovém místě s hexadecimální adresou 1A5Ø.

Upozornění:

V konkrétním programu nutno zadávat příslušnou adresu tak, že nejdříve zadáme dva nižší řády, tedy 5Ø a pak teprve dva vyšší řády - 1A.

Příklad:

Naplňte paměťové místo s adresou Ø6Ø1 hexadecimálním číslem A7 a pak toto číslo vložte pomocí strojového programu do střadače A.

Řešení:

Po zapnutí počítače vložíme v režimu MONITOR pomocí příkazu S na adresu Ø6Ø1 číslo A7. Pak se pomocí příkazu X v tomtéž režimu přesvědčíme, že střadač A je obsazen jiným číslem, než je A7 - většinou je tam 8A. Nyní v režimu MONITOR vložíme následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód
5ØØ	LDA, a	3A
5Ø1	Ø1	Ø1
5Ø2	Ø6	Ø6
5Ø3	RET	C9

Spustíme tento program příkazem C5ØØ v režimu MONITOR a opět pomocí příkazu X v tomtéž režimu zjistíme, že ve střadači A se objevilo skutečně číslo A7, které jsme vložili na paměťové místo s adresou Ø6Ø1. Pomocí příkazu D6ØØ v režimu MONITOR snadno dále zjistíme, že číslo A7 na adrese Ø6Ø1 zůstalo při činnosti instrukce LDA, a nezměněno.

Instrukce, která naplní dvojregistr HL obsahem dvou paměťových míst, vypadá takto:

LHLD, a s kódem 2A ,

kde a je hexadecimální adresa z intervalu <ØØØØ; FFFF> . Instrukce pracuje tak, že naplní registr L obsahem paměťového místa, jehož adresa je za instrukcí uvedena a registr H naplní obsahem paměťového místa sousedního, tedy s adresou o 1 vyšší.

Konkrétně

LHLD Ø6Ø1

naplní registr L obsahem paměťového místa s adresou Ø6Ø1 a registr H obsahem paměťového místa s hexadecimální adresou Ø6Ø2.

Upozornění:

V programech nutno opět psát u adresy nejdříve oba

nižší řády a pak oba vyšší řády obdobně, jako u LDA.

Příklad:

Naplňte paměťové místo s adresou Ø6Ø1 /hexadecimální/ hexadecimálním číslem AA a paměťové místo s hexadecimální adresou Ø6Ø2 hexadecimálním číslem B5. Pomocí strojového programu obsaďte dvojregistr HL hexadecimálním číslem B5AA.

Řešení:

Po zapnutí počítače vložíme v režimu MONITOR pomocí příkazu S na paměťové místo s hexadecimální adresou Ø6Ø1 hexadecimální číslo AA a na následující paměťové místo s hexadecimální adresou Ø6Ø2 hexadecimální číslo B5, dále se pomocí příkazu X ve stejném režimu přesvědčíme, že dvojregistr HL je obsazen jiným číslem než B5AA. Nyní pomocí příkazu S v režimu MONITOR vložíme následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód
5ØØ	LHLD, a	2A
5Ø1	Ø1	Ø1
5Ø2	Ø6	Ø6
5Ø3	RET	C9

Spustíme nyní tento program příkazem C5ØØ v režimu MONITOR a pomocí příkazu X v tomtéž režimu zjistíme, že se dvojregistr obsadil požadovaným způsobem. Pomocí příkazu D6ØØ se pro úplnost můžeme přesvědčit, že prováděním instrukce LHLD, a zůstala hexadecimální

Poznámka:

Naplnění jiných jednoduchých nebo dvojitých registrů obsahem paměťového místa o zvolené adrese nutno provádět již pomocí více než jedné instrukce - nejdříve přeneseme číslo do střadače A nebo dvojregistru HL pomocí uvedených instrukcí LDA nebo LHLD a pak použijeme vhodné přesunové instrukce typu MOV.

2.8 Uložení obsahu registrů na paměťová místa o daných adresách - přímé adresování

V minulé stati jsme se naučili naplnit registry obsahem zvoleného paměťového místa. Nyní si ukážeme opačný proces, kdy obsah některých registrů lze uložit na paměťové místo o zvolené adrese, která se opět uvádí za instrukcí. V programech se u adres uvádí zase nejdříve dva nižší řády, pak oba vyšší. Po provedení instrukcí tohoto typu zůstává vždy obsah registru zachován, pouze se překopíruje na jiné místo v paměti. Pomocí jediné instrukce lze do zvolených paměťových míst uložit pouze obsah střadače A a dvojregistru HL. Obsah jiných jednoduchých registrů nebo dvojregistrů je možno do paměti ukládat pomocí více instrukcí - nejprve pomocí instrukcí typu MOV přesuneme obsah takových registrů buď do střadače A nebo do dvojregistru HL a pak teprve provedeme uložení do paměťového místa.

Instrukce, která "okopíruje" obsah střadače do zvoleného paměťového místa s danou adresou a , je instrukce

STA, a s kódem 32 ,

kde adresa a je hexadecimální z intervalu ⟨ØØØØ; FFFF⟩ .

STA Ø95Ø

uloží obsah střadače A do paměťového místa s adresou Ø95Ø.

Příklad:

Pomocí strojového programu umístěte obsah střadače A na paměťové místo s adresou Ø7Ø5.

Řešení:

Po zapnutí počítače pomocí příkazu X v režimu MONITOR zjistíme, jakým číslem je obsazen střadač A. Dále pomocí příkazu D7ØØ v tomtéž režimu zjistíme, jakým hexadecimálním číslem je obsazeno paměťové místo s hexadecimální adresou Ø7Ø5. Většinou po zapnutí počítače je to FF. Nyní v režimu MONITOR pomocí příkazu S vložíme tento strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód
5ØØ	STA, a	32
5Ø1	Ø5	Ø5
5Ø2	Ø7	Ø7
5Ø3	RET	C9

Spustíme tento program příkazem C5ØØ v režimu MONITOR a pak pomocí příkazu D7ØØ v tomtéž režimu zjistíme, že se číslo z registru A objevilo na paměťovém místě s hexadecimální adresou Ø7Ø5. Pomocí příkazu X bychom se mohli dále přesvědčit, že při provádění instrukce STA se obsah střadače A nezměnil.

Instrukce, která uloží obsah dvojregistru HL do dvou

paměťových míst, vypadá takto:

SHLD, a **s kódem 22** ,

kde a je adresa jednoho z těchto paměťových míst, je hexadecimální z intervalu <ØØØØ; FFFF> . V programech nutno psát cpět nejdříve dva nižší řády adresy, pak teprve dva vyšší. Instrukce pracuje tak, že překopíruje obsah registru L do paměťového místa, jehož adresa je za instrukcí uvedena a obsah registru H okopíruje na sousední paměťové místo, jehož adresa je o 1 vyšší.

Konkrétně

SHLD, Ø5A1

překopíruje obsah registru L na paměťové místo s adresou Ø5A1 a obsah registru H překopíruje na paměťové místo s adresou Ø5A2.

Příklad:

Uložte pomocí strojového programu obsah dvojregistru HL do paměti tak, aby obsah registru L byl na adrese Ø8Ø5.

Řešení:

Po zapnutí počítače příkazem XH v režimu MONITOR vložíme do registru H hexadecimální číslo 1A a do registru L hexadecimální číslo 2B. Nyní se pomocí příkazu D8ØØ v tomtéž režimu přesvědčíme, že adresy Ø8Ø5 a Ø8Ø6 jsou obsazeny jinými čísly, nejčastěji po zapnutí počítače jsou to hexadecimální čísla FF. V režimu MONITOR vložíme následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód
5ØØ	SHLD, a	22
5Ø1	Ø5	Ø5
5Ø2	Ø8	Ø8
5Ø3	RET	C9

Po spuštění programu příkazem C5ØØ se podíváme pomocí příkazu D8ØØ /oboje v režimu MONITOR/ na obsazení paměťových míst s hexadecimálními adresami Ø8Ø5 a Ø8Ø6. Vidíme, že obsah registru L - tedy 2B - je na adrese Ø8Ø5 a obsah registru H je na adrese Ø8Ø6, jak úloha požadovala. Pomocí příkazu X v tomtéž režimu se lze přesvědčit, že při vykonávání instrukce SHLD zůstal obsah dvojregistru HL zachován.

2.9 Naplnění střadače A nebo jednoduchého registru obsahem paměťového místa, jehož adresa je uložena ve dvojregistru - tzv. nepřímé adresování

Jednoduché registry je možno kromě již popsaných způsobů naplnit rovněž obsahem paměťového místa, jehož adresa je uložena v některém dvojregistru HL, BC nebo DE.

1/ Střadač A lze naplnit obsahem paměťového místa, jehož hexadecimální adresa je uložena v dvojregistru HL nebo BC nebo DE pomocí následujících instrukcí:

kód	instrukce	poznámka
ØA	LDAX B	adresa uložena v BC
1A	LDAX D	adresa uložena v DE
7E	MOV A, M	adresa uložena v HL

Symbol M v poslední instrukci znamená fiktivní registr v paměti, jehož adresa je dána obsahem HL. Symboly B a D v prvních dvou instrukcích znamenají zkratky za dvojregistry - po řadě BC a DE.

Konkrétně tedy instrukce

LDAX B

znamená, že do střadače A bude překopírován obsah paměťového místa, jehož adresa je v okamžiku provádění instrukce uložena v dvojregistru BC. Instrukce

MOV A, M

znamená, že do střadače bude překopírován obsah paměťového místa, jehož adresa je uložena v dvojregistru HL.

2/ Jiný jednoduchý registr - B, C, D, E, H, a L - je možno obsadit číslem z paměťového místa, jehož adresa je pouze v dvojregistru HL. Není možno použít dvojregistrů DE a BC. Všechny instrukce jmenovaného typu jsou následující:

kód	instrukce	význam
46	MOV B, M	naplnění reg. B
4E	MOV C, M	naplnění reg. C
56	MOV D, M	naplnění reg. D
5E	MOV E, M	naplnění reg. E
66	MOV H, M	naplnění reg. H
6E	MOV L, M	naplnění reg. L

Konkrétně instrukce

MOV D, M

znamená, že se do jednoduchého registru D překopíruje obsah

paměťového místa, jehož adresa v okamžiku vykonávání instrukce byla uložena v dvojregistru HL.

Příklad:

Po zapnutí počítače obsaďte paměťové místo s adresou 6ØØ číslem A1, paměťové místo s adresou 6Ø1 číslem B2 /vše hexadecimálně/. Pak přesuňte obsah adresy 6ØØ do střadače a obsah adresy 6Ø1 do registru H.

Postup:

Nejdříve v režimu MONITOR pomocí příkazu S obsadíme adresy 6ØØ a 6Ø1 po řadě hexadecimálními čísly A1 a B2. Pomocí příkazu X v tomtéž režimu se přesvědčíme, že v registrech A a H jsou jiná čísla, než úlohou požadovaná. Nyní pomocí příkazu S vložíme tento strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód	význam
5ØØ	LXI H, w	21	vložení adresy 6ØØ do HL
5Ø1	ØØ	ØØ	
5Ø2	Ø6	Ø6	
5Ø3	XCHG	EB	přesun adresy z HL do DE
5Ø4	LXI H, w	21	vložení adresy 6Ø1 do HL
5Ø5	Ø1	Ø1	
5Ø6	Ø6	Ø6	
5Ø7	LDAX D	1A	přesun čísla do A adresa byla v DE
5Ø8	MOV H, M	66	přesun čísla do H adresa byla v HL
5Ø9	RET	C9	konec

Spustíme program příkazem C5ØØ v režimu MONITOR a pak pomocí příkazu X v tomtéž režimu zjistíme, že registry A a H byly obsazeny požadovaným způsobem. Dále se prostřednictvím příkazu D6ØØ můžeme přesvědčit, že hodnoty vložené na adresách 6ØØ a 6Ø1 se při provádění instrukcí LDAX D a MOV H, M nezměnily.

2.10 Uložení obsahu střadače nebo jednoduchého registru na paměťové místo, jehož adresa je uložená ve dvojregistru - nepřímé adresování

Obsahy jednoduchých registrů lze překopírovat do libovolného místa v paměti počítače, jehož adresa je uložena v některém z dvojregistrů BC, DE nebo HL. Jedná se vlastně o opačný postup, než je uveden v předchozí stati, kdy se číslo z paměťového místa přesouvalo do jednoduchého registru.

1/ Obsah střadače A lze "překopírovat" na paměťové místo, jehož adresa je v libovolném dvojregistru pomocí instrukcí:

kód	instrukce	význam
Ø2	STAX, B	adresa uložena v BC
12	STAX, D	adresa uložena v DE
77	MOV M, A	adresa uložena v HL

Zkratky B, D a M v instrukcích mají stejný význam, jako byl uveden v předchozím článku.

Konkrétně instrukce

STAX, D

znamená, že obsah střadače A bude překopírován do paměťového místa o adrese, která je uložena v okamžiku provádění instrukce v dvojregistru DE a instrukce

MOV M, A

znamená, že obsah střadače A bude překopírován na paměťové místo, jehož adresa je uložená v dvojregistru HL.

2/ Obsah libovolného jiného jednoduchého registru - B, C, D, E, H, a L - je možno překopírovat pomocí jediné instrukce do paměťového místa, jehož adresa je pouze v dvojregistru HL, nelze mít jako v minulém případě adresu uloženou ve dvojregistrech BC nebo DE. Všechny instrukce tohoto typu jsou následující:

kód	instrukce	poznámka
7Ø	MOV M, B	překopírování z reg. B
71	MOV M, C	překopírování z reg. C
72	MOV M, D	překopírování z reg. D
73	MOV M, E	překopírování z reg. E
74	MOV M, H	překopírování z reg. H
75	MOV M, L	překopírování z reg. L

M má opět známý význam - je to fiktivní registr v paměti, při překopírování obsahu jednoduchého registru do paměťového místa zůstává obsah v registru zachován.

Konkrétně instrukce

MOV M, C

znamená, že obsah jednoduchého registru C bude překopírován na paměťové místo, jehož adresa je uložena v dvojregistru HL v okamžiku provádění jmenované instrukce.

Příklad:

Vložte do střadače A číslo 21 a do registru B číslo

3F /obě hexadecimální/ a pak strojovým programem překopírujte obsah střadače na paměťové místo s hexadecimální adresou EC1Ø a obsah registru B na paměťové místo s adresou EC15 - rovněž hexadecimální.

Postup:

V režimu MONITOR pomocí příkazu S vložíme od hexadecimální adresy 5ØØ následující strojový program:

adresa	instrukce	kód	význam
5ØØ 5Ø1	MVI A, d 21	3E 21	obsazení A číslem 21
5Ø2 5Ø3	MVI B, d 3F	Ø6 3F	obsazení B číslem 3F
5Ø4 5Ø5 5Ø6	LXI H, w 1Ø EC	21 1Ø EC	obsazení dvojreg. HL číslem EC1Ø
5Ø7	XCHG	EB	přesun obsahu HL do DE
5Ø8 5Ø9 5ØA	LXI H, w 15 EC	21 15 EC	obsazení dvojreg. HL číslem EC15
5ØB	STAX, D	12	překop. obsahu A na místo, jehož adresa je v dvojreg. DE
5ØC	MOV M, B	7Ø	překop. obsahu B na místo, jehož adr. je v dvojreg. HL
5ØD	RET	C9	konec

Dále v režimu BASIC vložíme následující program:


```
1 CLS
2 CALL HEX ( 5ØØ )
3 END
```

Spustíme tento program příkazem RUN v režimu BASIC /druhý programový řádek spustí náš předchozí program ve strojovém kódu/. Obrazovka se nejdříve vyčistí - viz 1. řádek programu - dále se na prvním řádku obrazovky objeví vykřičník a otazník a nakonec se objeví nápis READY s blikajícím kurzorem, což znamená, že došlo k návratu ze strojového programu a chod programu v režimu BASIC byl ukončen příkazem END na třetím řádku. Vykřičník a otazník na obrazovce se objevily v důsledku chodu našeho strojového programu, v jehož působení se hexadecimální číslo 21 přesunulo na adresu EC1Ø, což je jedna z adres oblasti VIDEORAM. Je-li nějaká adresa oblasti VIDEORAM obsazena hexadecimálním číslem, objeví se na příslušném místě obrazovky znak, jehož hexadecimální kód je na této adrese - v našem případě vykřičník, jehož hexadecimální kód je právě 21. Obdobné zdůvodnění je i pro otazník na obrazovce.

3. Aritmetické instrukce

Pomocí aritmetických instrukcí lze provádět sečítání nebo odečítání v registrech. Aritmetické instrukce mají rovněž vliv na jednotlivé bity registru příznaků F podle pravidel dříve uvedených. Zatím žádné dříve vykládané instrukce tuto vlastnost neměly kromě CMC a STC, které přímo ovlivňovaly jednoduchým způsobem hodnotu příznaku CY. Dosavadní instrukce

pouze obsazovaly jednotlivé registry nebo jejich obsah přesouvaly bez působení na bity registru příznaků F.

3.1 Aritmetické instrukce pro přičítání nebo odečítání ve střadači A

1/ Instrukce pro sečítání pracují tak, že k původnímu obsahu střadače A přičtou určité číslo, které může být buď za instrukcí jako přímý operand /instrukce ADI/, nebo toto přičítané číslo může být v libovolném jednoduchém registru A, B, C, D, E, H, L /instrukce ADD/, nebo může být na libovolném místě v paměti, jehož adresa je uložena v dvojregistru HL /instrukce ADD M/.

Například instrukce

ADI 1A

přičte k obsahu střadače hexadecimální číslo 1A, instrukce

ADD C

přičte k původnímu obsahu střadače A číslo, které je uložené v registru C a instrukce

ADD M

přičte k původnímu obsahu střadače A číslo, které je na paměťovém místě, jehož adresa je uložena v dvojregistru HL.

Ve všech případech je výsledek operace uložen v registru A.

Podle výsledků operací jsou nastaveny všechny bity registru příznaků F v souladu s dříve uvedenými pravidly.

Všechny instrukce uvedených typů jsou následující:

kód	instrukce	význam
C6	ADI, d	přičtení čísla d k obsahu střadače A d ∈ ⟨ØØ; FF⟩
87 8Ø 81 82 83 84 85	ADD A ADD B ADD C ADD D ADD E ADD H ADD L	přičtení čísla z registru uvedeného za instrukcí k obsahu střadače A
86	ADD M	přičtení čísla, které je na paměťovém místě, jehož adresa je uložena v HL, k obsahu střadače A

Příklad:

Vložte do střadače A hexadecimální číslo 74 a do registru C hexadecimální číslo 87. Proveďte součet obou čísel ve střadači A.

Rozbor:

Ve střadači bude uloženo číslo:

hexadecimálně	decimálně	dvojkově
74	116	Ø111 Ø1ØØ

V registru C bude vloženo číslo:

hexadecimálně	decimálně	dvojkově
87	135	1ØØØ Ø111

Po sečtení obou čísel bude ve střadači A číslo:

hexadecimálně	decimálně	dvojkově
FB	251	1111 1Ø11

Všimneme si ještě, jak budou obsazeny jednotlivé bity registru příznaků F:

7. bit - S = 1 - v sedmém řádu dvojkového tvaru výsledku je cifra 1

6. bit - Z = Ø - výsledek ve střadači není nula

5. bit - Ø - stabilní obsazení

4. bit - AC = Ø - při sečítání nedošlo k přenosu ze 3. do 4. řádu dvojkového tvaru

3. bit - Ø - stabilní obsazení

2. bit - P = Ø - ve dvojkovém tvaru výsledku je lichý počet jedniček

1. bit - 1 - stabilní obsazení

Ø. bit - CY = Ø - nedošlo k přenosu ze 7. řádu dvojkového tvaru výsledku.

Po provedení instrukce sečítání bude tedy v registru příznaků F uloženo číslo, jehož dvojkový tvar bude:

1ØØØ ØØ1Ø

Postup:

Po zapnutí počítače zjistíme příkazem X v režimu MONITOR stávající obsazení všech registrů, které je jiné, než úloha požaduje. Nyní pomocí příkazu S v tomtéž režimu vložíme následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód	význam
5ØØ 5Ø1	MVI C, d 87	ØE 87	vložení hexad. čísla 87 do registru C
5Ø2 5Ø3	MVI A, d 74	3E 74	vložení hexad. čísla 74 do střadače A
5Ø4	ADD C	81	přičtení obsahu reg. C ke střadači A
5Ø5	RET	C9	konec

Nyní tento program spustíme příkazem C5ØØ v režimu MONITOR a pomocí příkazu X v tomtéž režimu se podíváme na obsazení všech registrů. Vidíme, že ve střadači A je nyní uložen součet - hexadecimální číslo FB a v registru příznaků F je hexadecimální číslo 82, což souhlasí s předchozím rozborem.

2/ Instrukce pro odečítání pracují tak, že od obsahu střadače A odečtou určité číslo /menšitele/. Tento menšitel může být uveden přímo za instrukcí odečítání jako operand /instrukce SUI/, nebo může být uložen v libovolném registru A, B, C, D, E, H, L /instrukce SUB/, nebo konečně může být na libovolném paměťovém místě, jehož adresa je uložena v okamžiku provádění instrukce v dvojregistru HL /instrukce SUB M/. Všimněte si analogie s předchozím případem sečítání.

Například instrukce

SUI 8C

odečte od hodnoty ve střadači A číslo 8C, instrukce

SUB L

odečte od hodnoty ve střadači A číslo, které je uloženo v registru L a instrukce

SUB M

odečte od obsahu střadače A číslo, které je na paměťovém místě, jehož adresa se nachází v okamžiku provádění jmenované instrukce v dvojregistru HL.

Výsledek odečítání je uložen vždy ve střadači A. Rovněž instrukce pro odečítání mění automaticky obsazení bitů registru příznaků F podle pravidel dříve uvedených. Nezapomeňte si zopakovat, že příznak CY se nastavuje při odečítání jinak, než při sečítání.

Proces odečítání probíhá tak, že se nejdříve vytvoří dvojkový doplněk menšitele a tento se přičte k původnímu obsahu registru A.

Všechny instrukce pro odečítání jsou následující:

kód	instrukce	význam
D6	SUI, d	odečtení čísla d od čísla ve střadači A d ∈ ⟨ØØ; FF⟩
97	SUB A	odečtení čísla, které je v registru uvedeném za instrukcí, od čísla ve střadači A
9Ø	SUB B	
91	SUB C	
92	SUB D	
93	SUB E	
94	SUB H	
95	SUB L	
96	SUB M	odečtení čísla, které je na pam. místě, od hodnoty v A - adresa místa v HL

Příklad:

Vložte na paměťové místo o hexadecimální adrese 6Ø1 hexadecimální číslo 2C, do střadače A hexadecimální číslo A7 a pomocí strojového programu proveďte ve střadači rozdíl A7 - 2C.

Rozbor:

Menšenec ve střadači A bude:

hexadecimálně	decimálně	dvojkově
A7	167	1Ø1Ø Ø111 .

Menšitel na pam. místě s adresou 6Ø1 bude:

hexadecimálně	decimálně	dvojkově
2C	44	ØØ1Ø 11ØØ .

Jednotkový doplněk menšitele bude:

11Ø1 ØØ11 .

Přičtením jedničky v nultém řádu dostaneme dvojkový doplněk menšitele ve tvaru:

11Ø1 Ø1ØØ .

Aritmetická instrukce sečte dále menšence s dvojkovým doplňkem menšitele, tedy provede:

```
  1Ø1Ø Ø111
+ 11Ø1 Ø1ØØ
-----------
1 Ø111 1Ø11 .
```

Protože střadač A má jako i kterékoliv místo v paměti 8 bitů, zbude v něm pouze číslo :

Ø111 1Ø11 ,

což je hexadecimálně 7B a decimálně 123.

Nyní se podíváme, jak budou obsazeny jednotlivé bity registru příznaků F:

7. bit - S = Ø - v 7. řádu výsledku je Ø
6. bit - Z = Ø - výsledek ve střadači není nula
5. bit - Ø - stabilní obsazení
4. bit - AC = Ø - nedošlo k přenosu ze 3. na 4. řád
3. bit - Ø - stabilní obsazení
2. bit - P = 1 - výsledek ve střadači má sudý počet jedniček
1. bit - 1 - stabilní obsazení
Ø. bit - CY = Ø - došlo při odečítání k přenosu ze 7. řádu výsledku - v tomto případě je CY = Ø.

Tedy v registru příznaků F bude číslo ØØØØ Ø11Ø, což je hexadecimálně číslo 6.

Postup:

Po zapnutí počítače přejdeme do režimu MONITOR a dále pomocí příkazu S obsadíme paměťové místo s hexadecimální adresou 6Ø1 hexadecimálním číslem 2C. Dále se pomocí příkazu X v tomtéž režimu podíváme na stávající obsazení všech registrů a zjistíme, že je rozdílné, než by mělo být po vyřešení příkladu.
Dále pomocí příkazu S v režimu MONITOR vložíme následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód	poznámka
5ØØ	MVI A, d	3E	vložení hexad. čísla A7 do A
5Ø1	A7	A7	
5Ø2	LXI H, w	21	vložení adresy 6Ø1 do dvjreg. HL
5Ø3	Ø1	Ø1	
5Ø4	Ø6	Ø6	
5Ø5	SUB M	96	odečetní v A
5Ø6	RET	C9	konec

Nyní spustíme tento program příkazem C5ØØ v režimu MONITOR a pak pomocí příkazu X v tomtéž režimu zjistíme, že ve střadači A je hexadecimální číslo 7B jako výsledek odečítání a v registru příznaků F je hexadecimální číslo 6, což souhlasí s dříve provedeným rozborem.

Nyní aniž vypnete počítač, vrátíte se do režimu BASIC příkazem R a vložíte tento jednoduchý program:

```
1 CLS
5 PRINT USR( HEX (5ØØ) )
9 END
```

Programový řádek číslo 5 vyvolá náš strojový program a po jeho provedení se počítač vrátí do stejného režimu, ze kterého vyšel a navíc napíše decimální tvar výsledku ve střadači A - viz vlastnosti funkce USR. Po startu našeho programu příkazem RUN se na čisté obrazovce objeví decimální číslo 123 a hlášení READY s blikajícím kurzorem, což je v souladu s našimi rozbory.

3.2 Aritmetické instrukce pro přičítání a odečítání čísla ve střadači se započtením příznaku přenosu

Takové instrukce pracují analogicky, jako instrukce pro obyčejné sečítání a odečítání ve střadači A, pouze ke druhému sčítanci nebo menšiteli se nejdříve přičítá hodnota příznaku přenosu CY, jaká byla před prováděním sečítací nebo odečítací instrukce. Po provedení takových aritmetických operací dojde opět k automatickému nastavení jednotlivých bitů registru příznaků F jako v minulé stati.

Druhý sčítanec, resp. menšitel může být buď jako přímý operand za instrukcí /instrukce ACI, resp. SBI/, nebo může být uložen v libovolném jednoduchém registru A, B, C, D, E, H, L /instrukce ADC, resp. SBB/, nebo může být uložen na libovolném místě v paměti, jehož adresa je uložena v dvojregistru HL /instrukce ADC M, resp. SBB M/. Ještě jednou si uvědomte, že po provedení sečítání se nastavuje příznak přenosu CY v registru příznaků F pomocí jiných pravidel; než je tomu při odečítání.

Konkrétně instrukce

ACI, 3B

znamená, že k číslu 3B bude nejdříve přičtena stávající hodnota příznaku přenosu CY - vznikne tedy číslo 3C, pokud CY bylo 1 - a teprve potom dojde ve střadači A k sečtení původního čísla a čísla 3C a k novému nastavení bitů v registru příznaků F podle výsledku sečítání.

Instrukce

SBB E

se provede tak, že se k obsahu registru E přičte nejdříve

stávající hodnota příznaku přenosu CY a pak takto upravený obsah registru E se odečte od hodnoty, která byla ve střadači, postupem podrobně ukázaným v předchozím článku.

Všechny instrukce pro sečítání nebo odečítání se započtením příznaku přenosu jsou následující:

kód	instrukce	význam
CE	ACI, d	přičtení čísla d a CY k obsahu střadače A, d∈⟨00; FF⟩
8F	ADC A	přičtení CY a čísla, které je v registru za instrukcí, k obsahu střadače A
88	ADC B	
89	ADC C	
8A	ADC D	
8B	ADC E	
8C	ADC H	
8D	ADC L	
8E	ADC M	přičtení CY a čísla, které je v paměti na adrese vložené v dvojregistru HL, k hodnotě v A
DE	SBI, d	přičtení CY k menšiteli d a odečtení takto vzniklé hodnoty od čísla ve střadači A
9F	SBB A	přičtení hodnoty CY k menšiteli, který je v registru uvedeném za instrukcí a odečtení takto vzniklého čísla od hodnoty ve střadači A
98	SBB B	
99	SBB C	
9A	SBB D	
9B	SBB E	
9C	SBB H	
9D	SBB L	
9E	SBB M	sečtení obsahu paměťového místa s adresou v HL a CY a následné odečt. od hodnoty v A

Pomocí strojového programu sečtěte hexadecimální čísla 2C9E a 3ABD.

Rozbor:

Všimněte si nejdříve, že obě čísla jsou větší, než hexadecimální číslo FF, proto nemůže být žádné z nich umístěno v jednoduchém registru, tedy například ve střadači A, kde by sečítání probíhalo. Sečítání můžeme realizovat jedině tak, že ve střadači A sečteme nejdříve nižší dva řády obou sčítanců, provedeme tedy

9E + BD ,

zjistíme výsledek a současně skutečnost, že součet těchto dvou nižších řádů přesáhne hexadecimální číslo FF, dojde tedy k přenosu ze 7. řádu výsledku. Tento přenos bude registrován pomocí příznaku přenosu CY registru příznaků F. Pak sečteme vyšší řády obou čísel, provedeme tedy

2C + 3A

a současně nesmíme zapomenout přičíst i přenos z nižších dvou řádů, tedy nutno přičíst ještě 1 jakožto příznak přenosu CY z předchozího sečítání nižších řádů. To nám právě zajistí instrukce ACI, která do výsledku započítává příznak přenosu CY z minulé operace.
Instrukce ADI by při sečítání vyšších dvou řádů vedla k chybnému výsledku v důsledku zanedbání přenosu při sečítání dvou nižších řádů.
Abychom mohli zobrazit celý výsledek, přesuneme výsledek součtu nižších řádů do registru L a součet vyšších řádů včetně přenosu do registru H a pak zobrazíme obsah dvojregistru HL.

Konkrétně:

sečítání nižších řádů:

1. sčítanec

hexadecimálně	dvojkově
9E	1ØØ1 111Ø

2. sčítanec

hexadecimálně	dvojkově
BD	1Ø11 11Ø1

součet

hexadecimálně	dvojkově
15B	1 Ø1Ø1 1Ø11

Bity registru příznaků F budou po tomto sčítání nastaveny takto:

7. bit - S = Ø - na 7. bitu výsledku je Ø
6. bit - Z = Ø - výsledek není nula
5. bit - Ø - stabilní obsazení
4. bit - AC = 1 - došlo k přenosu ze 3. do 4. řádu
3. bit - Ø - stabilní obsazení
2. bit - P = Ø - lichý počet jedniček ve výsledku, který je ve střadači A
1. bit - 1 - stabilní obsazení
Ø. bit - CY = 1 - při sečítání došlo k přenosu ze 7. řádu výsledku ve střadači A

V registru příznaků F bude po provedení tohoto sečí-

tání číslo ØØØ1 ØØ11, což je hexadecimálně 13 a decimálně 19. Výsledek sečítání bude přesunut do registru L.

sečítání vyšších řádů:

1. sčítanec

hexadecimálně	dvojkově
2C	ØØ1Ø 11ØØ

2. sčítanec

hexadecimálně	dvojkově
3A	ØØ11 1Ø1Ø

přenos

hexadecimálně	dvojkově
Ø1	ØØØØ ØØØ1

součet

hexadecimálně	dvojkově
67	Ø11Ø Ø111

K přenosu ze 7. řádu dvojkového tvaru výsledku tohoto sečítání nedošlo. Jednotlivé bity registru příznaků F jsou nastaveny po uvedené operaci takto:

7. bit - S = Ø - na 7. bitu výsledku je Ø
6. bit - Z = Ø - výsledek není nula
5. bit - Ø - stabilní obsazení
4. bit - AC = 1 - došlo k přenosu ze 3. do 4. řádu při sečítání
3. bit - Ø - stabilní obsazení
2. bit - P = Ø - výsledek má lichý počet jedniček
1. bit - 1 - stabilní obsazení
Ø. bit - CY = Ø - při sčítání nedošlo k přenosu ze 7. řádu

V registru příznaků F je nyní číslo ØØØ1 ØØ1Ø, což je hexadecimálně 12 a decimálně 18. Výsledek tohoto sečítání bude přesunut do registru H.
Pak tedy v dvojregistru HL je uložen výsledek celého sčítání ve tvaru:

decimálně	hexadecimálně
26459	675B

dvojkově

Ø11Ø Ø111 Ø1Ø1 1Ø11

Postup:

Po zapnutí počítače vložíme v režimu BASIC následující jednoduchý program:

```
5 CLS
1Ø PRINT WORD(HEX(7ØØ))
15 END
```

Druhý programový řádek číslo 1Ø spustí strojový program, který bude v paměti uložen od hexadecimální adresy 7ØØ a po proběhnutí strojového programu vytiskne výsledný obsah dvojregistru HL v desítkovém tvaru na obrazovku. Zopakujte si při této příležitosti význam a funkci slova WORD v jazyce BASIC.
Nyní pomocí tlačítka BR přejdeme do režimu MONITOR a pomocí příkazu S v tomto režimu vložíme následující strojový program do paměti od hexadecimální adresy 7ØØ:

adresa	instrukce	kód	význam
7ØØ 7Ø1	MVI A, d 9E	3E 9E	vložení nižších řádů 1. sčítance do střadače A
7Ø2 7Ø3	ADI, d BD	C6 BD	přičtení nižších řádů 2. sčítance ke střadači A
7Ø4	MOV L, A	6F	přesun součtu nižších řádů do L
7Ø5 7Ø6	MVI A, d 2C	3E 2C	vložení vyšších řádů 1. sčítance do střadače A
7Ø7 7Ø8	ACI, d 3A	CE 3A	přičtení vyšších řádů 2. sčítance a přenosu k A
7Ø9	MOV H, A	67	přesun součtu vyšších řádů do H
7ØA	RET	C9	konec

Nyní se pomocí příkazu R vrátíme do režimu BASIC a spustíme náš program příkazem RUN v tomtéž režimu. Po jeho proběhnutí se na obrazovce objeví očekávané decimální číslo 26459.

3.3 Inkrement, resp. dekrement

1/ Pomocí instrukcí tohoto typu můžeme zvyšovat /instrukce INR/, resp. snižovat /instrukce DCR/ číslo, které je v libovolném jednoduchém registru A, B, C, D, E, H, L nebo na libovolném paměťovém místě, jehož adresa je uložena v dvojregistru HL, o jedničku.

Například instrukce

INR B

zvýší číslo v registrú B o jednotku, příkaz

DCR H

sníží číslo v registru H o jednotku a instrukce

INR M

zvýší o jednotku číslo uložené na paměťovém místě, jehož adresa je uložena v dvojregistru HL.

Všechny instrukce uvedených typů jsou následující:

kód	instrukce	význam
3C	INR A	zvýšení obsahu registru uvedeného za instrukcí o jednotku
Ø4	INR B	
ØC	INR C	
14	INR D	
1C	INR E	
24	INR H	
2C	INR L	
34	INR M	zvýšení čísla na paměťovém místě, jehož adresa je uložena v HL, o jednotku

kód	instrukce	význam
3D	DCR A	snížení obsahu registru uvedeného za instrukcí o jednotku
Ø5	DCR B	
ØD	DCR C	
15	DCR D	
1D	DCR E	
25	DCR H	
2D	DCR L	
35	DCR M	snížení čísla na pam. místě, jehož adresa je v HL, o jednotku

Všechny uvedené instrukce nastavují jednotlivé bity registru příznaků F kromě příznaku přenosu CY, i když se nejedná pouze o střadač A.

Umístěte v registru C hexadecimální číslo EF a pak jej zvyšte o jednotku.

Rozbor:

Hexadecimální číslo v registru C bude mít před zvýšením o jednotku dvojkový tvar 111Ø 1111. Po zvýšení tohoto čísla o jednotku, tedy po přičtení ØØØØ ØØØ1, bude mít dvojkový výsledek tvar 1111 ØØØØ, bude to hexadecimálně FØ a bude uložen opět v registru C. V registru příznaků F je po zapnutí počítače hexadecimální číslo 86, což je číslo sudé, proto jeho nultý bit /příznak CY/ je obsazen nulou.

Po provedení instrukce INR C budou jednotlivé bity registru příznaků F nastaveny takto:

7. bit - S = 1 - na 7. bitu výsledku je cifra 1
6. bit - Z = Ø - výsledek není nula
5. bit - Ø - stabilní obsazení
4. bit - AC = 1 - došlo k přenosu ze 3. na 4. řád
3. bit - Ø - stabilní obsazení
2. bit - P = 1 - výsledek má sudý počet jedniček
1. bit - 1 - stabilní obsazení
Ø. bit - CY = Ø - instrukce INR nemění příznak CY

Tedy v registru příznaků bude číslo 1ØØ1 Ø11Ø, což je hexadecimálně 96.

Postup:

Po zapnutí počítače zjistíme pomocí příkazu X v režimu MONITOR, že registr C je obsazen hexadecimálním číslem EC a registr příznaků F hexadecimálním číslem 86. Nyní vložíme pomocí příkazu S v režimu MONITOR následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód	význam
5ØØ	MVI C, d	ØE	vložení čísla EF do reg. C
5Ø1	EF	EF	
5Ø2	INR C	ØC	zvýšení čísla v reg. C o 1
5Ø3	RET	C9	konec

Spustíme tento program pomocí příkazu C5ØØ v režimu MONITOR a pak se pomocí příkazu X v tomtéž režimu přesvědčíme, že nyní je v registru C hexadecimální

číslo FØ a v registru příznaků F hexadecimální číslo 96, což je ve shodě s předchozím rozborem.

Do registru C vložte hexadecimální číslo FF a zvyšte jej o jednotku.

Rozbor:

Dvojkový tvar čísla FF je 1111 1111, pokud toto číslo uložené v jednoduchém registru zvýšíme o jednotku, dostaneme výsledek ØØØØ ØØØØ s tím, že došlo k přenosu ze 7. řádu dvojkového tvaru výsledku. To ale nebude zaznamenáno na nultém bitu CY v registru příznaků - jak by tomu bylo v případě instrukcí pro sečítání - protože instrukce INR nepůsobí na jmenovaný bit. Obsazení jednotlivých bitů registru příznaků F bude následující:

7. bit - S = Ø - na 7. bitu výsledku je Ø
6. bit - Z = 1 - výsledek je nula
5. bit - Ø - stabilní obsazení
4. bit - AC = 1 - došlo k přenosu ze 3. na 4. řád
3. bit - Ø - stabilní obsazení
2. bit - P = 1 - sudý počet jednotek výsledku /nulový/
1. bit - 1 - stabilní obsazení
Ø. bit - CY = Ø - po zapnutí počítače je CY = Ø a instrukce INR tuto hodnotu nemění

Tedy v registru příznaků bude dvojkové číslo Ø1Ø1 Ø11Ø, což je hexadecimálně 56.

Postup:

Z minulého příkladu máte v počítači vložen strojový program. V režimu MONITOR pomocí příkazu S změníme pouze hexadecimální číslo EF na hexadecimální číslo FF /na hexadecimální adrese 5Ø1/. Takto upravený program spustíme pomocí příkazu C5ØØ v režimu MONITOR a pak se pomocí příkazu X v tomtéž režimu podíváme, že v registru C je nyní číslo ØØ a v registru příznaků je hexadecimální číslo 56, což souhlasí s předchozím rozborem.

2/ Pomocí instrukcí INX, resp. DNX můžeme zvyšovat, resp. snižovat o jednotku číslo, které je uloženo v některém dvojregistru BC, DE, HL, nebo v registru SP.

Například instrukce

INX D

zvýší číslo, které je uloženo v dvojregistru DE o jednotku, instrukce

DCX H

sníží číslo, které je uloženo v dvojregistru HL o jednotku. Uvědomte si při této příležitosti, že v dvojregistrech nebo v registru SP jsou celá hexadecimální čísla z intervalu < ØØØØ; FFFF >

Všechny dále uvedené instrukce typu INX a DCX nemění hodnotu žádného bitu v registru příznaků F.

Všechny instrukce uvedených typů jsou v přehledu uvedeny v následující tabulce.

kód	instrukce	význam
Ø3 13 23	INX B INX D INX H	zvýšení čísla v dvojregistru, jehož první písmeno je za instrukcí, o jednotku
33	INX SP	zvýšení v SP o 1
ØB 1B 2B	DCX B DCX D DCX H	snížení čísla v dvojregistru, jehož první písmeno je za instrukcí, o jednotku
3B	DCX SP	snížení v SP o 1

Příklad:

Pomocí programu ve strojovém kódu napište na čistou obrazovku vedle sebe písmena A, B, C, D, E na druhou polovinu prvního řádku.

Rozbor:

Hexadecimální kódy znaků A, B, C, D a E jsou po řadě 41, 42, 43, 44 a 45. Abychom splnili náš úkol, musíme hexadecimálním číslem 41 obsadit paměťové místo s hexadecimální adresou EC1Ø /šestnácté místo oblasti VIDEORAM/, adresa o jednotku větší má být obsazena číslem rovněž o jednotku větším atd., až hexadecimální číslo 45 bude umístěno na paměťovém místě s hexadecimální adresou EC14. Na šestnáctém místě oblasti VIDEORAM - tedy vlastně na prvním místě druhé poloviny prvního řádku obrazovky - se neobjeví hexadecimální číslo 41, ale znak A, jehož hexadecimálním kódem je 41. Obdobně se na následujících místech

druhé poloviny prvního řádku budou objevovat další písmena v souladu se zadáním úlohy.
Zvyšování hexadecimálních kódů jednotlivých znaků v rozmezí 41 až 45 bude probíhat ve střadači A pomocí instrukce INR A, zvyšování adresy v rozmezí EC1Ø až EC14 bude probíhat v dvojregistru HL pomocí instrukce INX H. Překopírováním obsahu střadače A na požadovaná paměťová místa splníme podmínky příkladu. Použijeme k tomu instrukce MOV M, A.

Postup:

Po zapnutí počítače vložíme v režimu BASIC následující jednoduchý program:

```
3 CLS
5 CALL HEX(6ØØ)
7 END
```

Tento program nejdříve vyčistí obrazovku - viz první programový řádek s číslem 3, pak vyvolá strojový program, který budeme mít vložen do paměti počítače od hexadecimální adresy 6ØØ - viz programový řádek číslo 5 - a nakonec chod počítače zastaví - viz poslední programový řádek s číslem 7.
Nyní pomocí tlačítka BR přejdeme do režimu MONITOR a vložíme do paměti následující strojový program od hexadecimální adresy 6ØØ:

adresa	instrukce	kód	význam
6ØØ 6Ø1	MVI A, d 41	3E 41	obsazení střadače hex. číslem 41
6Ø2 6Ø3 6Ø4	LXI H, w 1Ø EC	21 1Ø EC	obsazení dvojreg. HL hex. číslem EC1Ø
6Ø5	MOV M, A	77	překopírování obsahu střadače A na pam. místo s adr. EC1Ø
6Ø6 6Ø7	INR A INX H	3C 23	zvýšení čísla ve střadači a dvojreg. HL o jednotku
6Ø8	MOV M, A	77	překopírování obsahu střadače A na pam. místo s adr. EC11
6Ø9 6ØA	INR A INX H	3C 23	zvýšení čísla ve střadači a dvojreg. HL o jednotku
6ØB	MOV M, A	77	překopírování obsahu střadače A na pam. místo s adr. EC12
6ØC 6ØD	INR A INX H	3C 23	zvýšení čísla ve střadači a dvojreg. HL o jednotku
6ØE	MOV M, A	77	překopírování obsahu střadače A na pam. místo s adr. EC13
6ØF 61Ø	INR A INX H	3C 23	zvýšení čísla ve střadači A a dvojreg. HL o jednotku
611	MOV M, A	77	překopírování obsahu střadače A na pam. místo s adr. EC14

Dokončení tabulky:

adresa	instrukce	kód	význam
612	RET	C9	konec

Nyní se pomocí tlačítka R vrátíme do režimu BASIC a spustíme program příkazem RUN. Vidíme, že program vyhovuje podmínkám úlohy.

3.4 Sečítání obsahů dvojregistrů

Obsah libovolného dvojregistru BC, DE, HL a SP je možno přičíst k obsahu dvojregistru HL pomocí instrukcí typu DAD, za níž je vždy uvedeno první písmeno příslušného dvojregistru, jehož obsah bude přičten k obsahu dvojregistru HL.

Například instrukce

DAD B

přičte k obsahu dvojregistru HL obsah dvojregistru BC a výsledek uloží do dvojregistru HL.

Všechny instrukce jmenovaného typu jsou následující:

kód	instrukce	význam
Ø9 19 29	DAD B DAD D DAD H	přičtení obsahu dvojreg., jehož první písmeno je za instrukcí,k obsahu HL
39	DAD SP	přičtení obsahu registru SP k obsahu dvojregistru HL

Instrukce typu DAD ovlivňuje pouze hodnotu příznaku přenosu CY v registru příznaků F a to tak, že CY = 1, když součet v dvojregistru HL přesáhne hexadecimální hodnotu

FFFF, tedy když dojde k přenosu ze 7. bitu vyššího registru H dvojregistru HL. CY = Ø, jestliže k přenosu ze 7. bitu registru H při sečtení nedošlo.

Příklad:

Proveďte sečtení hexadecimálních čísel A1C5 a B516.

Rozbor:

Číslo A1C5 umístíme do dvojregistru HL, číslo B516 do dvojregistru BC. Instrukce DAD B provede následující sečtení:

```
  A 1 C 5
  B 5 1 6
-----------
  1 5 6 D B
```

Vidíme, že nastal přenos ze 7. řádu vyššího registru H dvojregistru HL, tedy CY = 1. V dvojregistru HL bude po sečtení číslo 56DB.

Postup:

Po zapnutí počítače pomocí příkazu X v režimu MONITOR zjistíme stávající obsazení dvojregistrů HL a BC a skutečnost, že v registru příznaků F je sudé číslo, což znamená, že CY = Ø. Nyní pomocí příkazu S v tomtéž režimu vložíme od hexadecimální adresy 8ØØ strojový program uvedený na následující stránce.
Spustíme tento program pomocí příkazu C8ØØ v režimu MONITOR a pak pomocí příkazu X v tomtéž režimu zjistíme, že v dvojregistru HL je hexadecimální číslo 56DB, což je nultý až třetí řád součtu obou čísel a v registru F je nyní liché číslo, tedy CY = 1.

Výpočet čtvrtého a pátého řádu výsledku by šel nyní realizovat pomocí instrukcí přičítajících ke střadači příznak přenosu CY - tedy např. pomocí ACI, ØØ . Pokud by byl střadač před operací vynulován, pak v důsledku instrukce ACI, ØØ se v něm objeví číslo Ø1, což je čtvrtý a pátý řád výsledku našeho sečítání.

Tabulka strojového programu pro úlohu:

adresa	instrukce	kód	význam
8ØØ	LXI H, w	21	vložení čísla A1C5 do HL
8Ø1	C5	C5	
8Ø2	A1	A1	
8Ø3	LXI B, w	Ø1	vložení čísla B516 do BC
8Ø4	16	16	
8Ø5	B5	B5	
8Ø6	DAD B	Ø9	přičtení obsahu dvojreg. BC k obsahu dvojr. HL
8Ø7	RET	C9	konec

3.5 Konverze obsahu střadače

Pomocí běžných operací můžeme sečítat ve střadači A čísla ve dvojkové soustavě, která se pak při výpisu v režimu MONITOR objevují v hexadecimálním tvaru. Je-li tedy například ve střadači číslo Ø1ØØ 1Ø11, objeví se na obrazovce po příkazu X v režimu MONITOR v registru A hexadecimální číslo 4B. Čísla do střadače nebo do jiných registrů jsme zatím vkládali pouze v hexadecimálním tvaru, výsledky jsme dostávali rovněž v hexadecimálním tvaru, pokud jsme nepoužili některé slovo z dvojice WORD a USR v režimu BASIC.

Instrukce, s níž se nyní seznámíme, nám bude umožňovat vkládat do registru A čísla v decimálním tvaru a výsledek ve střadači získat opět jako decimální číslo. Přeměnu /konverzi/ výsledku operace ve střadači A na decimální tvar zajišťuje instrukce:

DAA s kódem 27 ,

která se provádí takto:

1/ Jestliže je hodnota 4 nižších řádů dvojkového tvaru výsledku ve střadači větší než decimální číslo 9, nebo je-li příznak pomocného přenosu AC = 1, pak instrukce přičte k těmto čtyřem nižším řádům výsledku dvojkové číslo Ø11Ø, což je decimální číslo 6.

2/ Jestliže je dále hodnota 4 vyšších řádů dvojkového tvaru výsledku ve střadači A větší než decimální číslo 9, nebo je-li příznak přenosu CY = 1, pak přičte k těmto vyšším čtyřem řádům dvojkové číslo Ø11Ø, což je opět decimální číslo 6.

Instrukce DAA mění dále všechny bity registru příznaků F podle známých pravidel.

Najděte decimální tvar součtu dvou decimálních čísel 38 a 29.

Rozbor:

Číslo 38 vožíme do střadače A - toto vložení proběhne tak, jako kdyby toto číslo bylo hexadecimální - tedy jednotlivé bity střadače budou obsazeny takto:

ØØ11 1ØØØ .

Druhé číslo 29 přičteme k obsahu střadače A jako přímý operand pomocí instrukce ADI. Rovněž i toto číslo bude převzato počítačem jako číslo hexadecimální, proto bude vlastně přičteno číslo, jehož dvojkový tvar je

ØØ1Ø 1ØØ1 .

Prostřednictvím instrukce ADI proběhne toto sečtení:

```
  ØØ11 1ØØØ
  ØØ1Ø 1ØØ1
-------------
  Ø11Ø ØØØ1 ,
```

což je hexadecimální číslo 61.

Všimněte si, že při sečítání došlo k přenosu ze 3. do 4. řádu výsledku, tedy příznak pomocného přenosu AC = 1. Příznak přenosu CY = Ø. Sami si rozmyslete, že nikdy nemůže nastat případ, kdy AC = 1 a současně nižší 4 řády výsledku dávají větší číslo, než decimální číslo 9. Provedeme-li nyní instrukci DAA, protože AC = 1, přičte se ke 4 nižším řádům výsledku Ø11Ø, tedy:

```
  Ø11Ø ØØØ1
     Ø11Ø
-----------
  Ø11Ø Ø111 .
```

Protože 4 vyšší řády výsledku nepřesahují decimální číslo 9 a CY = Ø, působení instrukce DAA tímto končí. Výsledek ve střadači má nyní hexadecimální tvar 67, což je ale rovněž decimální tvar výsledku našeho sečítání decimálních čísel 38 a 29.

Postup:

Po zapnutí počítače vložíme v režimu MONITOR pomocí příkazu S následující strojový program od hexadecimální adresy 65Ø:

adresa	instrukce	kód	význam
65Ø	MVI A, d	3E	vložení dec. čísla 38 do střadače A
651	38	38	
652	ADI, d	C6	přičtení dec. čís. 29 ke střadači A
653	29	29	
654	DAA	27	konverze výsledku
655	RET	C9	konec

Tento program spustíme příkazem C65Ø v režimu MONITOR a pak se pomocí příkazu X v tomtéž režimu přesvědčíme, že ve střadači A je výsledek 67, což odpovídá decimálnímu tvaru výsledku sečítání.

4. Logické instrukce

Tyto instrukce provádějí tzv. logické operace s obsahem střadače A a nejčastěji s nějakým jiným číslem, které může být buď za instrukcí jako přímý operand, nebo v některém jednoduchém registru, nebo na libovolném paměťovém místě, jehož hexadecimální adresa je uložena v dvojregistru HL. Logické instrukce jsou následujících typů:

1/ Konjunkce

Konjunkce dosadí do určitého bitu výsledku jedničku, právě když na bitech odpovídajících si řádů obou porovnávaných čísel jsou jedničky. Jinak dosazuje nulu.

Příklad:

první číslo :	1Ø1Ø ØllØ
druhé číslo :	Ølll 1Ø11
konjunkce :	ØØ1Ø ØØ1Ø

2/ Disjunkce

Disjunkce dosadí v určitém bitu výsledku jedničku, jestliže alespoň v jednom čísle je na bitu odpovídajícího řádu jednička. Jinak dosazuje nuly.

Příklad:

první číslo :	11ØØ Ø1Ø1
druhé číslo :	Ø1ØØ 11ØØ
disjunkce :	11ØØ 11Ø1

3/ Nonekvivalence

Nonekvovalence dosadí v určitém bitu výsledku jedničku, je-li hodnota právě jediného bitu odpovídajících si řádů obou čísel rovna jedné. Jinak dosazuje nuly.

Příklad:

první číslo :	Ølll ØØ1Ø
druhé číslo :	11Ø1 Ø11Ø
nonekvivalence :	1Ø1Ø Ø1ØØ

4/ Negace

Negace vytváří jednotkový doplněk obsahu střadače - záměnní nuly za jedničky a naopak.

Příklad:

číslo v A :	Ø11Ø 11Ø1
negace:	1ØØ1 ØØ1Ø

Výsledky všech logických operací se ukádají opět do střadače A.

Logické operace nastavují hodnoty jednotlivých bitů registru příznaků F kromě instrukce pro negaci. Lze snadno ukázat, že po provedení libovolné logické instrukce je vždy příznak pomocného přenosu AC = Ø a příznak přenosu CY = Ø. Následuje přehled logických instrukcí podle jejich významu.

4.1 Instrukce pro konjunkci

Jsou ve tvaru ANA nebo ANI, v prvním případě následuje za instrukcí písmeno označující jednoduchý registr, v němž se druhé číslo nachází, nebo symbol M. Ve druhém případě následuje druhé porovnávané číslo jako přímý operand za instrukcí. Všechny instrukce uvedených typů jsou následující:

kód	instrukce	význam
A7	ANA A	provádí logickou konjunkci obsahu střadače A s číslem, které je uloženo v jednoduchém registru uvedeném za instrukcí
AØ	ANA B	
A1	ANA C	
A2	ANA D	
A3	ANA E	
A4	ANA H	
A5	ANA L	
A6	ANA M	provádí logickou konjunkci obsahu střadače A s číslem, které je na pam. místě, jehož adresa je v dvojr. HL
E6	ANI, d	provádí logickou konjunkci obsahu střadače A s číslem d uvedeným za instrukcí; d je hexad. z intervalu $\langle ØØ; FF \rangle$

Například tedy instrukce

ANA L

znamená, že bude provedena logická konjunkce obsahu střadače s obsahem registru L, instrukce

ANI, 7E

znamená, že bude provedena logická konjunkce obsahu střadače a čísla 7E.

4.2 Instrukce pro disjunkci

Jsou tvaru ORA nebo ORI, za první musí být buď jednoduchý registr nebo symbol M, za druhou přímý operand jako v minulém případě, přehled všech instrukcí je následující:

kód	instrukce	význam
B7	ORA A	provádí logickou disjunkci obsahu střadače a čísla, které je v jednoduchém reg. uvedeném za instrukcí
BØ	ORA B	
B1	ORA C	
B2	ORA D	
B3	ORA E	
B4	ORA H	
B5	ORA L	
B6	ORA M	provádí logickou disjunkci obsahu střadače A a čísla, které je na pam místě, jehož adresa je v dvojr. HL
F6	ORI, d	provádí logickou disjunkci obsahu střadače A s číslem, které je uvedeno za instrukcí $d \in \langle ØØ; FF \rangle$

Například instrukce

ORA C

znamená, že bude provedena logická disjunkce obsahu střadače a čísla, které je uloženo v registru C, instrukce

ORI,E5

znamená, že bude provedena logická disjunkce obsahu střadače a čísla E5.

4.3 Instrukce pro nonekvivalenci

Jsou tvaru XRA nebo XRI, za první musí být buď jednoduchý registr nebo symbol M, za druhou přímý operand, jako v minulých případech. Přehled všech instrukcí je následující:

kód	instrukce	význam
AF	XRA A	provádí logickou nonekvivalenci obsahu střadače A a čísla, které je v registru uvedeném za instrukcí
A8	XRA B	
A9	XRA C	
AA	XRA D	
AB	XRA E	
AC	XRA H	
AD	XRA L	
AE	XRA M	provádí logickou nonekvivalenci obsahu střadače A a čísla, které je na pam. místě, jehož adresa je v HL
EE	XRI, d	provádí logickou nonekvivalenci obsahu střadače A a čísla, které je za instrukcí uvedeno - d∈⟨00; FF⟩

Například instrukce

XRA M

znamená, že bude provedena logická nonekvivalence obsahu střadače a čísla, které je na paměťovém místě, jehož adresa je v době provádění instrukce v dvojregistru HL.

4.4 Instrukce pro negaci

Je to pouze jediná instrukce

CMA s kódem 2F ,

která provede záměnu jedniček za nuly a naopak ve všech bitech střadače A.

Upozornění: Tato instrukce narozdíl od předchozích logických instrukcí nemění hodnotu na žádném bitu registru příznaků F.

Příklad

Proveďte všechny logické operace na hexadecimálních číslech D6 a 72, výsledek konjunkce vložte do registru B, výsledek disjunkce do registru C, výsledek nonekvivalence vložte do registru D a negaci čísla D6 ponechte ve střadači.

Rozbor:

Hexadecimální čísla D6 a 72 si napíšeme ve dvojkovém tvaru a pak budeme provádět jednotlivé logické operace a současně převádět výsledky do hexadecimálního tvaru.

1. číslo	11Ø1 Ø11Ø	D6
2. číslo	Ø111 ØØ1Ø	72
konjunkce	Ø1Ø1 ØØ1Ø	52
disjunkce	1111 Ø11Ø	F6
nonekvival.	1Ø1Ø Ø1ØØ	A4
negace D6	ØØ1Ø 1ØØ1	29

Vytvořená čísla budeme přesunovat do předepsaných jednoduchých registrů pomocí přesunových instrukcí typu MOV.

Postup:

Po zapnutí počítače vložíme v režimu MONITOR pomocí příkazu S od hexadecimální adresy 7ØØ následující strojový program:

adresa	instrukce	kód	poznámka
7ØØ 7Ø1 7Ø2 7Ø3	MVI H, d D6 MVI L, d 72	26 D6 2E 72	vložení hexadec. čísel D6, resp. 72 do registru H, resp. L
7Ø4	MOV A, H	7C	přesun čísla D6 do střadače A
7Ø5	ANA L	A5	vytvoření konjunkce čísel ze střadače a reg. L ve střadači A

adresa	instrukce	kód	poznámka
7Ø6	MOV B, A	47	přesun výsledku do B
7Ø7	MOV A, H	7C	přesun čísla D6 znovu do A
7Ø8	ORA L	B5	vytvoření disjunkce čísel ze střadače a reg. L ve střadači
7Ø9	MOV C, A	4F	přesun výsledku do C
7ØA	MOV A, H	7C	přesun čísla D6 znovu do A
7ØB	XRA L	AD	vytvoření nonekv. čísel ze střadače a reg L ve střadači
7ØC	MOV D, A	57	přesun výsledku do D
7ØD	MOV A, H	7C	přesun čísla D6 znovu do A
7ØE	CMA	2F	vytvoření negace čísla v A
7ØF	RET	C9	konec

Spustíme tento program pomocí příkazu C7ØØ v režimu MONITOR a pomocí příkazu X v tomtéž režimu zjistíme, že nynější obsazení jednotlivých registrů souhlasí s předcházejícím rozborem. Pokuste se sami zdůvodnit číslo, které je nyní v registru příznaků F.

5. Instrukce pro posuvy obsahů bitů střadače

Pomocí těchto instrukcí lze posunout obsahy jednotlivých bitů střadače A o jedno místo vpravo nebo vlevo, přičemž u jednoho typu instrukcí se posuvu účastní i příznak přenosu CY - tedy poslední bit registru příznaků F. Instruk-

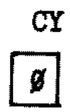
ce jsou celkem 4 - RLC, RRC, RAL a RAR. Všechny uvedené instrukce mění pouze příznak přenosu v registru příznaků F.

5.1 Instrukce RLC

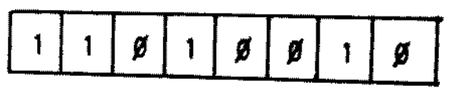
Instrukce při svém působení nejprve přepíše původní obsah příznaku přenosu hodnotou, která je na 7. bitu střadače. Potom posune obsah celého střadače o 1 bit vlevo a obsah 7. bitu přitom přejde na nultý bit. Instrukce je

RLC s kódem Ø7 .

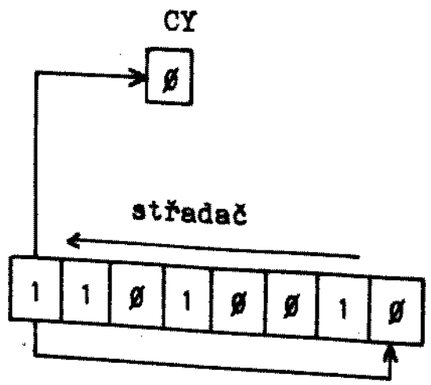
Výchozí stav:



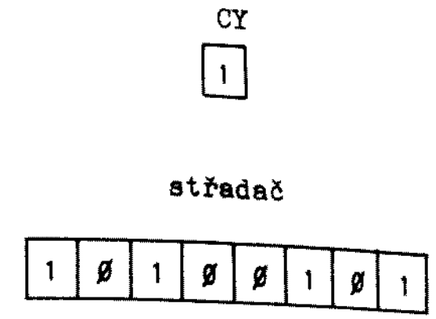
střadač



Provádění instrukce RLC:



Konečný stav:

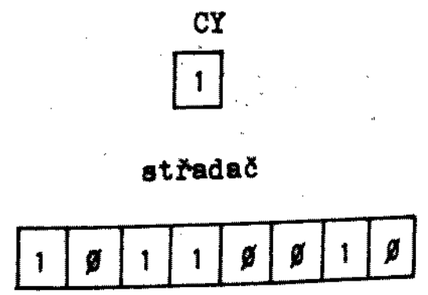


5.2 Instrukce RRC

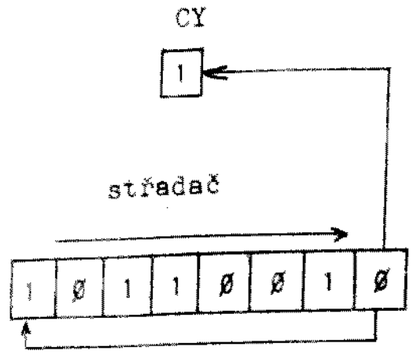
Instrukce při svém působení nejprve přepíše původní obsah příznaku přenosu hodnotou, která je na nultém bitu střadače. Potom posune obsah celého střadače o 1 bit vpravo a přitom obsah nultého bitu přejde na 7. bit střadače. Instrukce je:

RRC s kódem ØF .

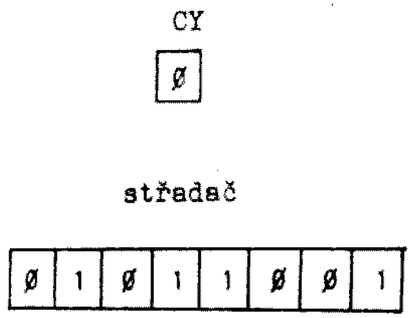
Výchozí stav:



Provádění instrukce RRC:



Konečný stav:

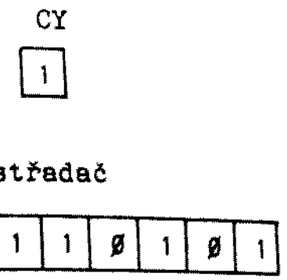


5.3 Instrukce RAL

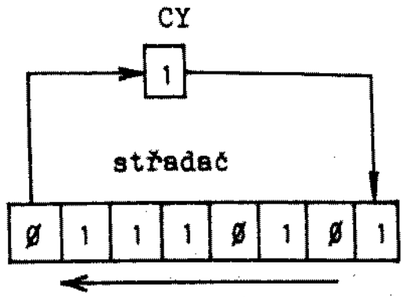
Tato instrukce působí obdobně jako RLC s tím rozdílem, že se kruhového posuvu vlevo účastní i nultý bit registru příznaků F. Názorně nám to ukáže následující příklad. Instrukce je

RAL s kódem 17 .

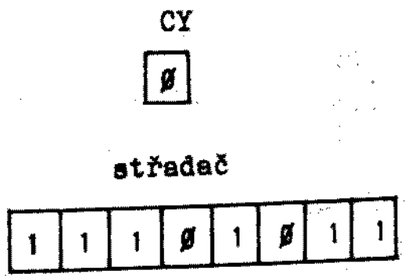
Výchozí stav:



Provádění instrukce RAL:



Konečný stav:



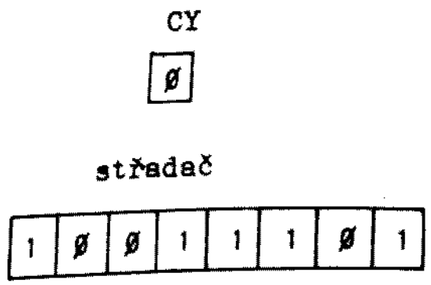
5.4 Instrukce RAR

Tato instrukce způsobí kruhový posuv obsahů jednotlivých bitů střadače a posledního bitu registru příznaků F opačným směrem, než předchozí instrukce RAL. Názorně nám to ukáže opět následující příklad. Instrukce je

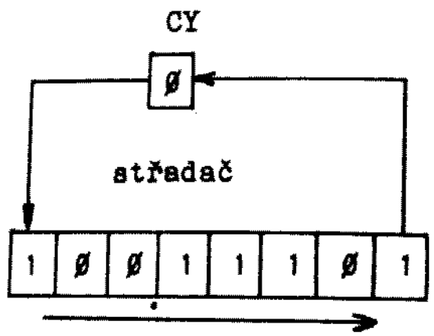
RAR s kódem 1F .

Příklad:

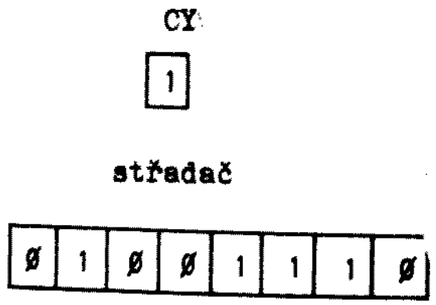
Výchozí stav:



Provádění instrukce RAR:



Konečný stav:



Poznámka:

Všimněte si, že dosadíte-li například do střadače číslo Ø1, jehož dvojkový tvar je ØØØØ ØØØ1 a příznak přenosu nastavíte roven Ø /což lze zajistit pomocí STC a CMC/, pak po provedení RAL dostaneme ve střadači hexadecimálně Ø2,

po další instrukci RAL hexadecimální číslo Ø4, pak Ø8, pak hexadecimálně 1Ø, pak hexadecimálně 2Ø, pak 4Ø, pak 8Ø a provedeme-li instrukci RAL poosmé, dostaneme ve střadači nulu a jedině nyní bude CY = 1.

6. Podmíněný a nepodmíněný skok na určitou adresu v paměti

Při programování v jazyce BASIC známe funkci příkazu GOTO, který způsobí nepodmíněný skok na programový řádek, jehož číslo je za příkazem uvedeno. Rovněž z programování v jazyce BASIC známe podmíněný skok na určitý programový řádek, který nastane, pokud je splněna určitá podmínka. Takový příkaz je ve tvaru:

IF (výrok) THEN GOTO (číslo řádku) .

Je-li výrok za IF pravdivý, nastane skok na programový řádek, jehož číslo je uvedeno za příkazem GOTO. V opačném případě - není-li výrok pravdivý - k přeskoku nedojde.

Obdobné skokové instrukce existují i při programování ve strojovém kódu. Umožňují různá větvení strojových programů. Příslušné skokové instrukce pro strojový program nemohou obsahovat pochopitelně číslo nějakého programového řádku - programové řádky ve strojovém programu neexistují - ale týkají se nějaké adresy, na níž začíná určitá část strojového programu.

Skok na určitou adresu může být nepodmíněný, což zajišťuje instrukce JMP, za níž je uvedena příslušná adresa. Adresa je vždy hexadecimální číslo z intervalu ⟨ØØØØ:, FFFF⟩. V programech se uvádí známým způsobem - nejdříve dva nižší řády, pak dva vyšší řády.

Skok na danou adresu může být i podmíněný; nastane pouze tehdy, je-li splněna určitá podmínka. Takové instrukce jsou například JC, JNC, JZ a další. Není-li podmínka splněna, skoková instrukce se neprovede a počítač zpracovává další instrukci, která za skokovou následuje.

Jaké je možno při strojovém programování uvádět podmínky? Budou pochopitelně jiné, než jsme zvyklí z programování v jazyce BASIC. Podmínky vycházejí z testování jednotlivých příznaků v registru F, zjišťuje se tedy, zda například je příznak znaménka roven jedné, zda je příznak přenosu roven nule apod.

Důležité upozornění:

Žádná skoková instrukce nemění při svém působení hodnotu žádného bitu v registru příznaků F.

Všechny skokové instrukce jsou následující - a znamená adresu, která je hexadecimální z intervalu ⟨ØØØØ; FFFF⟩ .

kód	instrukce	význam
C3	JMP a	nepodmíněný skok na adr. a
DA	JC a	je-li CY = 1, skok na adr. a
D2	JNC a	je-li CY = Ø, skok na adr. a
CA	JZ a	je-li Z = 1, skok na adr. a
C2	JNZ a	je-li Z = Ø, skok na adr. a
FA	JM a	je-li S = 1, skok na adr. a
F2	JP a	je-li S = Ø, skok na adr. a
EA	JPE a	je-li P = 1, skok na adr. a
E2	JPO a	je-li P = Ø, skok na adr. a

Naplňte pomocí strojového programu pravou polovinu obrazovky písmenem A.

Rozbor:

Protože celá televizní obrazovka zobrazuje část paměti od hexadecimální adresy ECØØ až do EFFF a má 32 řádků po 32 znacích /poslední dva údaje decimální/, snadno lze vypočítat, že druhá polovina horního řádku má hexadecimální adresy v rozmezí EC1Ø až EC1F, druhá polovina následujícího /druhého/ řádku má hexadecimální adresy v rozmezí od EC3Ø do EC3F atd. Takových polovin řádků bude celkem 32 decimálně, což je 2Ø hexadecimálně.

Adresy míst, která mají být obsazena hexadecimálním číslem 41 /jakožto kódem znaku A/, budou uchovávány v dvojregistru HL. Přechod k následujícímu místu na obrazovce se bude realizovat zvýšením obsahu dvojregistru HL o jednotku pomocí instrukce INX H, je-li to následující místo na témže řádku obrazovky a přičtením hexadecimálního čísla 1Ø, jestliže se z konce předchozího řádku máme dostat do poloviny řádku následujícího. To se bude realizovat přičtením obsahu dvojregistru BC k obsahu dvojregistru HL pomocí instrukce DAD B, přičemž na počátku obsazování každého řádku bude dvojregistr BC vynulován a při tisku každého písmena A vedle sebe na řádek se zvýší obsah dvojregistru BC o jednotku pomocí instrukce INX B. Bude-li druhá polovina libovolného řádku právě plně obsazena

požadovanými znaky A, bude v dvojregistru BC hexadecimální číslo 1Ø, které se bude přičítat k obsahu dvojregistru HL při přechodu na další řádek.

Nyní zvolíme ještě dva registry, které budou fungovat jako řídící pro postupný tisk předepsaného počtu znaků a pro předepsaný počet řádků. Bude to registr A, kam dosadíme hexadecimální číslo 1Ø a při každém tisku znaku A na obrazovku na daném řádku budeme jeho obsah snižovat o jednotku pomocí instrukce DCR A. Až v registru A vznikne nula /což nám bude signalizovat příznak nuly v registru příznaků F/, víme, že druhá polovina určitého řádku je znaky naplněna a bude možno přejít na obsazování dalšího řádku. V dalším jednoduchém registru D bude na počátku programu hexadecimální číslo 2Ø a po každém ukončeném obsazování určitého řádku znaky snížíme hodnotu v tomto registru o jednotku pomocí instrukce DCR D. Až v registru D bude nula /což nám bude opět signalizovat příznak nuly v registru F - bude v tomto případě Z = 1/, víme, že bylo obsazeno právě 32 řádků /decimálně/, což je 2Ø hexadecimálně.

Postup:

Po zapnutí počítače v režimu MONITOR vložíme pomocí příkazu S následující strojový program od hexadecimální adresy 8ØØ:

adresa	instrukce	kód	poznámka
8ØØ	LXI H, w	21	vložení hexadec. čísla EC1Ø do HL a vynulování BC
8Ø1	1Ø	1Ø	
8Ø2	EC	EC	
8Ø3	LXI B, w	Ø1	
8Ø4	ØØ	ØØ	
8Ø5	ØØ	ØØ	
8Ø6	MVI D, d	16	počet řádků vložen do D
8Ø7	2Ø	2Ø	
8Ø8	MVI A, d	3E	počet obsazovaných míst v řádku vložen do A
8Ø9	1Ø	1Ø	
8ØA	MVI M, d	36	obsazení adresy vložené do HL kódem znaku A
8ØB	41	41	
8ØC	INX H	23	zvýšení v HL a BC o 1, snížení v A o 1
8ØD	INX B	Ø3	
8ØE	DCR A	3D	
8ØF	JNZ a	C2	není-li v A nula, skok na adr. 8ØA
81Ø	ØA	ØA	
811	Ø8	Ø8	
812	DAD B	Ø9	je-li v A nula, přičtení obsahu BC k obsahu HL
813	LXI B, w	Ø1	vynulování BC
814	ØØ	ØØ	
815	ØØ	ØØ	
816	DCR D	15	snížení počtu neobsazených řádků v reg. D o 1

adresa	instrukce	kód	poznámka
817 818 819	JNZ a Ø8 ØØ8	C2 ØØ8 ØØ8	je-li nějaký řádek dosud neobsazen, tedy v D není ještě nula, skok na adr. 8ØØ8 hexad.
81A	RET	C9	jsou-li všechny řádky obsazeny, konec

Nyní se pomocí příkazu R vrátíme do režimu BASIC a vložíme následující krátký program, jehož funkce je vám již jistě jasná:

```
2 CLS
4 CALL HEX(8ØØ)
6 END
```

Po jeho spuštění příkazem RUN v režimu BASIC vidíme, že se pravá část obrazovky zaplnila předepsaným způsobem.

Poznámka:

Všimněte si, že umíme již ve strojových programech pomocí podmíněných skokových příkazů naprogramovat konečný cyklus pracující analogicky, jako konečný cyklus v jazyce BASIC řízený příkazy FOR, TO, STEP, NEXT. Řídící proměnná cyklu je uložena v registru A, resp. D a instrukce JNZ a působí obdobně, jako příkaz NEXT v jazyce BASIC. Krok, po němž se mění řídící proměnná u předchozího strojového programu, je vyjádřen pomocí instrukce DCR. Nekonečný cyklus ve strojových

programech jde realizovat pomocí instrukce pro nepodmíněný skok JMP.

7. Porovnávací instrukce

Dosud jsme měli možnost využívat v konečných cyklech strojových programů pouze nastavení příznaků v registru F, které bylo důsledkem nějaké aritmetické operace v registru. Nyní si ukážeme, že existují i instrukce, které mohou přímo porovnávat číslo ve střadači A s jiným číslem, které je buď jako přímý operand uvedeno za instrukcí, nebo je uloženo v některém jednoduchém registru A, B, C, D, E, H, L, nebo je na určitém místě v paměti, jehož adresa je vložena do dvojregistru HL. Všechny instrukce uvedených typů jsou následující:

kód	instrukce	význam
FE	CPI, d	porovná obsah střadače a hexadecimální číslo za instrukcí uvedené $d \in \langle ØØ; FF \rangle$
BF B8 B9 BA BB BC BD	CMP A CMP B CMP C CMP D CMP E CMP H CMP L	porovná obsah střadače a obsah registru za instrukcí uvedeného
BE	CMP M	porovná obsah střadače a obsah pam. místa, jehož adresa je uložena v HL

Jak probíhá toto porovnání dvou čísel ?

Ve vnitřních registrech mikroprocesoru, k nimž nemá programátor přístup, dojde k odečtení hodnoty, která je za instrukcí, nebo v registru, nebo v paměti na dané adrese, od hodnoty, která je ve střadači A. Podle výsledku tohoto odečtení jsou nastaveny jednotlivé bity v registru příznaků F tak, jako je tomu při běžném odečítání ve dvojkovém doplňkovém kódu.

Jestliže jsou tedy obě porovnávaná čísla stejná, jejich rozdíl je nula a v registru F bude Z = 1. Nejsou-li stejná, jejich rozdíl není nulový a v registru F je Z = Ø.

Je-li číslo ve střadači větší, pak rozdíl porovnávaných čísel je kladný a podle pravidel pro odečítání je v takovém případě příznak přenosu CY = Ø. Je-li ve střadači A číslo menší, je z analogických důvodů CY = 1.

Mimo to dochází k nastavení i všech ostatních hodnot na bitech registru příznaků F, jako při každém odečítání - většinou se však v programech využívá testování pouze příznaků Z a CY výše uvedených.

Jmenované instrukce rozšiřují možnosti pro využívání podmíněných skoků ve strojových programech pomocí nastavování hodnot na bitech registru příznaků F, na nichž závisí činnost skokových instrukcí.

Na prostřední řádek obrazovky vytiskněte pomocí strojového programu předepsaný počet bílých koleček, jejichž hexadecimální kód je ØØ.

Rozbor:

Hexadecimální adresy prostředního řádku obrazovky jsou v rozmezí EDEØ až EDFF. Adresa EDEØ bude vložena do dvojregistru HL a bude zvyšována pomocí instrukce INX H. Střadač A bude sloužit jako registr, kde bude uchovávána řídící proměnná konečného programového cyklu, na počátku bude jeho obsah vynulován, při každém tisku znaku bude jeho obsah zvýšen o 1 instrukcí INR A. Obsah střadače bude při každém průchodu cyklem porovnáván pomocí instrukce CPI se zadaným číslem a pokud si obě čísla budou rovna, program skončí. V opačném případě nastane chod programu opětně příkazy náležejícími do cyklu a dojde k tisku dalšího znaku do řady a zvýšení obsahu střadače A a dvojregistru HL o jedničku.

Postup:

Po zapnutí počítače v režimu MONITOR pomocí příkazu S vložíme do paměti následující strojový program od hexadecimální adresy 4ØØ:

adresa	instrukce	kód	poznámka
4ØØ	LXI H, w	21	vložení první adresy řádku do HL
4Ø1	EØ	EØ	
4Ø2	ED	ED	
4Ø3	MVI A, d	3E	vynulování střadače A
4Ø4	ØØ	ØØ	
4Ø5	MVI M, d	36	tisk znaku s kódem ØØ na adresu ulož. v HL
4Ø6	ØØ	ØØ	

adresa	instrukce	kód	poznámka
4Ø7 4Ø8	INR A INX H	3C 23	zvýšení obsahu A a HL o 1 v každém cyklu
4Ø9 4ØA	CPI, d 1Ø	FE 1Ø	porovnání obsahu A s hex. číslem 1Ø
4ØB 4ØC 4ØD	JNZ a Ø5 Ø4	C2 Ø5 Ø4	jestliže porovnávaná čísla nejsou stejná, skok na hex. adr. 4Ø5
4ØE	RET	C9	jsou-li porovnávaná čísla stejná, konec

Nyní se vrátíme pomocí příkazu R do režimu BASIC a vložíme následující příkaz /bez čísla programového řádku/:

CLS : CALL HEX(4ØØ)

a odešleme ho tlačítkem CR. Vídíme, že na čisté obrazovce se objeví 16 /decimálně/ grafických znaků, což odpovídá hexadecimálnímu číslo 1Ø, které bylo do strojového programu vloženo. Jeho změnou na adrese 4ØA je možno získat na obrazovce jiný počet grafických znaků.

8. Instrukce pro volání strojových podprogramů a pro návraty ze strojových podprogramů

V programech v jazyce BASIC známe příkazy, které způsobí přechod z hlavního programu na některý podprogram. Jsou to příkazy

GOSUB čř

a rovněž příkazy, které způsobí návrat z daného podprogramu do hlavního programu na místo, z něhož nastal odskok na zvolený podprogram. Je to příkaz

RETURN .

příkazy pro přechod na podprogram a zpět mohou být rovněž podmíněné - tedy ve tvaru

IF (výrok) THEN GOSUB čř

a rovněž

IF (výrok) THEN RETURN .

Obdobné přechody na podprogramy a příslušné návraty lze realizovat i ve strojových programech. Takové instrukce se opět nemohou odvolávat na nějaká čísla programových řádků, ale pouze na adresy paměťových míst, na nichž příslušné strojové podprogramy začínají. Rovněž podmínky pro podmíněné přechody na strojové podprogramy jsou odvozeny výhradně od obsazení bitů registru F - srovnej s kapitolou o nepodmíněných a podmíněných skocích ve strojových programech.

Všechny strojové instrukce pro přechod do podprogramů jsou následující: /uvědomte si: je-li v A nula, je Z = 1/

kód	instrukce	poznámka
CD	CALL a	nepodmíněný skok na podprogram, který začíná na adrese a
DC	CC a	skok na podprogram začínající na adr. a, jeli CY = 1

kód	instrukce	poznámka
CC	CZ a	skok na podprogram začínající na adr. a, je-li Z = 1
C4	CNZ a	skok na podprogram začínající na adr. a, je-li Z = Ø
D4	CNC a	skok na podprogram začínající na adr. a, je-li CY = Ø
FC	CM a	skok na podprogram začínající na adr. a, je-li S = 1
F4	CP a	skok na podprogram začínající na adr. a, je-li S = Ø
EC	CPE a	skok na podprogram začínající na adr. a, je-li P = 1
E4	CPO a	skok na podprogram začínající na adr. a, je-li P = Ø

Adresa a je vždy celé hexadecimální číslo z intervalu

< ØØØØ; FFFF >

a v konkrétních programech se uvádějí nejdříve dva nižší řády a pak teprve dva vyšší řády, jak již víme z dřívějšího výkladu instrukcí LXI, JMP apod.

Všechny instrukce strojového kódu pro návrat z podprogramu do hlavního programu jsou následující:

kód	instrukce	poznámka
C9	RET	nepodmíněný návrat z podprogramu do hlavního programu
D8	RC	návrat, je-li CY = 1
DØ	RNC	návrat, je-li CY = Ø
C8	RZ	návrat, je-li Z = 1

kód	instrukce	poznámka
CØ	RNZ	návrat, je-li Z = Ø
FØ	RP	návrat, je-li S = Ø
F8	RM	návrat, je-li S = 1
E8	RPE	návrat, je-li P = 1
EØ	RPO	návrat, je-li P = Ø

Důležitá upozornění:

Instrukce pro volání podprogramů a pro návraty z podprogramů nemění hodnotu žádného bitu v registru příznaků F, pouze tyto příznaky testují a podle jejich hodnoty řídí chod programu.

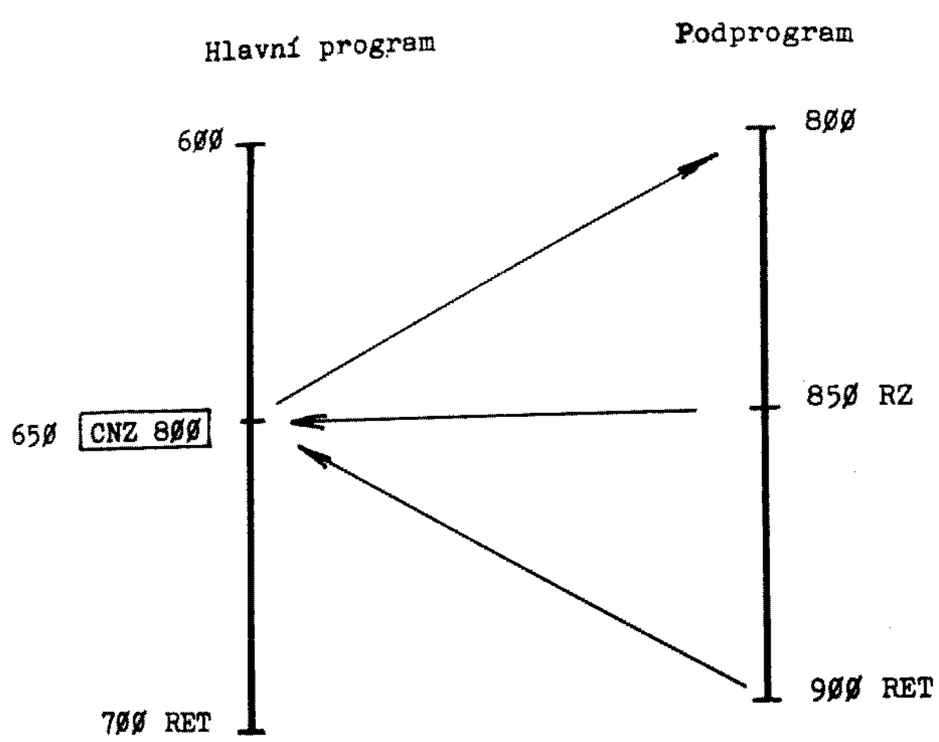
Všimněte si, že instrukce pro strojové programování CALL vypadá opticky stejně, jako příkaz CALL v jazyku BASIC, funkce příkazu v jazyce BASIC je však jiná - způsobuje přechod z programu v jazyce BASIC na program ve strojovém kódu.

Všimněte si dále, že instrukce RET způsobuje návrat ze strojového podprogramu do hlavního strojového programu a je-li na konci hlavního strojového programu, způsobuje návrat do programu v jazyce BASIC.

Není-li podmínka pro některou podmíněnou instrukci v programu splněna, instrukce se ignoruje a program pokračuje tak, jako kdyby této instrukce nebylo.

Dojde-li k návratu z podprogramu ve strojovém kódu, pokračuje chod hlavního strojového programu instrukcí, která je uvedena hned za příslušnou volací instrukcí hlavního programu a příslušnou adresou.

Prohlédněte si následující schema hlavního programu a podprogramu /oba jsou ve strojovém kódu/:



Program probíhá takto:
Od adresy 6ØØ do 65Ø probíhá bez větvení, je-li příznak nuly Z = Ø v okamžiku, kdy se zpracovává instrukce na adrese 65Ø, dojde k přechodu na podprogram, který začíná na adrese 8ØØ. Je-li příznak nuly Z = 1, k přechodu na podprogram vůbec nedojde a program skončí na adrese 7ØØ. Došlo-li k odskoku na podprogram, potom pokud je příznak nuly Z = 1 v okamžiku, kdy se zpracovává instrukce na adrese 85Ø, dojde k návratu do hlavního programu a začne se zpracovávat instrukce,

která je na adrese 653. Uvědomte si totiž, že na adrese 65Ø je skoková instrukce CNZ, k níž přísluší ještě adresa, obsazující další dvě paměťová místa s adresami 651 a 652. Proto následující instrukce strojového programu je skutečně až na adrese 653 - na ní se vrací chod programu po návratu z podprogramu a od ní pokračuje program až ke koncové adrese 7ØØ.
Jestliže je příznak nuly Z = Ø v okamžiku zpracovávání instrukce na adrese 85Ø, pak nedojde k návratu do hlavního programu z tohoto místa, program zpracovává instrukce na adresách od 851 do 9ØØ a teprve z adresy 9ØØ se vrací do hlavního programu na adresu 653 a pokračuje v něm až do jeho konce na adrese 7ØØ.

Poznámka:

Podle počtu paměťových míst, které určitá instrukce zaujímá všetně příslušné adresy nebo operandu, mluvíme o jedno- , dvou- a tříslabikových instrukcích. Jednoslabiková instrukce je například MOV A, H, dvouslabiková je například MVI B, d a tříslabiková je například JMP a.

9. Význam dvojregistru PC

Zatím jsme se nezabývali tímto registrem, protože se nepoužívá k přímým výpočtům jako ostatní dvojregistry, ale slouží k ovládání chodu programu. V tomto dvojregistru je uchovávána adresa instrukce strojového kódu, která se právě zpracovává - adresa je pochopitelně hexadecimální. Začíná-li nějaký strojový program na adrese 5ØØ, pak při jeho spuštění se dosadí do registru PC hexadecimální číslo 5ØØ.

Protože máme k dispozici instrukci, která způsobuje překopírování obsahu dvojregistru HL do PC - je to instrukce

PCHL s kódem E9 ,

můžeme tímto způsobem provádět i skoky v programu - obdobně, jako to provádí instrukce JMP. Ukážeme si to na následujícím příkladě.

Na konec prostředního řádku obrazovky napište znak A /jehož hexadecimální kód je 41/, nebo B /jehož hexadecimální kód je 42/ s využitím instrukce PCHL.

Postup:

Po zapnutí počítače vložíme v režimu MONITOR pomocí příkazu S následující strojový program od hexadecimální adresy 5ØØ:

adresa	instrukce	kód	poznámka
5ØØ 5Ø1 5Ø2	LXI H, w ØØ Ø6	21 ØØ Ø6	vložení skokové adr. do HL
5Ø3	PCHL	E9	překopírování skokové adr. do PC
6ØØ 6Ø1 6Ø2	LXI H, w FF ED	21 FF ED	vložení adresy zvoleného místa obrazovky do HL
6Ø3 6Ø4	MVI M, d 41	36 41	obsazení zvoleného místa na obr. kódem znaku A - adr. v HL
6Ø5	RET	C9	konec

adresa	instrukce	kód	poznámka
7ØØ 7Ø1 7Ø2	LXI H, w FF ED	21 FF ED	vložení adresy zvoleného místa obrazovky do HL
7Ø3 7Ø4	MVI M, d 42	36 42	obsazení zvoleného místa na obr. kódem znaku B - adr. v HL
7Ø5	RET	C9	konec

Spustíme tento program pomocí příkazu C5ØØ v režimu MONITOR, pomocí příkazu na adrese 5Ø3 se do registru PC dostane adresa 6ØØ a tedy se začne zpracovávat instrukce, která je na adrese 6ØØ, na obrazovce se objeví v předepsané pozici znak A a program končí na adrese 6Ø5.

Nyní pomocí příkazu S v režimu MONITOR změníme hexadecimální číslo na adrese 5Ø2 z Ø6 na Ø7. Spustíme-li takto opravený program příkazem C5ØØ v režimu MONITOR, pak instrukce na adrese 5Ø3 vloží do PC adresu 7ØØ, tedy zpracování programu je převedeno na tuto adresu a následující. V důsledku toho se na předepsaném místě obrazovky objeví znak B a program skončí na hexadecimální adrese 7Ø5.

Poznámky:

Na tomto principu /dosazení adresy do PC a převedení chodu programu od této adresy/ pracují všechny instrukce typu JMP, JC, JZ, JNZ apod.

Podíváme-li se na obsazení jednotlivých registrů po-

mocí příkazu X v režimu MONITOR po ukončení chodu našeho programu, je v PC nastavena první adresa volaného programu - v našem případě hexadecimálně 5ØØ, což způsobuje instrukce RET na konci strojového programu.

10. Zásobník

Při práci s programy ve strojovém kódu je možno používat tzv. zásobník. Je to oblast paměti, do níž lze ukládat různé údaje - návratové adresy, obsahy dvojregistrů - a ve vhodném časovém okamžiku je opět vyzvedávat a naplňovat jimi vhodné registry.

Zásobník je umístěn v okolí USR-oblasti v paměti RAM počítače - jeho poloha se tedy může měnit podle toho, jak je rozsáhlá USR-oblast, zavedená pomocí příkazu

CLEAR p_1 , p_2

v režimu BASIC programátorem.

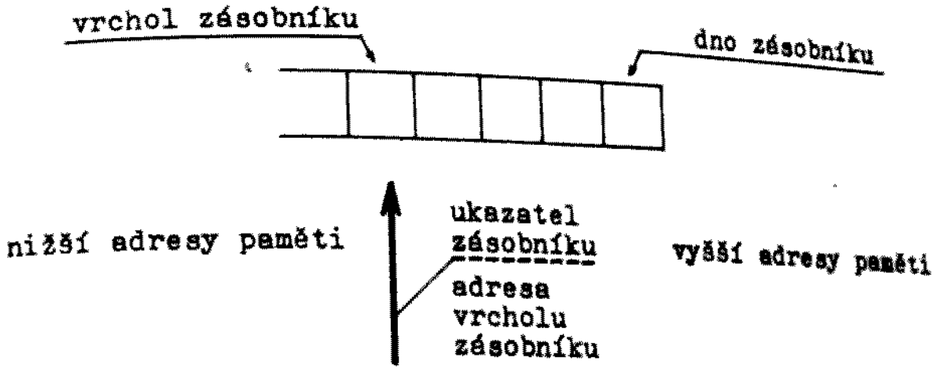
Ukládá-li se nějaký údaj do zásobníku, ukládá se na sousední nižší adrese, než údaj předchozí.

Poslední údaj, který byl do zásobníku vložen, tvoří tzv. vrchol zásobníku. Pokud se tento údaj ze zásobníku odebere, vrchol zásobníku se tím posune směrem k vyšším adresám v paměti.

Adresa paměťového místa, na němž je vrchol zásobníku, se uchovává v dvojregistru SP - zkráceně se značí S. Při vložení nějakého údaje do zásobníku se posune vrchol zásobníku směrem k nižším adresám v paměti o 2, protože adresa nebo obsah dvojregistru je vždy na dvou sousedních paměťových

místech a tedy se sníží o dvě jednotky i údaj v dvojregistru SP - tzv. ukazateli zásobníku. Při odebírání údaje ze zásobníku se hodnota v dvojregistru SP opět zvýší, protože se vrchol zásobníku posunul směrem k vyšším hodnotám adres.

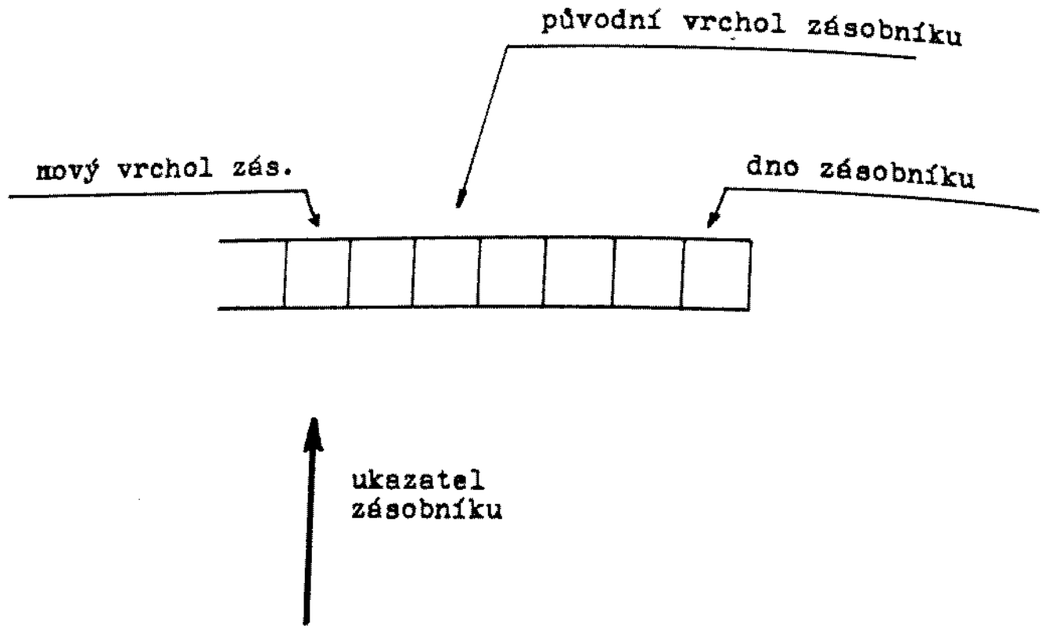
Příklad



Je-li v tomto stavu v ukazateli zásobníku SP hexadecimální číslo 7E24, pak po vložení obsahu dvojregistru HL do zásobníku bude vypadat situace tak, jak ukazuje obrázek na následující stránce.
Na nejbližší nižší adresy se vloží obsah dvojregistru HL, což způsobí, že se vrchol zásobníku přesune o dvě paměťová místa směrem k nižším adresám a ukazatel zásobníku ukazuje nyní nový vrchol zásobníku, tedy v dvojregistru SP bude hexadecimální adresa

7E22 .

Obsah dvojregistru HL se do zásobníku vkládá tak, že se nejdříve uloží obsah registru H a pak teprve obsah registru L - ten tvoří obsazení vrcholu zásobníku.



Při odebírání ze zásobníku je situace opačná, nejdříve se uloží údaj, který je na vrcholu zásobníku do registru L a údaj, který je "pod ním", do registru H. Přitom se posune ukazatel zásobníku o dvě paměťová místa, tedy v registru SP bude opět hexadecimální číslo 7E24.

10.1 Instrukce, které využívají při své činnosti zásobník

Existují instrukce, které již známe a které při své činnosti automaticky zásobník využívají a s ním manipulují. Jsou to všechny instrukce pro volání podprogramů /instrukce skupiny CALL - ať již podmíněné nebo nepodmíněné/ a rovněž všechny instrukce pro návrat z podprogramů /instrukce skupiny RET - ať již podmíněné nebo nepodmíněné/.

Instrukce skupiny CALL pracují tak, že uloží do zá-

sobníku návratovou adresu do hlavního programu, z něhož se provádí odskok na podprogram a pak teprve nastane skok na první adresu podprogramu, která je za instrukcí typu CALL uvedena. Instrukce skupiny RET vyzvedne tuto návratovou adresu z vrcholu zásobníku a vloží ji do dvojregistru PC, čímž dojde k návratu z konce podprogramu na příslušné místo hlavního strojového programu.

Zde vidíme, jaký je rozdíl mezi instrukcemi skupiny CALL a instrukcemi skupiny JMP. Instrukce skupiny JMP neukládají návratovou adresu do zásobníku a provádějí pouze skok na určenou adresu bez zajištění návratu.

Důležité upozornění:

Obsah tohoto článku je nutno mít vždy na paměti, abychom nevhodnou manipulací se zásobníkem neznemožnili správný návrat z podprogramů. Instrukce, s nimiž lze zasahovat do zásobníku "svévolně", jsou uvedeny v následující stati 10.2 a 10.3.

10.2 Instrukce měnící obsazení dvojregistru SP

Z předchozích kapitol již známe některé instrukce, které mění hodnotu v ukazateli zásobníku SP. Jsou to:

```
DCX SP
INX SP
LXI SP, w
SPHL
```

První dvě instrukce způsobí posuv ukazatele zásobníku o jedno paměťové místo - buď směrem k vyšším hodnotám adres /INX/, nebo směrem k nižším hodnotám adres /DCX/. To se projevuje zvýšením nebo snížením čísla v dvojregistru SP.

Třetí uvedená instrukce umožňuje přesunout ukazatel zásobníku tak, aby ukazoval na adresu w, která je za instrukcí uvedena. Hodnota w bude tedy v dvojregistru SP.

Obsah dvojregistru SP je možno změnit i tak, že do SP přesuneme obsah dvojregistru HL - to provádí poslední ve výčtu uvedená instrukce.

Obsah dvojregistru SP je možno také přičíst k obsahu dvojregistru HL, což zajišťuje již dříve uvedená instrukce DAD SP.

10.3 Další instrukce pracující se zásobníkem

Jsou to instrukce, které ukládají obsahy dvojregistrů do zásobníku a instrukce, které vložené údaje ze zásobníku odebírají a umísťují zpět do dvojregistrů. Všechny instrukce uvedených typů jsou následující:

kód	instrukce	poznámka
C5	PUSH B	vloží do zásobníku obsah dvojreg., jehož první písmeno je za instrukcí
D5	PUSH D	
E5	PUSH H	
C1	POP B	vloží obsah dvou míst vrcholu zásobníku do dvojreg., jehož první písmeno je za instrukcí
D1	POP D	
E1	POP H	

Například tedy instrukce

PUSH B

pracuje tak, že se posune nejdříve ukazatel zásobníku o jedno paměťové místo směrem k nižším adresám v paměti, na toto místo se uloží obsah registru B, pak se opět posune ukazatel

zásobníku o 1 místo stejným směrem a na toto místo se uloží obsah registru C. Vznikne tak nový vrchol zásobníku a hodnota v SP se sníží o 2.

Instrukce

POP D

pracuje tak, že údaj na vrcholu zásobníku přesune do registru E, pak posune ukazatel zásobníku o jedno místo v paměti směrem k vyšším adresám a obsah tohoto paměťového místa vloží do registru D. Pak opět posune ukazatel zásobníku o 1 místo stejným směrem. Hodnota v registru SP se při provádění jmenované instrukce zvýší o 2.

Dále lze analogickým způsobem uložit do zásobníku hodnotu ze střadače A a hodnotu z registru příznaků F - tzv. stavové slovo, značíme ho PSW - a v příhodném okamžiku je opět vyzvednout. Nikoliv ovšem buď registr A, nebo registr F, ale vždy obsahy obou registrů současně. To zajišťují instrukce:

kód	instrukce	poznámky
F5	PUSH PSW	uloží obsáhy A a F do zásobníku
F1	POP PSW	vloží údaje z vrcholu zásobníku do A a F

Obě jmenované instrukce pracují naprosto stejně, jako dříve uvedené instrukce typu PUSH a POP. PUSH PSW uloží do zásobníku nejdříve obsah střadače A a pak obsah registru F - ten je na vrcholu zásobníku. Instrukce POP PSW je v tomto pořadí odebírá z vrcholu zásobníku.

Další instrukce umožňuje výměnu obsahu dvojregistru HL s obsahem dvou paměťových míst na vrcholu zásobníku. Je to instrukce

XTHL s kódem E3

Instrukce pracuje tak, že se nejdříve vymění obsah registru L s obsahem paměťového místa, na něž ukazuje ukazatel zásobníku. Potom se ukazatel zásobníku posune o 1 paměťové místo směrem k vyšším adresám a vymění se obsah paměťového místa, na něž je nastaven ukazatel zásobníku nyní, s obsahem registru H.

11. Instrukce pro periferie počítače

Ve skupině instrukcí strojového kódu jsou i takové, pomocí nichž je možno přijmout a zpracovat údaje vyslané do počítače periferním zařízením, nebo naopak vyslat signály směrem k periferním zařízení.

Protože začátečník těchto instrukcí používat nebude, zmíníme se o nich pouze krátce a informativně pro úplnost přehledu všech instrukcí.

11.1 Instrukce přijímající nebo vysílající signál prostřednictvím portů

Pod pojmem port si zatím představíme určité místo v paměti počítače, přes nějž jde signál z počítače ven nebo naopak dovnitř. Každý port má svou hexadecimální adresu.

Instrukce strojového kódu, které vysílají nebo přijímají signál z portu o určité adrese, jsou následující:

kód	instrukce	poznámka
DB	IN a	instrukce naplní obsah střadače A hexad. číslem, které je na portu s adresou a
D3	OUT a	instrukce vyšle obsah střadače A na port, jehož adresa je za instrukcí uvedena

Adresa a je hexadecimální z intervalu ⟨ØØ; FF⟩.

Například port, který souvisí s klávesnicí, má hexadecimální adresu 85. Není-li stisknuto žádné černé tlačítko, je na tomto portu hexadecimální číslo FF. Jestliže je stisknuto určité černé tlačítko klávesnice, po dobu jeho stisknutí se na tomto portu objeví jiné hexadecimální číslo. Rovněž i tlačítka F1 až F5 mění hodnotu čísla na tomto portu.

Například:

při stisknutí tlačítka	je na portu s adr. 85 číslo	
	decimálně	hexadecim.
F1	239	EF
F2	223	DF
F3	127	7F
F4, F5	191	BF
4	254	FE
E	253	FD
P	223	DF
A	251	FB
L	239	EF

To snadno ověříme pomocí chodu programu

```
1 PRINT INP(HEX(85))
2 GOTO 1
```

v režimu BASIC.
Číslo z tohoto portu může být pomocí instrukce IN "vtaženo" do počítače a může s ním být pracováno ve strojovém programu.

Sestavte strojový program, který napíše na pravý okraj obrazovky písmeno A, není-li stisknuté žádné tlačítko klávesnice a napíše na totéž místo písmeno E, je-li stisknuto tlačítko písmene E.

Rozbor:

Hexadecimální kód znaku E je 45, znaku A je 41. Adresa v oblasti VIDEORAM, na níž se písmena budou objevovat, bude EDFF hexadecimálně a bude uložena v dvojregistru HL.
Strojový program bude ve tvaru nekonečného cyklu, při každém průchodu cyklem bude vloženo číslo z portu o hexadecimální adrese 85 do střadače A pomocí instrukce IN a porovnáno s hexadecimálním číslem FD, které se na tomto portu objeví po dobu stisknutí tlačítka s písmenem E. Podle výsledku porovnání bude program pokračovat po jedné ze dvou větví - první bude zajišťovat tisk znaku A na předepsaném místě obrazovky, druhá tisk znaku E na tomtéž místě. Po ukončení každé větve se program vrátí na začátek.

Postup:

Po zapnutí počítače v režimu MONITOR vložíme pomocí příkazu S následující strojový program od hexadecimální adresy 7ØØ:

adresa	instrukce	kód	poznámka
7ØØ	LXI H, w	21	vložení adresy místa na obrazovce do HL
7Ø1	FF	FF	
7Ø2	ED	ED	
7Ø3	IN a	DB	naplnění A číslem z portu s adr. a
7Ø4	85	85	
7Ø5	CPI, d	FE	porovnání čísla ve střadači s číslem FD
7Ø6	FD	FD	
7Ø7	JZ a	CA	v případě shody porovnávaných čísel skok na adr. 7ØF
7Ø8	ØF	ØF	
7Ø9	Ø7	Ø7	
7ØA	MVI M, d	36	umístění znaku A na místo, jehož adresa je v HL
7ØB	41	41	
7ØC	JMP a	C3	skok na začátek programu - na adr. 7Ø3
7ØD	Ø3	Ø3	
7ØE	Ø7	Ø7	
7ØF	MVI M, d	36	umístění znaku E na místo, jehož adresa je v HL
71Ø	45	45	
711	JMP a	C3	skok na začátek programu - na adr. 7Ø3
712	Ø3	Ø3	
713	Ø7	Ø7	

Nyní se pomocí příkazu R vrátíme do režimu BASIC a vložíme tento příkaz /bez čísla řádku !/:

CLS : CALL HEX (7ØØ)

Po jeho odeslání tlačítkem CR se na čisté obrazovce objeví písmeno A a po dobu stisku tlačítka s písmenem E se změní na znak E - rovněž tak i při stisku tlačítek s písmeny Q, W, R, T, Y, U aj., protože tato tlačítka obsazují port 85 stejným číslem.

Program lze zastavit buď stisknutím tlačítka BR, přičemž se na obrazovce objeví hexadecimální adresa, na níž byl program přerušen, nebo pomocí tlačítka RES, čímž se vrátíme do režimu BASIC, ale strojový program zůstane v paměti neporušen.

Poznámka:

Všimněte si, že náš strojový program realizuje přibližně totéž, co realizuje program v jazyce BASIC pomocí klíčového slova INKEY$ - vstup informace z klávesnice při chodu programu a další řízení chodu programu podle vyhodnocení této informace.

11.2 Zbývající instrukce

Aby byl přehled všech probíraných instrukcí úplný, uvádíme jen informativně zbývající, které začínající programátor nevyužije. Jsou to instrukce související například s činností periferních zařízení a jsou to:

kód	instrukce	poznámka
FB	EI	instrukce dovoluje zpracovat přerušení programu
F3	DI	instrukce zakazuje zpracovat přerušení programu až do doby, kdy se v programu objeví instrukce EI
76	HLT	instrukce zastaví chod mikroprocesoru a před opětným jeho spuštěním je očekáván vnější podnět.
C7	RST Ø	může se použít pro určité druhy větvení programů, většinou signály typu RST vysílají do počítače periferní zařízení a dožadují se vstupu svých hodnot
CF	RST 1	
D7	RST 2	
DF	RST 3	
E7	RST 4	
EF	RST 5	
F7	RST 6	
FF	RST 7	

12. Doplňky a závěrečné poznámky

12.1 Poznámky k funkci příkazu CALL v jazyce BASIC

Příkaz CALL v režimu BASIC vyvolá program ve strojovém kódu, jehož počáteční decimální adresa je za příkazem uvedená. Za touto adresou mohou být v příkazu CALL i jiné decimální parametry, oddělené navzájem čárkou. Tedy příkaz může vypadat:

CALL a, d_1, ... , d_n

V tomto případě se d_n ukládá do dvojregistru DE, d_{n-1} do dvojregistru BC a zbytek se uloží do zásobníku tak, že na vrcholu je návratová adresa a pak následují po řadě parametry d_i, přičemž d_1 je v zásobníku nejhlouběji. Platí samozřejmě, že

$$d_i \in \langle 0; 65535 \rangle \quad \text{decimálně}$$

pro všechny indexy i.

12.2 Zkrácení záznamů a instrukcí strojových programů

V tabulkových zápisech našich strojových programů jsme dosud z metodických důvodů dodržovali určitý způsob psaní některých instrukcí, adres a přímých operandů.

Například:

adresa	instrukce	kód	poznámka
...	.....	..	.....
...	.....	..	.....
716	LXI H, w	21	.....
717	08	08	.....
718	05	05	.....
719	MVI D, d	16	.....
71A	10	10	.....
71B	JMP a	C3	.....
71C	80	80	.....
71D	09	09	.....
71E	.....	..	.....
...	.....	..	.....
...	.....	..	.....

V našich i zahraničních příručkách se takový zápis většinou zkracuje, takže tabulka vypadá následovně:

adresa	instrukce	kód	poznámka
...	.....	.....	.....
...	.....	.....	.....
716	LXI H, 0508	21 08 05	.....
719	MVI D, 10	16 10	.....
71B	JMP 0980	C3 80 09	.....
71E	.....	.....	.....
...	.....	.....	.....

Z uvedeného příkladu je princip zkrácení patrný a čtenář nepodlehne dojmu, že některé adresy v levém sloupci tabulky nejsou obsazeny. Silně orámovaná část tabulky ukazuje způsob programování v tzv. jazyce symbolických adres.

12.3 Umístění strojového programu v paměti počítače

Víme, že program ve strojovém kódu je vhodné umístit do oblasti USR vhodného rozsahu, kde je zaručeno, že program ve strojovém kódu nebude poškozen chodem nebo vkládáním programu v jazyce BASIC. Postup jsme si podrobně ukázali na úvodním příkladu příručky.

Ve většině našich příkladů jsme umísťovali strojový program již od adresy 4000, 5000, 6000, 7000 apod., aniž strojový program byl v oblasti USR. Použili jsme skutečnosti, že současně se strojovým programem, je tak malého rozsahu, že

nemůže zasáhnout do oblasti, v níž je strojový program.

V obou jmenovaných případech při nahrávání programů na magnetofonový pásek je nutno nahrávat zvlášť program v jazyce BASIC příkazem MSAVE v režimu BASIC a zvlášť strojový program v režimu MONITOR příkazem W.

Ukážeme si nyní třetí způsob, jak umístit program v jazyce BASIC současně s programem strojovým do paměti počítače, aby se navíc strojový program přehrával současně s programem v jazyce BASIC na magnetofon. Využíváme k tomu příkazů DATA,READ a POKE v režimu BASIC. Víme, že vložit program ve strojovém kódu do paměti počítače znamená, že obsadíme určitý úsek paměti vhodnými čísly. V režimu MONITOR obsazujeme paměťová místa hexadecimálně, v režimu BASIC decimálně. Princip takového vložení strojového programu nám ukáže následující příklad.

Příklad:

Minulý příklad /demonstrace funkce strojové instrukce IN a / proveďte tak, že po vložení strojového programu použijete programových příkazů DATA, READ a POKE v režimu BASIC.

Řešení:

Nejdříve převedeme jednotlivé hexadecimální kódy strojových instrukcí na decimální. Program ve strojovém kódu bude vložem počínaje od hexadecimální adresy 6ØØ1, je tedy nutno upravit adresy za skokovými instrukcemi JMP. Strojový program bude vypadat následovně:

adresa hex.	instrukce	kód hex.	kód dec.
6ØØ1	LXI H, w	21	33
6ØØ2	FF	FF	255
6ØØ3	ED	ED	237
6ØØ4	IN a	DB	219
6ØØ5	85	85	133
6ØØ6	CPI, d	FE	254
6ØØ7	FD	FD	253
6ØØ8	JZ a	CA	2Ø2
6ØØ9	1Ø	1Ø	16
6ØØA	6Ø	6Ø	96
6ØØB	MVI M, d	36	54
6ØØC	41	41	65
6ØØD	JMP a	C3	195
6ØØE	Ø4	Ø4	4
6ØØF	6Ø	6Ø	96
6Ø1Ø	MVI M, d	36	54
6Ø11	45	45	69
6Ø12	JMP a	C3	195
6Ø13	Ø4	Ø4	4
6Ø14	6Ø	6Ø	96

Žádný strojový program do počítače nevkládáme, po jeho zapnutí vložíme v režimu BASIC následující program:

```
 2 DATA 33,255,237,219,133,254,253,2Ø2,16,96,
   54,65,195,4,96,54,69,195,4,96
 4 FOR I = HEX ( 6ØØ1) TO HEX ( 6Ø14)
 6 READ A : POKE I, A
 8 NEXT I
1Ø CLS
12 CALL HEX (6ØØ1)
```

Všimněte si programových řádků 2 až 8 - ty vlastně po startu programu příkazem RUN v režimu BASIC vytvoří strojový požadovaný program na paměťových místech s hexadecimálními adresami od 6ØØ1 do 6Ø14. Desátý programový řádek vyčistí obrazovku a dvanáctý vyvolá chod takto vloženého strojového programu.

Tento program lze přehrát na magnetofon pouze pomocí příkazu MSAVE v režimu BASIC. Strojový program již přehrávat nemusíme, sám se vytvoří při startu programu v jazyce BASIC. Nutno uváženě volit místo v paměti, kam se bude ukládat takový strojový program, aby nedošlo k porušení programu v jazyce BASIC vznikajícím strojovým programem.

12.4 Rychlé přehrávání programů

Protože už víme, jak se přehrává program ve strojovém kódu a program v jazyce BASIC, víme také, že každý strojový program je možno zahrnout do programu v jazyce BASIC - viz minulý článek - zbývá nyní ukázat, jak je možno podstatně /až 2,5x/ zkrátit dobu přehrávání programu v jazyce BASIC.

Víme, že program v jazyce BASIC je v paměti počítače

od hexadecimální adresy 16A až do nějaké koncové hexadecimální adresy, kterou označíme třeba VXYZ a najdeme ji pomocí příkazu D v režimu MONITOR - viz příručka-Feil: "Monitor IQ 151". Nezapomeneme, že záznam každého programu v paměti počítače je uzavřen několika nulami, které také k programu patří, tvoří jeho zakončení v paměti. Proto hexadecimální číslo VXYZ je adresou až některého následujícího paměťového místa za těmito nulami.

Před adresou 16A jsou ještě některé informace o rozsahu programu, které musí být nahrány současně s programem, takže "kompletní" záznam programu je v paměti počítače v oblasti od hexadecimální adresy DØ do adresy VXYZ.

Při pořizování rychlé nahrávky programu v jazyce BASIC na magnetofonový pásek postupujeme takto:

1/ Tlačítkem BR přejdeme do režimu MONITOR.

2/ Snížíme meziblokovou distanci na Ø2, tj. na hexadecimální adresu 1C vložíme pomocí příkazu S v režimu MONITOR číslo Ø2.

3/ V režimu MONITOR dáme na obrazovku příkaz

W DØ, VXYZ, Ø

a zapneme nahrávání magnetofonu. Po asi 5 sekundách nahrávání pilotního kmitočtu odešleme výše uvedený příkaz tlačítkem CR.

4/ Ukončení nahrávky nám signalizuje objevení blikajícího kurzoru na obrazovce.

Pokud naši rychlonahrávku chceme přehrát z magnetofonu do paměti počítače, pak po zapnutí počítače přejdeme do režimu MONITOR pomocí tlačítka BR, vložíme na obrazovku pří-

kaz L a nám známým způsobem záznam programu do paměti počítače v režimu MONITOR přehrajeme. Po ukončení nahrávky se pomocí příkazu R vrátíme do režimu BASIC a běžným příkazem jazyka BASIC program spustíme, nebo získáme na obrazovce jeho výpis.

Poznámka:

1/ Pokud při přehrávání z paměti počítače na magnetofon použijeme při tomto postupu příkaz

W DØ, WXYZ, CAD6

pak při zpětném přehrávání záznamu do paměti počítače se po ukončení nahrávání počítač automaticky vrátí do režimu BASIC, ušetříme si tedy stisknutí tlačítka R.

2/ Hexadecimální adresu paměťového místa, na němž končí program, je možno najít na adresách DØ a D1 v paměti RAM počítače. Obě adresy jsou samozřejmě hexadecimální. Na paměťovém místě s adresou DØ jsou dvě poslední cifry hexadecimální adresy místa, kde program končí - tedy YZ. Na paměťovém místě s adresou D1 jsou obě zbývající cifry adresy - tedy VX.

Tabulka převodů celých čísel v soustavě

decimální a hexadecimální

D - decimální H - hexadecimální

D	H	D	H	D	H	D	H	D	H	D	H	D	H
Ø	Ø	37	25	74	4A	111	6F	148	94	185	B9	222	DE
1	1	38	26	75	4B	112	7Ø	149	95	186	BA	223	DF
2	2	39	27	76	4C	113	71	15Ø	96	187	BB	224	EØ
3	3	4Ø	28	77	4D	114	72	151	97	188	BC	225	E1
4	4	41	29	78	4E	115	73	152	98	189	BD	226	E2
5	5	42	2A	79	4F	116	74	153	99	19Ø	BE	227	E3
6	6	43	2B	8Ø	5Ø	117	75	154	9A	191	BF	228	E4
7	7	44	2C	81	51	118	76	155	9B	192	CØ	229	E5
8	8	45	2D	82	52	119	77	156	9C	193	C1	23Ø	E6
9	9	46	2E	83	53	12Ø	78	157	9D	194	C2	231	E7
1Ø	A	47	2F	84	54	121	79	158	9E	195	C3	232	E8
11	B	48	3Ø	85	55	122	7A	159	9F	196	C4	233	E9
12	C	49	31	86	56	123	7B	16Ø	AØ	197	C5	234	EA
13	D	5Ø	32	87	57	124	7C	161	A1	198	C6	235	EB
14	E	51	33	88	58	125	7D	162	A2	199	C7	236	EC
15	F	52	34	89	59	126	7E	163	A3	2ØØ	C8	237	ED
16	1Ø	53	35	9Ø	5A	127	7F	164	A4	2Ø1	C9	238	EE
17	11	54	36	91	5B	128	8Ø	165	A5	2Ø2	CA	239	EF
18	12	55	37	92	5C	129	81	166	A6	2Ø3	CB	24Ø	FØ
19	13	56	38	93	5D	13Ø	82	167	A7	2Ø4	CC	241	F1
2Ø	14	57	39	94	5E	131	83	168	A8	2Ø5	CD	242	F2
21	15	58	3A	95	5F	132	84	169	A9	2Ø6	CE	243	F3
22	16	59	3B	96	6Ø	133	85	17Ø	AA	2Ø7	CF	244	F4
23	17	6Ø	3C	97	61	134	86	171	AB	2Ø8	DØ	245	F5
24	18	61	3D	98	62	135	87	172	AC	2Ø9	D1	246	F6
25	19	62	3E	99	63	136	88	173	AD	21Ø	D2	247	F7
26	1A	63	3F	1ØØ	64	137	89	174	AE	211	D3	248	F8
27	1B	64	4Ø	1Ø1	65	138	8A	175	AF	212	D4	249	F9
28	1C	65	41	1Ø2	66	139	8B	176	BØ	213	D5	25Ø	FA
29	1D	66	42	1Ø3	67	14Ø	8C	177	B1	214	D6	251	FB
3Ø	1E	67	43	1Ø4	68	141	8D	178	B2	215	D7	252	FC
31	1F	68	44	1Ø5	69	142	8E	179	B3	216	D8	253	FD
32	2Ø	69	45	1Ø6	6A	143	8F	18Ø	B4	217	D9	254	FE
33	21	7Ø	46	1Ø7	6B	144	9Ø	181	B5	218	DA	255	FF
34	22	71	47	1Ø8	6C	145	91	182	B6	219	DB		
35	23	72	48	1Ø9	6D	146	92	183	B7	22Ø	DC		
36	24	73	49	11Ø	6E	147	93	184	B8	221	DD		

Tabulka převodů vybraných celých čísel v soustavě decimální, hexadecimální a dvojkové

decimální	hexadecimální	dvojková
Ø	Ø	ØØØØ
1	1	ØØØ1
2	2	ØØ1Ø
3	3	ØØ11
4	4	Ø1ØØ
5	5	Ø1Ø1
6	6	Ø11Ø
7	7	Ø111
8	8	1ØØØ
9	9	1ØØ1
1Ø	A	1Ø1Ø
11	B	1Ø11
12	C	11ØØ
13	D	11Ø1
14	E	111Ø
15	F	1111

Tabulka znaků a jejich kódů

D - decimální H - hexadecimální

D	H	znak	D	H	znak	D	H	znak	D	H	znak	D	H	znak
Ø	Ø		26	1A		52	34	4	78	4E	N	1Ø4	68	h
1	1		27	1B		53	35	5	79	4F	O	1Ø5	69	i
2	2		28	1C		54	36	6	8Ø	5Ø	P	1Ø6	6A	j
3	3		29	1D		55	37	7	81	51	Q	1Ø7	6B	k
4	4		3Ø	1E		56	38	8	82	52	R	1Ø8	6C	l
5	5		31	1F		57	39	9	83	53	S	1Ø9	6D	m
6	6		32	2Ø	␣	58	3A	:	84	54	T	11Ø	6E	n
7	7		33	21	!	59	3B	;	85	55	U	111	6F	o
8	8		34	22	"	6Ø	3C	<	86	56	V	112	7Ø	p
9	9		35	23	#	61	3D	=	87	57	W	113	71	q
1Ø	A		36	24	$	62	3E	>	88	58	X	114	72	r
11	B		37	25	%	63	3F	?	89	59	Y	115	73	s
12	C		38	26	&	64	4Ø	@	9Ø	5A	Z	116	74	t
13	D		39	27	'	65	41	A	91	5B	[	117	75	u
14	E		4Ø	28	(	66	42	B	92	5C	\	118	76	v
15	F		41	29	)	67	43	C	93	5D	]	119	77	w
16	1Ø		42	2A	*	68	44	D	94	5E	↑	12Ø	78	x
17	11		43	2B	+	69	45	E	95	5F	-	121	79	y
18	12		44	2C	,	7Ø	46	F	96	6Ø	`	122	7A	z
19	13		45	2D	_	71	47	G	97	61	a	123	7B	{
2Ø	14		46	2E	.	72	48	H	98	62	b	124	7C	|
21	15		47	2F	/	73	49	I	99	63	c	125	7D	}
22	16		48	3Ø	Ø	74	4A	J	1ØØ	64	d	126	7E	~
23	17		49	31	1	75	4B	K	1Ø1	65	e	127	7F	DEL
24	18		5Ø	32	2	76	4C	L	1Ø2	66	f			
25	19		51	33	3	77	4D	M	1Ø3	67	g			

Tabulky zobrazení celých dekadických čísel

D-DV ve dvojkovém kódu

D-DD ve dvojkovém doplňkovém kódu

D-DP ve dvojkovém přímém kódu

Pro celá dekadická čísla do Ø do 127 jsou všechny tři kódy totožné.

Příklad:

Kódem 1Ø1Ø ØØØØ je vyjádřeno dekadické číslo 16Ø ve dvojkovém kódu /D-DV/, dekadické číslo -96 ve dvojkovém doplňkovém kódu /D-DD/ a dekadické číslo -32 v přímém dvojkovém kódu /D-DP/.

D-DV D-DD D-DP	kód	D-DV D-DD D-DP	kód	D-DV D-DD D-DP	kód	D-DV D-DD D-DP	kód
Ø	ØØØØ ØØØØ	32	ØØ1Ø ØØØØ	64	Ø1ØØ ØØØØ	96	Ø11Ø ØØØØ
1	ØØØØ ØØØ1	33	ØØ1Ø ØØØ1	65	Ø1ØØ ØØØ1	97	Ø11Ø ØØØ1
2	ØØØØ ØØ1Ø	34	ØØ1Ø ØØ1Ø	66	Ø1ØØ ØØ1Ø	98	Ø11Ø ØØ1Ø
3	ØØØØ ØØ11	35	ØØ1Ø ØØ11	67	Ø1ØØ ØØ11	99	Ø11Ø ØØ11
4	ØØØØ Ø1ØØ	36	ØØ1Ø Ø1ØØ	68	Ø1ØØ Ø1ØØ	1ØØ	Ø11Ø Ø1ØØ
5	ØØØØ Ø1Ø1	37	ØØ1Ø Ø1Ø1	69	Ø1ØØ Ø1Ø1	1Ø1	Ø11Ø Ø1Ø1
6	ØØØØ Ø11Ø	38	ØØ1Ø Ø11Ø	7Ø	Ø1ØØ Ø11Ø	1Ø2	Ø11Ø Ø11Ø
7	ØØØØ Ø111	39	ØØ1Ø Ø111	71	Ø1ØØ Ø111	1Ø3	Ø11Ø Ø111
8	ØØØØ 1ØØØ	4Ø	ØØ1Ø 1ØØØ	72	Ø1ØØ 1ØØØ	1Ø4	Ø11Ø 1ØØØ
9	ØØØØ 1ØØ1	41	ØØ1Ø 1ØØ1	73	Ø1ØØ 1ØØ1	1Ø5	Ø11Ø 1ØØ1
1Ø	ØØØØ 1Ø1Ø	42	ØØ1Ø 1Ø1Ø	74	Ø1ØØ 1Ø1Ø	1Ø6	Ø11Ø 1Ø1Ø
11	ØØØØ 1Ø11	43	ØØ1Ø 1Ø11	75	Ø1ØØ 1Ø11	1Ø7	Ø11Ø 1Ø11
12	ØØØØ 11ØØ	44	ØØ1Ø 11ØØ	76	Ø1ØØ 11ØØ	1Ø8	Ø11Ø 11ØØ
13	ØØØØ 11Ø1	45	ØØ1Ø 11Ø1	77	Ø1ØØ 11Ø1	1Ø9	Ø11Ø 11Ø1
14	ØØØØ 111Ø	46	ØØ1Ø 111Ø	78	Ø1ØØ 111Ø	11Ø	Ø11Ø 111Ø
15	ØØØØ 1111	47	ØØ1Ø 1111	79	Ø1ØØ 1111	111	Ø11Ø 1111
16	ØØØ1 ØØØØ	48	ØØ11 ØØØØ	8Ø	Ø1Ø1 ØØØØ	112	Ø111 ØØØØ
17	ØØØ1 ØØØ1	49	ØØ11 ØØØ1	81	Ø1Ø1 ØØØ1	113	Ø111 ØØØ1
18	ØØØ1 ØØ1Ø	5Ø	ØØ11 ØØ1Ø	82	Ø1Ø1 ØØ1Ø	114	Ø111 ØØ1Ø
19	ØØØ1 ØØ11	51	ØØ11 ØØ11	83	Ø1Ø1 ØØ11	115	Ø111 ØØ11
2Ø	ØØØ1 Ø1ØØ	52	ØØ11 Ø1ØØ	84	Ø1Ø1 Ø1ØØ	116	Ø111 Ø1ØØ
21	ØØØ1 Ø1Ø1	53	ØØ11 Ø1Ø1	85	Ø1Ø1 Ø1Ø1	117	Ø111 Ø1Ø1
22	ØØØ1 Ø11Ø	54	ØØ11 Ø11Ø	86	Ø1Ø1 Ø11Ø	118	Ø111 Ø11Ø
23	ØØØ1 Ø111	55	ØØ11 Ø111	87	Ø1Ø1 Ø111	119	Ø111 Ø111
24	ØØØ1 1ØØØ	56	ØØ11 1ØØØ	88	Ø1Ø1 1ØØØ	12Ø	Ø111 1ØØØ
25	ØØØ1 1ØØ1	57	ØØ11 1ØØ1	89	Ø1Ø1 1ØØ1	121	Ø111 1ØØ1
26	ØØØ1 1Ø1Ø	58	ØØ11 1Ø1Ø	9Ø	Ø1Ø1 1Ø1Ø	122	Ø111 1Ø1Ø
27	ØØØ1 1Ø11	59	ØØ11 1Ø11	91	Ø1Ø1 1Ø11	123	Ø111 1Ø11
28	ØØØ1 11ØØ	6Ø	ØØ11 11ØØ	92	Ø1Ø1 11ØØ	124	Ø111 11ØØ
29	ØØØ1 11Ø1	61	ØØ11 11Ø1	93	Ø1Ø1 11Ø1	125	Ø111 11Ø1
3Ø	ØØØ1 111Ø	62	ØØ11 111Ø	94	Ø1Ø1 111Ø	126	Ø111 111Ø
31	ØØØ1 1111	63	ØØ11 1111	95	Ø1Ø1 1111	127	Ø111 1111

D-DD	D-DP	D-DV	kód	D-DD	D-DP	D-DV	kód
-128	-Ø	128	1ØØØ ØØØØ	-96	-32	16Ø	1Ø1Ø ØØØØ
-127	-1	129	1ØØØ ØØØ1	-95	-33	161	1Ø1Ø ØØØ1
-126	-2	13Ø	1ØØØ ØØ1Ø	-94	-34	162	1Ø1Ø ØØ1Ø
-125	-3	131	1ØØØ ØØ11	-93	-35	163	1Ø1Ø ØØ11
-124	-4	132	1ØØØ Ø1ØØ	-92	-36	164	1Ø1Ø Ø1ØØ
-123	-5	133	1ØØØ Ø1Ø1	-91	-37	165	1Ø1Ø Ø1Ø1
-122	-6	134	1ØØØ Ø11Ø	-9Ø	-38	166	1Ø1Ø Ø11Ø
-121	-7	135	1ØØØ Ø111	-89	-39	167	1Ø1Ø Ø111
-12Ø	-8	136	1ØØØ 1ØØØ	-88	-4Ø	168	1Ø1Ø 1ØØØ
-119	-9	137	1ØØØ 1ØØ1	-87	-41	169	1Ø1Ø 1ØØ1
-118	-1Ø	138	1ØØØ 1Ø1Ø	-86	-42	17Ø	1Ø1Ø 1Ø1Ø
-117	-11	139	1ØØØ 1Ø11	-85	-43	171	1Ø1Ø 1Ø11
-116	-12	14Ø	1ØØØ 11ØØ	-84	-44	172	1Ø1Ø 11ØØ
-115	-13	141	1ØØØ 11Ø1	-83	-45	173	1Ø1Ø 11Ø1
-114	-14	142	1ØØØ 111Ø	-82	-46	174	1Ø1Ø 111Ø
-113	-15	143	1ØØØ 1111	-81	-47	175	1Ø1Ø 1111
-112	-16	144	1ØØ1 ØØØØ	-8Ø	-48	176	1Ø11 ØØØØ
-111	-17	145	1ØØ1 ØØØ1	-79	-49	177	1Ø11 ØØØ1
-11Ø	-18	146	1ØØ1 ØØ1Ø	-78	-5Ø	178	1Ø11 ØØ1Ø
-1Ø9	-19	147	1ØØ1 ØØ11	-77	-51	179	1Ø11 ØØ11
-1Ø8	-2Ø	148	1ØØ1 Ø1ØØ	-76	-52	18Ø	1Ø11 Ø1ØØ
-1Ø7	-21	149	1ØØ1 Ø1Ø1	-75	-53	181	1Ø11 Ø1Ø1
-1Ø6	-22	15Ø	1ØØ1 Ø11Ø	-74	-54	182	1Ø11 Ø11Ø
-1Ø5	-23	151	1ØØ1 Ø111	-73	-55	183	1Ø11 Ø111
-1Ø4	-24	152	1ØØ1 1ØØØ	-72	-56	184	1Ø11 1ØØØ
-1Ø3	-25	153	1ØØ1 1ØØ1	-71	-57	185	1Ø11 1ØØ1
-1Ø2	-26	154	1ØØ1 1Ø1Ø	-7Ø	-58	186	1Ø11 1Ø1Ø
-1Ø1	-27	155	1ØØ1 1Ø11	-69	-59	187	1Ø11 1Ø11
-1ØØ	-28	156	1ØØ1 11ØØ	-68	-6Ø	188	1Ø11 11ØØ
-99	-29	157	1ØØ1 11Ø1	-67	-61	189	1Ø11 11Ø1
-98	-3Ø	158	1ØØ1 111Ø	-66	-62	19Ø	1Ø11 111Ø
-97	-31	159	1ØØ1 1111	-65	-63	191	1Ø11 1111

D-DD	D-DP	D-DV	kód	D-DD	D-DP	D-DV	kód
-64	-64	192	11ØØ ØØØØ	-32	-96	224	111Ø ØØØØ
-63	-65	193	11ØØ ØØØ1	-31	-97	225	111Ø ØØØ1
-62	-66	194	11ØØ ØØ1Ø	-3Ø	-98	226	111Ø ØØ1Ø
-61	-67	195	11ØØ ØØ11	-29	-99	227	111Ø ØØ11
-6Ø	-68	196	11ØØ Ø1ØØ	-28	-1ØØ	228	111Ø Ø1ØØ
-59	-69	197	11ØØ Ø1Ø1	-27	-1Ø1	229	111Ø Ø1Ø1
-58	-7Ø	198	11ØØ Ø11Ø	-26	-1Ø2	23Ø	111Ø Ø11Ø
-57	-71	199	11ØØ Ø111	-25	-1Ø3	231	111Ø Ø111
-56	-72	2ØØ	11ØØ 1ØØØ	-24	-1Ø4	232	111Ø 1ØØØ
-55	-73	2Ø1	11ØØ 1ØØ1	-23	-1Ø5	233	111Ø 1ØØ1
-54	-74	2Ø2	11ØØ 1Ø1Ø	-22	-1Ø6	234	111Ø 1Ø1Ø
-53	-75	2Ø3	11ØØ 1Ø11	-21	-1Ø7	235	111Ø 1Ø11
-52	-76	2Ø4	11ØØ 11ØØ	-2Ø	-1Ø8	236	111Ø 11ØØ
-51	-77	2Ø5	11ØØ 11Ø1	-19	-1Ø9	237	111Ø 11Ø1
-5Ø	-78	2Ø6	11ØØ 111Ø	-18	-11Ø	238	111Ø 111Ø
-49	-79	2Ø7	11ØØ 1111	-17	-111	239	111Ø 1111
-48	-8Ø	2Ø8	11Ø1 ØØØØ	-16	-112	24Ø	1111 ØØØØ
-47	-81	2Ø9	11Ø1 ØØØ1	-15	-113	241	1111 ØØØ1
-46	-82	21Ø	11Ø1 ØØ1Ø	-14	-114	242	1111 ØØ1Ø
-45	-83	211	11Ø1 ØØ11	-13	-115	243	1111 ØØ11
-44	-84	212	11Ø1 Ø1ØØ	-12	-116	244	1111 Ø1ØØ
-43	-85	213	11Ø1 Ø1Ø1	-11	-117	245	1111 Ø1Ø1
-42	-86	214	11Ø1 Ø11Ø	-1Ø	-118	246	1111 Ø11Ø
-41	-87	215	11Ø1 Ø111	-9	-119	247	1111 Ø111
-4Ø	-88	216	11Ø1 1ØØØ	-8	-12Ø	248	1111 1ØØØ
-39	-89	217	11Ø1 1ØØ1	-7	-121	249	1111 1ØØ1
-38	-9Ø	218	11Ø1 1Ø1Ø	-6	-122	25Ø	1111 1Ø1Ø
-37	-91	219	11Ø1 1Ø11	-5	-123	251	1111 1Ø11
-36	-92	22Ø	11Ø1 11ØØ	-4	-124	252	1111 11ØØ
-35	-93	221	11Ø1 11Ø1	-3	-125	253	1111 11Ø1
-34	-94	222	11Ø1 111Ø	-2	-126	254	1111 111Ø
-33	-95	223	11Ø1 1111	-1	-127	255	1111 1111

Jazykový původ instrukcí strojového programování

Většina instrukcí - například SUB, LDA, XCHG, JNZ a další - vznikla z určitých charakteristických písmen anglických frází, které popisují činnost dané instrukce. Konkrétně instrukce JNZ vznikla z anglického výrazu "Jump if Not Zero" /čti: džamp if not zerou/. V následujícím abecedním přehledu všech takových instrukcí uvádíme anglickou frázi, její fonetický přepis a překlad v následujícím tvaru:

instrukce	fonetický přepis angl. fráze
anglická fráze	česká varianta

Při hledání české varianty anglické fráze je nutno volit někdy volnější překlad. Pro zajímavost v některých případech uvádíme i doslovný překlad.

Záměrně se ve fonetických přepisech vyhýbáme fonetickým značkám, které se uvádějí v anglických slovnících, abychom začátečníkovi nekomplikovali příliš situaci a v přepisech používáme běžných písmen tak, aby přepis byl co nejvěrnější. Výjimku tvoří znak -th- , který v přepisu budeme někdy značit đ a číst -dz- a jindy θ a číst jako souhlásku foneticky mezi f a s.

Ostatní instrukce je nutno chápat spíše jako zjednodušená symbolická vyjádření - třeba SPHL vznikla spojením značek pro registry SP a HL, nebo k instrukcím vzniklým z anglických frází byl přidán další znak - viz LDA a LDAX - aby se vystihl rozdíl mezi způsobem adresování apod.

K takovým instrukcím je možno počítat:

DCX, INX, LDAX, PCHL, SPHL, STAX, XTHL .

ACI add with carry immediate	éd wid kery imidjét přičti s přenosem přímý operand doslovně: nejbližší
ADC add with carry	éd wid kery přičti s přenosem
ADI add immediate	éd imidjét přičti přímý operand
ANA logical "AND"	lodžikl end logická konjunkce
ANI logical "AND" immediate	lodžikl end imidjét logická konjunkce s přímým operandem
CAAL call	kól vyvolej /podprogram/
CC call if carry set	kól if kery set vyvolej podprogram, je-li přenos 1 /doslovně: je-li přenos nastaven/
CM call if sign negative	kól if sain negetiv vyvolej, je-li znaménko záporné
CMA complement accumulator	kompliment ekjúmjuleite doplněk /jedničkový/ střadače
CMC complement carry	kompliment kery doplněk /jedničkový/ přenosu
CMP compare	kempér srovnej

CNC call if not carry set	kól if not kery set vyvolej, není-li přenos 1 /doslovně: není-li přenos nastaven/
CNZ call if not zero	kól if not zerou vyvolej, pokud není nula
CP call if sign positive	kól if sain pozitiv vyvolej, je-li znaménko kladné
CPE call if parity even	kól if parity ívn vyvolej, je-li parita sudá
CPI compare immediate	kempér imidjét srovnej s přímým operandem
CPO call if parity odd	kól if parity od vyvolej, je-li parita lichá
CZ call if zero	kól if zerou vyvolej, je-li nula
DAA decimal adjust accumulator	desiml edžast ekjúmjuleite decimální nastavení střadače
DAD double add	dabl éd dvojité sečítání
DCR decrement	dikriment snížení
DI disable interrupt	diseibl interapt zákázání přerušení /doslovně: neschopnost přerušit/

EI enable interrupt	ineibl interapt přerušení povoleno
HLT halt	hélt zastav se
IN input	inpat vstup /údaje/
INR increment	inkriment zvýšení
JC jump if carry set	džamp if kery set skoč, je-li přenos 1
JM jump if sign negative	džamp if sain negetiv skoč, je-li znaménko záporné
JMP jump	džamp skoč
JNC jump if not carry set	džamp if not kery set skoč, není-li přenos 1
JNZ jump if not zero	džamp if not zerou skoč, není-li nula
JP jump if sign pozitive	džamp if sain pozitiv skoč, je-li znaménko kladné
JPE jump if parity even	džamp if parity ívn skoč, je-li parita sudá
JPO jump if parity odd	džamp if parity od skoč, je-li parita lichá

JZ jump if zero	džamp if zerou skoč, je-li nula
LDA load accumulator	loud ekjúmjuleite naplň střadač /akumulátor/
LHLD load HL	loud HL naplň HL
LXI load immediate	loud imidjét naplň přímým operandem
MOV move	múv přesuň
MVI move immediate	múv imidjét přesuň přímý operand
NOP no operation	nou operejšn žádná operace
ORA logical "OR"	lodžikl ór logické "nebo"
ORI logical "OR" immediate	lodžikl "ór" imidjét logické "nebo" s přímým operandem
OUT uotput	autpat výstup
POP pop	pop vyjmi

PUSH push	puš strč, vlož
RAL rotate left throw carry	roteit left θrú kery rotace vlevo s vynesením přenosu
RAR rotate right throw carry	roteit rait θrú kery rotace vpravo s vynesením přenosu
RC return if carry set	ritén if kery set vrať se, je-li přenos 1
RET return	ritén vrať se
RLC rotate left circular	roteit left sékjule otoč /posuň/ vlevo kruhově
RM return if sign negative	ritén if sain negetiv vrať se, je-li znaménko záporné
RNC return if not carry set	ritén if not kery set vrať se, není-li přenos 1
RNZ return if not zero	ritén if not zerou vrať se, není-li nula
RP return if sign positive	ritén if sain pozitiv vrať se, je-li znaménko kladné
RPE return if parity even	ritén if parity ívn vrať se, je-li parita sudá

RPO return if parity odd	ritén if parity od vrať se, je-li parita lichá
RRC rotate right circular	roteit rait sékjule otoč /přesuň/ vpravo kruhově
RST restart	ristárt znovuspuštění
RZ return if zero	ritén if zerou vrať se, je-li nula
SBB subtract with carry	sebtrékt wið kery odečti s přenosem
SBI subtract with carry immediate	sebtrékt wið kery imidjét odečti přímý operand s přenosem
SHLD store HL	stó HL ulož obsah HL
STA store accumulator	stó ekjúmjuleite ulož obsah střadače /akumulátoru/
STC set carry	set kery nastav přenos /na 1/
SUB subtract	sebtrékt odečti
SUI subtract immediate	sebtrékt imidjét odečti přímý operand

XCHG	ikščeindž
exchange	výměna
XRA	iksklúsiv "ór"
exclusive "OR"	vylučovací "nebo"
XRI	iksklúsiv "ór" imidjét
exclusive "OR" immediate	vylučovací "nebo" s přímým operandem

Označení registrů

A	ekjúmjuleite
accumulator	střadač, akumulátor
F	flég
flag	registr příznaků /doslovně: vlajkový registr/
PC	prougrem kaunter
program counter	čítač programových instrukcí
SP	sték pointer
stak pointer	ukazatel zásobníku

Značení příznaků

AC	ógziliery kery
auxiliary carry	pomocný přenos
C	kery
carry	přenos
P	parity
parity	parita
S	sain
sign	znaménko
Z	zerou
zero	nula

Tabulky instrukcí strojového kódu

V jedné tabulce jsou instrukce uspořádány abecedně, ve druhé jsou uspořádány podle svého kódu v hexadecimální číselné soustavě. K určitým hexadecimálním kódům neexistuje instrukce - jsou to: Ø8, 1Ø, 18, 2Ø, 28, 3Ø, 38, CB, D9, DD, ED, FD /hexadecimálně/. Instrukcí je tedy pouze 244.

Vysvětlivky:

značka	význam
d	konstanta 8 bitů
w	konstanta 2 x 8 bitů
a	adresa 2 x 8 bitů
"	nastavuje všechny bity CY, Z , S , P , AC registru příznaků F
+	nastavuje pouze CY registru příznaků F
!	nastavuje bity Z , S , P , AC, nikoliv CY registru příznaků F
M	adresa uložená v dvojregistru HL

ACI d	CE	"	DCX D	1B		MOV D,B	5Ø		POP PSW	F1	"
ADC A	8F	"	DCX H	2B		MOV D,C	51		PUSH B	C5	
ADC B	88	"	DCX SP	3B		MOV D,D	52		PUSH D	D5	
ADC C	89	"	DI	F3		MOV D,E	53		PUSH H	E5	
ADC D	8A	"	EI	FB		MOV D,H	54		PUSH PSW	F5	
ADC E	8B	"	HLT	76		MOV D,L	55		RAL	17	+
ADC H	8C	"	IN d	DB		MOV D,M	56		RAR	1F	+
ADC L	8D	"	INR A	3C	!	MOV E,A	5F		RC	D8	
ADC M	8E	"	INR B	Ø4	!	MOV E,B	58		RET	C9	
ADD A	87	"	INR C	ØC	!	MOV E,C	59		RLC	Ø7	+
ADD B	8Ø	"	INR D	14	!	MOV E,D	5A		RM	F8	
ADD C	81	"	INR E	1C	!	MOV E,E	5B		RNC	DØ	
ADD D	82	"	INR H	24	!	MOV E,H	5C		RNZ	CØ	
ADD E	83	"	INR L	2C	!	MOV E,L	5D		RP	FØ	
ADD H	84	"	INR M	34	!	MOV E,M	5E		RPE	E8	
ADD L	85	"	INX B	Ø3		MOV H,A	67		RPO	EØ	
ADD M	86	"	INX D	13		MOV H,B	6Ø		RRC	ØF	+
ADI d	C6	"	INX H	23		MOV H,C	61		RST Ø	C7	
ANA A	A7	"	INX SP	33		MOV H,D	62		RST 1	CF	
ANA B	AØ	"	JC a	DA		MOV H,E	63		RST 2	D7	
ANA C	A1	"	JM a	FA		MOV H,H	64		RST 3	DF	
ANA D	A2	"	JMP a	C3		MOV H,L	65		RST 4	E7	
ANA E	A3	"	JNC a	D2		MOV H,M	66		RST 5	EF	
ANA H	A4	"	JNZ a	C2		MOV L,A	6F		RST 6	F7	
ANA L	A5	"	JP a	F2		MOV L,B	68		RST 7	FF	
ANA M	A6	"	JPE a	EA		MOV L,C	69		RZ	C8	
ANI d	E6	"	JPO a	E2		MOV L,D	6A		SBB A	9F	"
CALL a	CD		JZ a	CA		MOV L,E	6B		SBB B	98	"
CC a	DC		LDA a	3A		MOV L,H	6C		SBB C	99	"
CM a	FC		LDAX B	ØA		MOV L,L	6D		SBB D	9A	"
CMA	2F		LDAX D	1A		MOV L,M	6E		SBB E	9B	"
CMC	3F	+	LHLD a	2A		MOV M,A	77		SBB H	9C	"
CMP A	BF	"	LXI B,w	Ø1		MOV M,B	7Ø		SBB L	9D	"
CMP B	B8	"	LXI D,w	11		MOV M,C	71		SBB M	9E	"
CMP C	B9	"	LXI H,w	21		MOV M,D	72		SBI d	DE	"
CMP D	BA	"	LXI SP,w	31		MOV M,E	73		SHLD a	22	
CMP E	BB	"	MOV A,A	7F		MOV M,H	74		SPHL	F9	
CMP H	BC	"	MOV A,B	78		MOV M,L	75		STA a	32	
CMP L	BD	"	MOV A,C	79		MVI A,d	3E		STAX B	Ø2	
CMP M	BE	"	MOV A,D	7A		MVI B,d	Ø6		STAX D	12	
CNC a	D4		MOV A,E	7B		MVI C,d	ØE		STC	37	+
CNZ a	C4		MOV A,H	7C		MVI D,d	16		SUB A	97	"
CP a	F4		MOV A,L	7D		MVI E,d	1E		SUB B	9Ø	"
CPE a	EC		MOV A,M	7E		MVI H,d	26		SUB C	91	"
CPI d	FE	"	MOV B,A	47		MVI L,d	2E		SUB D	92	"
CPO a	E4		MOV B,B	4Ø		MVI M,d	36		SUB E	93	"
CZ a	CC		MOV B,C	41		NOP	ØØ		SUB H	94	"
DAA	27	"	MOV B,D	42		ORA A	B7	"	SUB L	95	"
DAD B	Ø9	+	MOV B,E	43		ORA B	BØ	"	SUB M	96	"
DAD D	19	+	MOV B,H	44		ORA C	B1	"	SUI d	D6	"
DAD H	29	+	MOV B,L	45		ORA D	B2	"	XCHG	EB	
DAD SP	39	+	MOV B,M	46		ORA E	B3	"	XRA A	AF	"
DCR A	3D	!	MOV C,A	4F		ORA H	B4	"	XRA B	A8	"
DCR B	Ø5	!	MOV C,B	48		ORA L	B5	"	XRA C	A9	"
DCR C	ØD	!	MOV C,C	49		ORA M	B6	"	XRA D	AA	"
DCR D	15	!	MOV C,D	4A		ORI d	F6	"	XRA E	AB	"
DCR E	1D	!	MOV C,E	4B		OUT d	D3		XRA H	AC	"
DCR H	25	!	MOV C,H	4C		PCHL	E9		XRA L	AD	"
DCR L	2D	!	MOV C,L	4D		POP B	C1		XRA M	AE	"
DCR M	35	!	MOV C,M	4E		POP D	D1		XRI d	EE	"
DCX B	ØB		MOV D,A	57		POP H	E1		XTHL	E3	

ØØ NOP	44 MOV B,H	81 ADD C	BE CMP M
Ø1 LXI B,w	45 MOV B,L	82 ADD D	BF CMP A
Ø2 STAX B	46 MOV B,M	83 ADD E	CØ RNZ
Ø3 INX B	47 MOV B,A	84 ADD H	C1 POP B
Ø4 INR B	48 MOV C,B	85 ADD L	C2 JNZ a
Ø5 DCR B	49 MOV C,C	86 ADD M	C3 JMP a
Ø6 MVI B,d	4A MOV C,D	87 ADD A	C4 CNZ a
Ø7 RLC	4B MOV C,E	88 ADC B	C5 PUSH B
Ø9 DAD B	4C MOV C,H	89 ADC C	C6 ADI d
ØA LDAX B	4D MOV C,L	8A ADC D	C7 RST Ø
ØB DCX B	4E MOV C,M	8B ADC E	C8 RZ
ØC INR C	4F MOV C,A	8C ADC H	C9 RET
ØD DCR C	5Ø MOV D,B	8D ADC L	CA JZ a
ØE MVI C,d	51 MOV D,C	8E ADC M	CC CZ a
ØF RRC	52 MOV D,D	8F ADC A	CD CALL a
11 LXI D,w	53 MOV D,E	9Ø SUB B	CE ACI d
12 STAX D	54 MOV D,H	91 SUB C	CF RST 1
13 INX D	55 MOV D,L	92 SUB D	DØ RNC
14 INR D	56 MOV D,M	93 SUB E	D1 POP D
15 DCR D	57 MOV D,A	94 SUB H	D2 JNC a
16 MVI D,d	58 MOV E,B	95 SUB L	D3 OUT d
17 RAL	59 MOV E,C	96 SUB M	D4 CNC a
19 DAD D	5A MOV E,D	97 SUB A	D5 PUSH D
1A LDAX D	5B MOV E,E	98 SBB B	D6 SUI d
1B DCX D	5C MOV E,H	99 SBB C	D7 RST 2
1C INR E	5D MOV E,L	9A SBB D	D8 RC
1D DCR E	5E MOV E,M	9B SBB E	DA JC a
1E MVI E,d	5F MOV E,A	9C SBB H	DB IN d
1F RAR	6Ø MOV H,B	9D SBB L	DC CC a
21 LXI H,w	61 MOV H,C	9E SBB M	DE SBI d
22 SHLD a	62 MOV H,D	9F SBB A	DF RST 3
23 INX H	63 MOV H,E	AØ ANA B	EØ RPO
24 INR H	64 MOV H,H	A1 ANA C	E1 POP H
25 DCR H	65 MOV H,L	A2 ANA D	E2 JPO a
26 MVI H,d	66 MOV H,M	A3 ANA E	E3 XTHL
27 DAA	67 MOV H,A	A4 ANA H	E4 CPO a
29 DAD H	68 MOV L,B	A5 ANA L	E5 PUSH H
2A LHLD a	69 MOV L,C	A6 ANA M	E6 ANI d
2B DCX H	6A MOV L,D	A7 ANA A	E7 RST 4
2C INR L	6B MOV L,E	A8 XRA B	E8 RPE
2D DCR L	6C MOV L,H	A9 XRA C	E9 PCHL
2E MVI L,d	6D MOV L,L	AA XRA D	EA JPE a
2F CMA	6E MOV L,M	AB XRA E	EB XCHG
31 LXI SP,w	6F MOV L,A	AC XRA H	EC CPE a
32 STA a	7Ø MOV M,B	AD XRA L	EE XRI d
33 INX SP	71 MOV M,C	AE XRA M	EF RST 5
34 INR M	72 MOV M,D	AF XRA A	FØ RP
35 DCR M	73 MOV M,E	BØ ORA B	F1 POP PSW
36 MVI M,d	74 MOV M,H	B1 ORA C	F2 JP a
37 STC	75 MOV M,L	B2 ORA D	F3 DI
39 DAD SP	76 HLT	B3 ORA E	F4 CP a
3A LDA a	77 MOV M,A	B4 ORA H	F5 PUSH PSW
3B DCX SP	78 MOV A,B	B5 ORA L	F6 ORI d
3C INR A	79 MOV A,C	B6 ORA M	F7 RST 6
3D DCR A	7A MOV A,D	B7 ORA A	F8 RM
3E MVI A,d	7B MOV A,E	B8 CMP B	F9 SPHL
3F CMC	7C MOV A,H	B9 CMP C	FA JM a
4Ø MOV B,B	7D MOV A,L	BA CMP D	FB EI
41 MOV B,C	7E MOV A,M	BB CMP E	FC CM a
42 MOV B,D	7F MOV A,A	BC CMP H	FE CPI d
43 MOV B,E	8Ø ADD B	BD CMP L	FF RST 7

OBSAH

str.:

Dodatek - tabulka pro přepočet hexadecimálních čísel na čísla decimální

HEX	X...	.X..	..X.	...X
1	4Ø96	256	16	1
2	8192	512	32	2
3	12288	768	48	3
4	16384	1Ø24	64	4
5	2Ø48Ø	128Ø	8Ø	5
6	24576	1536	96	6
7	28672	1792	112	7
8	32768	2Ø48	128	8
9	36864	23Ø4	144	9
A	4Ø96Ø	256Ø	16Ø	1Ø
B	45Ø56	2816	176	11
C	49152	3Ø72	192	12
D	53248	3328	2Ø8	13
E	57344	3584	224	14
F	6144Ø	384Ø	24Ø	15

Postup:

1. cifra je B, ve sloupci X... najdeme 45Ø56
2. cifra je 1, ve sloupci .X.. najdeme 256
3. cifra je Ø, sepíšeme Ø
4. cifra je F, ve sloupci ...X najdeme 15

Součet všech vyhledaných čísel je 45327

Výsledek:

Hexadecimální číslo B1ØF má decimální tvar 45327.

Strojový kód IQ 151

Autor: RNDr. Miloslav Feil, CSc.

Recenze: RNDr. František Lustig

PaedDr. Jan Kuchař

Schválilo ministerstvo školství ČSR dne 3. 1. 1986

č. j. 7068/86 - 211 jako učební pomůcku k počítači

IQ 151 pro střední školy

Vydalo Komenium, n. p. jako učební pomůcku pro

střední školy pod ČKL 02076 v roce 1986

Odpovědný redaktor: PhDr. Emílie Petráková

Technický redaktor: Martina Mášová

Číslo publikace: 57-172-86

Náklad: 10000 kusů

Vydání: 1.