Program mikroprocesoru, instrukce

Program se skládá z jednotlivých příkazů – ***instrukcí***. V paměti je uložen ve formě strojového kódu. Ten vznikne překladem zdrojového textu programu napsaného v asembleru nebo v některém vyšším programovacím jazyce.

Rozlišujeme procesory:

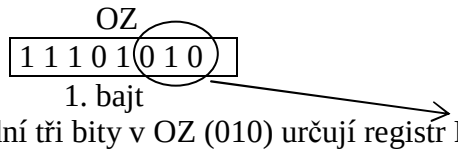
- s úplným souborem instrukcí (architektura CISC – complet instruction set computer)
- s omezeným souborem instrukcí (architekura RISC – reduce instruktion set computer)

Některé procesory mají prvky obou architektur.

U procesorů, které mají paměť (paměť programu) organizovanou po slabikách, jsou instrukce různé délky (např. jednočipový mikropočítač 8051 má instrukce jedno, dvou a tříbajtové).

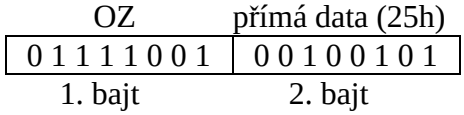
Příklady instrukcí (8051):

MOV A,R2 přesun obsahu registru R2 do registru A (akumulátor)
instrukce je jednobajtová, obsahuje pouze operační znak (OZ):



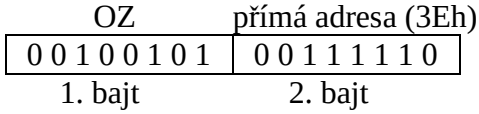
poslední tři bity v OZ (010) určují registr R2

MOV R1,#25h naplnění registru R1 číslem (přímými daty) 25h
instrukce je dvoubajtová:

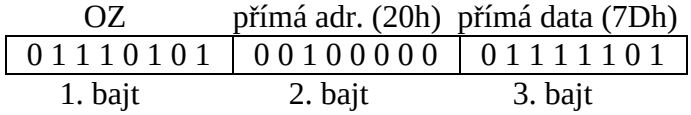


poslední tři bity v OZ (001) určují registr R1

ADD A,3Eh součet obsahu akumulátoru a obsahu paměťového místa o adrese 3Eh, výsledek bude v akumulátoru ((A) + (3Eh) → A)
instrukce je dvoubajtová:



MOV 20H,#7Dh naplnění paměťového místa o adrese 20h číslem 7Dh
instrukce je tříbajtová:



CJNE A,#30h,NAV porovnání obsahu akumulátoru s číslem 30h a skok při *nerovnosti* na adresu danou návěštím NAV
instrukce je tříbajtová:

OZ	přímá data (30h)	relativní adresa (10h)
1 0 1 1 0 1 0 1	0 0 1 1 0 0 0 0	0 0 0 1 0 0 0 0
1. bajt	2. bajt	3. bajt

Relativní adresa je dána rozdílem adresy určené návěštím NAV a adresy, na které je umístěn OZ instrukce následující za instrukcí CJNE. Je-li např. OZ instrukce CJNE umístěn např. na adrese 100h, pak OZ následující instrukce leží na adrese 103h. Jestliže návěští NAV ukazuje na adresu 113h, pak relativní adresa je 113h – 103h = 10h.

	adresy:	obsahy paměťových míst:	
	100h	OZ	instrukce CJNE A,#30h,NAV
	101h	30h	
	102h	10h	
sem ukazuje PC po načtení instrukce CJNE →	103h	OZ následující instrukce	
NAV:	113h		cílová adresa skoku

Relativní adresu vypočítá překladač při překladu programu podle umístění překládané instrukce skoku a polohy návěští odkazujícího na cílovou adresu skoku. Vypočítanou adresu pak vloží do strojového kódu programu.

Tuto relativní adresu při realizaci skoku přičte mikroprocesor k aktuální hodnotě *čítače instrukcí PC*, která je právě 103h, neboť v době vykonávání instrukce čítač instrukcí ukazuje na adresu, na které leží OZ následující instrukce. Tak dojde ke skoku na adresu danou návěštím NAV, tj. další instrukce bude čtena právě z této adresy.

Pokud není podmínka skoku splněna (v tomto případě obsah akumulátoru se rovná 30h), mikroprocesor relativní adresu k obsahu čítače instrukcí PC nepřičte a program bude pokračovat na adrese 103h, tj. instrukcí následující za instrukcí skoku CJNE.

Rozdělení instrukcí:

1. Přesuny dat
2. Aritmetické operace
3. Logické operace a rotace
4. Bitové operace
5. Větvení programu (skoky)
6. Speciální instrukce

Instrukce pro větvení programu:

- instrukce skupiny JUMP (skoky bez uložení návratové adresy)
  - nepodmíněné (skok nastane vždy)
  - podmíněné (skok nastane při splnění určité podmínky)
- instrukce skupiny CALL (skoky s uložením *návratové adresy* do *zásobníku* – volání podprogramu)
- instrukce skupiny RETURN (skoky na adresu naposledy vloženou do zásobníku – návraty z podprogramu)

Větvení programu realizuje instrukce skoku změnou obsahu čítače instrukcí PC. Po načtení jakékoliv instrukce ukazuje PC na následující instrukci. Instrukce skoku však změní obsah PC tak, aby ukazoval na cílovou adresu skoku. Tím bude další instrukce čtena z této adresy, dojde tedy ke skoku v programu.

Instrukce pro větvení programu buď přepíše celý obsah PC cílovou adresou skoku, pak hovoříme o *absolutní adrese*, nebo přičte k stávajícímu obsahu PC *relativní adresu*.

Příklady podmíněných skoků:

Instrukce podmíněných skoků používají relativní adresu dále označenou *rel*, tj. osmibitové číslo se znaménkem (rozsah –128 až +127). V případě splnění podmínky se tato adresa přičte k aktuálnímu obsahu čítače instrukcí PC, tj. k adrese následující instrukce. Tím dojde ke skoku na adresu vzdálenou o *rel* od této aktuální adresy.

JC rel	skok, je-li CY = 1
JNC rel	skok, je-li CY = 0
JZ rel	skok, je-li (A) = 0
JNZ rel	skok, je-li (A) ≠ 0
JB bit, rel	skok, je-li hodnota určeného bitu jedna
CJNE A,#d8,rel	skok, je-li (A) ≠ d8

V programu ovšem místo relativní adresy *rel* píšeme nejčastěji návěští, které určuje cílovou adresu skoku. Relativní adresu pak vypočítá a vloží do strojového kódu programu překladač.

Příklad provedení instrukce JZ NAV:

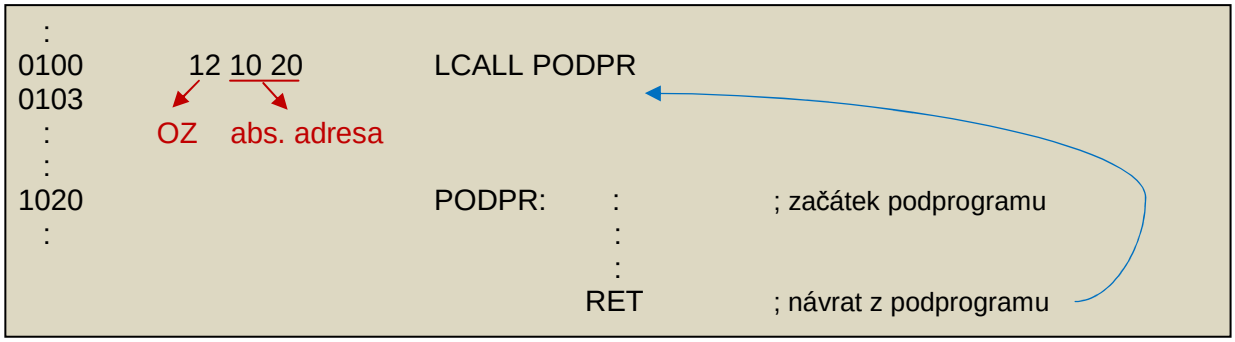
```
adresy:     strojový kód:     text programu:
  :
0100        40 5A                   JZ NAV     ; překladač spočítal relativní adresu jako
0102                                           ; 15Ch – 102h = 5Ah
  :         OZ   rel. adresa
  :
015C                          NAV:  ADD A,#5   ; instrukce ADD A,#5 jen jako příklad
```

Provedení instrukce JZ NAV:

- mikroprocesor načte OZ instrukce JZ rel a zvýší obsah čítače instrukcí PC o jedničku
- mikroprocesor načte druhý bajt instrukce a zvýší obsah čítače instrukcí PC o jedničku, tj. PC ukáže na adresu následující instrukce (102h)
- je-li (A) = 0, přičte se k obsahu PC relativní adresa: 102h + 5Ah = 15Ch, program bude pokračovat na adrese 15Ch

- je-li (A) ≠ 0, relativní adresa se k obsahu PC nepřičte a program bude pokračovat na adrese 102h, tj. na následující instrukci

Volání podprogramu a návrat z podprogramu:



Instrukce LCALL používá 16bitovou absolutní adresu.

Provedení instrukce LCALL a RET:

- mikroprocesor načte OZ instrukce LCALL a následně další dva bajty instrukce, tj. cílovou adresu skoku (1020h)
- uschová do zásobníku *návratovou adresu*, tj. adresu následující instrukce (103h)
- vloží do PC cílovou adresu skoku (1020h), tím dojde ke skoku na první instrukci podprogramu
- instrukce RET na konci podprogramu vyjme návratovou adresu z vrcholku zásobníku a vloží ji do PC – tím dojde ke skoku na instrukci následující za LCALL, tj. k návratu z podprogramu

Otázky a úkoly:

1. Co rozumíme pod pojmem instrukce? Z čeho se instrukce skládá?
2. Co představují zkratky CISC a RISC?
3. Co je uloženo v prvním bajtu instrukce? Co může obsahovat její druhý, příp. třetí bajt?
4. Na kolika paměťových místech je uložena tříbajtová instrukce?
5. Kolikabajtová je instrukce MOV R1,#25h a co představují její jednotlivé bajty?
6. Kolikabajtová je instrukce MOV A,3Eh a co představují její jednotlivé bajty?
7. Co rozumíme pod pojmem relativní adresa?
8. Instrukce skoku SJMP používá relativní adresu. Instrukce je umístěna na adrese 20h, skok má nastat na adresu 2Ah. Určete odpovídající relativní adresu.
9. Vysvětlete rozdíl mezi instrukcemi skoků skupiny JUMP a CALL.
10. Vysvětlete pojem podmíněný skok.
11. Uveďte příklad instrukce podmíněného skoku a popište činnost mikroprocesoru při jejím vykonání.
12. Popište činnost mikroprocesoru při vykonání instrukce CJNE A,#3Ah,Skok.
13. Popište činnost mikroprocesoru při vykonání instrukce LCALL (instrukce používá šestnáctibitovou absolutní adresu).
14. Popište činnost mikroprocesoru při vykonání instrukce RET (návrat z podprogramu).
15. Uvažte, v jakém případě může dojít k chybnému návratu z podprogramu (na špatnou adresu). Je možné použít k návratu z podprogramu instrukci typu JUMP místo RETURN?