Jazyk symbolických adres (asembler)

Jazyk symbolických adres používá ***dva druhy příkazů***:

- instrukce mikroprocesoru v symbolickém zápisu
- pseudoinstrukce

Symbolický zápis instrukce se skládá ze ***symbolu operačního znaku a operandu***.

Např. v zápisu instrukce `MOV A,B` je `MOV` symbolem operačního znaku a `A,B` jsou operandy. Některé instrukce však nepracují s operandem, např. `NOP`, nebo je operand implicitně obsažen v symbolu operačního znaku, např. instrukce `RET` pracuje se dvěma bajty na vrcholu zásobníku.

Pseudoinstrukce jsou příkazy pro práci překladače během překladu.

Zdrojový program je posloupnost příkazů jazyka symbolických adres (instrukcí a pseudoinstrukcí) jak jej napíše programátor.

Cílový program je zdrojový program přeložený do strojového kódu mikroprocesoru překladačem. Překladač jazyka symbolických adres se často nazývá *asembler* (stejně tak jako jazyk samotný).

Řádka zdrojového programu

Řádka zdrojového programu v jazyce symbolických adres se skládá ze čtyř polí:

[NÁVĚŠTÍ:] nebo [JMÉNO]	SYMBOL OZ INSTRUKCE nebo SYMBOL PSEUDOINSTRUKCE	[OPERAND(Y)]	[; KOMENTÁŘ]

Příklady:

```
                MOV             A,#5
ZAC:            MOV             @R0,A       ;začátek cyklu
; komentář může být na samostatném řádku
```

Jednotlivá pole musí být ***oddělena alespoň jednou mezerou, pole komentáře středníkem***.

Pole návěští – obsahuje ***návěští*** nebo ***jméno*** nebo je pole prázdné. Návěští nebo jméno se může skládat z písmen abecedy, číslic 0 až 9 a znaků _, ?, @, nesmí však začínat číslicí. Návěští je zakončeno *dvojtečkou* (jméno nikoliv), při překladu je mu přiřazena konkrétní adresa. Určité návěští se může v poli návěští vyskytnout pouze jednou.

Návěští představuje odkaz na určité místo v programu, *jméno* je symbolické označení konstanty, paměťového místa, bitu, příp. registru.

Pole operačního znaku – obsahuje symbol operačního znaku instrukce nebo symbol pseudoinstrukce (mnemotechnický zápis operace).

Pole operandu – obsahuje operandy instrukce. Vyžaduje-li instrukce operandy dva nebo tři, musí se napsat v předepsaném pořadí a oddělit čárkou (cílový operand, zdrojový operand).

Pole komentáře – komentář musí být oddělen středníkem. Nemá žádný vliv na překlad ani cílový program. Převádí se do výpisu programu a slouží k orientaci v programu.

Způsoby zápisu přímých operandů

Přímými operandy v instrukci či pseudoinstrukci mohou být celá čísla v rozsahu 0 až 255, příp. 0 až 65535 (mohou být zadávána i čísla se znaménkem, záporná čísla překladač převede do dvojkového doplňku). Představují přímá data, přímé adresy v datové paměti nebo v oblasti SFR registrů, relativní nebo absolutní adresy (v případě instrukcí skoků).

Přímé operandy je možné zapsat některým z následujících způsobů:

Hexadecimální (šestnáctkové) číslo – může obsahovat pouze číslice 0 až F, pokud začíná znakem A až F, musí mu být předřazena nula (tak překladač rozpozná, že se jedná o číslo a ne o symbol). Musí být zakončeno znakem H.

Příklad:
```
MOV R2,#0D5H
```

Desítkové číslo – může obsahovat pouze číslice 0 až 9, může být zakončeno znakem D (není povinný, pokud je u překladače desítková soustava nastavena jako výchozí).

Příklad:
```
MOV B,19
```
nebo
```
MOV B,19D
```

Osmičkové číslo – může obsahovat číslice 0 až 7, musí být zakončeno znakem O nebo Q.

Dvojkové číslo – může obsahovat číslice 0 a 1, musí být zakončeno znakem B.

Příklad:
```
ADD A,#11010011B
```

Znaková konstanta ASCII (řetězec) – tvoří ji jeden nebo více znaků uzavřených do apostrofů.

Příklad:
```
MOV R1,#'C'     ;kód znaku C (tj. 43H) do
                ;registru R1
```

Počitadlo adres – hodnotou počitadla adres $ je adresa, na kterou je umísťována právě překládaná instrukce.

Příklad:
```
SKOK:  SJMP $+8        ;skok na adresu o osm vyšší, než
                       ;představuje návěští SKOK
```

Jméno s přiřazenou hodnotou je symbolické označení konstanty, registru nebo paměťového místa či bitu. Hodnota se jménu přiřazuje pseudoinstrukcí EQU, SET, DATA, XDATA, IDATA, BIT.

```
Příklad:  P_ADR  EQU 20H          ;jménu P_ADR je přiřazena
                                  ;hodnota 20H
                    :
                  MOV R0,#P_ADR   ;do registru R0 je vložena
                                  ;hodnota 20H
```

Všude, kde je na místě přímého operandu zapsán symbol P_ADR, překladač dosadí hodnotu 20H. Používání jmen zpřehledňuje program a umožňuje snadno změnit hodnotu konstanty ve všech instrukcích, kde je použita, jejím přepsáním v přiřazovacím příkazu EQU.

Návěští – hodnotou návěští je adresa prvního bajtu instrukce, u které je návěští definováno (tj. uvedeno v poli návěští).

```
Příklad:          LCALL ZPOZD     ;skok na návěští ZPOZD
                                  ;(volání podprogramu)
                    :
          ZPOZD: MOV R2,#0FFH
                    :
```

Výrazy – uvedené přímé operandy je možné spojovat pomocí operátorů do výrazů, které mohou být zapsány v instrukcích či pseudoinstrukcích na místě přímých operandů.

Překladač během překladu programu vypočte jednotlivé výrazy a nahradí je v poli operandů jejich hodnotami.

Ve výrazech je možné použít **operátory**:

- aritmetické

  +, -, *, / (celočíselné dělení), MOD (zbytek po dělení)

- logické

  NOT, AND, OR, XOR

- relační – porovnání operandů

  =, <>, >, >=, <, <= (výsledkem porovnání jsou samé nuly při nesplnění nebo samé jedničky při splnění podmínky)

- pro posuny

  y SHR x – posun operandu y o x bitových pozic doprava
  y SHL x - posun operandu y o x bitových pozic doleva

- pro izolaci slabik

  HIGH – vyšší bajt 16bitového slova
  LOW – nižší bajt 16bitového slova

Pseudoinstrukce

Pseudoinstrukce slouží k *řízení překladače* během překladu zdrojového programu. Dělíme je na:

- příkazy pro alokaci programů a dat (umístění programu a dat v paměti)
- definiční příkazy
- příkazy pro rezervaci paměťových míst
- příkazy pro řízení překladu a jeho výpisu

Přehled nejdůležitějších pseudoinstrukcí

ORG nastavení počitadla adres $ (ORIGIN)

```
[návěští:]  ORG  výraz
```

– překladač dosadí do počitadla adres hodnotu výrazu, tzn., že následující instrukce nebo data budou umístěna na adrese dané tímto výrazem
– je-li uvedeno návěští, je mu přiřazena hodnota počitadla adres před zpracováním pseudoinstrukce `ORG`
– každý paměťový prostor (segment) má svoje počitadlo

Příklad:

```
        ORG  1000H   ;umístění OZ další instrukce
                     ;nebo dat v paměťovém místě o
                     ;adrese 1000H
```

EQU přiřazení hodnoty jménu (definice jména) (EQUATE)

```
jméno       EQU  výraz
```

– pseudoinstrukce přiřadí jménu hodnotu výrazu, toto přiřazení *nelze* v průběhu programu měnit

Příklad:

```
DELKA       EQU  0F4H      ;přiřaď jménu DELKA hodnotu 0F4H
              :
            MOV  R0,#DELKA ;vlož do registru R0 hodnotu
                           ;0F4H
```

– pseudoinstrukcí `EQU` je též možno přiřadit jiné jméno registru, např.:

```
BASE        EQU  R0
```

SET přiřazení hodnoty jménu opakovaně (SET)

```
jméno    SET  výraz
```

– přiřazení lze v průběhu programu měnit:

```
KONST    SET  15
          :
KONST    SET  17
```

DATA pojmenování paměťového místa ve vnitřní paměti dat (definice jména adresy)

Příklad:

```
P_ADR    DATA  40H
```

Podobně pojmenujeme paměťová místa v *ostatních paměťových prostorech (segmentech)* pseudoinstrukcemi:

IDATA - definuje jméno adresy paměťového místa v *nepřímo adresovatelné paměti dat*
XDATA - definuje jméno adresy paměťového místa ve *vnější paměti dat*
CODE - definuje jméno adresy paměťového místa v *paměti programu*

BIT pojmenování bitu v bitově adresovatelném prostoru (definice jména bitu)

```
jméno    BIT  výraz
```

– pseudoinstrukce přiřadí jménu adresu přímo adresovatelného bitu, příp. rezervované označení bitu

Příklady:

```
PRVNI    BIT  34H      ;bit o adrese 34H bude
                       ;pojmenován PRVNI
VYSTUP   BIT  TxD
CLK      BIT  P1.0
```

DB definice obsahu paměťových míst v paměti programu (definice bajtů) (DEFINE BYTE)

```
[návěští:]  DB  pol1 [,pol2]  .......
```

– za `pol1`, `pol2`, atd. můžeme dosadit *výraz* (jeho hodnota musí být v rozsahu 0 až 255) nebo *řetězec znaků* uzavřený v apostrofech
– počet položek je omezen jen délkou řádku

Příklad:

```
TAB:    DB 'AAB',7EH,6DH,14,0   ;ulož na paměťová
        ;místa od symbolické adresy TAB čísla 41H,
        ;42H, 42H, 7EH, 6DH, 0EH, 00H
```

DW definice obsahu paměťových míst v paměti programu (definice dvojic bajtů – slov) (DEFINE WORD)

```
[návěští:]  DW  výraz1 [,výraz2]  .......
```

- výraz musí nabývat hodnoty v rozsahu 0 až 65535
- za výraz může být dosazen též řetězec obsahující jeden nebo dva znaky
- počet výrazů je omezen jen délkou řádku

Příklad:

```
ADR:        DW  2010H       ;ulož na symbolickou adresu ADR
                            ;číslo 20H a na symbolickou
                            ;adresu ADR+1 číslo 10H
```

DS rezervování paměťové oblasti (DEFINE STORE)

```
[návěští:]  DS  výraz
```

- hodnota výrazu určuje počet rezervovaných paměťových míst

Příklad:

```
POLE:       DS  10          ;rezervuj 10 paměťových míst
                            ;počínaje adresou POLE
```

Výběr segmentu paměti

CSEG výběr paměti programu (výchozí)

BSEG výběr bitově adresovatelného prostoru dat

DSEG výběr přímo adresovatelného prostoru dat

ISEG výběr nepřímo adresovatelného prostoru dat

Příklad:

```
            DSEG            ; datový segment
            ORG  20H        ; nastavení lokálního počitadla
                            ; adres (BUFFER bude začínat na
                            ; adrese 20H)- totéž co
                            ; DSEG AT 20H
BUFFER:     DS  32          ; buffer o velikosti 32 bajtů
X:          DS  1           ; jednobajtová proměnná
```

INCLUDE vkládání souboru

```
INCLUDE  soubor
```

- pseudoinstrukce vloží do překládaného programu obsah jiného souboru
- pseudoinstrukce může být použita na více místech v tomtéž programu

IF, ELSE, ENDIF podmíněný překlad

```
IF  výraz
příkazy
ELSE
příkazy
ENDIF
```

– je-li výraz různý od nuly, přeloží se příkazy za `IF`, za `ELSE` se ignorují, při nulové hodnotě výrazu je tomu naopak

END konec překladu

```
END
```

– pseudoinstrukce ukončí překlad, případný text za `END` je ignorován

Makroinstrukce

Makroinstrukce je zkrácený zápis předem definované posloupnosti instrukcí. Každý zápis makroinstrukce v textu programu způsobí vložení této posloupnosti do zdrojového programu.

Definice makroinstrukce začíná pseudoinstrukcí `MACRO` a končí pseudoinstrukcí `ENDM`. Instrukce uzavřené těmito pseudoinstrukcemi tvoří *tělo makroinstrukce:*

```
jméno     MACRO  seznam  ;seznam = seznam formálních
                          ;parametrů
            :
          ENDM
```

Při vyvolání makroinstrukce jsou formální parametry nahrazeny skutečnými.

Příklad:

```
PRUMER    MACRO  X,Y        ; uvozující pseudoinstrukce
          MOV  A,X
          MOV  B,A
          MOV  A,Y
          ADD  A,B
          RR  A
          ENDM              ; závěrečná pseudoinstrukce
           :
          PRUMER  20H,21H   ; vyvolání makroinstrukce se
          ; skutečnými parametry 20h a 21H
           :
          PRUMER  30H,31H   ; vyvolání makroinstrukce se
          ; skutečnými parametry 30h a 31H
```

V uvedeném příkladu mají parametry makroinstrukce význam přímé adresy.

Otázky a úkoly:

1. Co jsou to pseudoinstrukce?
2. Jaký je rozdíl mezi instrukcí a pseudoinstrukcí?
3. Vysvětlete rozdíl mezi zdrojovým a cílovým programem.
4. Jakou funkci má návěští a k čemu slouží jméno?
5. Jakým způsobem se přiřadí hodnota návěští a jakým jménu?
6. V následujícím výpisu části programu doplňte adresu v instrukci `JNZ` zapsanou s využitím počitadla adres tak, aby skok nastal na instrukci `XCH A,R1`:

```
        JNZ  ....
        MOV  A,#POCET
        MOV  DPTR,#1100H
        SJMP JINAM
        XCH  A,R1
          :
```

7. Vyčíslete výrazy na místech přímých operandů:
   a) `MOV B,#100 MOD 15` b) `MOV TL0,#LOW 15536`
   c) `MOV TH0,#HIGH 15536` d) `MOV R0,#10/3`
   e) `MOV A,#00011000B SHR 3` f) `MOV A,#NOT(48)+1`
8. Nalezněte chyby v zápisech přímých operandů:
   a) `MOV C8H,A` b) `SUBB A,#11010011` c) `MOV DPTR,#A56DH`
   d) `ADD A,#0A51H` e) `MOV 20H,#1FFH`
9. V čem spočívá rozdíl mezi pseudoinstrukcemi `SET` a `EQU`?
10. Ve zdrojovém textu programu jsou zapsány řádky:

```
        MOV  DPTR,#TAB
          :
        ORG  800H
TAB:    DB   25H,34H,0AH
```

    Jaký bude obsah registru DPTR po vykonání instrukce `MOV DPTR,#TAB`?
11. Nalezněte chyby v zápisu řádky programu:
    a) `DB 0B6H,3A,134,-10,FFH,'ABCD',15AH,01101B,1101`
    b) `DB 25,D4H,1011B,251H,-5,11011,142,'XYZ',352`
12. Vypište obsahy adres `ADR` až `ADR+7` přiřazené pseudoinstrukcí `DW`:

```
ADR: DW 2010H,100H,2110H,20H
```

13. V jakém případě bude zápis `MOV B,D6H` správný a v jakém chybný?
14. Vysvětlete rozdíl mezi makroinstrukcí a podprogramem.