Smyčky

Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

ADCONS

- asembler nám dokáže vrátit absolutní nebo relativní adresu location counteru
- jde o tzv. ADCON – adressing constant
- s touto adresou pak lze provádět různé operace
- je několik typů ADCONů – A, Y, S, R a V
- nyní si ukážeme A

ADCON typu A

```
1  00000400  X       DC    A(X)
```

- konstanta X (fullword) se nastaví na hodnotu své adresy
  - je-li program načten od adresy 1000, bude mít X hodnotu 1400
  - je-li program načten od adresy 2000, bude mít X hodnotu 2400

ADCON typu A

```
1  00000400  X       DC    A(X-BEGIN)
```

- konstanta X se nastaví na hodnotu své adresy, od které je ale odečtena adresa počátku programu
  - je-li program načten od adresy 1000, bude mít X hodnotu 400
  - je-li program načten od adresy 2000, bude mít X hodnotu 400

Tabulky

```
1  TABLE  DC  1000A(((*-TABLE)/L'TABLE)+1)
```

- zde máme tabulku s 1000 prvky
- asembler generuje tabulku tak, že pro každý prvek vyčíslí daný ADCON
- * znamená adresu daného prvku tabulky
- `L'TABLE` znamená velikost jednoho prvku tabulky

Tabulky

```
1  NUMENT    DC    F'4'
2  TABLE     DC    CL5'AAAAA'
3            DC    CL5'BBBBB'
4            DC    CL5'CCCCC'
5            DC    CL5'DDDDD'
6            DC    CL5'EEEEE'
```

- zde si v NUMENT ukládáme index posledního prvku při číslování od nuly
- lze to ale udělat lépe

Tabulky

```
1  NUMENT     DC    A((TABLEEND-TABLE)/L'TABLE)
2  TABLE      DC    CL5'AAAAA'
3             DC    CL5'BBBBB'
4             DC    CL5'CCCCC'
5             DC    CL5'DDDDD'
6             DC    CL5'EEEEE'
7  TABLEEND   EQU   *
```

- NUMENT se teď počítá automaticky, takže její hodnotu nemusíme měnit ručně vždy, když přidáme další prvek
- EQU se používá pro definování aliasů pro asembler
- TABLEEND tedy není uložen v kódu programu, je to jen konstanta pro asembler

Příklad

```
 1             L       R4,NUMENT              LOAD NUMBER OF ENTRIES TO R4
 2             SR      R5,R5                  RESET R5
 3             LA      R6,TABLE               LOAD ADRESS OF TABLE TO R6
 4 LOOP        DS      0H
 5             A       R5,0(,R6)              ADD VALUE OF A TABLE ENTRY TO R5
 6             LA      R6,L'TABLE(,R6)        SET R6 TO THE ADRESS OF THE NEXT ENTRY
 7             BCT     R4,LOOP                LOOP UNTIL ALL DONE
 8 NUMENT      DC    A((TABLEEND-TABLE)/L'TABLE)
 9 TABLE       DC    100A((*-TABLE)/L'TABLE+1)
10 TABLEEND    EQU     *
```

Příklad

```
L R1,D2(X2,B2)
```

- instrukce L sloučí k načtení adresy do registru
- je typu RX

```
LA R1,D2(X2,B2)
LA R6,L'TABLE(,R6)
```

- do R6 se načte adresa daná bázovým registrem R6 a posunem L‘TABLE
- ve výsledku se tak hodnota v R6 zvětší o velikost L‘TABLE, čímž se posuneme na další prvek

Instrukce BCT

```
BCT R1,D2(X2,B2)
```

- obsah registru R1 se zmenší o jedna
- je-li výsledek nulový, pokračuje se další instrukcí
- jinak se provede skok na adresu D2(X2,B2)

Příklad

```
 1                L       R4,NUMENT              LOAD NUMBER OF ENTRIES TO R4
 2                SR      R5,R5                  RESET R5
 3                SR      R6,R6                  RESET R6
 4  LOOP          DS      0H
 5                A       R5,TABLE(R6)           ADD VALUE OF A TABLE ENTRY TO R5
 6                LA      R6,L'TABLE(,R6)        SET R6 TO THE ADRESS OF THE NEXT ENTRY
 7                BCT     R4,LOOP                LOOP UNTIL ALL DONE
 8  NUMENT        DC    A((TABLEEND-TABLE)/L'TABLE)
 9  TABLE         DC    100A((*-TABLE)/L'TABLE+1)
10  TABLEEND      EQU   *
```

Příklad

```
1                LM      R4,R7,LOOPDATA        INITIALIZE THE REGISTERS R4 – R7
2  LOOP          DS      0H
3                A       R5,0(,R4)             ADD VALUE OF A TABLE ENTRY TO R5
4                BXLE    R4,R6,LOOP            LOOP UNTIL ALL DONE
5  LOOPDATA      DC      A(TABLE,0,L'TABLE,TABLEEND–L'TABLE)
6  TABLE         DC      100A((*–TABLE)/L'TABLE+1)
7  TABLEEND      EQU     *
```

- R4 - adresa tabulky
- R5 - součet
- R6 - velikost prvku / inkrement indexování
- R7 - adresa posledního prvku

Instrukce BXLE

Branch on index lower or equal

`BXLE R1,R3,D2(B2)`

- obsah registru R1 se zvětší o hodnotu v registru R3
- je-li pak hodnota v R1 menší nebo rovna hodnotě v registru R3+1, provede se skok na zadanou adresu

Příklad

```
1                LM      R4,R7,LOOPDATA        INITIALIZE THE REGISTERS R4 - R7
2                LCR     R6,R6                 MAKE R6 NEGATIVE
3  LOOP          DS      0H
4                A       R5,0(,R4)             ADD VALUE OF A TABLE ENTRY TO R5
5                BXH     R4,R6,LOOP            LOOP UNTIL ALL DONE
6  LOOPDATA      DC      A(TABLEEND-L'TABLE,0,L'TABLE,TABLE-L'TABLE)
7  TABLE         DC      100A((*-TABLE)/L'TABLE+1)
8  TABLEEND      EQU     *
```

- R4 - adresa posledního prvku v tabulce
- R5 - součet
- R6 - velikost prvku / inkrement indexování
- R7 - adresa prvního prvku

Instrukce LCR

Load complement register

`LCR R1,R2`

- do registru R1 načte hodnotu z registru R2 a změní znaménko

`LNR R1,R2`

- do registru R1 načte hodnotu z registru R2 a změní znaménko na záporné

`LPR R1,R2`

- do registru R1 načte hodnotu z registru R2 a změní znaménko na kladné

Instrukce BXH

Branch on index higher

`BXH R1,R3,D2(B2)`

- k obsah registru R1 se přičte hondota v registru R3
- je-li pak hodnota v R1 větší než hodnota v registru R3+1, provede se skok na zadanou adresu