Aritmetika v assembleru

Tomáš Oberhuber

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

Příklad

Napíšeme program pro výpočet 54321-12345

- dekadicky: `54321-12345=41976`
- hexadecimálně: `x'0000D431'- x'00003039' = x'0000A3F8'`

Příklad na odečítání

```
1   PGM2        SETUP
2               L           2,A
3               S           2,B
4               ST          2,DIFF
5               SNAPSHOT    PGM2,ENDPGM
6   ENDIT
7   A           DC          F'54321'
8   B           DC          F'12345'
9   DIFF        DS          F
10  ENDPGM      DS          D
11              END
```

Příklad na odečítání

- použili jsme instrukci `S` – subtract
- `S R1, D2(X2,B2)`
  - ta odečítá hodnotu na adrese `D2(X2,B2)` od hodnoty v registru `R1`
- nyní můžeme získat záporný výsledek

Kódování záporných čísel

- znaménko celého čísla v binárním zápisu se pozná podle nejvyššího bitu
  - $0 \rightarrow +$
  - $1 \rightarrow -$
- čísla `x'00000000` až `x'7FFFFFFF` jsou kladná
- čísla `x'80000000` až `x'FFFFFFFF` jsou záporná
- aby bylo sčítání kladného a záporného čísla snazší, zachovává se vlastnost růstu čísel od `x'80000000` do `x'FFFFFFFF'`
- proto `x'80000000' = -2,147,483,648` je nejmenší záporné číslo a `x'FFFFFFFF' = -1` je největší záporné číslo
- pokud sečtu `x'00000002' + x'FFFFFFFF'` tj. $2 + (-1)$, dojde k přetečení, ale získáme korektní výsledek `x'00000001'` tj. 1

Kódování záporných čísel

- skutečnou hodnotu záporného binárního čísla získám odečtením od `x'100000000'`
  - Příklad: `x'100000000' - x'FFFFFFFF' = x'00000001'`, tedy hodnota je -1

Násobení

Napíšeme program pro výpočet 54321 * 12345

- dekadicky: `54321 * 12345 = 670592745`
- hexadecimálně: `x'0000D431'* x'00003039' = x'0000000027F86EE9'`

Násobení

```
 1  PGM3        SETUP
 2              L          3,A
 3              M          2,B
 4              ST         2,PROD
 5              ST         3,PROD+4
 6              SNAPSHOT   PGM3,ENDPGM
 7  ENDIT
 8  A           DC         F'54321'
 9  B           DC         F'12345'
10  PROD        DS         D
11  ENDPGM      DS         D
12              END
```

Násobení

- použili jsme instrukci `M R1, D2(X2,B2)`
  - násobí hodnotu v registru `R1` hodnotou na adrese `D2(X2,B2)`
  - výsledek se ukládá do registrů `R1` a `R2`
  - `R1` musí být sudý registr

Dělení

Napíšeme program pro výpočet 54321 / 12345

- dekadicky: `54321 / 12345 = 4 + 4941`
- hexadecimálně: `x'0000D431'/ x'00003039' = x'00000004' (+ x'0000134D)'`

Dělení

```
 1  PGM4        SETUP
 2              L           3,A
 3              SR          2,2
 4              D           2,B
 5              STM         2,3,QUOT
 6              SNAPSHOT    PGM4,ENDPGM
 7  ENDIT
 8  A           DC          F'54321'
 9  B           DC          F'12345'
10  QUOT        DS          D
11  ENDPGM      DS          D
12              END
```

Dělení

Nové instrukce:

- `SR R1,R2` – subtract register
  - odečte od registru `R1` registr `R2`
  - `SR 2,2` – nuluje registr 2
- `D R1,D2(X2,B2)` – divide
  - vydělí číslo v registrech `R1` a `R1+1` (`R1` je sudé) číslem na adrese `D2(X2,B2)`, výsledek se uloží do registru `R1+1` a zbytek po dělení do reistru `R1`
- `STM R1,R3,D2(B2)` – store multiple
  - uloží obsah registrů `R1` až `R3` na adresu `D2(B2)`

Jiné možnosti nulování registru

- `XR 2,2` – exclusive OR
- `LA 2,0` – load adress
- `L 2,=F'0'` – load
- `LHI 2,0` – load halfword immediate

EBCDIC kódování

- systém z/OS používá pro ukládání textu kódování EBCIDIC
- jde o jednobajtové kódování $\Rightarrow$ 256 znaků
- např. mezera má kód `x'40'`
- velká písmena abecedně `x'C1'` – `x'E9'`
- číslice `x'F0'` – `x'F9'`
- číslo `1234`, se zapíše jako `F1 F2 F3 F4`

Formát zoned decimal

- od tohoto zápisu se odvíjí formát *zoned decimal*
- ten přidává na poslední pozici i znaménko
  - x'C' - kladné číslo
  - x'D' - záporné číslo
  - x'F' - bez znaménka (považuje se za kladné)
- příklady
  - `1234` → `x'F1 F2 F3 C4`
  - `-1234` → `x'F1 F2 F3 D4`
- u tohoto formátu je vždy polovina bajtu nevyužita
- toho využívá formát packed decimal

Formát PACKED DECIMAL

- formát vznikl "kompresí" formátu zoned decimal
- tento formát umožňuje pracovat s celými čísly různé délky
- délka se udává explicitně každé instrukci
- kromě nejnižšího bajtu kóduje každý bajt dvě číslice
- nejnižší bajt kóduje jednu číslici a znaménko
  - x'C' - kladné číslo
  - x'D' - záporné číslo
  - x'F' - bez znaménka (považuje se za kladné)
- maximální délka je 16 bajtů tj. 31 číslic

Formát PACKED DECIMAL - ukázka

```
1  A        DC    F'3509'
2  B        DC    P'3509'
3  C        DC    P'2450940'
4  D        DC    P'-94'
```

V paměti se uloží toto:

- A → 00 00 0D 85
- B → 03 50 9C
- C → 24 50 94 0C
- D → 09 4D

Formát PACKED DECIMAL - příklad

```
1   PGM1        SETUP
2               ZAP     SUM,A
3               AP      SUM,B
4               SNAPSHOT  PGM1,ENDPGM
5               ENDIT
6   A           DC      P'54321'
7   B           DC      P'12345'
8   SUM         DS      PL8
9   ENDPGM      DS      F
10              END     PGM1
```

Nové instrukce

- `ZAP D1(L1,B1),D2(L2,B2)` – Zero and Add Pakced decimal
  - druhý operand se překopíruje na místo prvního
- `AP D1(L1,B1),D2(L2,B2)` – Add Packed decimal
  - k prvnímu operandu přičte druhý

- `DS PL8` znamená, že jde o typ packed decimal s velikostí 8 bajtů
- v případě `DC` si určí velikost překladač sám

Formát PACKED DECIMAL - příklad

```
 1  PGM1        SETUP
 2              ZAP     DIFF,A
 3              SP      DIFF,B
 4              SNAPSHOT PGM1,ENDPGM
 5              ENDIT
 6  A           DC      P'54321'
 7  B           DC      P'12345'
 8  DIFF        DS      PL8
 9  ENDPGM      DS      F
10              END     PGM1
```

Nové instrukce

- `SP D1(L1,B1),D2(L2,B2)` – Subtract Packed decimal
  - od prvního operandu odečte druhý

Formát PACKED DECIMAL - příklad

```
 1  PGM1        SETUP
 2              ZAP     PROD,A
 3              MP      PROD,B
 4              SNAPSHOT  PGM1,ENDPGM
 5              ENDIT
 6  A           DC      P'54321'
 7  B           DC      P'12345'
 8  PROD        DS      PL8
 9  ENDPGM      DS      F
10              END     PGM1
```

Nové instrukce

- `MP D1(L1,B1),D2(L2,B2)` – Multiply Packed decimal
  - první operand vynásobí druhým
  - druhý operand může mít maximálně 15 číslic a musí být kratší než první operand
  - první operand musí mít vlevo alespoň tolik nul, kolik má druhý operand nenul

Formát PACKED DECIMAL - příklad

```
 1  PGM1        SETUP
 2              ZAP     QUOT,A
 3              DP      QUOT,B
 4              SNAPSHOT  PGM1,ENDPGM
 5              ENDIT
 6  A           DC      P'54321'
 7  B           DC      P'12345'
 8  QUOT        DS      PL8
 9  ENDPGM      DS      F
10              END     PGM1
```

Nové instrukce

- `DP D1(L1,B1),D2(L2,B2)` – Divide Packed decimal
  - první operandu vydělí druhým
  - výsledek je uložen nejvíce vlevo v prvním operandu a jeho délka je `L1-L2`
  - zbytek je uložen nejvíce vpravo v prvním operandu a má velikost druhého operandu
  - délka dělitele (`L2`) musí být $\leq$ 15 cifer
  - délka dělitele (`L2`) musí být *V* než `L1`