Příznakový registr, podmínky, cykly a aritmetické instrukce

ISU-cv04

Ing. Jakub Husa

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole
ihusa@fit.vutbr.cz

1. února 2020

Opakování

Datový segment "`section .data`":

- Slouží pro alokaci proměnných.
- Při alokaci proměnné musíme vždy zadat:
  identifikátor, datový typ (`db`, `dw`, `dd`, `dq`), a počáteční hodnotu.
- Pole vytvoříme zadáním několika hodnot oddělených čárkou.

Adresování:

- Každá proměnná má svoji (32b) adresu v paměti počítače.
- Pro přístup k obsahu paměti slouží hranaté závorky "`[]`".
- NELZE přistupovat na dvě adresy současně.
- Pro adresování můžeme používat obecné a indexové registry:
  `eax`, `ebx`, `ecx`, `edx`, `esi`, `edi`.

Řízení toku programu:

- Adresa aktuální instrukce je uložena v registru `eip`.
- Obsah registru můžeme změnit skokem na nějaké `návěští`.
- Podmíněné skoky se rozhodují podle hodnoty příznaků.

Příznakový registr

Příznaky jsou ukládány do speciálního 32b registru `eflags`:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

- Každý bit registru určuje aktuální hodnotu jiného příznaku, jejich pořadí a význam jsou dány architekturou procesoru.
- Příznakový registr je chráněn proti přímému zápisu.

```
mov  eflags, 0 ;toto nepujde prelozit
```

- Příznaky jsou nastavovány podle výsledku operace jako vedlejší efekt některých instrukcí.

```
add  al, bl ;pokud al+bl = 0 tak ZF = 1, jinak ZF = 0
```

- Které instrukce nastavují které příznaky je uvedeno v tabulce základních instrukcí.

```
sub  ax, ax ;ax = 0, ZF = 1
mov  ax, 0  ;ax = 0, ZF = ZF (stejny jako predtim)
```

Důležité příznaky

Overflow Flag (OF) – příznak přetečení

- Výsledek operace je neplatný v doplňkovém kódu.
- 100 + 100 = -56 (pokud jsme použili 8b registry)

Carry Flag (CF) – příznak přenosu

- Došlo k přenosu do (nebo výpůjčce z) nejvyššího řádu.
- -1 + -1 = -2

Sign Flag (SF) – příznak znaménka

- Výsledkem operace bylo záporné číslo.
- Používán podmíněnými skoky pro “větší něž” a “menší než”.

Zero Flag (ZF) – příznak nuly

- Výsledkem operace byla nula.
- Používán podmíněným skokem “rovno s”.

Direction Flag (DF) – příznak směru

- Používán pro nastavení řetězových instrukcí (9. cvičení).

Úkol:

- Načtěte ze vstupu jedno 8b bez-znaménkové číslo.
- Na výstup vypište čísla od vstupu až po nulu.
- Program napište bez použití instrukce `cmp`.

Například:

- Vstup “5” => výstup “5 4 3 2 1 0”.
- Vstup “0” => výstup “0”.

Nápověda:

- Instrukce `add` a `sub` nastavují příznak ZF.
- Instrukce `mov` a `call` příznak ZF nenastavují.
- Užitečné skoky:

```
jz   nav ;skoc na "nav" pokud ZF == 1
jnz  nav ;skoc na "nav" pokud ZF == 0
jmp  nav ;skoc na "nav" pokazde
```

Základní aritmetické instrukce

Inkrementace a dekrementace:

```
inc   DEST              ;DEST++
dec   DEST              ;DEST--
```

Sčítání a odčítání, příznak CF umožňuje zohlednit výsledek předchozího součtu nebo rozdílu:

```
add   DEST, SOURCE      ;DEST = DEST + SOURCE
sub   DEST, SOURCE      ;DEST = DEST - SOURCE
adc   DEST, SOURCE      ;DEST = DEST + SOURCE + CF
sbb   DEST, SOURCE      ;DEST = DEST - SOURCE - CF
```

Negace a porovnání ARITMETICKÝCH hodnot:

```
neg   DEST              ;DEST = 0 - DEST
cmp   DEST, SOURCE      ;DEST - SOURCE  (vysledek se nikam
                        ;nezapise, pouze se nastavi priznaky)
```

Otázka:

- Jaký je rozdíl mezi “`add al, 1`” a “`inc al`”?

Násobení bez znaménka

Bez-znaménkové násobení povedeme instrukcí `mul`:

- Problém, výsledek může být řádově vetší než oba operandy a musí proto být uložen ve větším registru (8b * 8b => 16b).
- Podobně jako ostatní instrukce `mul` má tři varianty.
- První operand a místo kam se uloží výsledek NEJSOU volitelné, jsou určeny velikostí druhého operandu.

```
mul  SRC  ; 8b =>       AX =  AL * SRC
mul  SRC  ;16b =>   DX:AX =  AX * SRC
mul  SRC  ;32b => EDX:EAX = EAX * SRC
```

- Zápis "`dx:ax`" znamená "`dx` konkatenováno s `ax`", tzn. horních 16b výsledku je v `dx` a spodních 16b je v `ax`.

Například:

```
mov  al, 10       ;AL = 10
mov  bl, 10       ;BL = 20
mul  bl           ;AX = AL * BL = 200
call WriteInt16 ;vypis obsah registru AX
```

Dělení bez znaménka

Bez-znaménkové dělení povedeme instrukcí `div`:

- Problém, výsledkem celočíselného dělení je zbytek a podíl.
- Podobně jako u násobení, mají u dělení operandy různou velikost, a nemůžeme si zvolit a kam se uloží výsledek.

```
div  SRC   ; 8b =>  AL =       AX / SRC
           ; 8b =>  AH =       AX % SRC
div  SRC   ;16b =>  AX =    DX:AX / SRC
           ;16b =>  DX =    DX:AX % SRC
div  SRC   ;32b => EAX = EDX:EAX / SRC
           ;32b => EDX = EDX:EAX % SRC
```

- Dělení nulou způsobí chybu (SIGFPE).

Například:

```
mov  ax, 100      ;AX = 100
mov  bl, 20       ;BL = 20
div  bl           ;AL = AX / BL = 5
                  ;AH = AX % BL = 0
call WriteInt8    ;vypis obsah registru AL
```

Úkol:

- Načtěte ze vstupu tři 8b bez-znaménková čísla, A, B a C.
- Vypište následující hodnoty:
  - X = A*B
  - Y = (A*B)/C
  - Z = (A*B)%C

Na co si dát pozor:

- Pokud se výsledek dělení nevejde do registrů, dojde k chybě (SIGFPE), vyzkoušejte A = 100, B = 100, C = 10.
- Zamyslete se nad tím jaký je počáteční obsah registrů.
- Znovu se zamyslete nad počátečním obsahem registru `edx`, a co to pro vás znamená když používáte instrukci `div`.

Při kopírování záporné hodnoty z menšího registru do většího dochází ke ztrátě znaménka:

```
mov  ax, 0    ;AX = 0b0000000000000000 = 0
mov  al, -10  ;AL =         0b11110110 = -10
              ;AX = 0b0000000011110110 = 246
```

Aby byla hodnota správně reprezentována i ve vetším registru, musíme znaménko rozšířit, pomocí speciálních instrukcí:

```
cbw         ; AL => AX
cwde        ; AX => EAX
cwd         ; AX => DX:AX
cdq         ;EAX => EDX:EAX
```

Násobení a dělení se znaménkem

Velmi podobné práci bez znaménka:

- Násobení – instrukce `imul`.
- Dělení – instrukce `idiv`.
- Rozmístění operandů a výsledků je stejné jako pro bez-znaménkové varianty.

Pro násobení a dělení se znaménkem vždy platí rovnice:

$$A = B * D + M$$

A	B	D = A / B	M = A % B
114	5	22	4
-114	5	-22	-4
114	-5	-22	4
-114	-5	22	-4

Bonusová úloha

Úkol:

- Ze vstupu načtěte dvě bez-znaménková 16b čísla, X a Y.
- pokud X == Y, tak vypište řetězec oznamující chybu.
- V opačném případě vypište výsledek rovnice:

$$F = \frac{X * Y}{|X - Y|}$$

Například:

- vstup X = 7, Y = 5 => výstup = 17
- vstup X = 10, Y = 15 => výstup = 30
- vstup X = 20, Y = 20 => výstup = “neplatne vstupy”

Na co si dát pozor:

- Z jaké adresy vypisuje funkce `WriteString`?
- Instrukce pro výpočet absolutní hodnoty neexistuje, musíte použít podmíněné skoky.