Programování na strojové úrovni

Cvičení 5: Logické instrukce, posuvy, rotace

Tomáš Goldmann, igoldmann@fit.vutbr.cz
www.fit.vutbr.cz/~igoldmann/courses

Rev.: 1A/2018

Rozcvička ☺

Spočítejte kolik **0** je v registru EAX.

```
MOV ECX, ___

         ___________

CYCLE:   ___________

         ___________

         INC EBX

NOT_INC:

         _____ CYCLE
```

Rozcvička ☺ - řešení

Spočítejte kolik je **0** v registru EAX.

```
MOV ECX, 32
            MOV EBX, 0
CYCLE:      SHL EAX, 1
            JC NOT_INC
            INC EBX
NOT_INC:
            LOOP CYCLE
```

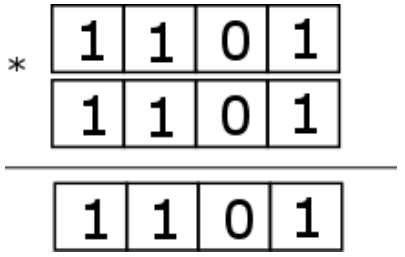
Logické instrukce - součet, součin, bitová inverze

OR operand1 (cíl), operand2 (zdroj)

- Provede logický bitový součet (neprovádějí se přenosy do vyšších řádů)

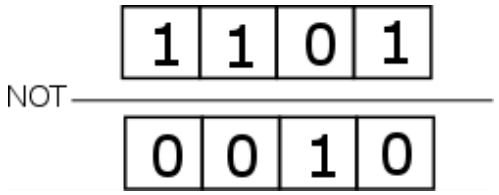
AND operand1 (cíl), operand2 (zdroj)

- Provede logický součin



NOT operand1 (neplést s instrukci NEG!)

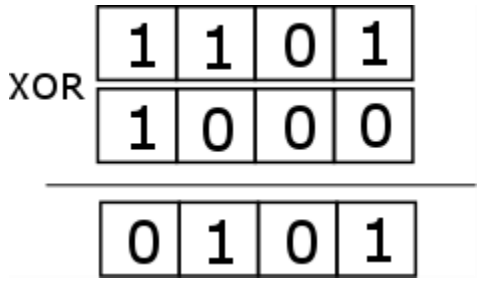
- Provede inverzi jednotlivý bitů (jedničkový komplement)



Logické instrukce – exkluzivní OR

XOR operand1 (cíl), operand2 (zdroj)

- Operace exkluzivní OR – pokud jsou bity rozdílné ->výsledek 1



Instrukce pro posuvy- SHL, SHR

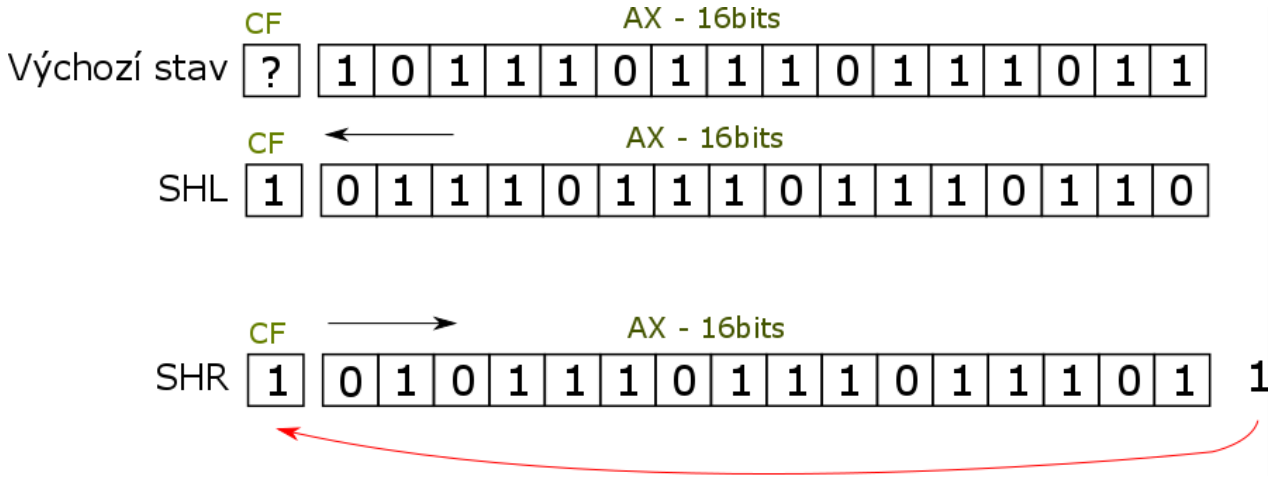
Posuvy slouží ke změně pozic bitů se zachováním bitové posloupnost. Bity určené prvním operandem jsou pomocí instrukcí SHL, SHR, SAL, SAR posunuty o n bitů (určených druhým operandem doleva nebo doprava.

SHL operand1 (cíl), operand2 (zdroj)

Provádí posun v rámci paměťového místa/registru doleva

SHR operand1 (cíl), operand2 (zdroj)

Provádí posun v rámci paměťového místa/registru doprava



Instrukce pro posuvy- SAL, SAR

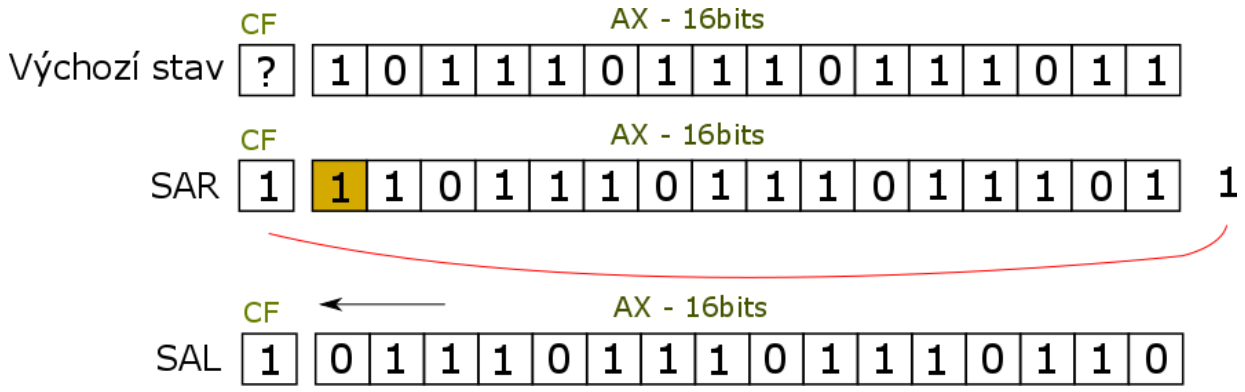
Tyto instrukce slouží pro aritmetické posuny (**posuny se zachováním znaménka**)

SAL operand1 (cíl), operand2 (zdroj)

Provádí **aritmetický** posun v rámci paměťového místa/registru doleva

SAR operand1 (cíl), operand2 (zdroj)

Provádí **aritmetický** posun v rámci paměťového místa/registru doprava



Instrukce pro rotace- ROL, ROR

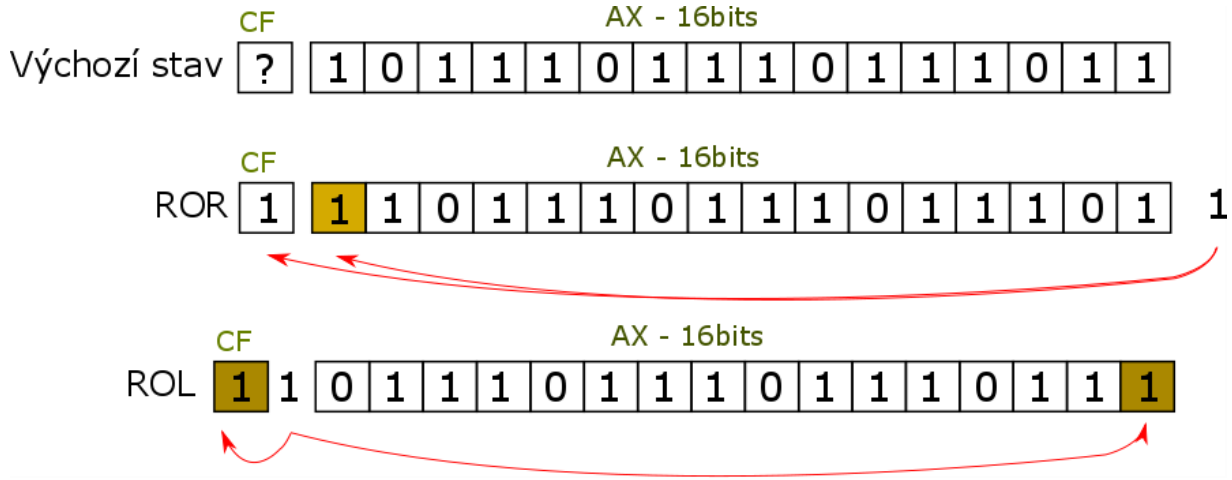
Rotace na rozdíl od posuvů provádějí nedestruktivní změnu pořadí bitů. Při rotacích nedochází ke ztrátě informace, jelikož se hodnoty pixelů pohybují cyklicky v rámci místa určeného operandem 1.

ROL operand1 (cíl), operand2 (zdroj)

Provádí **rotaci** v rámci paměťového místa/registru doleva

ROR operand1 (cíl), operand2 (zdroj)

Provádí **rotaci** v rámci paměťového místa/registru doprava



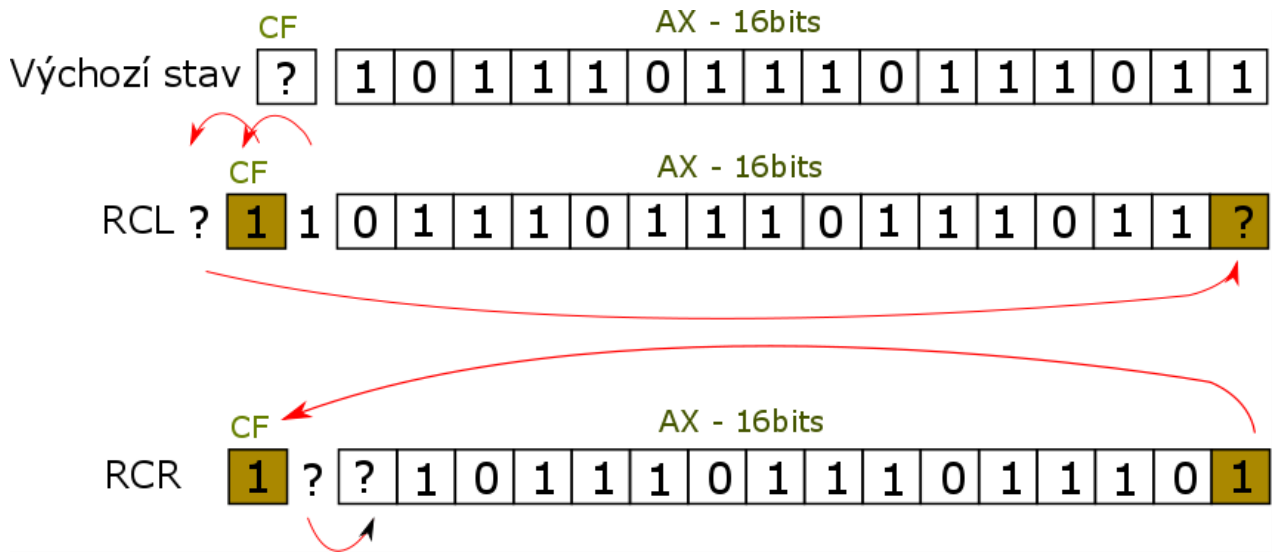
Instrukce pro rotace- RCL, RCR

RCL operand1 (cíl), operand2 (zdroj)

Provádí **rotaci přes CF** v rámci paměťového místa/registru doleva

RCR operand1 (cíl), operand2 (zdroj)

Provádí **rotaci přes CF** posun v rámci paměťového místa/registru doprava



Podmíněné a nepodmíněné skoky

Nepodmíněný skok

Jedná se o typ skoku, který se vždy provede. Nezávisí na výsledku předchozí instrukce. Instrukce JMP. Tato instrukce má jeden operand, který určuje na které návěští se bude skákat.

JMP <návěští>

Podmíněný skok

Jedná se o typ skoku, který se provádí na základě příznaků v EFLAGS. Instrukce podmíněného skoku obvykle začínají na J__

J__ <návěští>

Společné opakování

```
array_1 dw 0x12FF, 0x1, 0x8FFF,0xF123
array_2 db 0x12, 0x9, 0x10, 0xFF
array_3 dd 200, 300, 356, 400,0x0000FFFF
array_4 dq 0x00
```

Generování příznaků

Načtěte z pole array_1 0xA takové hodnoty, které vygenerují prostřednictvím instrukcí *adc a neg* jednotlivě příznaky *ZF, OF, CF, SF*

Součet prvků polí

Sečtěte první tři hodnoty z polí array_1 a array_2, výsledek uložte do registru EAX.

Společné opakování

Aritmetické operace s různými datovými typy

Sečtěte hodnoty v registrech BX=5 a ECX=0xFFF100; Vynásobte hodnoty v registrech BL=10 a EAX= 0xFFF100

```
xor ebx,ebx
mov bx,5
mov ecx, 0xFFF100
add ecx,ebx
```

Bezznaménkové výpočty

Vypočtěte bezznaménkově výraz array_4[0]=(array_3[0]+(array_1[0]/array_1[1])*4)*array_3[4]

Bezznaménkově -> pokud bude dělení možná budeme muset dělat nulování registru DX,EDX

Znaménkové výpočty

Vypočtěte znaménkově výraz array_3[0]=array_1[2]/array_1[0]-array_3[3]/array_3[0]

Znaménkově -> pokud se vyskytne operace dělení budeme muset udělat znaménkové rozšíření

Shifty

Vypočtěte znaménkově výraz array_1[0]*array_1[3] výsledek uložte do registru EDX

Úkoly

1. Do registru EBX uložte hodnotu 0xA1FF. Aplikujte masku 0x00FF na registr EBX. Uložte horní polovinu registru EBX do registru AX a pomocí instrukce `call WriteUInt16` vypište hodnotu.

2. Zaměňte horní polovinu registru EBX s dolní polovinou registru EAX.

3. Vyzkoušejte násobení a dělení celočíselných datových typů pomocí posuvu.

4. Vypočítejte příklad (-16/2 + -5*3), výsledek převeďte na kladné číslo pomocí logických instrukcí (nápověda: vzpomeňte si, jak se převádí číslo do doplňkového kódu).

5. Vypočítejte průměrnou hodnotu pole (pole db 10,20,30,15,16,17,18,19,20,21,18)