Logické instrukce, programový čítač a podmínky

ISU-cv05

Ing. Jakub Husa

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

8. března 2023

Logické instrukce

Logické instrukce (AND, OR, XOR, NOT) umožňují upravovat jednotlivé bity registru:

- Logické operace nám umožňují maskovat hodnotu konkrétních bitů.
- Pozor, složené podmínky, jak je znáte z vyšších programovacích jazyků, se implementují jinými instrukcemi!

```
1  and  eax, ebx       ;EAX = EAX & EBX  ;EAX a zaroven EBX
2  or   eax, ebx       ;EAX = EAX | EBX  ;EAX a   nebo  EBX
3  xor  eax, ebx       ;EAX = EAX ^ EBX  ;bud EAX nebo  EBX
4  not  eax            ;EAX = !EAX       ;invertuj EAX
```

Vyzkoušejte si:

```
5  ;AND vynucuje nulu
6  mov al, 0b00110011 ;AL = 00110011
7  and al, 0b00001111 ;AL = 00000011
8
9  ;OR  vynucuje jednicku
10 mov bl, 0b00110011 ;BL = 00110011
11 or  bl, 0b00001111 ;BL = 00111111
```

```
12 ;XOR vynucuje zmenu
13 mov cl, 0b00110011 ;CL = 00110011
14 xor cl, 0b00001111 ;CL = 00111100
15
16 ;NOT invertuje bity
17 mov dl, 0b00110011 ;DL = 00110011
18 not dl             ;DL = 11001100
```

Na procvičení

Vyzkoušejte si:

- Vytvořte si dostatečně dlouhé pole a načtěte do něj řetězec tří znaků.
- Předpokládejte že načtené znaky budou malá nebo velká písmena a jejich binární hodnoty (viz. tabulka ASCII) upravte tak aby první písmeno bylo malé, druhé velké, a třetí změnilo svoji velikost.
- Upravený řetězec vypište.

Například:

- abc => aBC
  ABC => aBc

Programový čítač

Programový čítač

Programový čítač ovládá pořadí vykonávaných instrukcí:

- Čítač obsahuje adresu aktuální instrukce která má být procesorem provedena.
- Jeho počáteční hodnotou je adresa první instrukce funkce main.
- Provedení libovolné instrukce do čítače (jako vedlejší efekt) automaticky nahraje adresu následující instrukce.

Hodnota programového čítače je uložena v registru EIP:

- Obsah registru je chráněn proti přímé manipulaci.

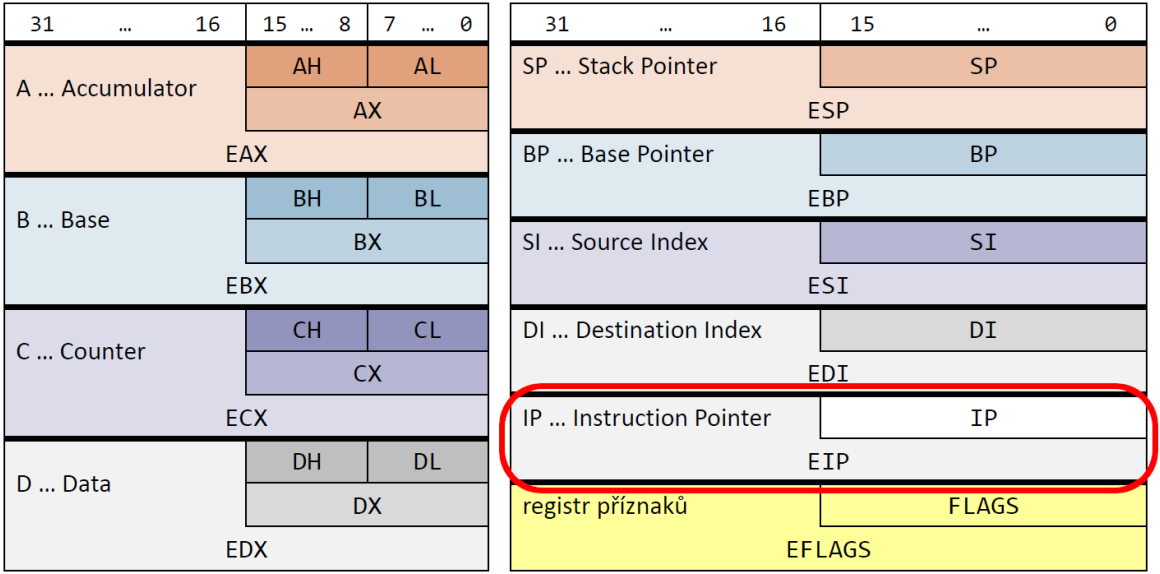
```
1     mov  eip, 0     ;CHYBA - do registru EIP nelze zapsat
```

Hodnotu čítače můžeme měnit pouze instrukcemi skoku:

- Skoky umožňují přejít na libovolnou instrukci označenou návěštím (label).
- Pomocí návěští označujeme funkce, podmínky i cykly.

```
2   label:          ;navesti jmenem LABEL
3     mov  eax, 10  ;instrukce oznacena navestim LABEL
```

Registry procesoru - programový čítač



Nepodmíněný skok

Nepodmíněný skok (JMP) umožňuje některé instrukce vynechat nebo opakovat:

```
1  _main:
2      mov   eax, 10          ;EAX = 10
3      jmp   label            ;skoc na navesti LABEL
4      mov   eax, 20          ;EAX = 20 (bude preskoceno)
5    label:                   ;navesti LABEL
6      call WriteInt32        ;vypis EAX (10)
7      ret
```

Program v nekonečné smyčce bude vypisovat rostoucí posloupnost čísel:

```
8  _main:
9      mov   eax, 0           ;EAX = 0
10   label:                   ;navesti LABEL
11     call WriteInt32        ;vypis EAX
12     call WriteNewLine      ;vypis konec radku
13     inc   eax              ;EAX = EAX + 1
14     jmp   label            ;skoc na navesti NAVESTI
15     ret                    ;konec funkce main (nikdy se nestane)
```

Podmíněné skoky

Podmíněné skoky jsou skoky které se provedou pouze při splnění nějaké podmínky:

- Pro každou z možných podmínek existuje samostatná instrukce.
- Pokud podmínka splněna není, program pokračuje následující instrukcí.

Skoky mohou být podmíněny hodnotu konkrétního příznaku (JO, JC, JS, JZ):

```
1 jo   label ;skoc pokud OF == 1 (Jump if Overflow)
2 jc   label ;skoc pokud CF == 1 (Jump if Carry)
3 js   label ;skoc pokud SF == 1 (Jump if Sign)
4 jz   label ;skoc pokud ZF == 1 (Jump if Zero)
```

Podmínky skoku mohou být negované (JNO, JNC, JNS, JNZ):

```
5 jno  label ;skoc pokud OF != 1 (Jump if Not Overflow)
6 jnc  label ;skoc pokud CF != 1 (Jump if Not Carry)
7 jns  label ;skoc pokud SF != 1 (Jump if Not Sign)
8 jnz  label ;skoc pokud ZF != 1 (Jump if Not Zero)
```

Program přeskočí dělení a výpis, pokud výsledek součtu byl nula:

```
1    mov  ax, 100    ;delenec
2    mov  bl, [X]    ;delitel
3    add  bl, 0      ;soucet nastavi priznaky
4
5    jz   preskoc1   ;skoc, pokud soucet nastavil ZF == 1
6    div  bl         ;deleni (necim co neni nula)
7    call WriteInt8  ;vypis (muze byt preskocen)
8  preskoc1:         ;navesti
```

Program přeskočí inkrementaci, pokud při posuvu nedošlo k přenosu:

```
9    mov  al, 0      ;pocitadlo
10   mov  bl, [Y]    ;posouvane cislo
11   shr  bl, 1      ;posun nastavi priznaky
12
13   jnc  preskoc2   ;skoc, pokud posun nastavil CF != 1
14   inc  al         ;inkrementace (pokud doslo k prenosu)
15 preskoc2:         ;navesti
16   call WriteInt8  ;vypis (vzdy se provede)
```

Vyzkoušejte si:

- Ze vstupu načtěte dvě 8b čísla (X a Y) a vypište jejich součet.
- Program upravte tak aby výpis přeskočil pokud při součtu došlo k přetečení.
- Program upravte tak aby výsledku obrátil znaménko pokud byl záporný.

Například:

- 10, 20 => 30
  100, 50 =>
  10, -20 => 10

Podmínky

Porovnávání hodnot

Příznaky můžeme úmyslně nastavit instrukcemi porovnání (CMP, TEST):

- Instrukce provedou výpočet ale nezapíší výsledek, pouze nastaví příznaky.

```
1 cmp  eax, ebx ;EAX - EBX (aritmeticke porovnani)
2 test eax, ebx ;EAX & EBX (  logicke   porovnani)
```

Aritmetické porovnání (CMP) příznaky připraví pro aritmetické skoky:

- Pro znaménková čísla.

```
3  je   lbl ;EAX == EBX (Equal)
4  jg   lbl ;EAX >  EBX (Greater)
5  jl   lbl ;EAX <  EBX (Lesser)
6  jge  lbl ;EAX >= EBX (.. or Equal)
7  jle  lbl ;EAX <= EBX (.. or Equal)
8
9  jne  lbl ;EAX != EBX (Not Equal)
10 jng  lbl ;EAX <= EBX (Not Greater)
11 jnl  lbl ;EAX >= EBX (Not Lesser)
12 jnge lbl ;EAX <  EBX (.. or Equal)
13 jnle lbl ;EAX >  EBX (.. or Equal)
```

- Pro bez-znaménková čísla.

```
14 je   lbl ;EAX == EBX (Equal)
15 ja   lbl ;EAX >  EBX (Above)
16 jb   lbl ;EAX <  EBX (Below)
17 jae  lbl ;EAX >= EBX (.. or Equal)
18 jbe  lbl ;EAX <= EBX (.. or Equal)
19
20 jne  lbl ;EAX != EBX (Not Equal)
21 jna  lbl ;EAX <= EBX (Not Above)
22 jnb  lbl ;EAX >= EBX (Not Below)
23 jnae lbl ;EAX <  EBX (.. or Equal)
24 jnbe lbl ;EAX >  EBX (.. or Equal)
```

Podmínky typu if-else

Podmínku if-else zapíšeme kombinací podmíněných a nepodmíněných skoků:

- Před tělo if dáme podmínku jestli ho chceme přeskočit.
- Pokud ho přeskočit chceme, skočíme na tělo else.
- Na konec těla if dáme nepodmíněný skok na konec celé podmínky.

Program porovná jestli je hodnota registrů EAX a EBX stejná:

```
1  if:                      ; navesti if
2    cmp  eax, ebx          ; porovnani hodnot nasastavuje priznaky
3    jne  else              ; podminka preskoceni tela if
4    ;cisla byla stejna     ; telo if
5    jmp  end               ; po provedeni if preskoc else
6  else:                    ; navesti else
7    ;cisla byla rozdilna   ; telo else
8  end:                     ; konec podminky
```

Vyzkoušejte si:

- Ze vstupu načtěte dvě bez-znaménková 8b čísla (X a Y).
- Napište program který ze zadaných čísel vypíše to větší.
- Bez-znaménková čísla načteme a vypíšeme funkcemi ReadUInt8 a WriteUInt8.

Například:

- 10, 20 => 20
  200, 100 => 200

Složené podmínky sestavíme kombinací několika podmíněných skoků:

- Instrukce skoku hodnoty příznaků neupravují.
- Volání funkce (CALL) příznaky nastaví na neznámou hodnotu.

Program porovná jestli je hodnota registru EAX v rozsahu od 1 do 10 (včetně):

```
1  if:                      ; navesti if
2    cmp  eax,  1           ; porovnani hodnot nastavuje priznaky
3    jl   else              ; podminka preskoceni tela if
4    cmp  eax, 10           ; porovnani hodnot nastavuje priznaky
5    jg   else              ; podminka preskoceni tela if
6    ;cislo je v rozsahu    ; telo if
7    jmp  end               ; po provedeni tela skoc na konec
8  else:                    ; navesti else
9    ;cislo v rozsahu neni; telo else
10 end:                     ; konec podminky
```

Vyzkoušejte si:

- Vytvořte si dvě inicializované 16b proměnné (X a Y).
- Napište program který podle hodnoty proměnných vypíše řetězec ”obe cisla jsou suda” nebo ”alespon jedno cislo je liche”.
- Sudá (even) a lichá (odd) čísla poznáte podle hodnoty jejich nejnižšího bitu.

Například:

- 10, 20 => obe cisla jsou suda
  11, 20 => alespon jedno cislo je liche
  10, 21 => alespon jedno cislo je liche
  11, 21 => alespon jedno cislo je liche

Podmínky typu if-else-if

Zřetězené podmínky typu if-else-if:

- Před každé tělo dáme podmínku jestli ho chceme přeskočit.
- Pokud ho přeskočit chceme, skočíme na následující podmínku.
- Na konec každého těla dáme nepodmíněný skok na konec celé podmínky.

Program porovná jestli je hodnota registru EAX a kladná, záporná nebo nula:

```
1   if:                     ; navesti if
2     cmp  eax, 0           ; porovnani hodnot nastavuje priznaky
3     jng  elseif           ; podminka preskoceni tela if
4     ;cislo bylo kladne    ; telo if
5     jmp  end              ; po provedeni tela skoc na konec
6   elseif:                 ; navesti else-if
7     jnl  else             ; podminka preskoceni tela else-if
8     ;cislo bylo zaporne   ; telo else-if
9     jmp  end              ; po provedeni tela skoc na konec
10  else:                   ; navesti else
11    ;cislo bylo nula      ; telo else
12  end:                    ; konec podminky
```

Vyzkoušejte si:

- Ze vstupu načtěte dvě znaménková 8b čísla (X a Y).
- Napište program který spočítá následující rovnici a vypíše její výsledek.
- Program upravte tak aby vypsal "overflow" pokud při součtu došlo k přetečení.
- Program upravte tak aby vypsal "NaN" pokud by při podílu došlo k dělení nulou.

$$f = \frac{1000}{X + Y}$$

Například:

$$10, 20 => \frac{1000}{30} = 33$$

$$100, 100 => \frac{1000}{-56} = \text{overflow} \qquad 10, -10 => \frac{1000}{0} = \text{NaN}$$