Cykly a řetězové instrukce

ISU-cv07

Ing. Jakub Husa

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

ihusa@fit.vutbr.cz

22. března 2023

Cykly

Cykly typu while

Cyklus pro opakování nějakých instrukcí vytvoříme skokem vzad:

- Cyklus while se na začátku těla ptá jestli cyklus chceme ukončit.

```
1  condition:
2    cmp  eax, 0     ;porovnanim hodnot nastavime priznaky
3    je   end        ;podminkou rozhodujeme o ukonceni cyklu
4  while:
5    ;telo WHILE
6    jmp  condition ;ne-podminenym skokem se vracime na podminku
7  end:
```

- Cyklus do-while se na konci těla ptá jestli cyklus chceme opakovat.

```
8  while:
9    ;telo WHILE
10 condition:
11   cmp  eax, 0     ;porovnanim hodnot nastavime priznaky
12   jne  while      ;podminkou rozhodujeme o opakovani cyklu
13 end:
```

Vyzkoušejte si:

- Program bude ze vstupu načítat 32b čísla tak dlouho dokud nezadáme nulu, a poté vypíše jejich sumu:

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2
3  section .text            ;kodovy segment
4  _main:
5      mov  ebx, 0          ; EBX = pocatecni hodnota sumy
6
7    while:
8      call ReadInt32       ; EAX = vstup
9      add  ebx, eax        ; suma = suma + vstup
10   condition:
11     cmp  eax, 0          ; porovnanim nastavime priznaky
12     jne  while           ; pokum vstup nebyl 0 tak se cyklus bude opakovat
13   end:
14
15     mov  eax, ebx        ; EAX = suma
16     call WriteInt32      ; vypis
17     ret
```

Vyzkoušejte si:

- Ze vstupu načtěte dvě 16b znaménková čísla (X a Y).
- Vypište všechny hodnoty v rozsahu od prvního do druhého z nich (včetně).

Například:

- 5, 10 => 5, 6, 7, 8, 9, 10
  5, 5 => 5
  10, 5 =>

Cyklus typu for

Cyklus typu for vytvoříme instrukcí LOOP (smyčka):

- Smyčka nejprve dekrementuje hodnotu registru ECX, a pokud není rovna nule (JNZ) provede skok na odpovídající návěští.

Vyzkoušejte si:

- Program ve smyčce vypíše čísla od 10 do 1:

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2
3  section .text            ;kodovy segment
4  _main:
5      mov  ecx, 10         ; ECX = pocet opakovani
6    for:
7      mov  eax, ecx        ; EAX = ECX
8      call WriteInt32      ; vypis EAX
9      call WriteNewLine    ; vypis prazdy radek
10     loop for             ; smycka opakovani for (ECX--, jnz for)
11
12     ret
```

Vyzkoušejte si:

- Ze vstupu načtěte jedno 32b číslo a vypište počet jedniček v jeho binárním zápisu.

Například:

- 0 => 0
  1 => 1
  255 => 8
  -1 => 32

Průchod polem

Při adresování položek pole můžeme kromě konstanty přičítat také 32b registr:

- Hodnotu přičítaného registru můžeme také násobit konstantami 1, 2 a 4.
- Pozor, na rozdíl od konstanty registr od adresy nelze odečíst.

```
1  mov  [pole + eax], al        ;ok     - pricitame 32b registr
2  mov  [pole + ebx*4 - 4], al  ;ok     - konstantu lze odecitat
3  mov  [pole +  cx], al        ;CHYBA  - pricitany registr nema 32b
4  mov  [pole - edx], al        ;CHYBA  - registr nelze odecitat
```

Vyzkoušejte si – program inicializuje všechny (16b) prvky pole na hodnotu -1:

```
5  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
6  section .bss             ;ne-inicializovany segment
7      arr resw 5
8  section .text            ;kodovy segment
9  _main:
10     mov  ecx, 5          ; ECX = pocet opakovani
11     mov  ebx, 0          ; EBX = index prvku pole
12   for:
13     mov  word [arr + ebx*2], -1  ;prvek pole = -1
14     inc  ebx             ; EBX = nasledujici prvek
15     loop for             ; smycka opakovani for (ECX--, jnz for)
16     ret
```

Vyzkoušejte si:

- Vytvořte si inicializované pole pěti 32b znaménkových čísel.
- Napište program který projde přes všechny prvky pole, a vypíše jejich maximum.

Například:

- 1, 5, 2, 4, 3 => 5
  -1, -5, -2, -4, -3 => -1

Řetězové instrukce

Řetězové instrukce - MOVS

Pro práci s poli můžeme použít také řetězové instrukce a prefix opakování:

- Usnadňují kopírování a porovnávání řetězců dat.
- Používají nepřímé adresování přes registry ESI a EDI.

Kopírování z pole do pole provedeme instrukcemi MOVSB, MOVSW, MOVSD:

- Instrukce zkopíruje jednu položku z adresy v ESI na adresu v EDI, a hodnoty ESI a EDI posune na další položku.
- Velikost položky i posuvu a je dána jménem instrukce (B, W, D).

```
1 movsb   ;zkopiruj  8b z adresy ESI do adresy EDI
2 movsw   ;zkopiruj 16b z adresy ESI do adresy EDI
3 movsd   ;zkopiruj 32b z adresy ESI do adresy EDI
```

Pro opakování řetězové instrukce můžeme použití prefix opakování REP:

- Prefix můžeme umístit před libovolnou řetězovou instrukci.
- REP dekrementuje hodnotu ECX, a dokud není nula instrukci bude opakovat.

Vyzkoušejte si:

- Program zkopíruje pět položek z pole src do pole dst:

```
1  section .bss                ;ne-inicializovane segment
2      dst resw 5              ; rezervujeme peti 16b hodnot
3
4  section .data               ;inicializovany segment
5      src dw 10,20,30,40,50 ; definujeme pole peti 16b hodnot
6
7  section .text               ;kodovy segment
8  _main:
9
10     mov  esi, src           ; ESI = zdrojove pole
11     mov  edi, dst           ; EDI =  cilove  pole
12     mov  ecx, 5             ; ECX = pocet polozek pole
13
14     rep  movsw              ; opakuj presun 16b polozek
15
16     ret
```

Kopírování z pole do registru provedeme instrukcemi LODSB, LODSW, LODSD:

- Instrukce zkopíruje data z adresy v ESI do registru AL, AX, EAX a hodnotu ESI posune na další položku.

```
1 lodsb   ;zkopiruj  8b z adresy ESI do  AL
2 lodsw   ;zkopiruj 16b z adresy ESI do  AX
3 lodsd   ;zkopiruj 32b z adresy ESI do EAX
```

Kopírování z registru do pole provedeme instrukcemi STOSB, STOSW, STOSD:

- Instrukce zkopíruje data z registru AL, AX, EAX na adresu v EDI a hodnotu EDI posune na další položku.

```
4 stosb   ;zkopiruj  8b z  AL do adresy EDI
5 stosw   ;zkopiruj 16b z  AX do adresy EDI
6 stosd   ;zkopiruj 32b z EAX do adresy EDI
```

Vyzkoušejte si:

- Program ze vstupu načte pět 8b hodnot a uloží je do pole dst:

```
1  %include "rw32-2018.inc"   ;knihovna pro vstup a vystup
2
3  section .bss               ;ne-inicializovane segment
4      dst resb 5             ; rezervujeme peti 8b hodnot
5
6  section .text              ;kodovy segment
7  _main:
8      mov  edi, dst          ; EDI = cilove pole
9      mov  ecx, 5            ; ECX = pocet opakovani
10
11   for:
12     call ReadInt8          ; AL  = vstup
13     stosb                  ; AL uloz na adresu v EDI
14     loop for               ; opakuj cyklus doku ECX != 0
15
16     ret
```

Vyzkoušejte si:

- Vytvořte si inicializované pole pěti 32b znaménkových čísel.
- Napište program který všechny prvky pole převede na jejich absolutní hodnotu.
- Výsledný obsah pole zobrazte v debuggeru.

Například:

- 100, -200, 300, -400, 500 => 100, 200, 300, 400, 500

Porovnání pole s polem provedeme instrukcemi CMPSB, CMPSW, CMPSD:

- Instrukce porovnává data z adresy v ESI s daty na adrese v EDI, a hodnoty ESI a ESI posune na další položku.

```
1 cmpsb    ;porovnej  8b z adresy ESI a adresy EDI
2 cmpsw    ;porovnej 16b z adresy ESI a adresy EDI
3 cmpsd    ;porovnej 32b z adresy ESI a adresy EDI
```

Porovnání pole s registrem provedeme instrukcemi SCASB, SCASW, SCASD:

- Instrukce porovnává data z registru AL, AX, EAX s daty na adrese v EDI.

```
4 scasb    ;porovnej  8b z adresy ESI a  AL
5 scasw    ;porovnej 16b z adresy ESI a  AX
6 scasd    ;porovnej 32b z adresy ESI a EAX
```

Směr a výsledek porovnávání

Při porovnávání používáme podmíněné prefixy opakování REPE a REPNE:

```
1 repe    ;instrukci opakuj dokud se polozky shoduji
2 repne   ;instrukci opakuj dokud se polozky odlisuji
```

Pokud podmínka opakování není splněna, porovnávání se ukončí předčasně:

- Předčasné ukončení poznáme dle hodnoty příznaku ZF (skoky JZ, JNZ, JE, JNE).
- Pozici nevyhovující položky poznáme dle hodnoty registru ECX.

Upravit můžeme také směr posuvu, řízený příznakem DF (Direction Flag):

- Hodnotu příznaku nastavujeme instrukcemi STD a CLD.

```
3 cld     ;clear direction flag (DF = 0), posun z leva do prava
4 std     ; set  direction flag (DF = 1), posun z prava do leva
```

- Pozor, příznak DF používají i knihovní funkce, a před jejich voláním ho tak vždy musíme nastavit zpět na nulu.

Vyzkoušejte si:

- Program porovná dvě pole a vypíše index poslední položky ve které se odlišují, nebo hodnotu -1 pokud se jejich obsah shoduje:

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2  section .data            ;inicializovany segment
3     src dw 1, 2, 3, 4, 5  ; prvni porovnavane pole
4     dst dw 1, 8, 3, 9, 5  ; druhe porovnavane pole
5  section .text            ;kodovy segment
6  _main:
7      mov  esi, src + 2*4  ; ESI = posledni znak zdrojoveho pole
8      mov  edi, dst + 2*4  ; EDI = posledni znak   ciloveho pole
9      mov  ecx, 5          ; ECX = pocet opakovani
10
11     std                 ; DF = 1, smer posuvu = z prava do leva
12     repe cmpsw          ; dokud se znaky shodujei, porovnavej retezce
13     cld                 ; DF = 0, priznak DF vracime na puvodni hodnotu
14
15     jne  skip           ; pokud retezce byli rozdilne preskoc nasledujici radek
16     mov  ecx, -1        ; pokud retezce byli stejne nastav citac na -1
17   skip:
18
19     mov  eax, ecx       ; EAX = stav citace
20     call WriteInt32     ; vypis EAX
21     ret
```

Vyzkoušejte si:

- Vytvořte si inicializované pole obsahující řetězec "Hello, World!".
- Napište program který ze vstupu načte jeden znak (funkce ReadChar), a první výskyt tohoto znaku v řetězci nahradí pomlčkou '–'.
- Upravený řetězec vypište na výstup.

Například:

- 'H' => –ello, World!
  'e' => H–llo, World!
  'l' => He–lo, World!
  'o' => Hell–, World!
  'X' => Hello, World!