Rekurze, konvence volání a externí funkce

ISU-cv09

Ing. Jakub Husa

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

ihusa@fit.vutbr.cz

6. dubna 2023

Rekurze

Rekurze nastává ve chvíli kdy nějaká funkce volá sebe sama:

- Při každém volání se na zásobník ukládá nová návratová adresa.
- Každé volání si na zásobníku může vytvořit svůj vlastní zásobníkový rámec, a mít na něm svoje vlastní parametry a lokální proměnné.

Na rozdíl od cyklu, rekurze vždy musí mít nějakou ukončující podmínku:

- Nekonečná rekurze způsobí přetečení zásobníku.

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2  section .text            ;kodovy segment
3  _main:
4      mov  eax, 0          ; EAX = pocitadlo
5      call func            ; zavolej funkci func
6      ret
7  func:
8      inc  eax             ; inkrementuj pocitadlo
9      call WriteInt32      ; vypis EAX
10     call WriteNewLine    ; vypis prazdny radek
11     call func            ; zavolej sebe sama
12     ret                  ; konec funkce (nikdy se nestane)
```

Na vyzkoušení

Vyzkoušejte si:

- Fukce faktorial spočítá hodnotu faktoriálu definovaného rekurzivně:

  0! = 1
  n! = (n-1)! * n

- Vstupem funkce je hodnota n předávaná přes zásobník.
- Horních polovinu výsledku 32b násobení (EDX) funkce ignoruje.

```
1  %include "rw32-2018.inc" ;knihovna
2  section .text            ;kodovy segment
3  _main:
4      call ReadInt32 ;nacti vstup
5      push eax       ;predej param
6      call faktorial ;zavolej funkci
7      add  esp, 4    ;odstran param
8      call WriteInt32;vypis vysledek
9      ret
```

```
9  faktorial:
10     push ebp       ;uloz stare dno
11     mov  ebp, esp  ;vytvor nove dno
12   condition:
13     cmp  dword [ebp+8],0;je n nula?
14     je   if        ;pokud ano tak IF
15     jmp  else      ;jinak ELSE
16   if:
17     mov  eax, 1    ;0! = 1
18     jmp  end       ;skoc na konec
19   else:
20     mov  eax, [ebp+8];EAX = n
21     dec  eax          ;EAX = n-1
22     push eax          ;predej param
23     call faktorial    ;EAX =(n-1)!
24     add  esp, 4       ;odstran param
25     mul  dword [ebp+8];EAX=(n-1)!*n
26     jmp  end       ;skoc na konec
27   end:
28     pop  ebp       ;obnov stare dno
29     ret
```

Vyzkoušejte si:

- Ze vstupu načte jedno 32b číslo (n) a zavolejte funkci vypisRekurze.
- Pro n == 0 funkce vypíše n.
- Pro n != 0 funkce vypíše n, zavolá vypisRekurze pro n-1 a znovu vypíše n.
- Vstupem funkce je hodnota n předávaná v registru EAX.

```
1 %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2 section .text            ;kodovy segment
3 _main:
4     call ReadInt32       ; nacitame vstup
5     call vypisRekurze    ; volame funkci
6     ret
```

Například:

- 0 => 0
- 2 => 21012
- 5 => 54321012345

Na procvičení

Vyzkoušejte si:

- Ze vstupu načte jedno 32b číslo, a pomocí rekurze vypočítejte hodnotu odpovídajícího pyramidového čísla:

$$pc(0) = 0 \qquad pc(n) = n^2 + pc(n-1)$$

- Vstupem funkce je hodnota n předávaná přes zásobník.
- Horních 32b výsledku násobení můžete ignorovat.

Například:

- 0 => 0
- 1 => 1
- 2 => 5
- 3 => 14
- 4 => 30
- 5 => 55

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2
3  section .text            ;kodovy segment
4  _main:
5      call ReadInt32       ; nacitame vstup
6      push eax             ; predavame parametr
7      call pc              ; volame funkci
8      add  esp, 4          ; parametr odstranujeme ze zasobniku
9      call WriteInt32      ; vypisujeme vysledek
10     ret
```

Konvence volání

Konvence volání

Pořadí předávaných parametrů je definováno konvencemi volání:

- Konvence jsou odvozeny od implementace vyšších programovacích jazyků.
- Konvence také určuji jestli parametry odstraňuje volající nebo volaná funkce.
- Volaná funkce parametry odstraňuje použitím nepovinného operandu instrukce RET který se po návratu z funkce přičte k vrcholu zásobníku (ESP).

```
1    ret      ;konec funkce
2    ret  4   ;konec funkce a "add esp, 4" (odstran jeden parametr)
```

Konvence volání	Parametry funkce	Zásobník uklízí	Dekorace jmen (pro jazyk C)	Použito v
pascal	zleva doprava	volaný	symbol	Pascal
cdecl	zprava doleva	volající	_symbol	Jazyk C
stdcall	zprava doleva	volaný	_symbol@4	Win32 API
fastcall	první dva parametry v ECX a EDX, zbytek zprava doleva	volaný	@symbol@8	různé

Volání funkce dle CDECL:

```
1  %include "rw32-2018.inc" ;knihovna
2  section .text            ;kodovy segment
3  _main:
4      push dword 20        ;druhy parametr
5      push dword 10        ;prvni parametr
6      call soucet          ;zavolej soucet
7      add  esp, 8          ;odstran param
8      call WriteInt32      ;vypis vysledek
9      ret
10
11 soucet:                  ;CDECL
12     push ebp             ;zalohuj stare
13     mov  ebp, esp        ;vytvor  nove
14
15     mov  eax, [ebp + 8]  ;prvni
16     add  eax, [ebp +12]  ;druhy
17
18     pop  ebp             ;obnov stare
19     ret                  ;konec funkce
20                          ;
```

Volání funkce dle STDCALL:

```
21 %include "rw32-2018.inc" ;knihovna
22 section .text            ;kodovy segment
23 _main:
24     push dword 20        ;druhy parametr
25     push dword 10        ;prvni parametr
26     call soucet          ;zavolej soucet
27
28     call WriteInt32      ;vypis vysledek
29     ret
30
31 soucet:                  ;STDCALL
32     push ebp             ;zalohuj stare
33     mov  ebp, esp        ;vytvor  nove
34
35     mov  eax, [ebp + 8]  ;prvni
36     add  eax, [ebp +12]  ;druhy
37
38     pop  ebp             ;obnov stare
39     ret  8               ;konec funkce a
40                          ;odstran param
```

Volání funkce dle PASCAL:

```
1  %include "rw32-2018.inc" ;knihovna
2  section .text            ;kodovy segment
3  _main:
4      push dword 10        ;prvni parametr
5      push dword 20        ;druhy parametr
6      call soucet          ;zavolej soucet
7
8      call WriteInt32      ;vypis vysledek
9      ret
10
11 soucet:                  ;PASCAL
12     push ebp             ;zalohuj stare
13     mov  ebp, esp        ;vytvor  nove
14
15     mov  eax, [ebp +12]  ;prvni
16     add  eax, [ebp + 8]  ;druhy
17
18     pop  ebp             ;obnov stare
19     ret  8               ;konec funkce a
20                          ;odstran param
```

Volání funkce dle FASTCALL:

```
21 %include "rw32-2018.inc" ;knihovna
22 section .text            ;kodovy segment
23 _main:
24     mov  ecx, 10         ;prvni parametr
25     mov  edx, 20         ;druhy parametr
26     call soucet          ;zavolej soucet
27
28     call WriteInt32      ;vypis vysledek
29     ret
30
31 soucet:                  ;FASTCALL
32     push ebp             ;zalohuj stare
33     mov  ebp, esp        ;vytvor  nove
34
35     mov  eax, ecx            ;prvni
36     add  eax, edx            ;druhy
37
38     pop  ebp             ;obnov stare
39     ret  0               ;konec funkce a
40                          ;odstran param
```

Vyzkoušejte si:

- Vytvořte si inicializované pole několika 16b bez-znaménkových čísel.
- Napište funkci pocetMensich která spočítá počet prvků s hodnotu menší jak n.
- Funkci definujte a volejte dle konvence STDCALL a následující hlavičky:

```
1    ;int pocetMensich (short* pole, int delka, int n);
```

- Funkci pocetMensich zavolejte z funkce main a vypište její výsledek.

Například:

- *(10, 20, 30, 40, 50), 5, 25 => 2
- *(10000, 50000), 2, 20000 => 1

Externí funkce

Externí funkce

Externí funkce jsou funkce implementované mimo náš zdrojový soubor:

- Externí funkce deklarujeme direktivou extern (_extern, CEXTERN).
- Funkce z knihovny "rw32-2018.inc" nejsou externí protože direktivou %include do našeho zdrojového souboru vkládáme celou knihovnu.

Externí jsou například funkce ze standardních knihoven jazyka C:

- Externí funkce dekorujeme dle konvencí daného jazyka (CDECL pro jazyk C).
- Při předávání parametrů a volání funkce se také řídíme konvencemi.
- Pozor, externí funkce nezálohují hodnoty obecných registrů (EAX, EBX, ECX, EDX)!

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2      extern _getchar      ; externi funkce "int getchar ( void );"
3      extern _putchar      ; externi funkce "int putchar ( int character );"
4  section .text            ;kodovy segment
5  _main:
6      call _getchar        ; EAX = navratova hodnota funkce getchar
7      push eax             ; predavame prvni parametr funkce putchar
8      call _putchar        ; volame funkci putchar (vypise znak)
9      add  esp, 4          ; predany parametr odstranujeme ze zasobniku
10     ret
```

Vyzkoušejte si:

- Program ze vstupu načte dvě čísla a externí funkcí printf vypíše jejich součet.

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2      extern _printf       ; "int printf ( const char * format, ... );"
3  section .data            ;inicializovany segment
4      msg db "%i + %i = %i", 10, 0 ;vypisovany retezec
5  section .text            ;kodovy segment
6  _main:
7      call ReadInt32       ; nacti vstup (X)
8      mov  ecx, eax        ; ECX = X
9      call ReadInt32       ; nacti vstup (Y)
10     mov  ebx, eax        ; EBX = Y
11     add  eax, ecx        ; EAX = Y+X
12
13     push eax             ; treti "%i"
14     push ebx             ; druhe "%i"
15     push ecx             ; prvni "%i"
16     push msg             ; adresa vypisovaneho retezce
17     call _printf         ; volame funkci printf
18     add  esp, 16         ; odstranujeme predane parametry
19     ret
```

Vyzkoušejte si:

- Ze vstupu načtěte jedno 32b číslo a pokuste se o dynamickou alokaci odpovídajícího množství paměti na hromadě (malloc).
- Pokud se alokace paměti podařila vypište její adresu (printf) v řetězci "Alokovana adresa je %i" a uvolněte ji (free).
- Pokud se alokace paměti nepodařila vypište řetězec "Alokace selhala".

```
1  extern _malloc ; funkce "void* malloc (size_t size);"
2  extern _free   ; funkce "void    free (void* ptr);"
3  extern _printf ; funkce "int   printf ( const char * format, ... );"
```

Například:

- 20 => Alokovana adresa je 11300896
- -1 => Alokace selhala

Vyzkoušejte si:

- Externí funkcí z knihovny jazyka C zjistěte aktuální čas (time).
- Čas použijte k inicializaci pseudo-náhodného generátoru (srand).
- V cyklu vygeneruje a vypište pět náhodných čísel (rand).

```
1   extern _time   ; externi funkce:  time_t  time (time_t* timer);
2   extern _srand  ; externi funkce:    void srand (unsigned int seed);
3   extern _rand   ; externi funkce:     int  rand (void);
```

Například:

- => 4900, 19769, 23118, 4036, 27957.
- => 4903, 30517, 8214, 28099, 5504.