Konvence volání, externí funkce a lokální proměnné

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

4. dubna 2024

Konvence volání

Konvence volání

Způsob předávání parametrů přes zásobník se řídí konvencemi volání:

- Konvence jsou odvozeny od implementace vyšších programovacích jazyků.
- Konvence určují také jestli parametry odstraňuje volající nebo volaná funkce.
- Volaná funkce parametry může odstranit 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 32b 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.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 dno
13     mov  ebp, esp   ;vytvor  dno
14
15     mov  eax, [ebp + 8] ;prvni
16     add  eax, [ebp +12] ;druhy
17
18     pop  ebp        ;obnov dno
19     ret             ;konec funkce
20                     ;
```

Volání funkce dle STDCALL:

```
21 %include 'rw32.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 dno
33     mov  ebp, esp   ;vytvor  dno
34
35     mov  eax, [ebp + 8] ;prvni
36     add  eax, [ebp +12] ;druhy
37
38     pop  ebp        ;obnov dno
39     ret  8          ;konec funkce a
40                     ;odstran param
```

Volání funkce dle PASCAL:

```
1  %include 'rw32.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 dno
13     mov  ebp, esp   ;vytvor  dno
14
15     mov  eax, [ebp +12] ;prvni
16     add  eax, [ebp + 8] ;druhy
17
18     pop  ebp        ;obnov dno
19     ret  8          ;konec funkce a
20                     ;odstran param
```

Volání funkce dle FASTCALL:

```
21 %include 'rw32.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 dno
33     mov  ebp, esp   ;vytvor  dno
34
35     mov  eax, ecx       ;prvni
36     add  eax, edx       ;druhy
37
38     pop  ebp        ;obnov dno
39     ret  0          ;konec funkce a
40                     ;odstran param
```

Na procvičení

Vyzkoušejte si:

- Napište funkci InicializujPole která prvky pole 16b nastaví na danou hodnotu.
- Parametry funkce jsou pole (arr) jeho délka (N) a hodnota (X), předané odkazem.
- Funkce je volána dle konvence STDCALL.
- Výsledný obsah pole si zkontrolujte v debuggeru.

Například:

```
1  %include 'rw32.inc'        ;knihovna pro vstup a vystup
2  section .bss               ;ne-inicializovany datovy segment
3      arr resw 5             ; pole peti 16b polozek
4  section .data              ;inicializovany datovy segment
5      N dd 5                 ; pocet prvku pole
6      X dw 1000              ; hodnota pouzita pro inicializaci
7  section .text              ;kodovy segment
8  main:
9      push X                 ; predavame treti parametr
10     push N                 ; predavame druhy parametr
11     push arr               ; predavame prvni parametr
12     call InicializujPole   ; volame funkci
13     ret
```

Vyzkoušejte si:

- Vytvořte si inicializované pole 8b hodnot obsahující textový řetězec.
- Napište funkci PocetCislic která spočítá kolik daný řetězec obsahuje číslic.
- Parametrem funkce je adresa pole, a funkce je volána dle konvence FASTCALL.
- Konec řetězce poznáte podle ukončujícího znaku NUL (v ASCII hodnota 0).

Například:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2  section .data                ;inicializovany datovy segment
3      msg1 db '14:00-19:50/N104', 0   ; 11 cislic
4      msg2 db '0123456789', 0         ; 10 cislic
5      msg3 db 'Hello, World!', 10, 0 ;  0 cislic
6  section .text                ;kodovy segment
7  main:
8      mov  ecx, msg1           ; do ECX nastav adresu retezce (parametr)
9      call PocetCislic         ; zavolej funkci PocetCislic
10     call WriteInt32NewLine ; vypis EAX (navratova hodnota)
11     ret
```

Externí funkce

Externí funkce

Externí funkce jsou funkce implementované mimo náš zdrojový soubor:

- Externí funkce deklarujeme direktivou extern (_extern, CEXTERN).
- Funkce z knihovny 'rw32.inc' nejsou externí protože direktivou %include do našeho zdrojového souboru vkládáme zdrojový kód knihovny.

Externí jsou například funkce ze standardních knihoven jazyka C:

- V některých vývojových prostředích je dekorujeme dle konvencí daného jazyka.
- Při předávání a mazání parametrů se řídíme odpovídajícími konvencemi.
- Pozor – externí funkce **NEZÁLOHUJÍ** hodnoty registrů EAX, EBX, ECX a EDX!

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2      extern strlen            ; funkce "size_t strlen (const char* str);"
3  section .data                ;inicializovany datovy segment
4      msg db "Hello, World!", 10, 0 ; textovy retezec
5  section .text                ;kodovy segment
6  main:
7      push msg                 ; predavame parametr funkce (retezec)
8      call strlen              ; EAX = navratova hodnota funkce strlen
9      add  esp, 4              ; predany parametr mazeme ze zasobniku
10     call WriteInt32NewLine   ; vypis
11     ret
```

Program ze vstupu načte dvě 32b hodnoty a externí funkcí printf vypíše jejich součet:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2      extern printf            ; "int printf (const char* format, ...);"
3  section .data                ;inicializovany datovy segment
4      msg db "%i + %i = %i", 10, 0 ; textovy retezec
5  section .text                ;kodovy segment
6  main:
7      call ReadInt32NewLine ; nacti vstup (X)
8      mov  ecx, eax         ; ECX = X
9      call ReadInt32NewLine ; nacti vstup (Y)
10     mov  ebx, eax         ; EBX = Y
11     add  eax, ecx         ; EAX = Y+X
12
13     push eax              ; ctvrty parametr - %i (X+Y)
14     push ebx              ; treti  parametr - %i (Y)
15     push ecx              ; druhy  parametr - %i (X)
16     push msg              ; prvni  parametr - adresa vypisovaneho retezce
17     call printf           ; volame funkci printf
18     add  esp, 16          ; ctyri predane parametry mazeme ze zasobniku
19     ret
```

Na procvičení

Vyzkoušejte si:

- Ze vstupu načtěte jedno 32b číslo a na hromadě se pokuste o dynamickou alokaci odpovídajícího množství paměti (malloc).
- Předpokládejte že alokace uspěje – při neúspěšné alokaci ISU-HUB havaruje.
- Vypište řetězec "Na adrese %i je alokovano %i bajtu" (printf).
- Alokovanou paměť před ukončením programu nezapomeňte uvolnit (free).
- Všechny funkce jazyka C se vždy volají dle konvence CDECL.

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

Například:

- 20 => Na adrese 266768 bylo alokovano 20 bajtu

Lokální proměnné

Lokální proměnné

Lokální proměnné alokujeme po vytvoření nového dna (EBP) posunutím vrcholu zásobníku (ESP) směrem dolů:

- Pro každou proměnnou alokujeme 32b paměti.

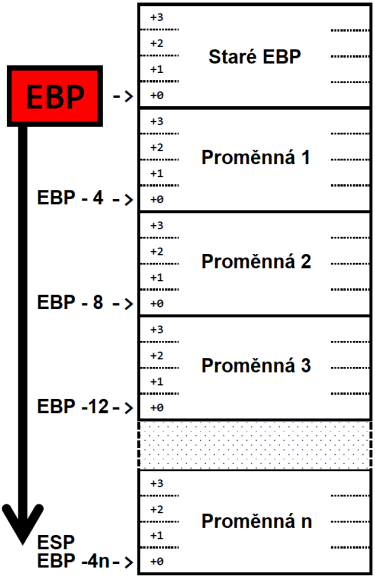
```
1  push ebp        ; zalohuj stare dno
2  mov  ebp, esp   ; vytvor nove dno
3  sub  esp, 8     ; alokujeme dve lokalni promenne
```

- K lokálním proměnným přistupujeme přes (neměnnou) adresu dna zásobníku (EBP):

```
4  [ebp - 0]       ; stare dno
5  [ebp - 4]       ; prvni lokalni promenna
6  [ebp - 8]       ; druha lokalni promenna
```

- Na konci funkce, před obnovením starého dna (EBP), alokované lokální proměnné musíme také uvolnit.

```
7  mov  esp, ebp   ; uvolni vsechny lokalni promenne
8  pop  ebp        ; obnov stare dno
9  ret             ; konec funkce
```



Funkce (volaná dle konvence CDECL) v řetězci vyhledá ukončující znak NUL, tím spočítá jeho délku, a jako počítadlo při tom použije lokální proměnnou:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2
3  section .data                ;inicializovany datovy segment
4      msg db "Hello, World!", 10, 0 ; textovy retezec
5
6  section .text                ;kodovy segment
7  main:
8      push msg                 ; predavame prvni parametr (adresa pole)
9      call DelkaRetezce        ; volame funkci
10     add  esp, 4              ; mazeme predany parametr
11     call WriteInt32NewLine   ; vypis EAX (navratova hodnota funkce)
12     ret
```

Na vyzkoušení (2/2)

```
13 DelkaRetezce:
14     push ebp                    ; zalohujeme stare dno
15     mov  ebp, esp               ; vytvarime  nove dno
16     sub  esp, 4                 ; alokujeme misto pro jednu lokalni promennou
17                                 ; [EBP - 4], kterou pouzijeme jako pocitadlo
18     mov  [ebp - 4], dword 0     ; pocitadlo = 0
19     mov  edi, [ebp + 8]         ; EDI = prohledavany retezec (parametr)
20     mov   al, 0                 ;  AL = hledany znak NUL (v ASCII hodnota 0)
21     cld                         ;  DF = 0 (posun doprava)
22
23   while:
24     scasb                       ; AL porovnej se znakem z retezce
25     je   end                    ; pokud se rovnaji, ukonci cyklus
26   do:                           ; jinak (AL != znaku z retezce)
27     inc  dword [ebp-4]          ; inkrementuj pocitadlo
28     jmp  while                  ; opakuj cyklus
29   end:
30     mov  eax, [ebp-4]           ; EAX = pocitadlo (navratova hodnota)
31
32     mov  esp, ebp               ; uvolnujeme vsechny lokalni promenne
33     pop  ebp                    ; obnovujeme stare dno
34     ret
```

Na procvičení

Vyzkoušejte si:

- Napište funkci ObratVypis která ze vstupu načte několik 32b hodnot, a v obráceném pořadí je vypíše na výstup.
- Funkci voláme dle konvence PASCAL.

Například:

- 10, 20, 30, 40, 50 => 50, 40, 30, 20, 10

```
1  %include 'rw32.inc' ;knihovna
2
3  section .text       ;kodovy segment
4  main:
5      push dword 5    ;predavame parametr
6      call ObratVypis ;volame funkci
7      ret
```

```
8  ObratVypis:
9    push ebp          ;stare dno
10   mov  ebp, esp     ;nove  dno
11
12   mov  eax, [ebp+8];EAX = N
13   sal  eax, 2       ;EAX = N*4
14   sub  esp, eax     ;alokujeme
15   ;misto pro pole nekolika
16   ;32b lokalnich promennych
17
18   ;??????????????????????
19
20   mov  esp, ebp     ;uvolnujeme
21   ;vsechny lokalni promenne
22
23   pop  ebp  ;obnovujeme dno
24   ret  4    ;koncime funkci a
25             ;mazeme parametr
```