Funkce, zálohování registrů, předávání parametrů přes zásobník a lokální proměnné

ISU-cv08

Ing. Jakub Husa

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

30. března 2023

Funkce

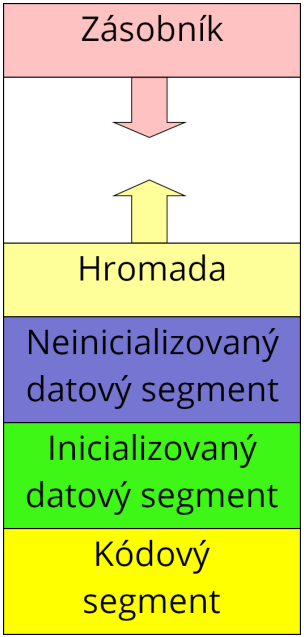
Zásobník

Zásobník je paměťový segment ukládající dočasná data (návratové adresy, parametry funkcí a lokální proměnné):

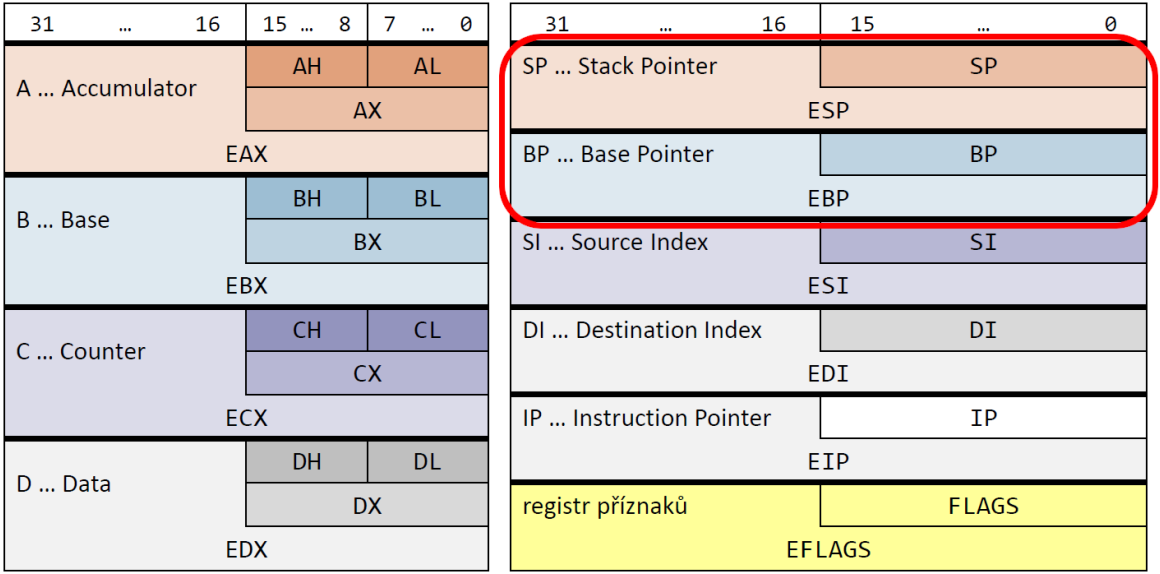
- Velikost zásobníku je (na rozdíl od datových segmentů) proměnlivá a závisí na množství právě uložených položek.
- Maximální velikost zásobníku je omezena pouze množstvím dostupné paměti (v 32b režimu 4 GiB).
- Pokud místo dojde nastává přetečení zásobníku.

Obsah zásobníku **ROSTE SMĚREM DOLŮ**:

- Adresu dna uchovává registr EBP (Base Pointer).
- Adresu vrcholu uchovává registr ESP (Stack Pointer).
- Vrchol má vždy stejnou nebo nižší adresu než dno!



Registry procesoru - vrchol a dno zásobníku



Funkce (podprogram) je úsek kódu označený návěštím,
volaný instrukcí CALL a zakončený instrukcí RET:

- CALL – na zásobník uloží adresu následující instrukce (návratová adresa)
  a do programového čítače (EIP) nahraje adresu volané funkce.
- RET – návratovou adresu odstraní ze zásobníku a nahraje do EIP.

Parametry můžeme funkcím předávat třemi způsoby:

- V registrech – uložíme je do EAX, EBX, ECX, EDX (hodnoty) nebo ESI, EDI (adresy).
- V paměti – uložíme je do globálních proměnných (.data, .bss).
- V zásobníku – uložíme je na zásobník (jak si ukážeme za chvíli).

Návratovou hodnotu funkce vracíme ve výstupním registru EAX:

- Každá funkce by měla končit právě jednou instrukcí RET.
- Každá instrukce programu by měla být součástí nějaké funkce.

Vyzkoušejte si:

- Funkce zdvojnásobí hodnotu čísla předaného v registru EAX.

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2
3  section .text            ;kodovy segment
4  _main:                   ;navesti funkce "_main" (zacatek programu)
5
6      mov  eax, 10         ; EAX = 10
7      call zdvojnasob      ; volani funkce zdvojnasob
8      call WriteInt32      ; vypis vysledek (EAX = 20)
9      call WriteNewLine
10
11     mov  eax, 15         ; EAX = 15
12     call zdvojnasob      ; volani funkce zdvojnasob
13     call WriteInt32      ; vypis vysledek (EAX = 30)
14     call WriteNewLine
15     ret                  ; konec funkce _main (konec programu)
16
17 zdvojnasob:              ;navesti funkce zdvojnasob
18     add eax, eax         ; EAX = EAX + EAX
19     ret                  ; konec funkce zdvojnasob (navrat do funkce main)
```

Na procvičení

Vyzkoušejte si:

- Vytvořte si pole 32b znaménkových čísel s nějakými počátečními hodnotami.
- Napište funkci sumaPole která jako vstup dostane adresu pole (ESI) a počet jeho prvků (EAX), a jako výstup vrátí jejich sumu (EAX).
- Funkci sumaPole zavolejte z funkce main, a vypište její výsledek.

Například:

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2
3  section .data            ;inicializovany segment
4      arr dd 10, 20, 30, 40, 50
5
6  section .text            ;kodovy segment
7  _main:
8      mov  esi, arr        ; ESI = adresa pole
9      mov  eax, 5          ; EAX = pocet jeho prvku
10     call sumaPole        ; volani funkce
11     call WriteInt32      ; vypis (EAX = 150)
12     ret
```

Zálohování registrů

Funkce by měla zachovávat hodnotu všech používaných registrů:

- Hodnoty registrů zálohujeme a obnovujeme instrukcemi PUSH a POP.
- PUSH – posune vrchol zásobníku a uloží na něj novou položku.
- POP – ze zásobníku odebere poslední uloženou položku a posune jeho vrchol.

```
1     push eax       ; hodnotu EAX uloz na zasobnik
2     pop  eax       ; hodnotu EAX odeber ze zasobniku
```

Položky ze zásobníku odebíráme v opačném pořadí než ve kterém jsme je uložili:

```
3  func:
4      push ebx       ; zalohujeme registr EBX
5      push ecx       ; zalohujeme registr ECX
6      push edx       ; zalohujeme registr EDX
7      ;telo funkce
8      pop  edx       ; obnovujeme registr EDX
9      pop  ecx       ; obnovujeme registr ECX
10     pop  ebx       ; obnovujeme registr EBX
11     ret
```

Položky uložené na zásobníku z něj před ukončením funkce musíme také odebrat:

- Pokud je neodebereme RET do EIP nahraje neplatnou adresu a program havaruje.

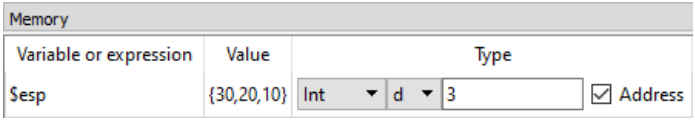
```
1 func:
2     push dword 10    ; na zasobnik ukladame 32b konstantu 10
3     ret              ; CHYBA - do EIP nahraje adresu "10"
```

Na zásobník vždy ukládáme pouze 32b hodnoty:

- K položkám zásobníku nám to umožní přistupovat jako k položkám pole.

```
4     push eax         ; hodnotu EAX uloz na zasobnik
5     mov  ebx, [esp]  ; do EBX uloz hodnotu z vrcholu zasobniku (EAX)
6     push ax          ; CHYBA - rozbijeme si adresovani zasobniku
```

Obsah zásobníku si můžeme zobrazit v debuggeru ($esp se zaškrtnutým address):



Vyzkoušejte si:

- Jak se bude měnit obsah zásobníku při provádění následujících instrukcí?

```
1  %include "rw32-2018.inc"      ;knihovna pro vstup a vystup
2
3  section .data                 ;inicializovany segment
4      var dd 222                ; 32b globalni promenna
5
6  section .text                 ;kodovy segment
7  _main:                        ;[ESP+0] [ESP+4] [ESP+8]
8      mov  eax, 111             ;    ?       ?       ?
9      push eax                  ;  111       ?       ?
10     push dword [var]          ;  222     111       ?
11     push dword 333            ;  333     222     111
12     pop  eax                  ;  222     111       ?
13     mov  dword [esp + 4], 444;  222     444       ?
14     push dword [esp + 4]      ;  444     222     444
15     pop  eax                  ;  222     444       ?
16     pop  eax                  ;  444       ?       ?
17     pop  eax                  ;    ?       ?       ?
18     ret
```

Na procvičení

Vyzkoušejte si:

- Napište program který ze vstupu načte 8b číslo (X) a X-krát zavolá funkci vypisRadek.
- Funkce vypisRadek vypíše X hvězdiček a jeden konec řádku.
- Vstupem funkce bude hodnota X, předávaná v registru EAX.
- Funkce musí zachovat hodnotu všech používaných registrů.

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2  section .text            ;kodovy segment
3  _main:
4      call ReadInt8        ; AL = vstup
5      cbw                  ; AL -> AX
6      cwde                 ; AX -> EAX
7      mov  ecx, eax        ; ECX = pocet opakovani
8    for:
9      call vypisRadek      ; volani funkce vypisRadek
10     loop for             ; volani X-krat opakuj
11     ret
```

Například:

- 1 => *
- 2 => **
**
- 3 => ***
***
***
- 4 => ****
****
****
****

Předávání parametrů přes zásobník

Zásobníkový rámec

Zásobníkový rámec (stack frame, oblast mezi EBP a ESP) na zásobníku ohraničuje místo které patří aktuální funkci:

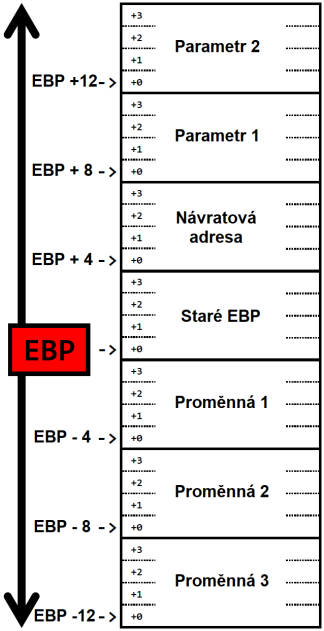
- Zásobníkový rámec nám umožňuje funkcím přes zásobník předávat parametry a alokovat pro ně lokální proměnné.

Každá funkce si potřebuje vytvořit svůj vlastní ráme:

- Na začátku zálohujeme staré dno volající funkce a vytvoříme nové dno volané funkce.
- Na konci obnovíme staré dno volající funkce.

```
1  func:
2      push ebp          ; zalohuj stare dno
3      mov  ebp, esp     ; vytvor nove dno
4      ;telo funkce
5      pop  ebp          ; obnov stare dno
6      ret               ; konec funkce
```

- Jinak adresu zásobníkového dna nikdy neupravujeme.



Předávání parametrů

Volající funkce parametry ukládá na zásobník (PUSH):

- Volání funkce (CALL) uloží návratovou adresu.
- Volaná funkce uloží staré dno (PUSH).

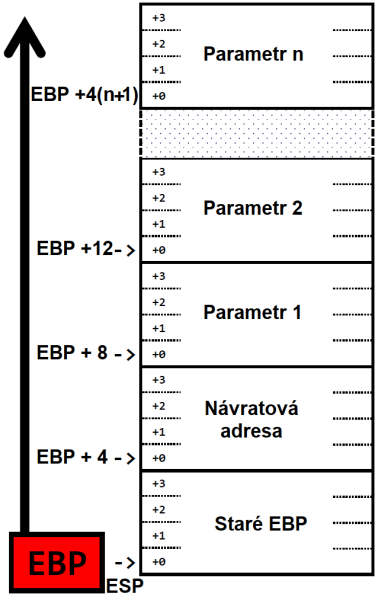
Protože se adresa nového dna (EBP) se nikdy nemění můžeme ji použít k adresování uložených parametrů:

```
1  [ebp + 12]     ; druhy parametr
2  [ebp +  8]     ; prvni parametr
3  [ebp +  4]     ; navratova adresa
4  [ebp +  0]     ; stare dno
```

Parametry můžeme ze zásobníku odstranit instrukcí POP, nebo úpravou adresy vrcholu zásobníku (ESP):

```
5  add  esp, 4    ; odstran jeden parametr
```

- Funkce svoje parametry nesmí přepisovat.
- Návratovou hodnotu stále vracíme přes EAX.



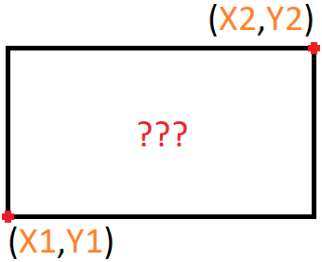
Vyzkoušejte si:

- Funkce spočítá součet dvou parametrů předaných přes zásobník.

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2
3  section .text            ;kodovy segment
4  _main:
5      push dword 20        ; predevame druhy parametr
6      push dword 10        ; predavame prvni parametr
7      call soucet          ; volame funkci soucet
8      add  esp, 8          ; oba dva paremetry odstranujeme ze zasobniku
9      call WriteInt32      ; vypis (EAX = 30)
10     ret
11
12 soucet:
13     push ebp             ; zalohuj stare dno
14     mov  ebp, esp        ; vytvor nove dno
15     mov  eax, [ebp + 8]  ; EAX = prvni parametr
16     add  eax, [ebp +12]  ; EAX = EAX + druhy parametr
17     pop  ebp             ; obnov stare dno
18     ret                  ; konec funkce
```

Vyzkoušejte si:

- Napište funkci obsah která spočítá obsah obdélníku ze souřadnic (X1, Y1) a (X2, Y2) předávaných přes zásobník.
- Funkce musí zachovat hodnotu všech registrů vyjma výstupního EAX (horních 32b výsledku můžete ignorovat).



Například:

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2  section .text            ;kodovy segment
3  _main:
4      push dword 20        ; ctvrty parametr (Y2)
5      push dword 10        ; treti  parametr (X2)
6      push dword 5         ; druhy  parametr (Y1)
7      push dword 0         ; prvni  parametr (X1)
8      call obsah           ; volame funkci
9      add  esp, 16         ; ze zasobniku odstranujeme 4 parametry
10     call WriteInt32      ; vypis (EAX = 150)
11     ret
```

Lokální proměnné

Lokální proměnné

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

- Pro každou proměnnou rezervujeme 32b paměti.

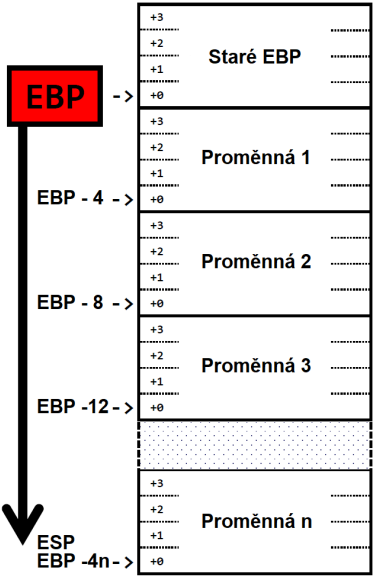
```
1  push ebp        ; zalohuj stare dno
2  mov  ebp, esp   ; vytvor nove dno
3  sub  esp, 4     ; rezervuj jednu lokalni promennou
```

- 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, lokální proměnné musíme také uvolnit.

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



Vyzkoušejte si:

- Funkce spočítá délku načteného řetězce.

```
1  %include "rw32-2018.inc"
2
3  section .bss            ;datovy segment
4      arr resb 30
5
6  section .text           ;kodovy segment
7  _main:
8      push ebp            ;zalohujeme stare
9      mov  ebp, esp       ;vytvarime nove
10
11     mov  edi, arr       ;cilove pole
12     mov  ebx, 29        ;max pocet znaku
13     call ReadString     ;nacti retezec
14
15     mov  esi, arr       ;zdrojove pole
16     call delkaRetezce;volej funkci
17     call WriteInt32     ;vypis
18
19     pop  ebp            ;obnovujeme stare
20     ret
```

```
21 delkaRetezce:
22     push ebp        ;zalohujeme stare
23     mov  ebp, esp   ;vytvarime nove
24     sub  esp, 4     ;rezervujeme jednu
25                     ;lokalni promennou
26     mov  dword [ebp-4], 0 ;suma = 0
27
28   condition:
29     lodsb           ;AL = znak
30     cmp  al, 0      ;je znak null?
31     je   end        ;pokud ano, konec
32   while:
33     inc  dword [ebp-4] ;suma++
34     jmp  condition     ;opakuj
35   end:
36     mov  eax, [ebp-4]  ;eax = vysledek
37
38     mov  esp, ebp ;uvolnujeme lokalni
39     pop  ebp      ;obnovujeme stare
40     ret
```

Na procvičení

Vyzkoušejte si:

- Napište funkci obratVypis která ze vstupu načte několik 32b hodnot a vypíše je na výstup v obráceném pořadí.
- Funkce nesmí používat globální proměnné.
- Funkce nemusí zálohovat hodnoty používaných registrů.

Například:

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

```
1  %include "rw32-2018.inc" ;knihovna pro vstup a vystup
2  section .text            ;kodovy segment
3  _main:
4      push ebp             ; zalohujeme stare dno
5      mov  ebp, esp        ; vytvarime nove dno
6
7      push dword 5         ; predavame prvni parametr
8      call obratVypis      ; volame funkci
9      add  esp, 4          ; parametr odstranujeme ze zasobniku
10
11     pop  ebp             ; obnovujeme stare dno
12     ret
```