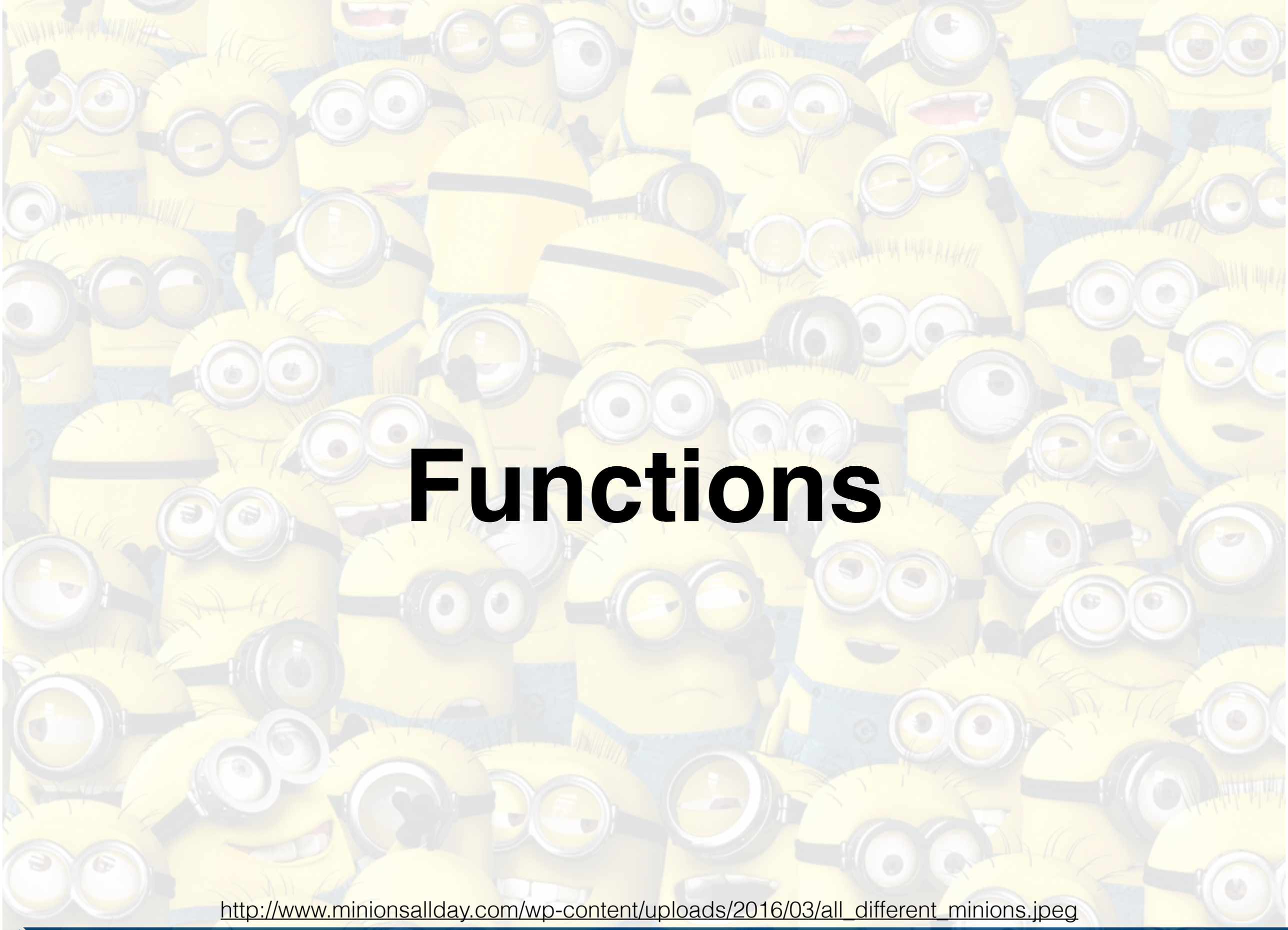
Smith College
Computer Science

CSC231 — Assembly

Week #11 — Spring 2017

Dominique Thiébaut
dthiebaut@smith.edu

Functions



http://www.minionsallday.com/wp-content/uploads/2016/03/all_different_minions.jpeg

eax

ebx

ecx

edx

esi

edi

eip

esp

CF | PF | ZF | SF | OF | DF

ALU

eax ??

Memory

?? <— esp

??

??

??

??

??

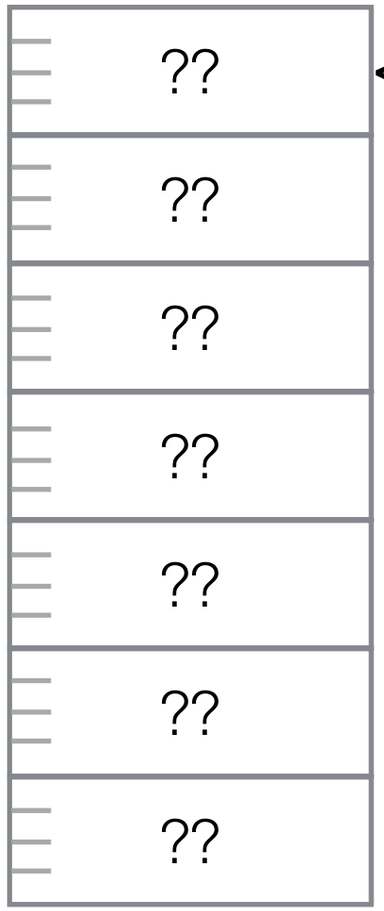
??

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004    Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eip—>

eax 10

Memory

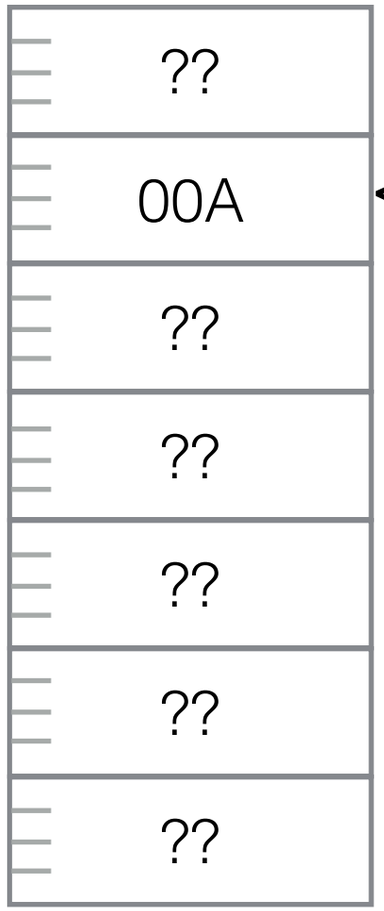



```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eip—>

eax 10

Memory

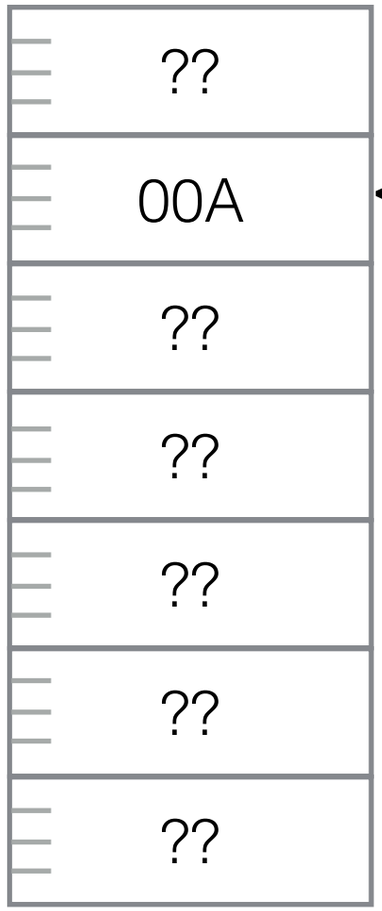


eip—>

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eax ~~10~~ 11

Memory



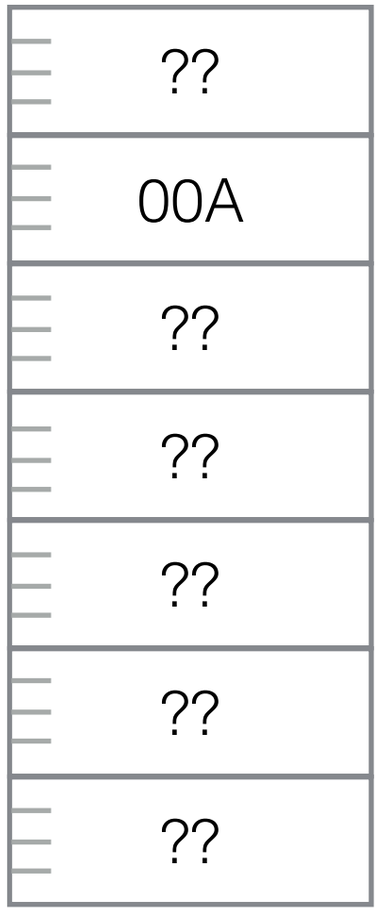

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret
```

eip—>

```
004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eax ~~10~~ 11

Memory

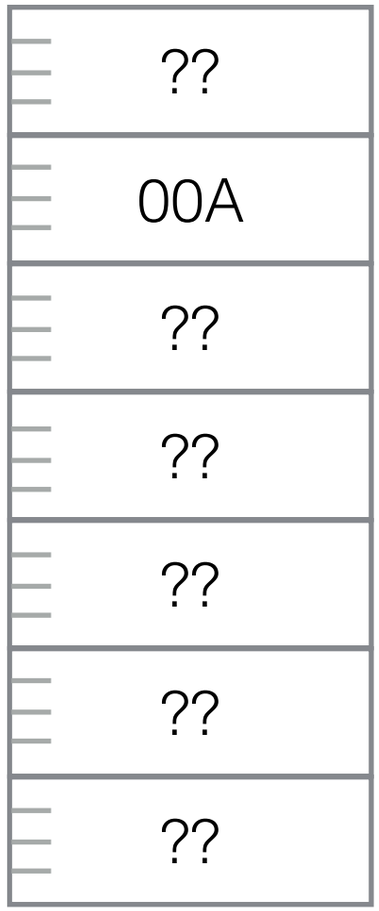



```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eip—>

eax ~~10~~ ~~11~~ 6

Memory

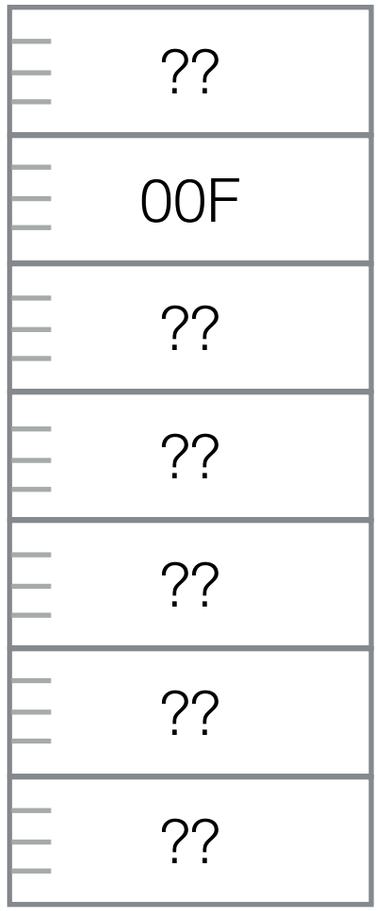



```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eip—>

eax ~~10~~ ~~11~~ 6

Memory



eip—>

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eax ~~10~~ ~~11~~ ~~6~~ 7

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret
```

eip—>

```
004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

Memory

eax ~~10~~ ~~11~~ ~~6~~ 7

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
```

eip—>

Memory

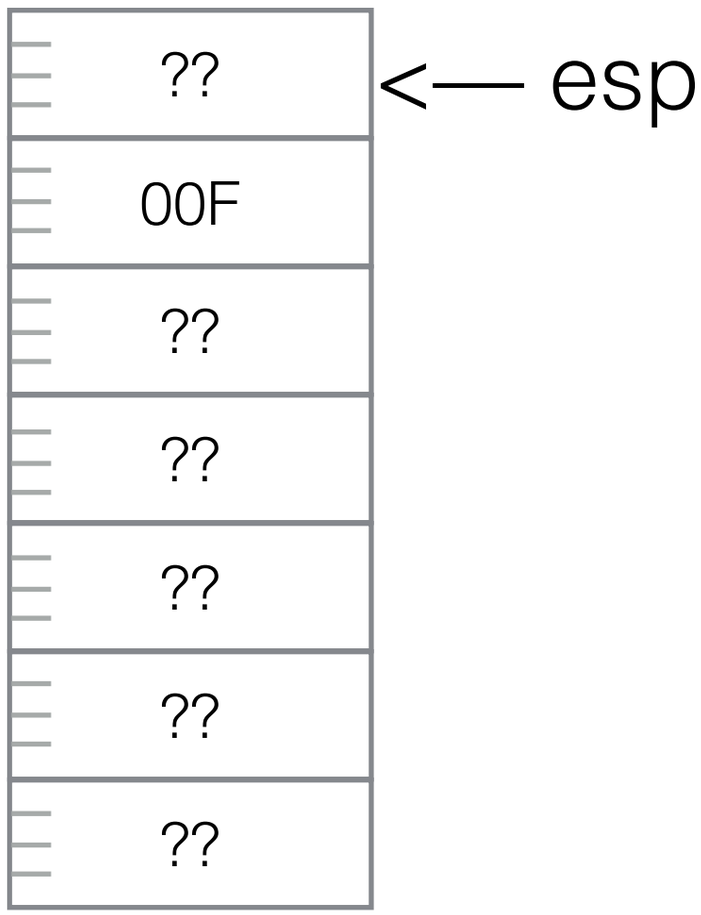
??	<— esp
00F	
??	
??	
??	
??	
??	
??	

eax 10 ~~11~~ ~~67~~ 10

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            ret

004   ; main()
004   _Start:
004            mov      eax, 10
007            call     add1
00A            sub      eax, 5
00C            call     add1
00F            add      eax, 3
012            ???      ???
```

eip—>

Memory



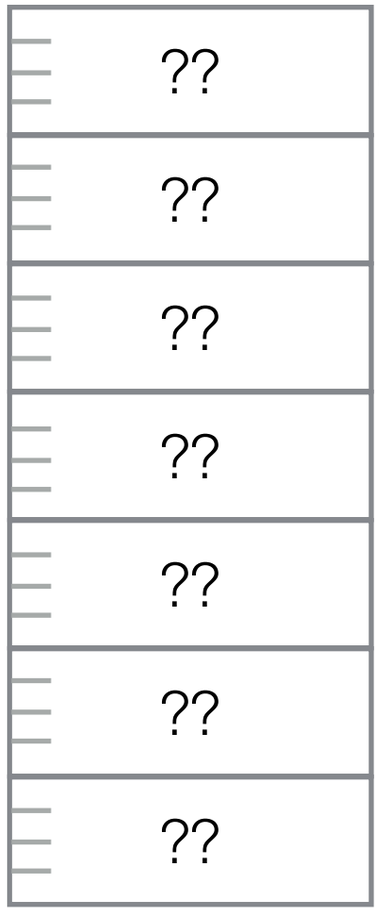
func: *inst*1
inst2
ret → pop into eip

call func → { push addr of inst3
jmp func
inst3

What about Nested Function Calls?

eax ??

Memory

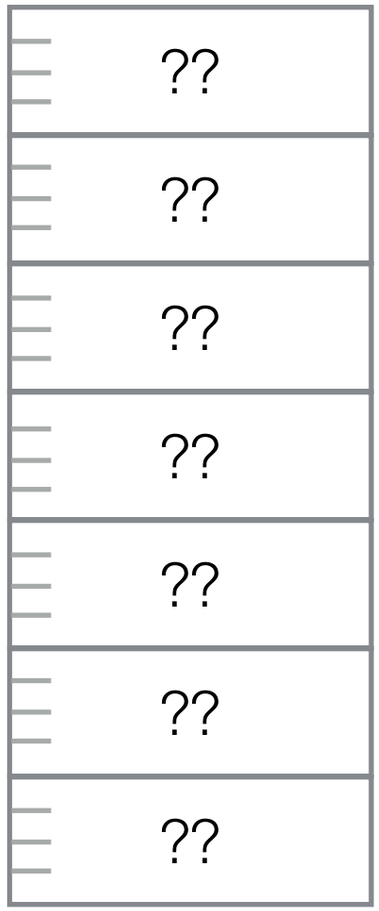



```
001   ; add1(): adds 1 to eax
001   add1:   inc     eax
002           call    sub2
003           ret

004   sub2:   sub     eax, 2
008           ret

00A   Start:
00A           mov     eax, 10
00F           call    add1
011           sub     eax, 5
```

eip—>

eax 10

Memory

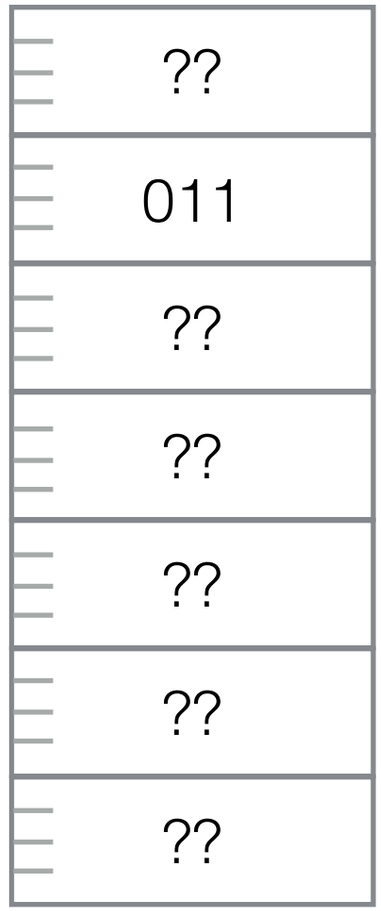



```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eip—>

eax 10

Memory



eip—>

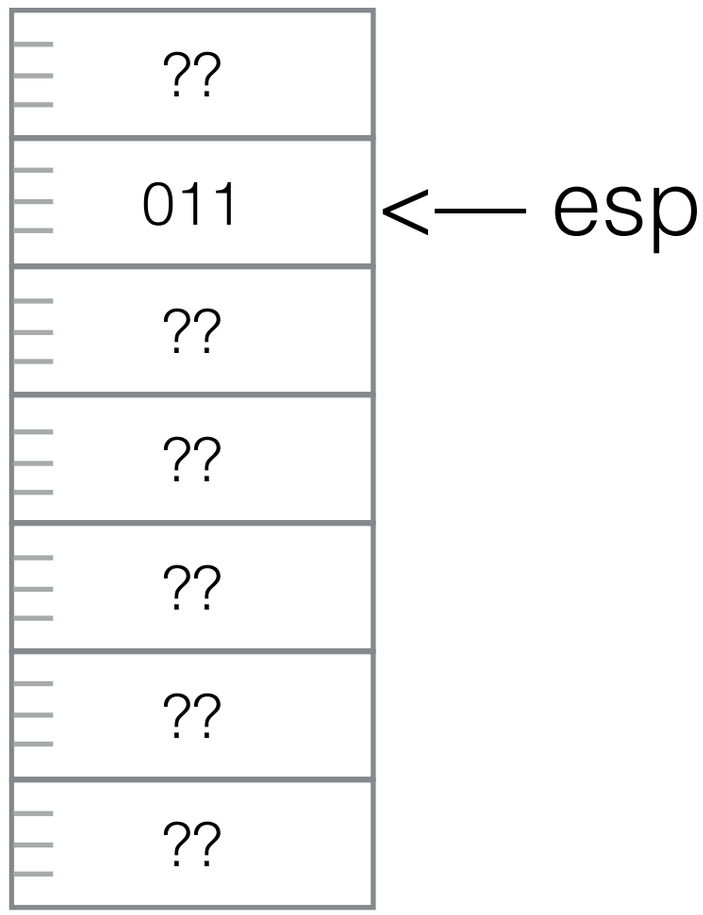
```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eax 10 11

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eip—>

Memory

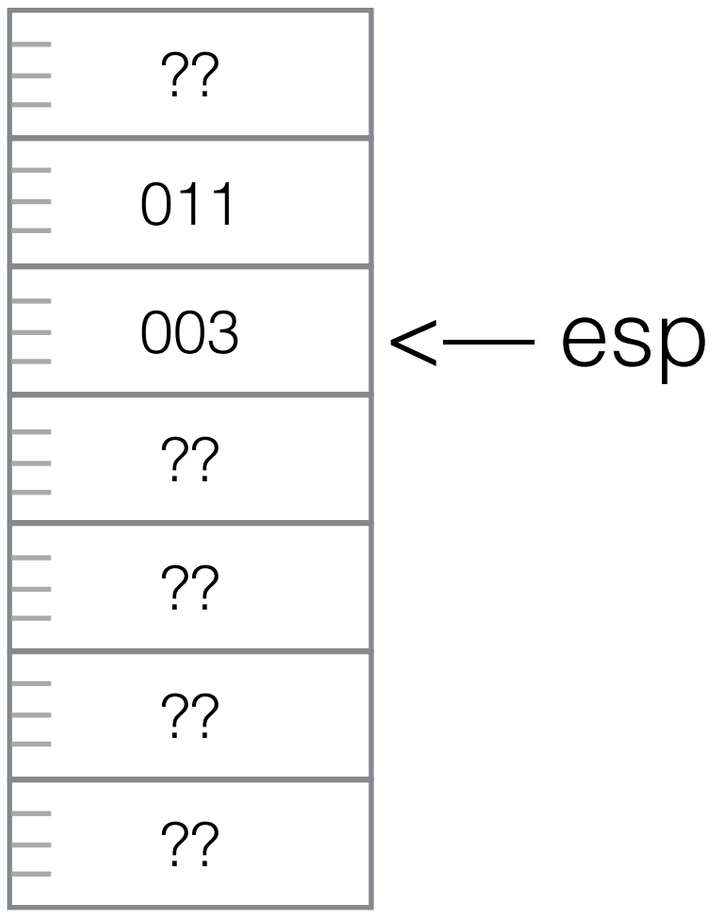


eax ~~10~~ 11

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eip—>

Memory

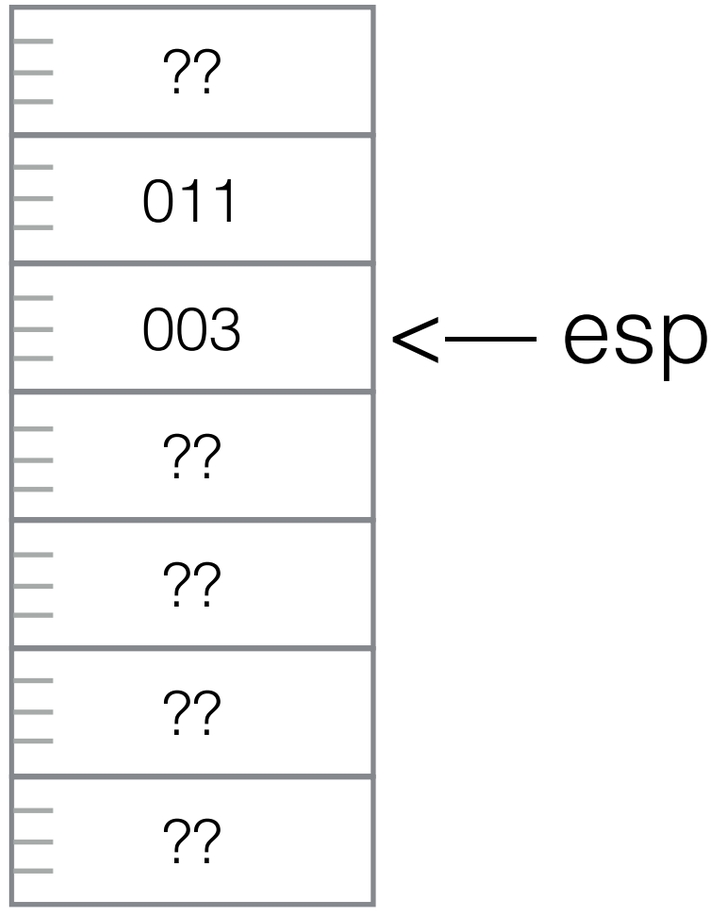


eax ~~10~~ ~~11~~ 9

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eip—>

Memory



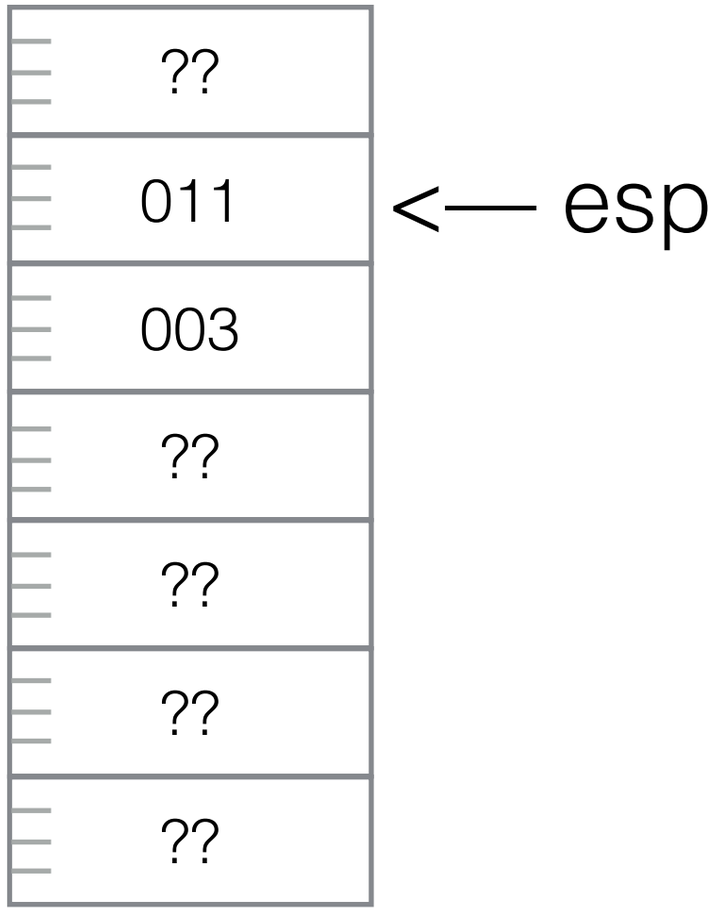
eax ~~10~~ ~~11~~ 9

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret
```

eip—>

```
004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

Memory



eax ~~10~~ ~~11~~ 9

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A  _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eip—>

Memory

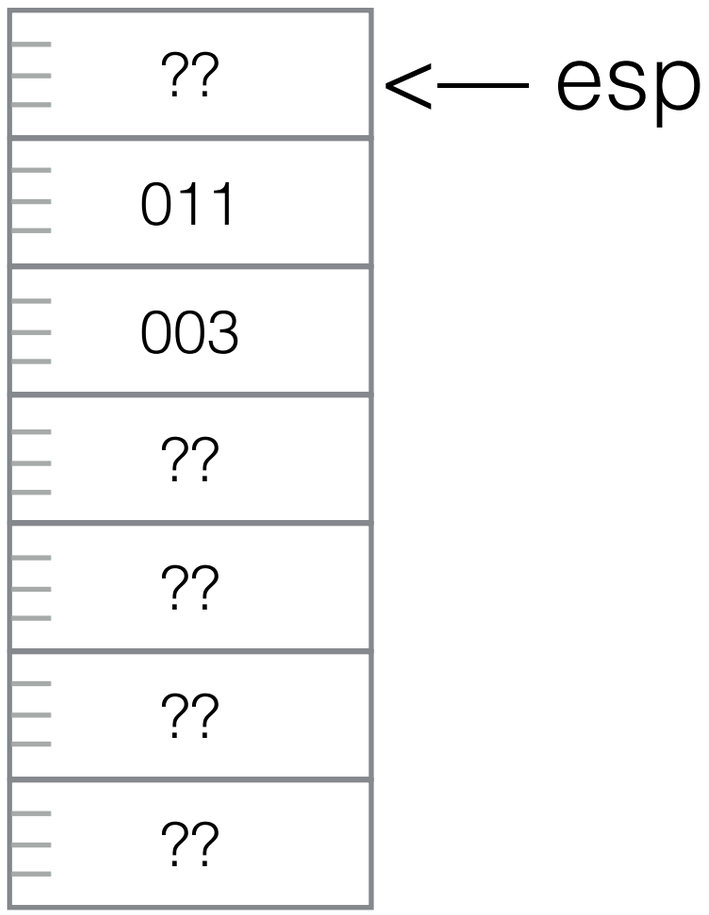
??	<— esp
011	
003	
??	
??	
??	
??	

eax ~~10~~ ~~11~~ ~~9~~ 4

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
002            call     sub2
003            ret

004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eip—>

Memory



Exercise

- Modify the Game of Life game developed in class and add two functions:
  - one to print the dish
  - one to compute life

http://www.science.smith.edu/dftwiki/index.php/CSC231_Developing_the_Game_of_Life_in_Assembly

Passing Parameters to Functions

- Passing through **registers**
- Passing through the **stack**
- Passing by **Value**
- Passing by **Reference**

Passing Parameters Via Registers

```
001   ; add1(): adds 1 to eax
001   add1:    inc      eax
003            call     sub2
003            ret

004   ; sub2: subtracts 2 from ex
004   sub2:    sub      eax, 2
008            ret

00A _Start:
00A            mov      eax, 10
00F            call     add1
011            sub      eax, 5
```

eax

ebx

ecx

edx

esi

edi

eip

esp

ebp

CF | PF | ZF | SF | OF | DF

ALU

Two New Instructions:

Push & Pop

Passing Parameters Through the Stack

```
void printSumXY( int x, int y ) {
   print( x+y );
}
```

```
int a, b;
a = 3;
b = 5;
printSumXY( a, b ); // prints 8
```

```
                section .data
a               dd      0
b               dd      0

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
```

```
                section .data
a               dd      0
b               dd      0

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A   Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
```

<— eip

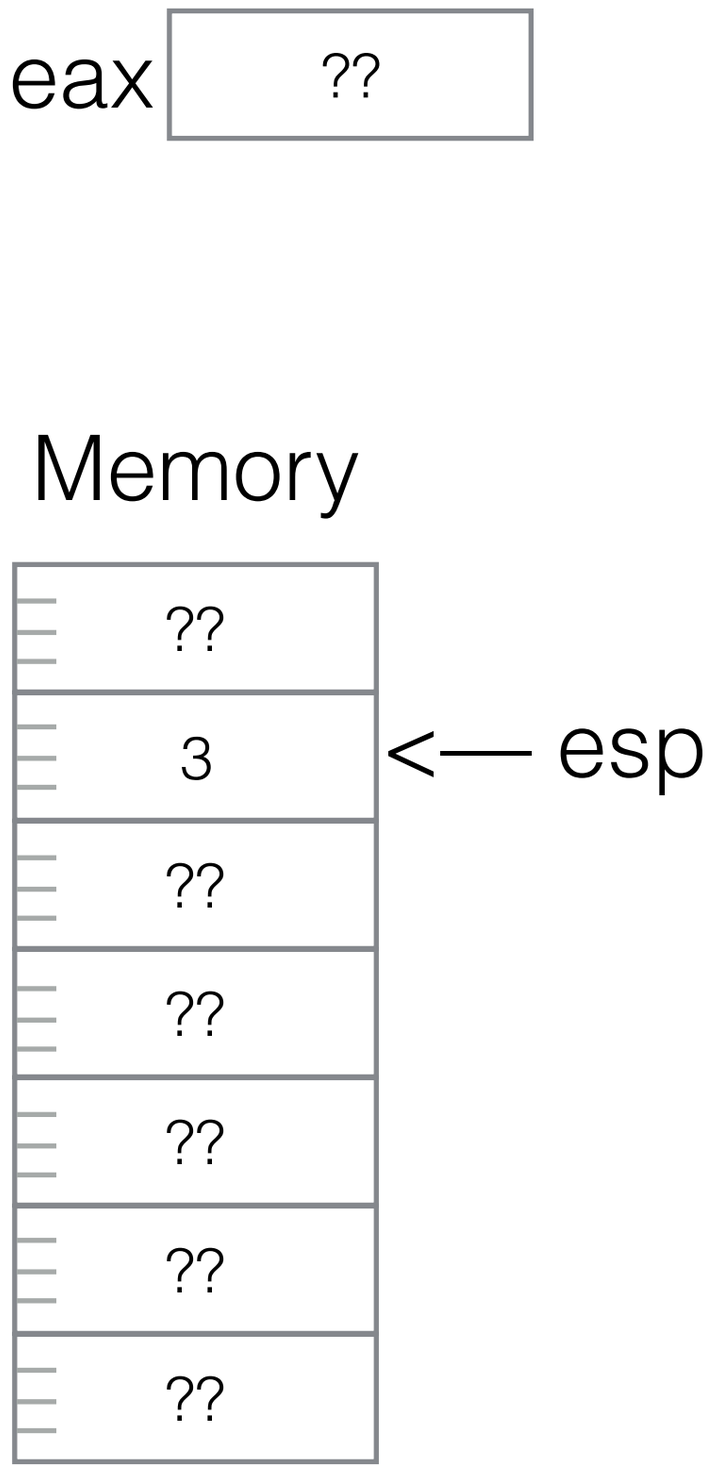
```
                section .data
a               dd      0 3
b               dd      0

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
```

<— eip

```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
```

<— eip

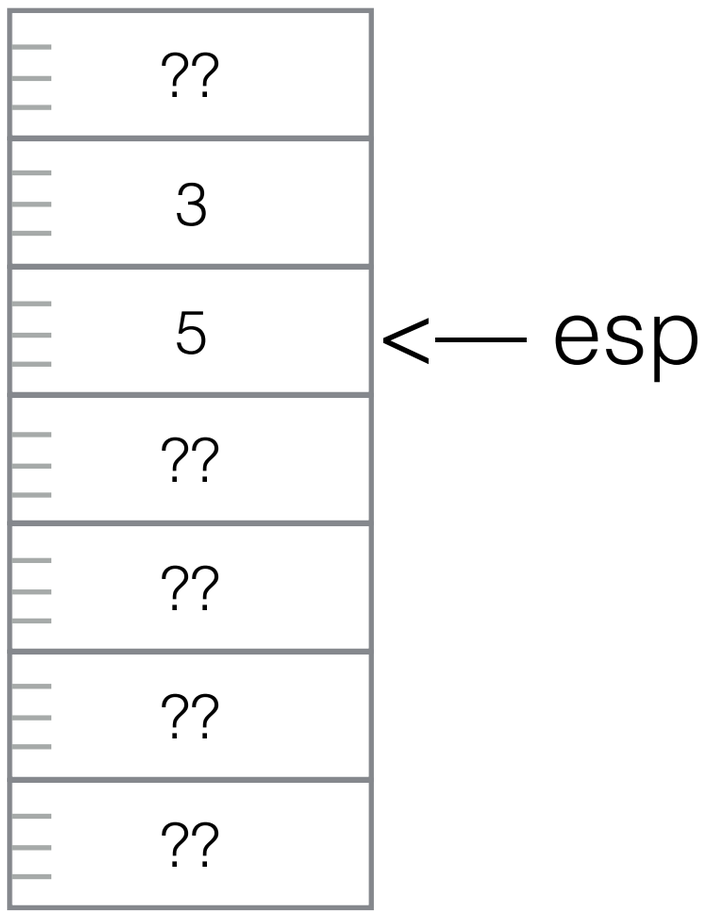
```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]        <— eip
025             call    printSumXY
```



```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY      <— eip
027
```

eax ??

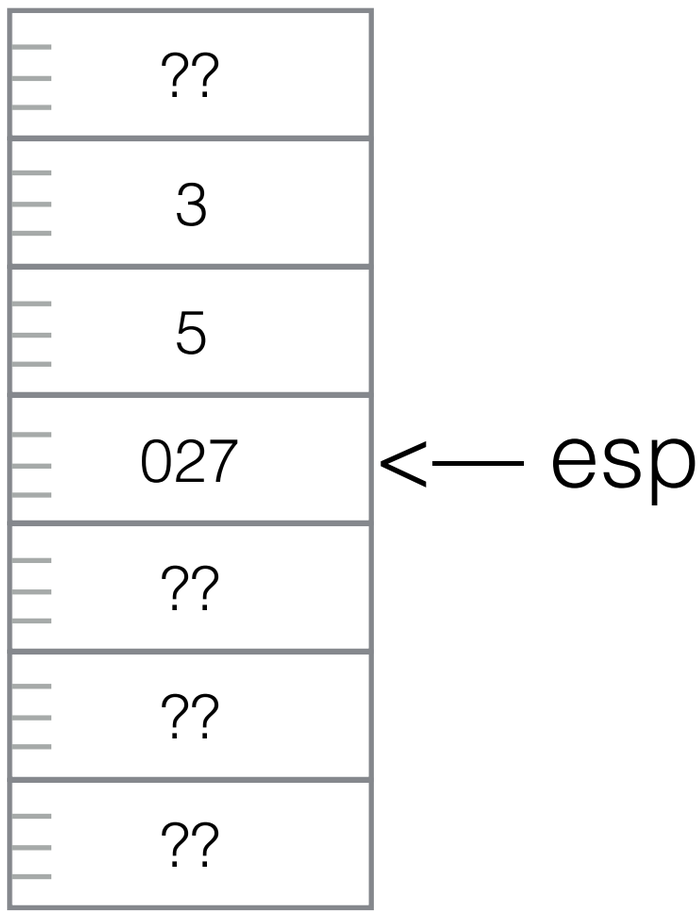
Memory



```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp                     <— eip
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

eax ??

Memory

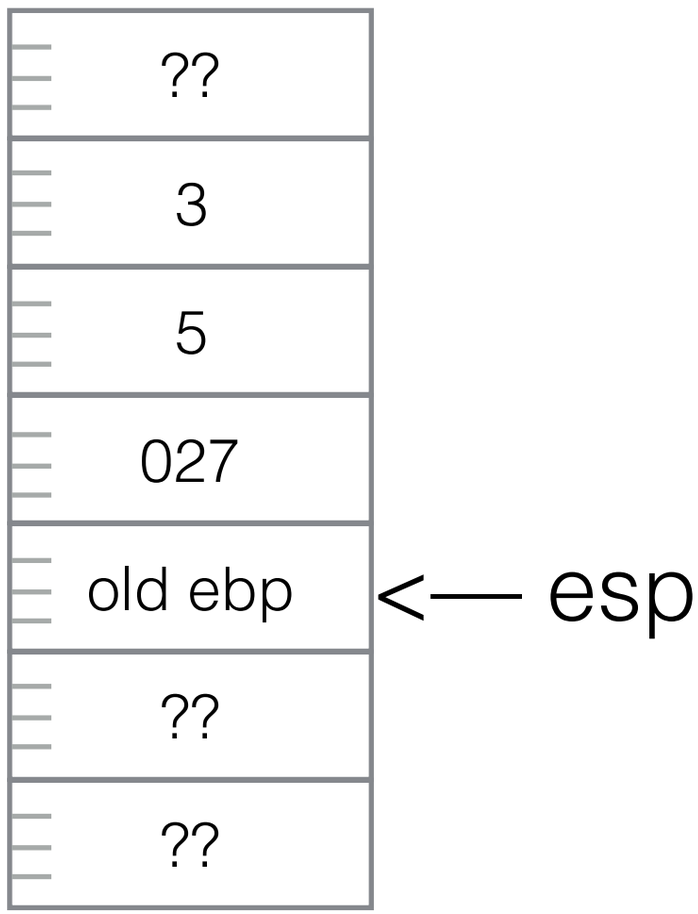


```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

<— eip

eax ??

Memory



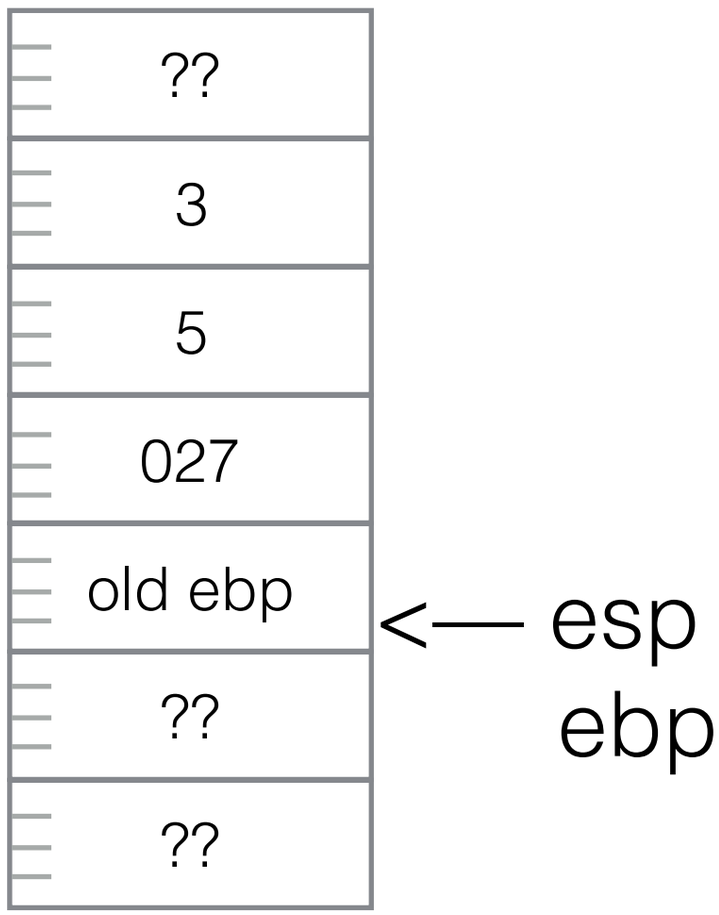
<— esp

```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

<— eip

eax ??

Memory



<— esp
ebp

```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]   <— eip
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

eax | 5

Memory

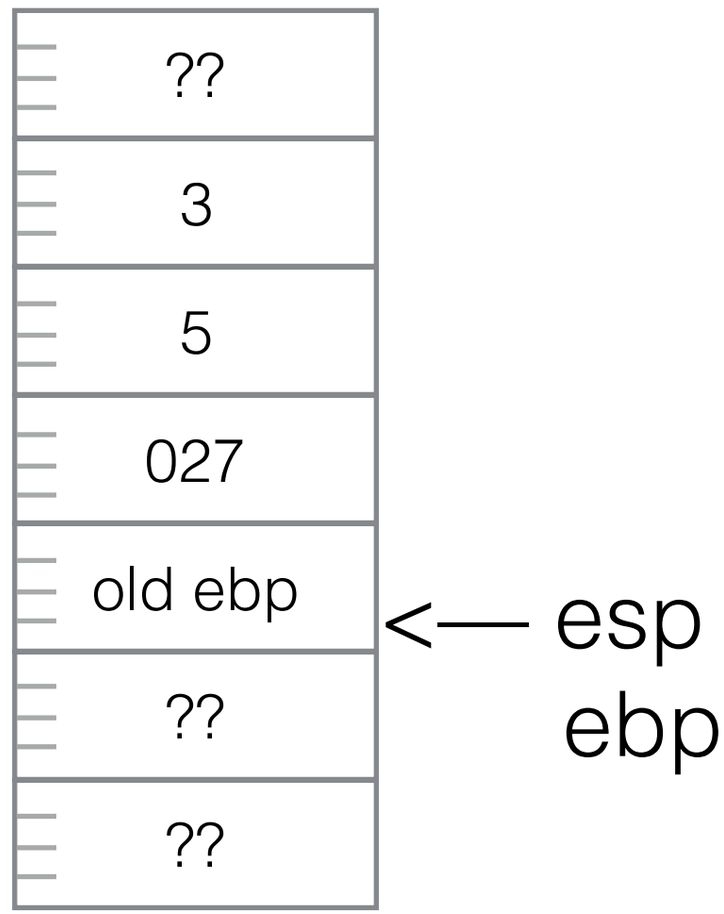
??
3
5
027
old ebp
??
??

<— esp
ebp

```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt          <— eip
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

eax: ~~5~~ 8

Memory



somewhere where _printInt lives <— eip

eax ~~5~~ 8

```
                section .data
a               dd      ~~0~~ 3
b               dd      ~~0~~ 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

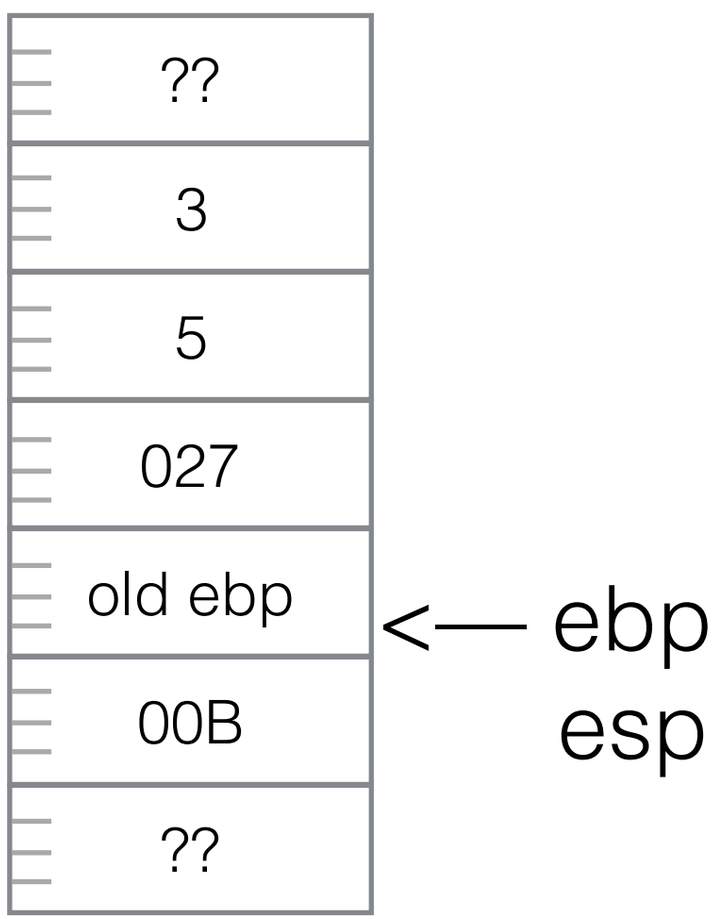
Memory

```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

<— eip

eax: ~~0~~ 8

Memory



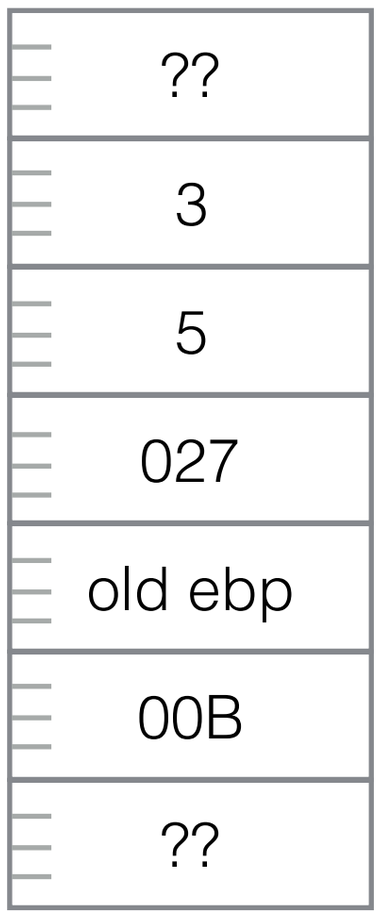
```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8

01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027
```

<— eip

eax: ~~0~~ 8

<— ebp

Memory



<— esp

```
                section .data
a               dd      0 3
b               dd      0 5

                section .text
001   ; printSumXY(x,y): prints x+y
001   printSumXY:
001             push    ebp
003             mov     ebp, esp
005             mov     eax, dword[ebp+8]
007             add     eax, dword[ebp+12]
009             call    _printInt
00B             pop     ebp
00C             ret     8


01A _Start:
01A             mov     dword[a], 3
01F             mov     dword[b], 5
021             push    dword[a]
023             push    dword[b]
025             call    printSumXY
027             …
```

<— eip

eax: ~~5~~ 8

<— ebp

Memory

??	<— esp
3	
5	<— esp
027	
old ebp	
00B	
??	

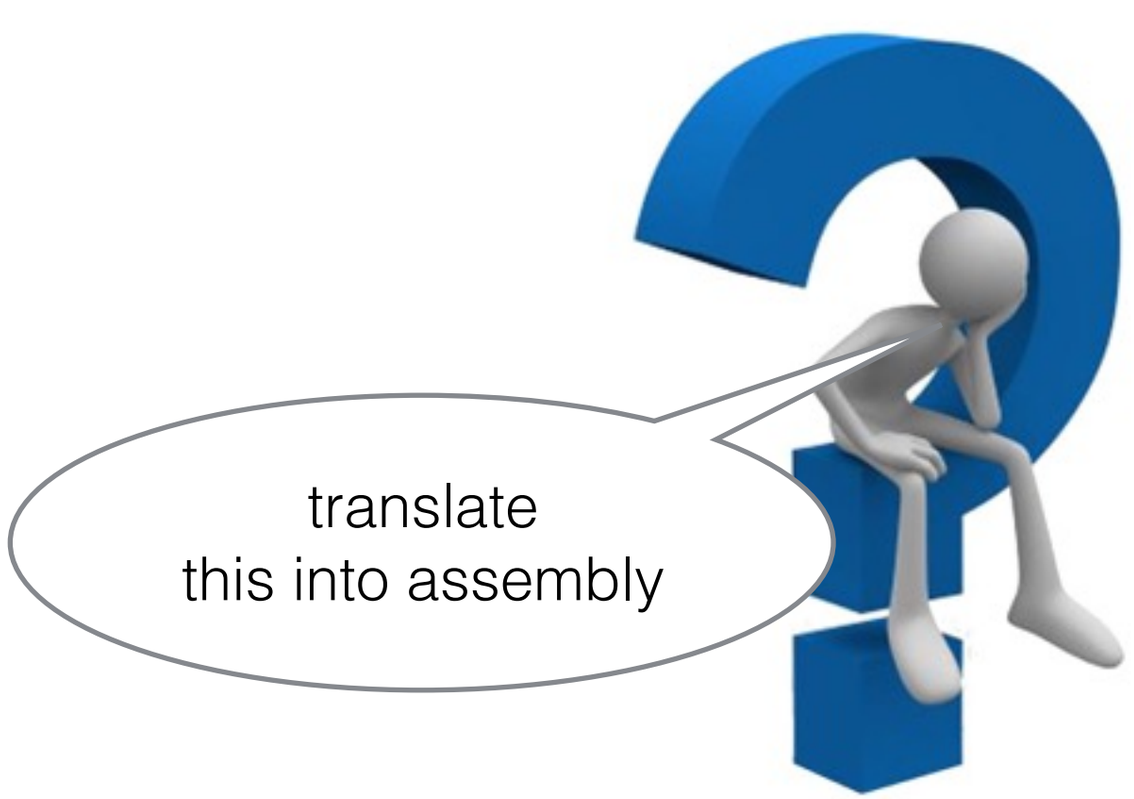
We stopped here last time…

http://john.do/wp-content/uploads/2012/07/rest-and-relaxation-r-and-r-570x320.jpg

Exercise

```
void func( int x ) {
   return x*3 + 2;
}
```

```
int a, b, c, d;
a = 3;
b = 5;
c = func( a );
d = func( 30 );
```



Summary

```
;;; FUNCTION SIDE

function:   push    ebp             ;save old ebp
            mov     ebp, esp        ;make ebp point
                                    ;to stack frame

            xxx     dword[ebp+8]    ;access paramN
            xxx     dword[ebp+12]   ;access paramN-1


            pop     ebp             ;restore ebp
            ret     4N              ;return and pop
                                    ;4N bytes from stack
```

```
;;; CALLER SIDE

            push    param1
            push    param2
            …
            push    paramN
            call    function
```

Summary
(clean function)

```
;;; FUNCTION SIDE

function:    push    ebp             ;save old ebp
             mov     ebp, esp        ;make ebp point
                                     ;to stack frame
             push-registers-you-will-use

             xxx     dword[ebp+8]  ;access param1
             xxx     dword[ebp+12];access param2

             pop-register-you-used

             pop     ebp             ;restore ebp
             ret     4N              ;return and pop
                                     ;4N bytes from stack

;;; CALLER SIDE

                push    param1
                push    param2
                …
                push    paramN
                call    function
```

Great Instructions for Functions

```
pushad              ;push EAX, ECX, EDX,
                    ;EBX, original ESP,
                    ; EBP, ESI, and EDI.

popad               ;pop them back in
                    ;reverse order
```

Example 1

```
;;; ; ------------------------------------------------------------
;;; ; _printString:         prints a string whose address is in
;;; ;                      ecx, and whose total number of chars
;;; ;                      is in edx.
;;; ; Examples:
;;; ; Assume a string labeled msg, containing "Hello World!",
;;; ; and a constant MSGLEN equal to 12.  To print this string:
;;; ;
;;; ;           mov        ecx, msg
;;; ;           mov        edx, MSGLEN
;;; ;           call       _printSTring
;;; ;
;;; ; REGISTERS MODIFIED:  NONE
;;; ; ------------------------------------------------------------

;;; ;save eax and ebx so that they are not modified by the function

_printString:
                push            eax
                push            ebx

                mov             eax,SYS_WRITE
                mov             ebx,STDOUT
                int             0x80

                pop             ebx
                pop             eax
                ret
```

Example 2

```
;;; ;-----------------------------------------------------
;;; ;-----------------------------------------------------
;;; ; getInput: gets a numerical input from the keyboard.
;;; ;           returns the resulting number in eax (32 bits).
;;; ;           recognizes - as the first character of
;;; ;           negative numbers.  Does not skip whitespace
;;; ;           at the beginning.  Stops on first not decimal
;;; ;           character encountered.
;;; ;
;;; ; NO REGISTERS MODIFIED, except eax
;;; ;
;;; ; Example of call:
;;; ;
;;; ;          call  getInput
;;; ;          mov   dword[x], eax ; put integer in x
;;; ;
;;; ;-----------------------------------------------------
;;; ;-----------------------------------------------------
_getInput:
                section .bss
buffer          resb    120
intg            resd    1
isneg           resb    1

                section .text
                pushad                  ; save all registers

                mov     esi, buffer     ; eci --> buffer
                mov     edi, 0          ; edi = counter of chars
.loop1:
                mov     eax, 03         ; input
```

Dot-Labels

```
;;; ;-----------------------------------------------------
;;; ;-----------------------------------------------------
printLine:      mov     esi, buffer     ; esi --> buffer
                mov     edi, 0          ; edi = counter of chars
.loop1:
                mov     eax, 03         ; input
                mov     ebx, 0          ; stdin
                mov     ecx, esi        ; where to put the next char
                loop    .loop1

.for1:          mov     …

.for2:          mov     …
                . . .
                ret

;;; ;-----------------------------------------------------
;;; ;-----------------------------------------------------
printReg:       mov     esi, buffer
                mov     edi, 0

.for1:          mov     …

.for2:          mov     …
                . . .
                . . .
                ret
```

Dot-Labels

```
;;; ;-------------------------------------------------------
;;; ;-------------------------------------------------------
printLine:      mov     esi, buffer     ; esi --> buffer
                mov     edi, 0          ; edi = counter of chars
.loop1:
                mov     eax, 03         ; input
                mov     ebx, 0          ; stdin
                mov     ecx, esi        ; where to put the next char
                loop    .loop1

.for1:          mov     …

.for2:          mov     …
                . . .
                ret

;;; ;-------------------------------------------------------
;;; ;-------------------------------------------------------
printReg:       mov     esi, buffer
                mov     edi, 0

.for1:          mov     …

.for2:          mov     …
                . . .
                . . .
                ret
```

printLine.for2:

printReg.for2:

- Passing through **registers** ✓
- Passing through the **stack** ✓
- Passing by **Value** ✓
- Passing by **Reference**