Notes on Stack

**Push Instruction**:

Places the operand of TOS, and decrements SP by 2

Syntax:

Push imme

Push [mem]

Push reg

**Pop Instruction:**

Pops TOS element in destination operand

Systax:

Pop reg

Pop [mem]

**CALL:**

Place a call to function. Changes IP to address of first instruction of function and pushes the return address on TOS

Sytax:

Call functionName

**RET:**

Used to return from function to caller. Pops TOS to IP so make sure that the TOS is returning address.

Syntax:

Ret

**RET N:**

Used to return from function to caller and discard input args from stack. Pops TOS to IP so make sure that the TOS is returning address. It also subtracts n from SP.

Syntax:

Ret n

**Template of Caller:**

;Make space of output arguments on stack using sub sp,m where m=2\*number of output args

; push input arguments to stack

;Call function

;pop output arguments from stack

**Template of Function:**

FunctionName:

Push bp

Mov bp,sp

; make space for local variables using sub sp, n

; save registers by pushing in stack

; Body of function

; restore registers using pop

; Release space of local variables by using add sp, n

; pop bp

;ret p where p is size of input arguments in bytes.

|  |
| --- |
|  |
| Registers value |
| Local variables |
| bp |
| Returning address |
| Input arguments |
| Ouput arguments |
| …. |

**Stack Frame of function:**

bp

**How to access input arguments output arguments and local variables in function**

Input/output arguments are accessed using:

[bp-x] where x is offset it will depend on which argument you are accessing

Local variable are accessed using

[bp+x] where x is offset, it will depend on which local variable you are accessing.

**Example:**

**Write a function that takes three numbers as input argument and return the highest number as output argument. The arguments (input/output) are to be passes to function on stack.**

**Create a local variable in function to keep track of max number.**

**All the values of regusters should be saved and restored before and after call.**

**After return from function pop the output in ax.**

**Solution:**

Jmp start

|  |
| --- |
|  |
| Local variable |
| bp |
| Return add |
| Arg 3 |
| Arg 2 |
| Arg 1 |
| Output (max) |

findMax:

push bp

mov bp,sp

sub sp, 2; space for local variable

push ax ; save registers

push bx

push cx

; body

Mov ax, [bp-4]

Mov bx, [bp-6]

Mov cx, [bp-8]

Cmp ax, bx

Ja AGrB:

Cmp bx,cx

Ja BGrAndC

Jmp CGrAandB

AGrB:

Cmp ax, cx

Ja AGrBandC

CGrAandB

Mov [bp+2], cx

Jmp return

AGrBandC:

Mov [bp+2], ax

Jmp return

BGrAAndC:

Mov [bp+2], bx

Return:

Mov ax, [bp+2]

Mov [bp+10], ax ; copy local variable to output variable

; restore registers

Pop cx

Pop bx

Pop ax

; remove local variables

Add sp, 2

Pop bp

Ret 6

**Start:**

Sub sp, 2 ; make space for output

Push 5 ; place arg 1 on stack

Push 10; place arg 2 on stack

Push 17; place arg 3 on stack

Call findMax

Pop ax; pop the output in register.