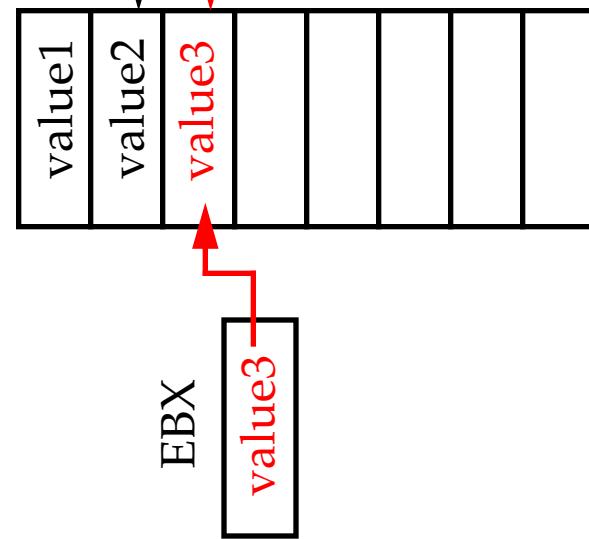


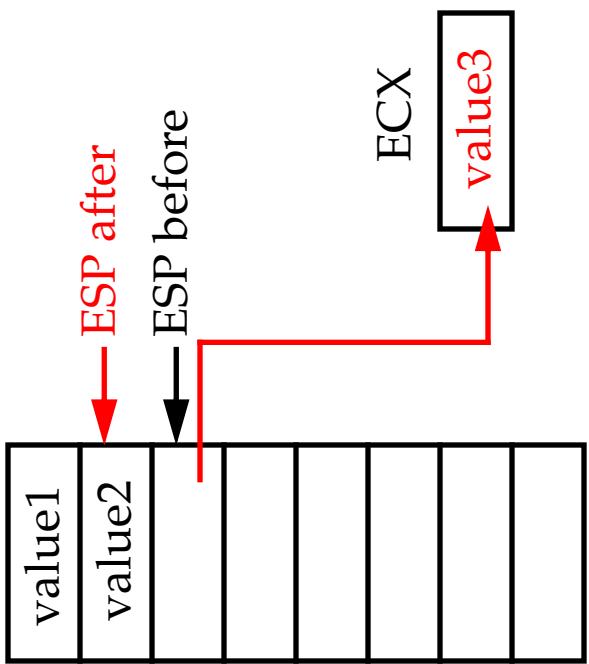
Purpose of Stack

- Memory used to pass parameters to procedures.
- Memory used for allocating space for local variables.
- Save return address in procedure calls.
- Save registers to be preserved across procedure calls.

PUSH EBX



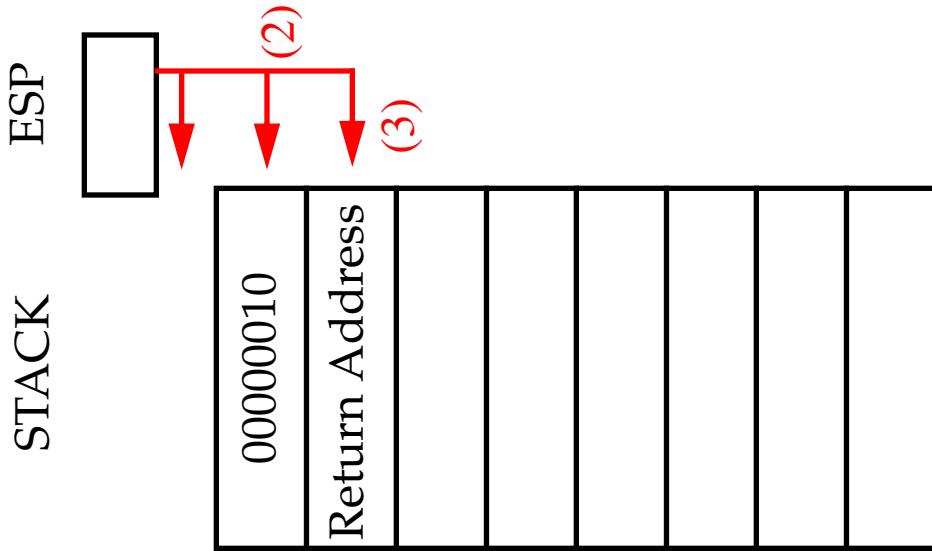
POP ECX



Passing Parameters to Procedures

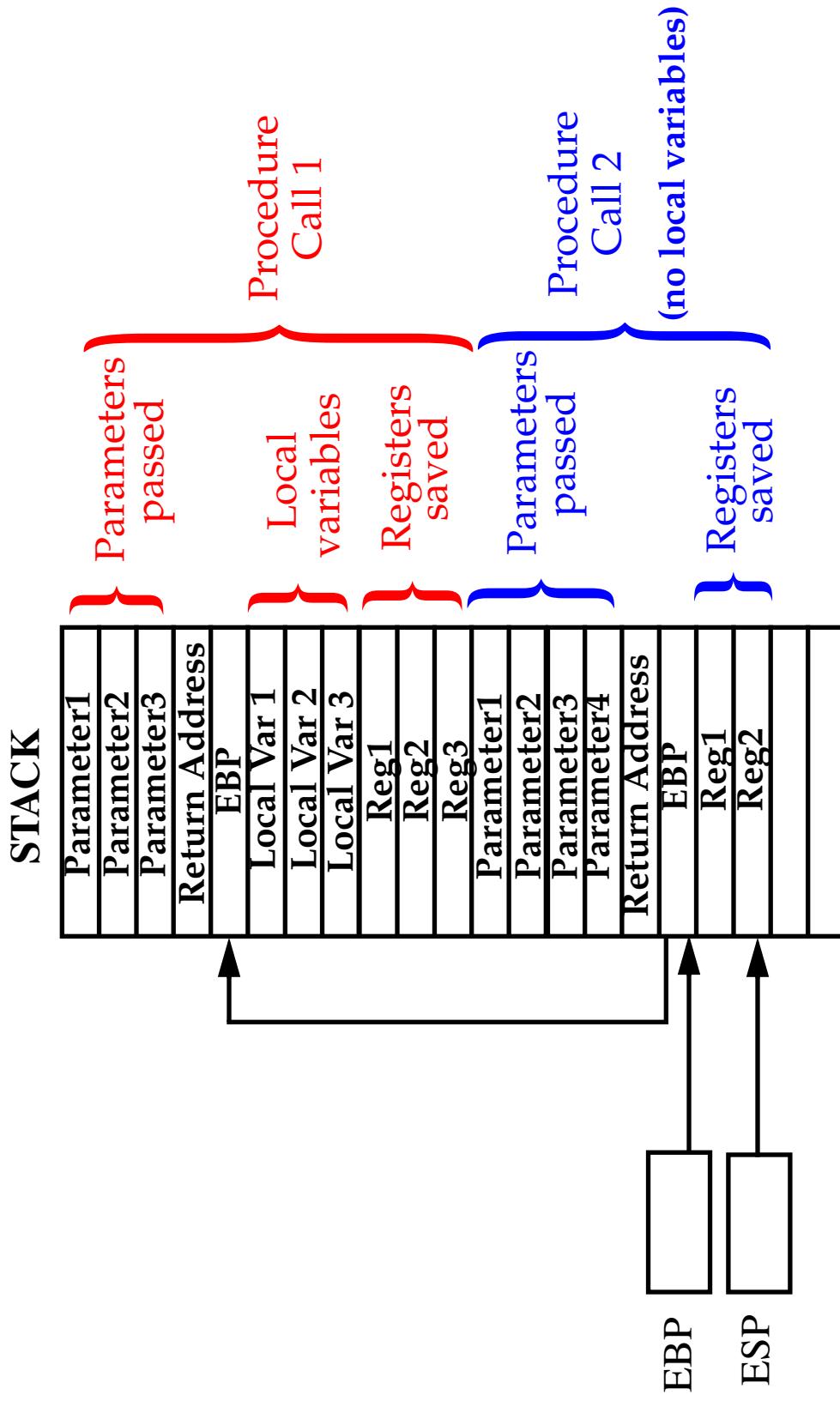
```
section .data  
input_filename_ptr : dd 0  
  
section .text  
  
main:  
    push dword input_filename_ptr (1)  
    call GetCommandLine (2)  
    add esp, 4 (3)
```

- (1) *input_filename_ptr* :
 00000010 Pointer to the filename
- (2) Push the address of the pointer to the filename
- (3) Return address pushed to the stack.
 Address of the add instruction.



Call Frames

One call frame created per procedure call



Setting up Call Frames

GetCommandLine:

Enter 0

Push_Regs ebx, ecx, edx

(1) (2)

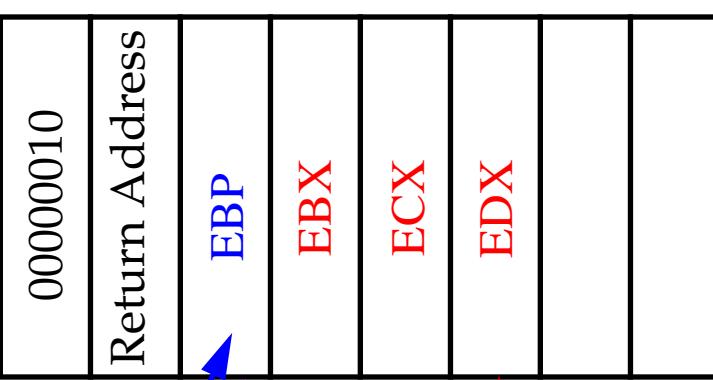
```
%macro Enter 1  
    push ebp  
    mov ebp, esp  
    sub esp, %1  
%endmacro
```

(1) Push EBP

Move ESP into EBP
i.e. EBP points to the pushed ESP

Allocate space for local variables
(none in this example)

(2) Push the registers that are to be saved
EBX, ECX and EDX in this example



Reading Arguments

```
mov ebx, [ebp + 8]
mov [ebx], dword 0
```

```
mov ecx, [ebp + 16]
```

```
cmp ecx, 2
```

```
if ne
```

```
jmp gcl_done
endif
```

Exactly 2 arguments required

Program name and input file name

ELSE ERROR!!

REGISTERS

ECX

argc

EBX

00000010

EBP

[EBX]

DATA

00000010

STACK

argc (# of arg)

EBP+16

00000010

Return Address

EBP

EBX

ECX

EDX

Reading Arguments

```

    mov ecx, [ebp + 20]
    mov ebx, [ecx]
    DEBUG
    stuff
    printf
    mov ecx, [ebp + 20]
    mov ebx, [ecx + 4]

```

REGISTERS

ECX

Pointer to args.
pointers

EBX

Pointer to
program name
Pointer to
input file name

EBP

EBX

ECX

EDX

STACK

Pointer to args.
pointers

EBP+20

argc (# of arg)

00000010

Return Address

EBP
EBX
ECX
EDX

DATA

[ECX]

Pointer to
program name

[ECX + 4]

Pointer to
input file name

program name

input file name

Get argument and Return

```
mov edx, [ebp + 8]
mov [edx], ebx
```

Pop_Regs ebx,ecx,edx

Leave
ret

