香港中文大學
The Chinese University of Hong Kong

CSCI2510 Computer Organization

Tutorial 07: MASM Subroutines

Tsun-Yu YANG

yangty@cse.cuhk.edu.hk

Outline

- Processor Stack
  - Processor Stack Review
  - How to use Processor Stack
  - Subroutine linkage & Parameter Passing in Process Stack
- Subroutine
  - Subroutine Review
  - Why Subroutine?
  - How to write subroutine code in MASM
- Example of MASM Subroutine Code

Processor Stack Review

- Modern processors usually provide native support to stacks (called **processor stack**)

- Stack is useful data structure because of the FILO (First In Last Out) feature:
  - Useful in doing subroutine linkage
  - Useful in parameter passing

- **Processor stack** is managed by 2 special registers:
  - ESP: Current Stack Pointer
  - EBP: Base Pointer for Current Stack Frame

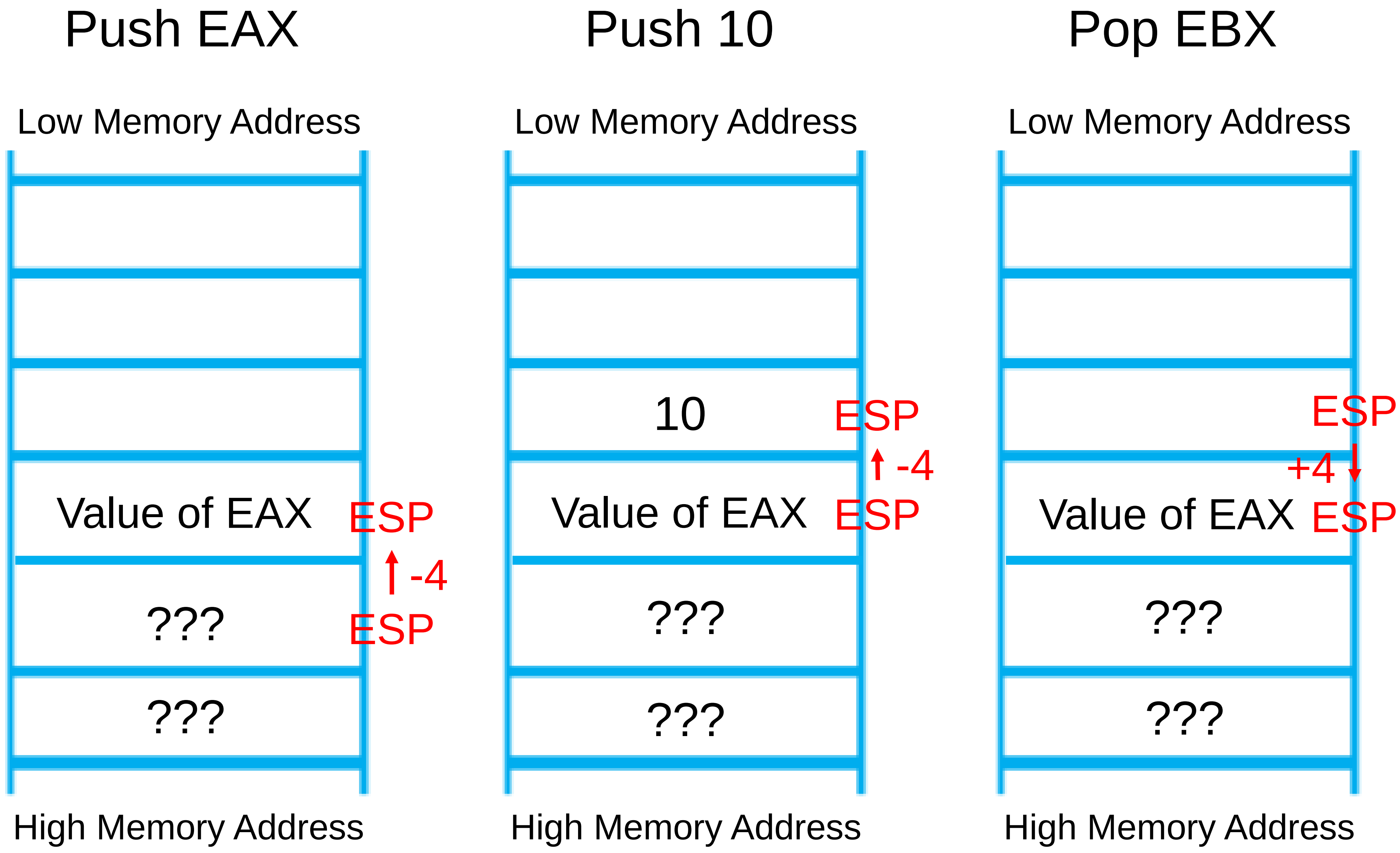
How to use Processor Stack?

- **Push** and **Pop** are 2 commands that user can directly access the processor stack.

- **Push** syntax:
  - Push reg/m16 or m32
  - Push imm32
- **Pop** syntax:
  - Pop reg/m16 or m32

- Whenever a new value is pushed (or popped) to the stack, the ESP will also be updated.
  - Please note that ESP points to the top of the process stack

Examples of Push & Pull



Push EAX

Low Memory Address

Value of EAX

ESP

-4

???

ESP

???

High Memory Address

Push 10

Low Memory Address

10

ESP

-4

Value of EAX

ESP

???

???

High Memory Address

Pop EBX

Low Memory Address

ESP

+4

Value of EAX

ESP

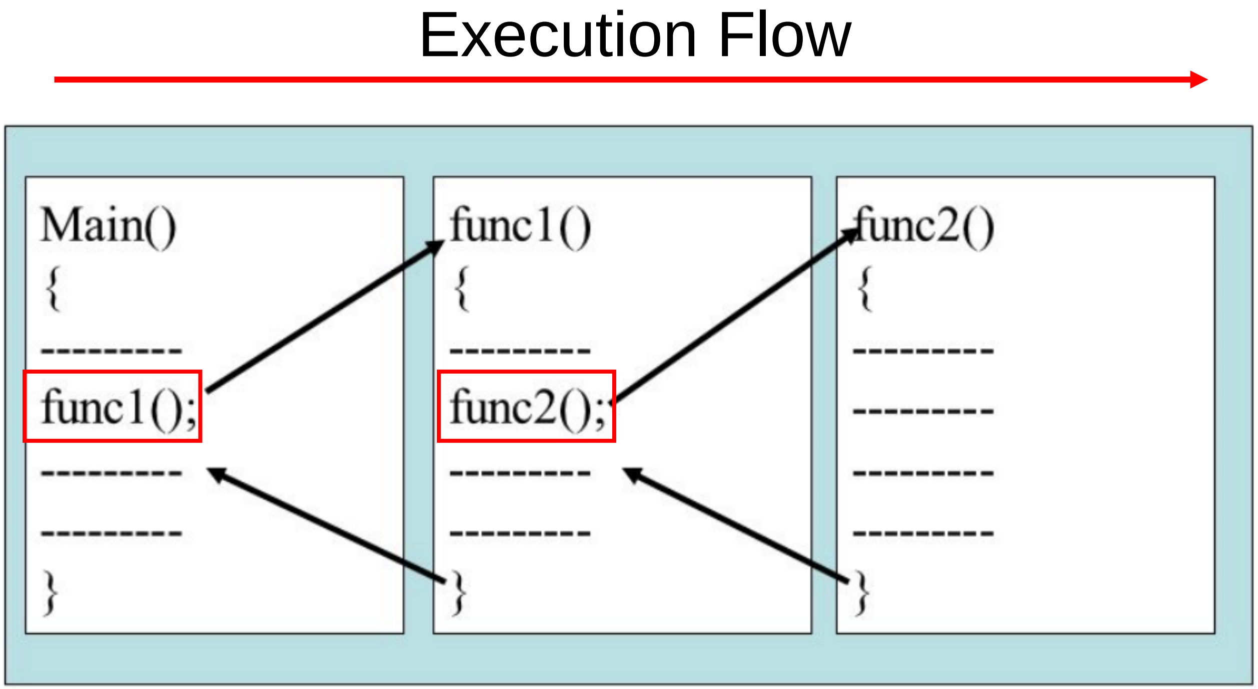
???

???

High Memory Address

Processor Stack in Subroutine Linkage

- Let’s see an example to do the Subroutine Linkage with Processor Stack

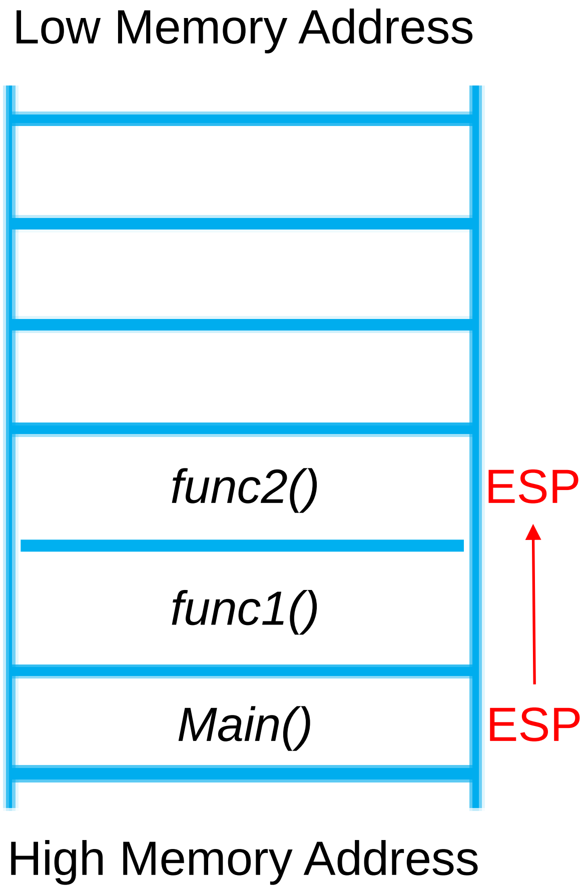


Execution Flow

```
Main()
{
---------
func1();
---------
---------
}
```

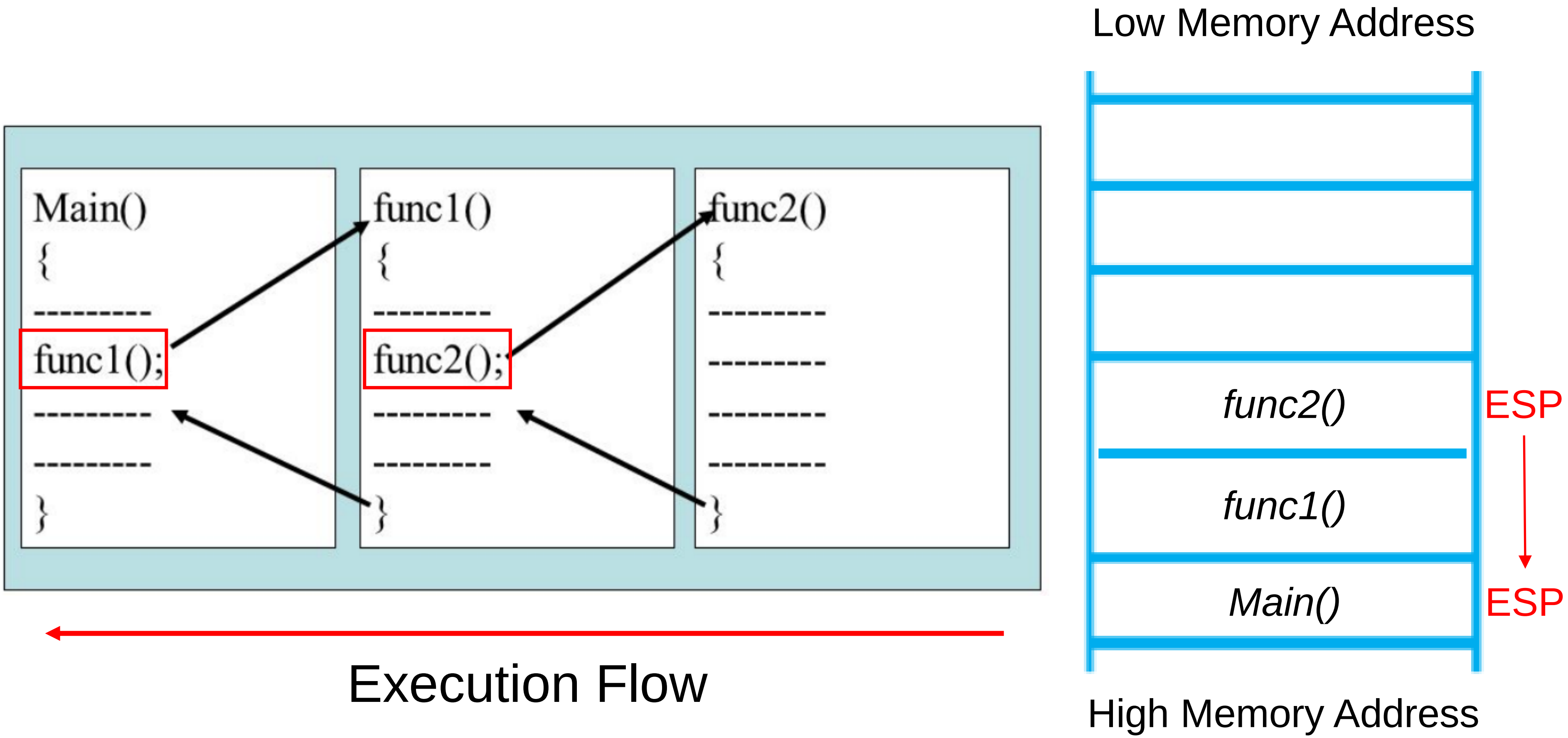
```
func1()
{
---------
func2();
---------
---------
}
```

```
func2()
{
---------
---------
---------
---------
}
```



Processor Stack in Subroutine Linkage

- Let’s see an example to do the Subroutine Linkage with Processor Stack



Processor Stack in Parameter Passing 1

- If we use registers for passing the parameters, we can only pass 4 parameters. (because EAX ~ EDX)
- We can use **processor stack** to pass more parameters

Example Pseudo Code:

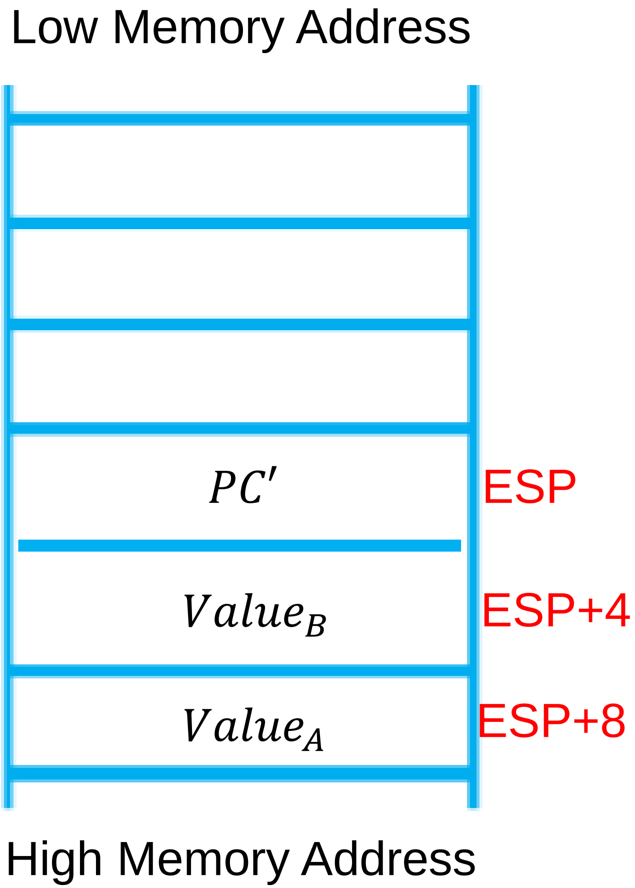
```
    Push $Value_A$ → First parameter
    Push $Value_B$ → Second parameter
    Call Func  ⟵ PC (Current Instr.)
    ……….       ⟵ PC' (Next Instr.)
```

To Extract the parameters in Func:

[ESP+8] is the first parameter

[ESP+4] is the second parameter

What if we push value to processor stack in Func?

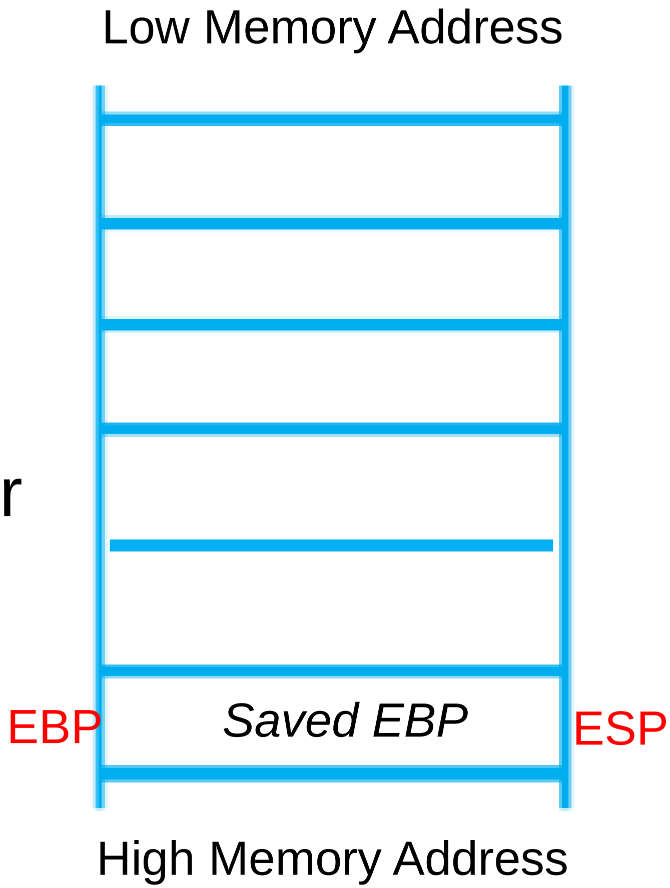


Processor Stack in Parameter Passing 2

- EBP (base pointer) register is introduced to solve the above problem.
  - EBP stores the base address

Example Pseudo Code:

```
    Push EBP
→   Mov EBP, ESP
    Push $Value_A$ → First parameter
    Push $Value_B$ → Second parameter
    Call Func  ⟵ PC (Current Instr.)
    Pop EBP    ⟵ PC’ (Next Instr.)
```

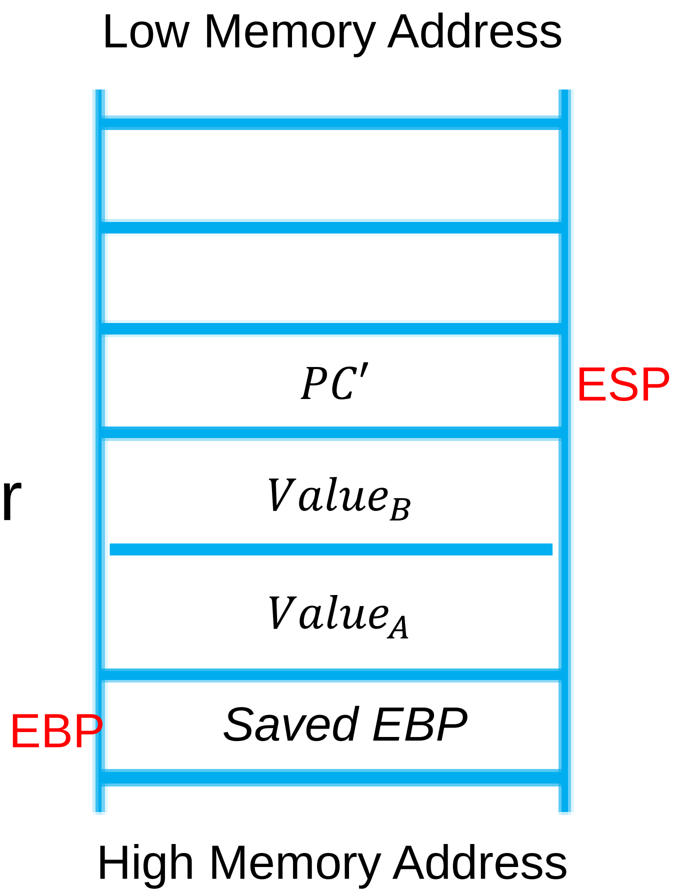


Processor Stack in Parameter Passing 2

- EBP (base pointer) register is introduced to solve the above problem.
  - EBP stores the address address

Example Pseudo Code:

```
    Push EBP
    Mov EBP, ESP
    Push $Value_A$ → First parameter
    Push $Value_B$ → Second parameter
→   Call Func  ⟵ $PC$ (Current Instr.)
    Pop EBP    ⟵ $PC'$ (Next Instr.)
```



The first parameter is always at [EBP-4] !!!
The second parameter is always at [EBP-8] !!!

Processor Stack in Parameter Passing 2

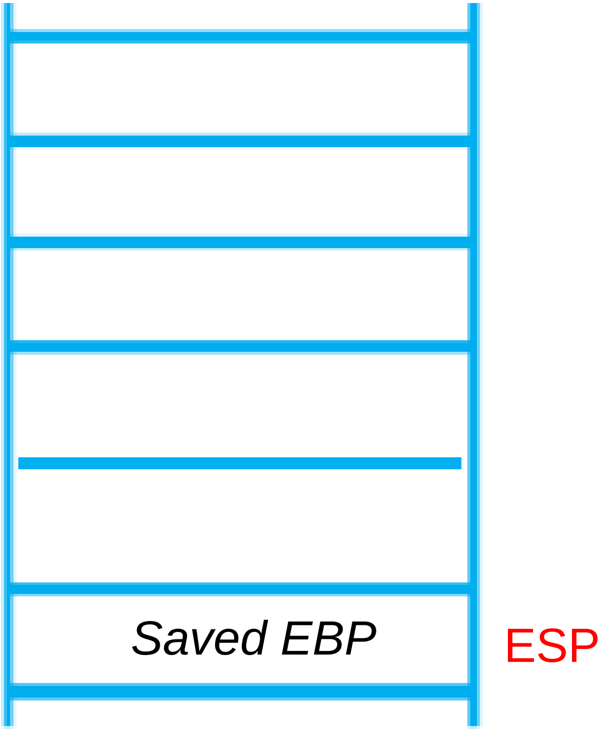
- EBP (base pointer) register is introduced to solve the above problem.
  - EBP stores the return address

Example Pseudo Code:

```
    Push EBP
    Mov EBP, ESP
    Push $Value_A$ → First parameter
    Push $Value_B$ → Second parameter
    Call Func  ⟵ PC (Current Instr.)
→   Pop EBP    ⟵ PC’ (Next Instr.)
```

The first parameter is always at [EBP-4] !!!
The second parameter is always at [EBP-8] !!!

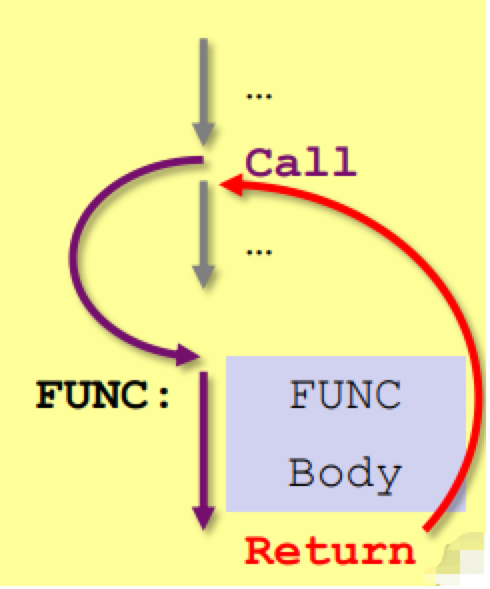
Low Memory Address



High Memory Address

Subroutine Review

- Basic concepts:
  - When a program branches to a subroutine, we say that it is **calling** the subroutine.
  - After a subroutine calling, the subroutine is said to **return** to the program that called it.
    - Immediately continuing executing after the instruction that called the subroutine.
  - Provision must be made for returning to the appropriate location.
    - the contents of the PC must be saved by the call instruction to enable correct return



Why Do We Need Subroutine?

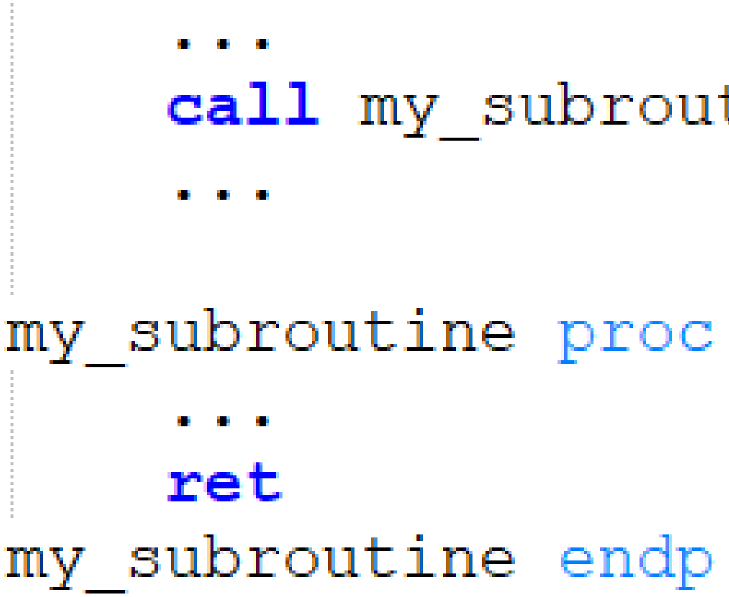
- Subroutines are the basic building blocks of programs. They are usually small and used to perform particular tasks.

- Subroutine is not needed in small size program, but it’s strongly recommended to be used in large program.

- Two advantages to use Subroutine in your program:
  - Reuse the codes to make your program clean
  - Clearly define a logical structure for your program

How to write Subroutine in MASM

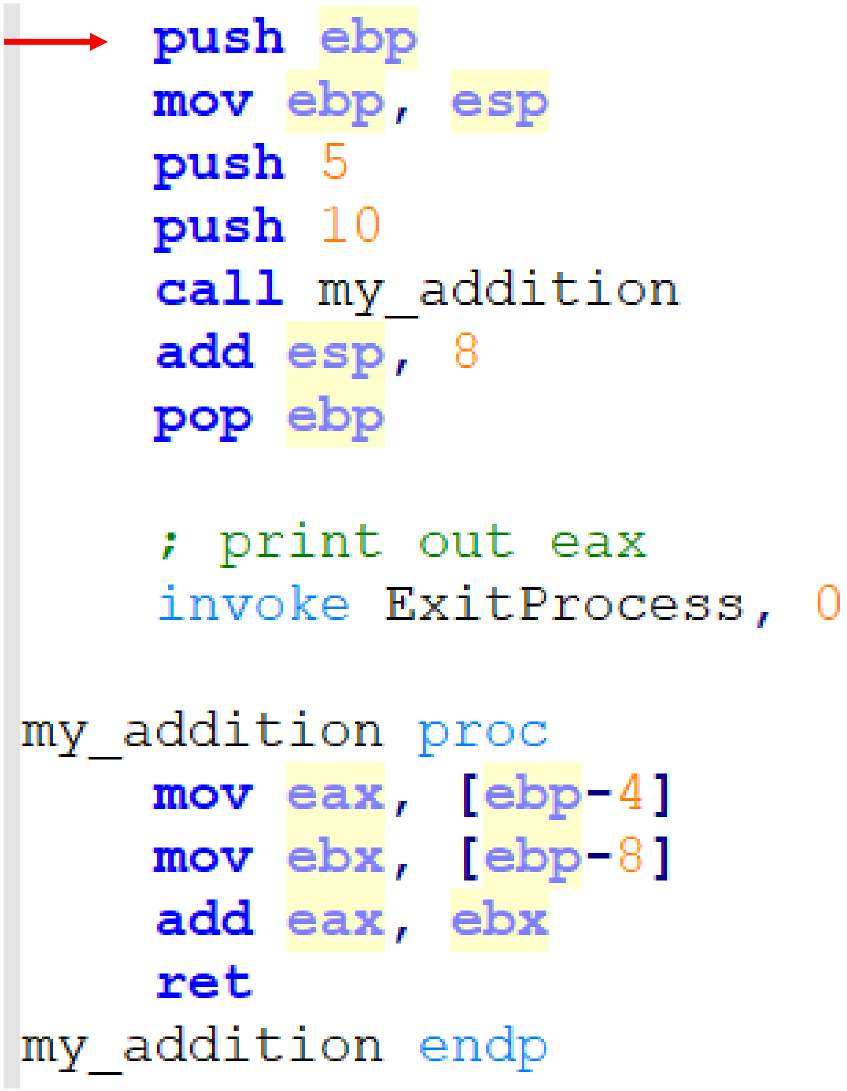
- Two instructions are used in implementing a subroutine:
  - **Call**: Push the offset of next instruction on the stack & Jump to the location of the subroutine
  - **Ret**: Pop the top of stack & Jump back to the saved address



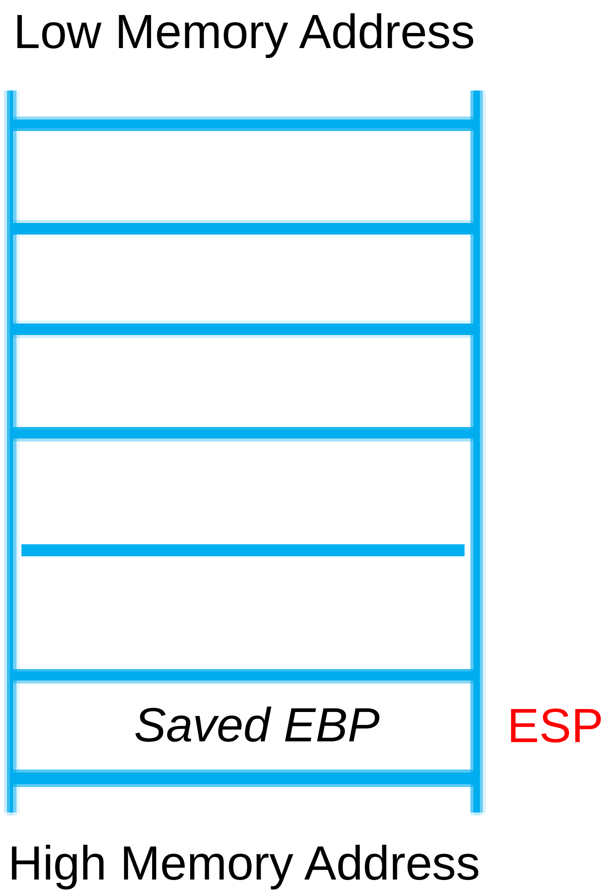
```
    ...
    call my_subroutine
    ...

my_subroutine proc
    ...
    ret
my_subroutine endp
```

Indicate the region of your subroutine

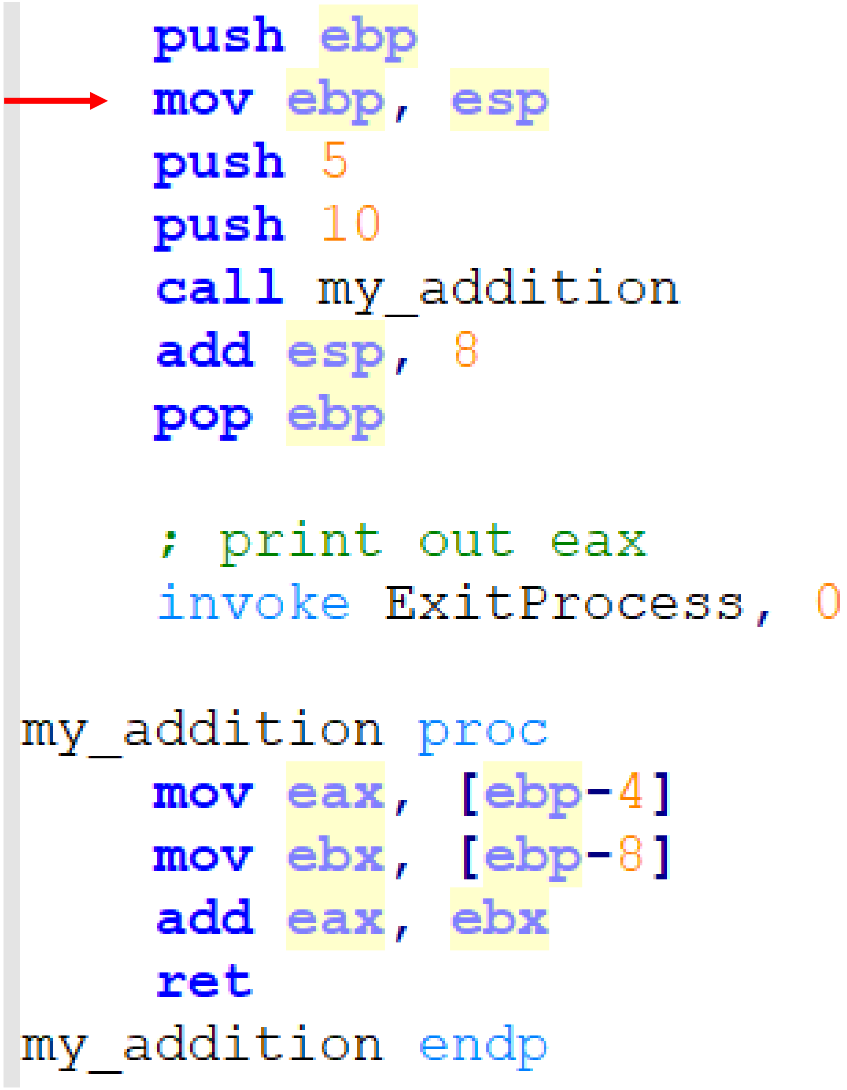
Example of MASM Subroutine Code



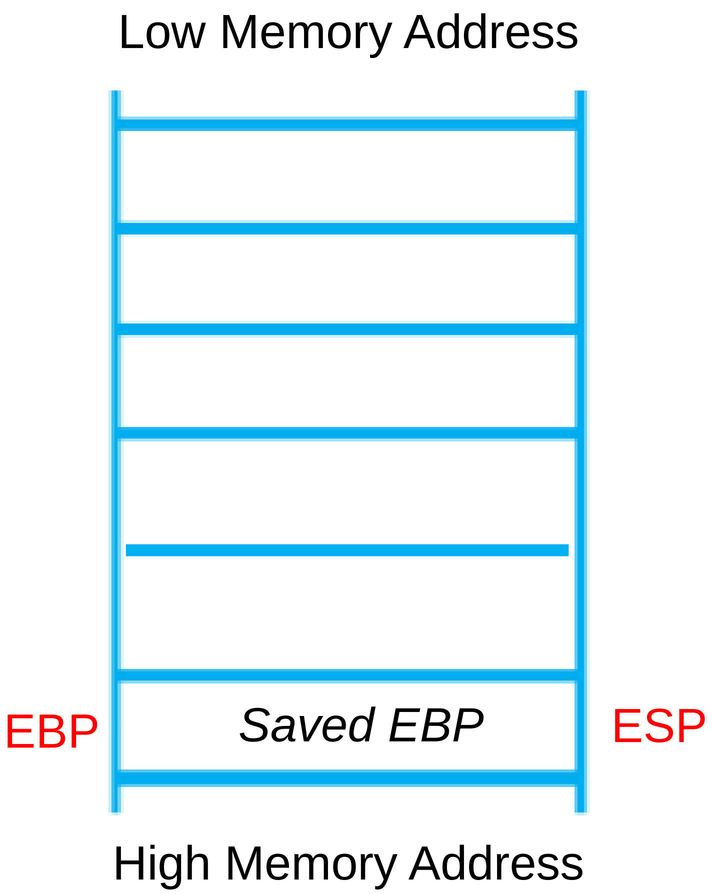
```
    push ebp
    mov ebp, esp
    push 5
    push 10
    call my_addition
    add esp, 8
    pop ebp

    ; print out eax
    invoke ExitProcess, 0

my_addition proc
    mov eax, [ebp-4]
    mov ebx, [ebp-8]
    add eax, ebx
    ret
my_addition endp
```



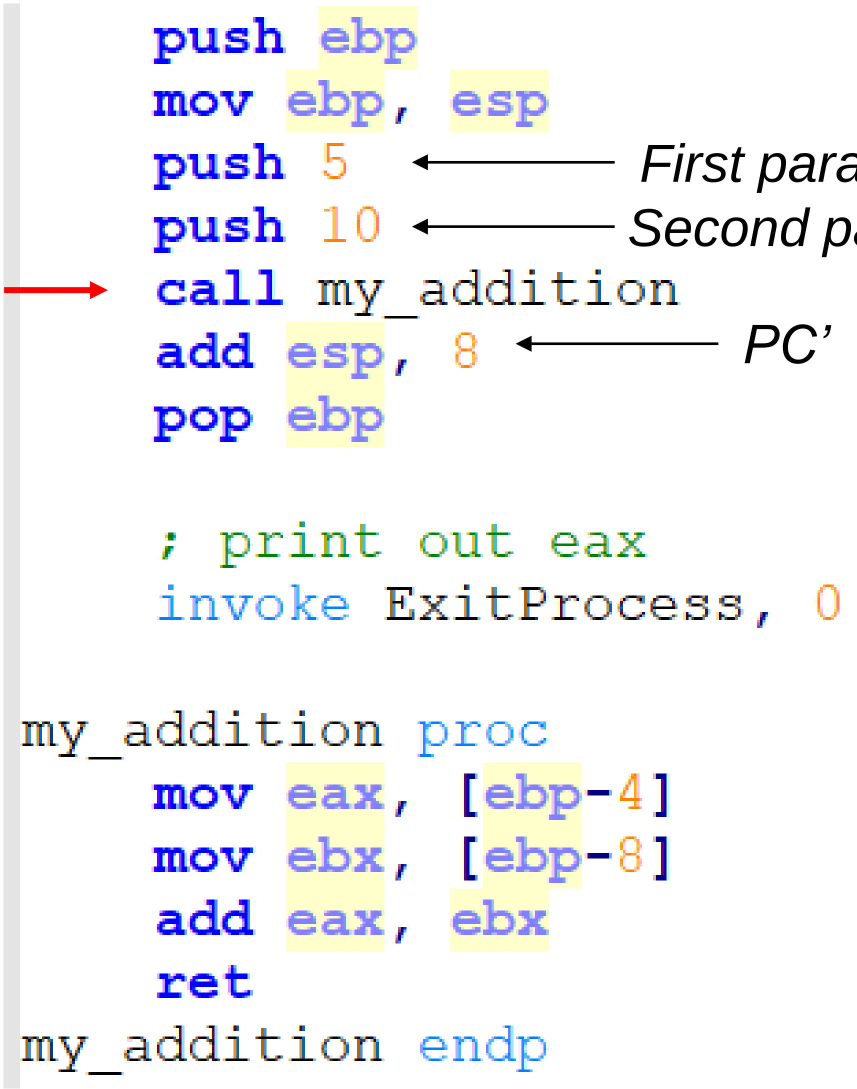
Example of MASM Subroutine Code



```
    push ebp
    mov ebp, esp
    push 5
    push 10
    call my_addition
    add esp, 8
    pop ebp

    ; print out eax
    invoke ExitProcess, 0

my_addition proc
    mov eax, [ebp-4]
    mov ebx, [ebp-8]
    add eax, ebx
    ret
my_addition endp
```



Example of MASM Subroutine Code

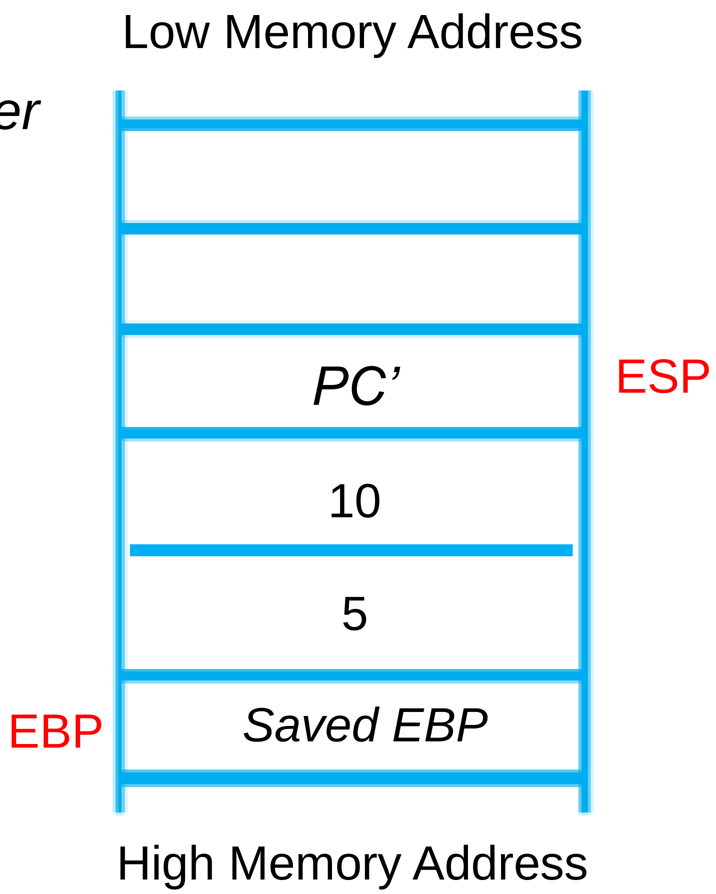


```
        push ebp
        mov ebp, esp
        push 5
        push 10
        call my_addition
        add esp, 8
        pop ebp

        ; print out eax
        invoke ExitProcess, 0

my_addition proc
        mov eax, [ebp-4]
        mov ebx, [ebp-8]
        add eax, ebx
        ret
my_addition endp
```

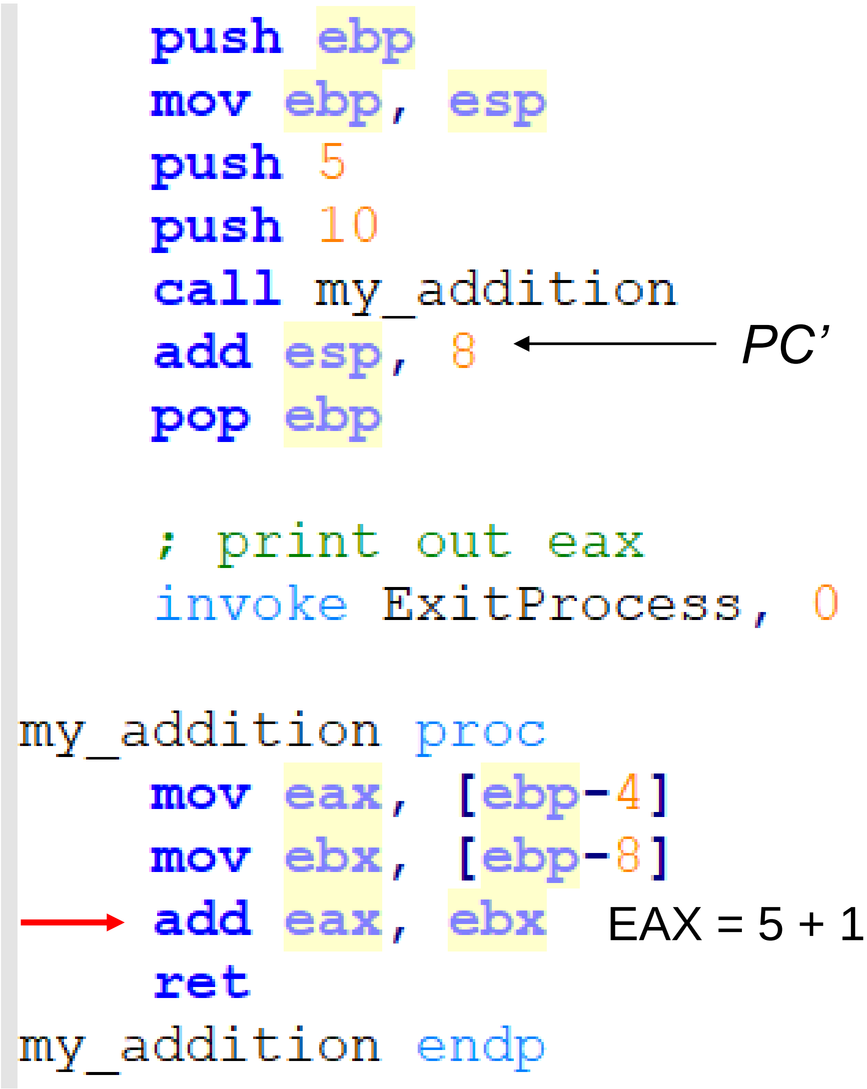
First parameter

Second parameter

PC’



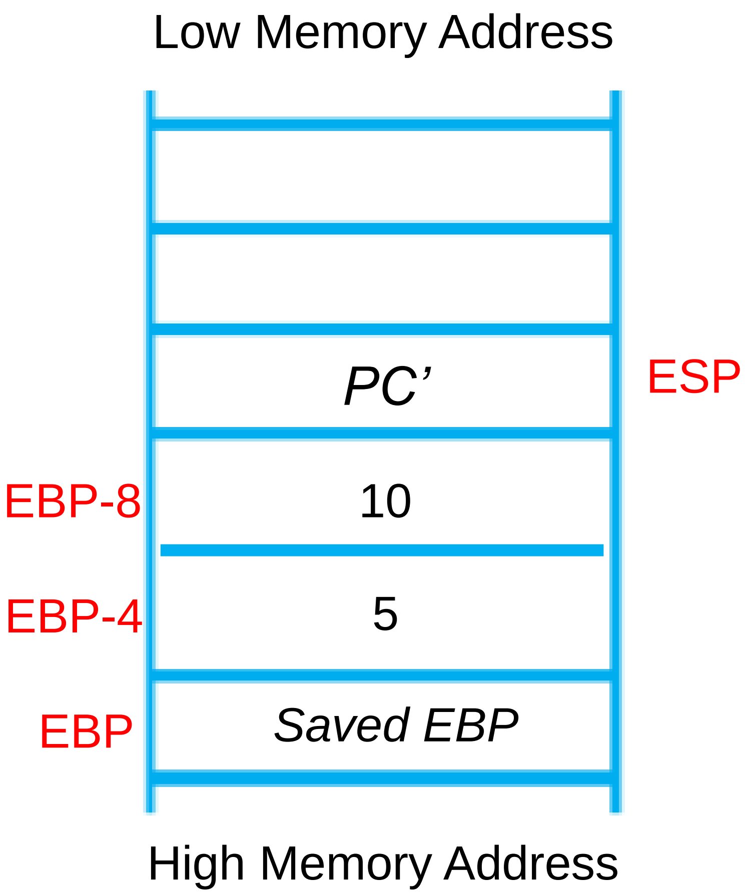
Example of MASM Subroutine Code



```
      push ebp
      mov ebp, esp
      push 5
      push 10
      call my_addition
      add esp, 8
      pop ebp

      ; print out eax
      invoke ExitProcess, 0

my_addition proc
      mov eax, [ebp-4]
      mov ebx, [ebp-8]
      add eax, ebx
      ret
my_addition endp
```

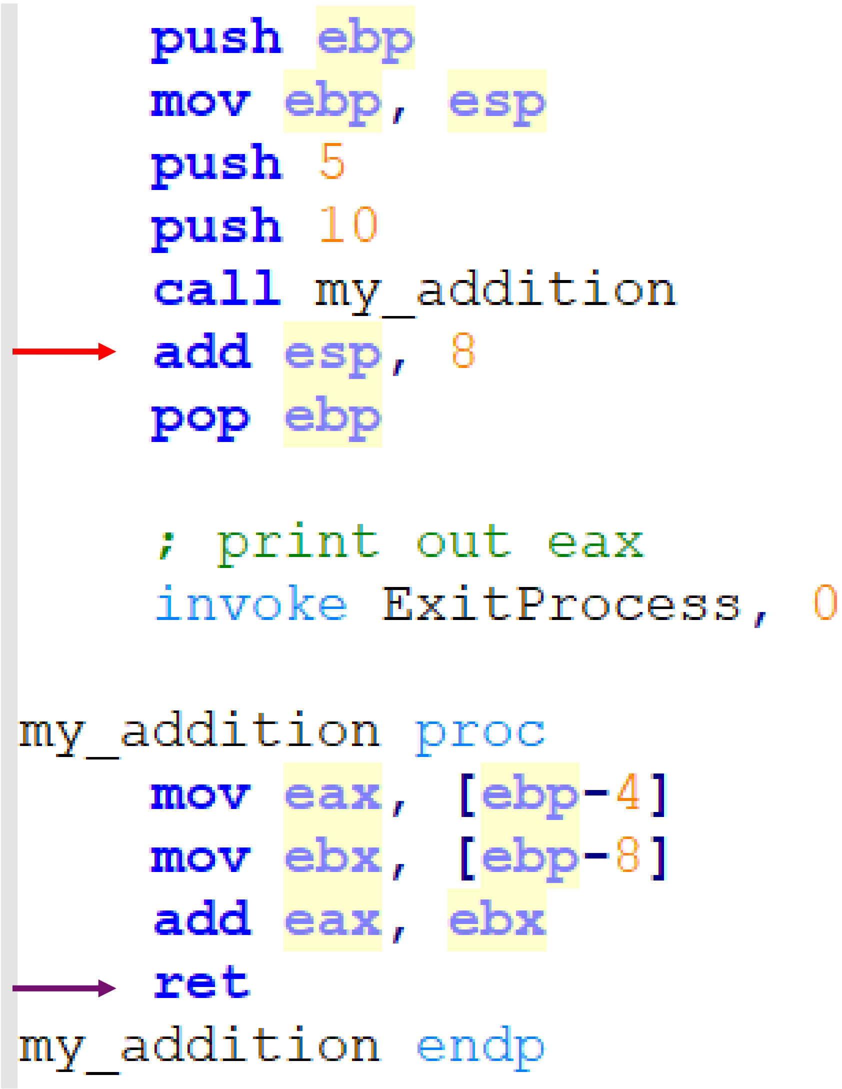
PC'

EAX = 5 + 10

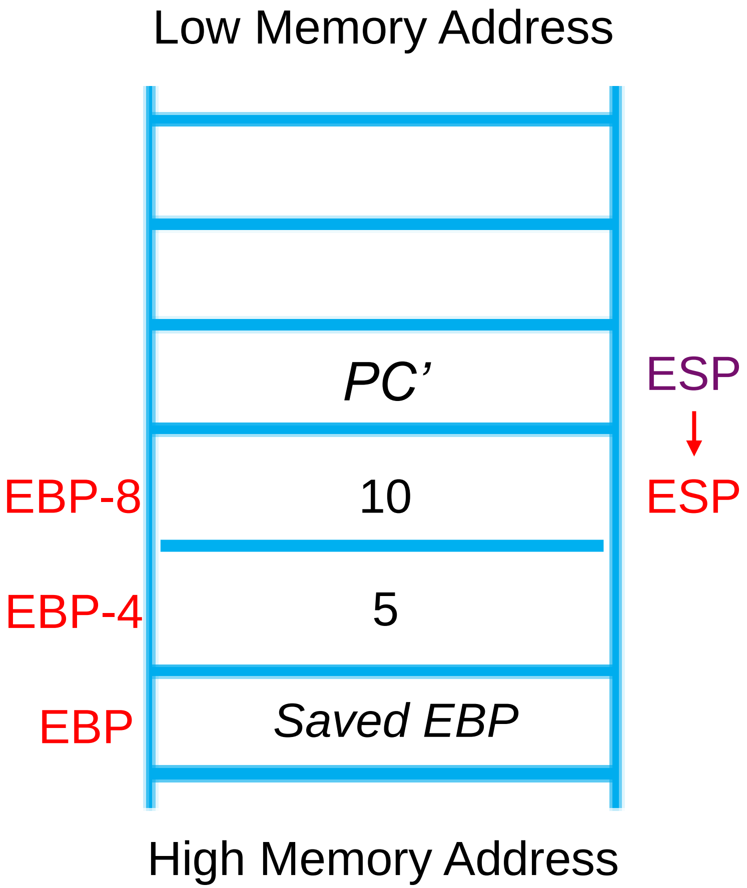


Low Memory Address	
PC'	ESP
EBP-8	10
EBP-4	5
EBP	*Saved EBP*
High Memory Address	

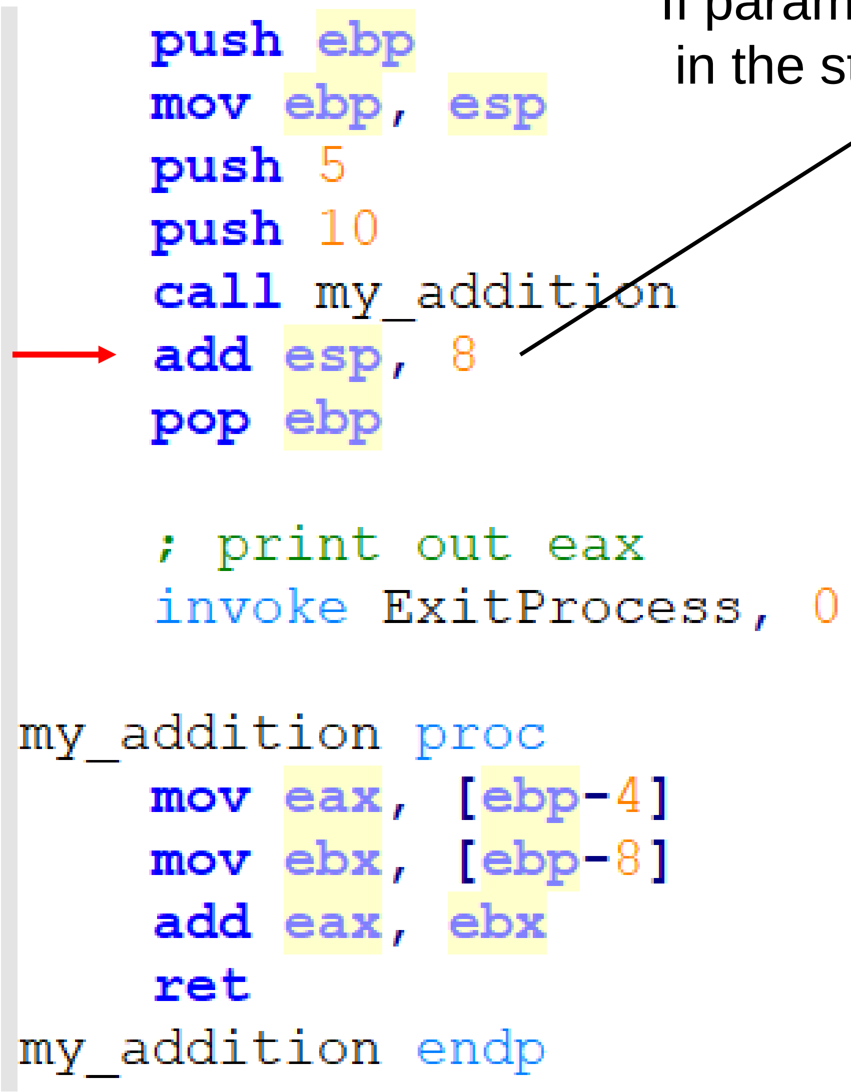
Example of MASM Subroutine Code



```
    push ebp
    mov ebp, esp
    push 5
    push 10
    call my_addition
    add esp, 8
    pop ebp

    ; print out eax
    invoke ExitProcess, 0

my_addition proc
    mov eax, [ebp-4]
    mov ebx, [ebp-8]
    add eax, ebx
    ret
my_addition endp
```

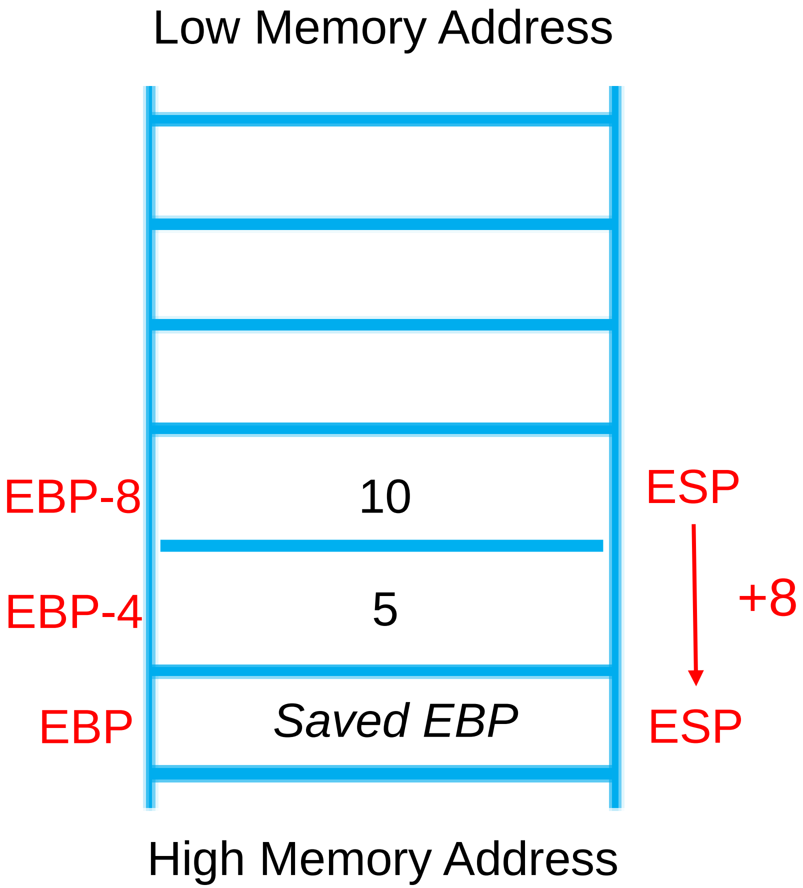


Example of MASM Subroutine Code

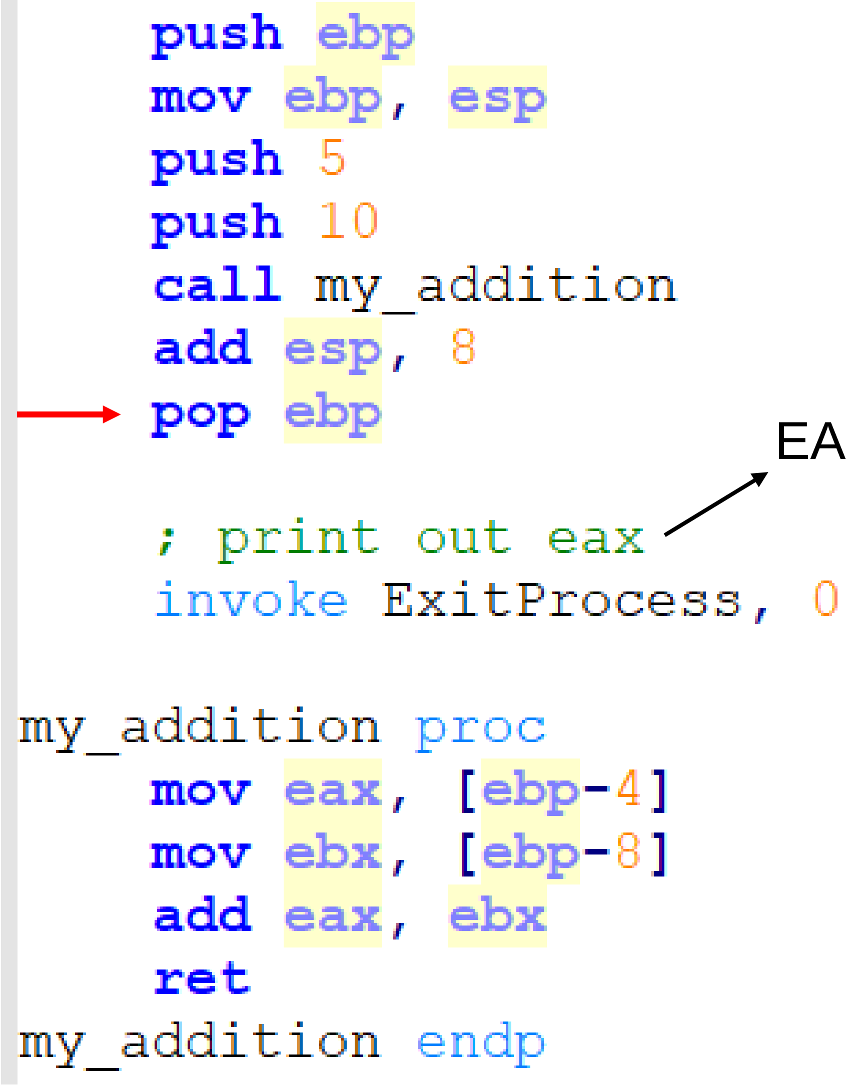


```
    push ebp
    mov ebp, esp
    push 5
    push 10
    call my_addition
    add esp, 8
    pop ebp

    ; print out eax
    invoke ExitProcess, 0

my_addition proc
    mov eax, [ebp-4]
    mov ebx, [ebp-8]
    add eax, ebx
    ret
my_addition endp
```

If parameters are still in the stack, clean it

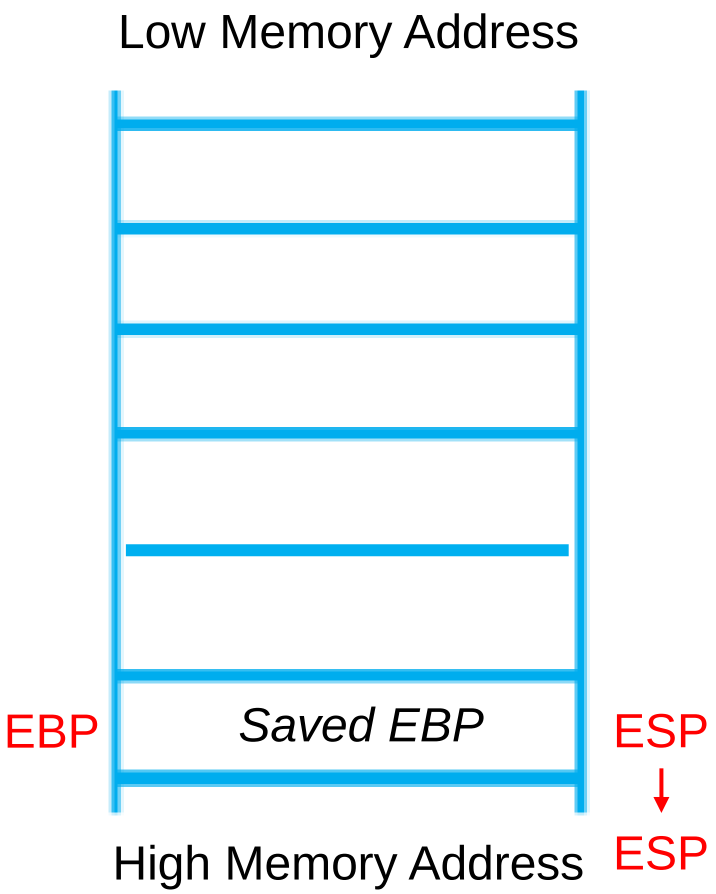


Example of MASM Subroutine Code



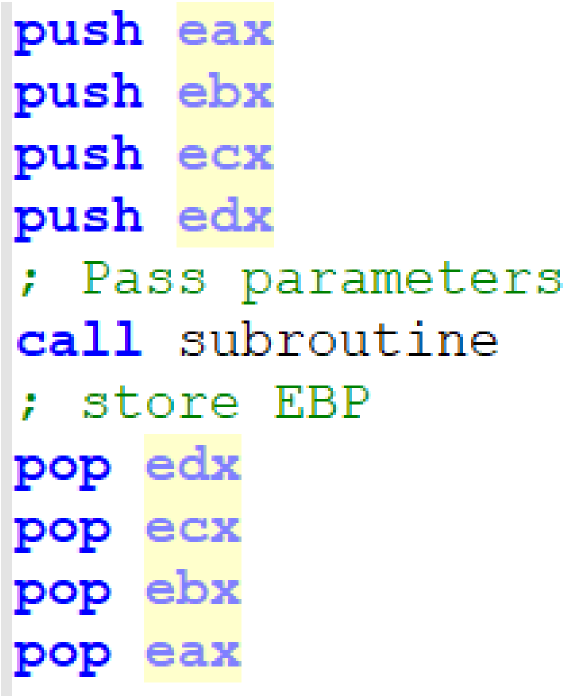
```
    push ebp
    mov ebp, esp
    push 5
    push 10
    call my_addition
    add esp, 8
    pop ebp

    ; print out eax
    invoke ExitProcess, 0

my_addition proc
    mov eax, [ebp-4]
    mov ebx, [ebp-8]
    add eax, ebx
    ret
my_addition endp
```

EAX=15



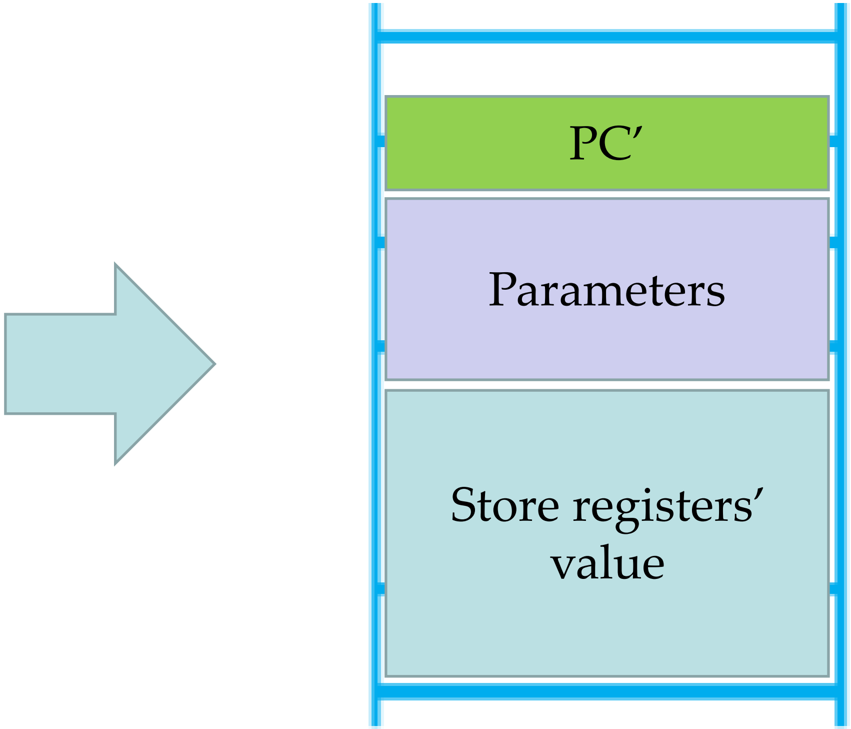
More Examples

- Processor stack can also be used for temporarily storing the registers’ value.
- If you have important data in registers before calling subroutine or instructions:



```
push eax
push ebx
push ecx
push edx
; Pass parameters
call subroutine
; store EBP
pop edx
pop ecx
pop ebx
pop eax
```

Low Memory Address



High Memory Address

Summary

- Processor Stack
- Subroutine
- Example of MASM Subroutine Code