Example Stack Instructions

- Assuming that ESP=00001000h
Example Stack Instructions

- Assuming that ESP=00001000h

push    dword 1     ; ESP = 00000FFCh

Increasing addresses:
Example Stack Instructions

- Assuming that ESP=00001000h

  push    dword  1    ; ESP = 00000FFCh
  push    dword  2    ; ESP = 00000FF8h
Example Stack Instructions

- Assuming that ESP=00001000h

  push  dword  1  ; ESP = 00000FFCh
  push  dword  2  ; ESP = 00000FF8h
  push  dword  3  ; ESP = 00000FF4h
Example Stack Instructions

- Assuming that ESP=00001000h

  ```
  push    dword  1    ; ESP = 00000FFCh
  push    dword  2    ; ESP = 00000FF8h
  push    dword  3    ; ESP = 00000FF4h
  ```

  ```
  pop     eax        ; EAX = 3
  pop     ebx        ; EBX = 2
  pop     ecx        ; ECX = 1
  ```
A Full Example

L      dd  42, 43, 44, 45, 56
...
push   dword L
call   func
add    esp, 4
call   print_int
...
func:
push   ebp
mov    ebp, esp
push   [ebp+8]
push   8
call   reference
add    esp, 8
add    eax, 10
pop    ebp
ret

reference:
push   ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
A Full Example

L dd 42, 43, 44, 45, 56

... push    dword L
call    func
add    esp, 4
call    print_int
...

func:
    push    ebp
    mov     ebp, esp
    push    [ebp+8]
    push    8
    call    reference
    add     esp, 8
    add     eax, 10
    pop     ebp
    ret

reference:
    push    ebp
    mov     ebp, esp
    mov     eax, [ebp+12]
    mov     eax, [ebp+8]
    mov     eax, [eax]
    pop     ebp
    ret
A Full Example

```
L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
```
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
A Full Example

L dd 42, 43, 44, 45, 56
...
push dword L
call func
add esp, 4
call print_int
...
func:
push ebp
mov ebp, esp
push [ebp+8]
push 8
call reference
add esp, 8
add eax, 10
pop ebp
ret
reference:
push ebp
mov ebp, esp
mov eax, [ebp+12]
mov eax, [ebp+8]
mov eax, [eax]
pop ebp
ret
A Full Example

```
L     dd  42, 43, 44, 45, 56
...
push  dword L
call  func
add   esp, 4
call  print_int
...
func:
push  ebp
mov   ebp, esp
push   [ebp+8]
push  8
call  reference
add   esp, 8
add   eax, 10
pop   eax
ret

reference:
push  ebp
mov   ebp, esp
mov   eax, [ebp+12]
add   eax, [ebp+8]
mov   eax, [eax]
pop   ebp
ret```

XXX

L
ret @
saved ebp
L
EBP
ESP
A Full Example

L dd 42, 43, 44, 45, 56
...
push dword L
call func
add esp, 4
call print_int
...
func:
push ebp
mov ebp, esp
push [ebp+8]
push 8
call reference
add esp, 8
add eax, 10
pop ebp
ret
reference:
push ebp
mov ebp, esp
mov eax, [ebp+12]
mov eax, [ebp+8]
mov eax, [eax]
pop ebp
ret
A Full Example

L dd 42, 43, 44, 45, 56
... 
push dword L 
call func  
add esp, 4  
call print_int
...

func:
    push ebp
    mov ebp, esp
    push [ebp+8]
    push 8
    call reference
    add esp, 8
    add eax, 10
    pop eax
    ret

reference:
    push ebp
    mov ebp, esp
    mov eax, [ebp+12]
    mov eax, [ebp+8]
    mov eax, [eax]
    pop ebp
    ret
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
A Full Example

```
L dd  42, 43, 44, 45, 56
...
push   dword L
call   func
add    esp, 4
call   print_int
...
func:
push   ebp
mov    ebp, esp
push   [ebp+8]
push   8
call   reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push   ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
```
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret

EAX = L
A Full Example

L dd 42, 43, 44, 45, 56
...
push dword L
call func
add esp, 4
call print_int
...
func:
push ebp
mov ebp, esp
push [ebp+8]
push 8
call reference
add esp, 8
add eax, 10
pop ebp
ret
reference:
push ebp
mov ebp, esp
mov eax, [ebp+12]
add eax, [ebp+8]
mov eax, [eax]
pop ebp
ret

EAX = L + 8
A Full Example

L dd 42, 43, 44, 45, 56

... push dword L call func add esp, 4 call print_int ...

func:
push ebp
mov ebp, esp
push [ebp+8]
push 8
call reference
add esp, 8
add eax, 10
pop ebp
ret

reference:
push ebp
mov ebp, esp
mov eax, [ebp+12]
add eax, [ebp+8]
mov eax, [eax]
pop ebp
ret

EAX = [L + 8] = 44
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...

func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret

reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret

EAX = 44

XXX
L
ret @
saved ebp
8
ret @

EBP
ESP
A Full Example

```
L    dd  42, 43, 44, 45, 56
...
push   dword L
call   func
add    esp, 4
call   print_int
...
func:
push   ebp
mov    ebp, esp
push    [ebp+8]
push    8
call   reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push   ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
```

EBP

ESP

EAX = 44
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret

EBP
ESP
saved ebp
XXX
L
ret @

EAX = 44
A Full Example

L  dd  42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
    push    ebp
    mov    ebp, esp
    push    [ebp+8]
    push    8
    call    reference
    add    esp, 8
    add    eax, 10
    pop    ebp
    ret

reference:
    push    ebp
    mov    ebp, esp
    mov    eax, [ebp+12]
    add    eax, [ebp+8]
    mov    eax, [eax]
    pop    ebp
    ret

EBP  ESP  saved ebp

EAX  = 44 + 10 = 54
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add     esp, 4
call    print_int
...
func:
push    ebp
mov     ebp, esp
push    [ebp+8]
push    8
call    reference
add     esp, 8
add     eax, 10
pop     ebp
ret

reference:
push    ebp
mov     ebp, esp
mov     eax, [ebp+12]
add     eax, [ebp+8]
mov     eax, [eax]
pop     ebp
ret

ESP  ret @

EAX  = 54
A Full Example

L dd 42, 43, 44, 45, 56
...
push    dword L
call    func
add    esp, 4
call    print_int
...
func:
push    ebp
mov    ebp, esp
push    [ebp+8]
push    8
call    reference
add    esp, 8
add    eax, 10
pop    ebp
ret
reference:
push    ebp
mov    ebp, esp
mov    eax, [ebp+12]
add    eax, [ebp+8]
mov    eax, [eax]
pop    ebp
ret
A Full Example

L dd 42, 43, 44, 45, 56
...
push   dword L
call   func
add    esp, 4
call   print_int
...
func:
    push   ebp
    mov    ebp, esp
    push   [ebp+8]
    push   8
    call   reference
    add    esp, 8
    add    eax, 10
    pop    ebp
    ret
reference:
    push   ebp
    mov    ebp, esp
    mov    eax, [ebp+12]
    add    eax, [ebp+8]
    mov    eax, [eax]
    pop    ebp
    ret

ESP → XXXX

EAX = 54
A Full Example

L          dd  42, 43, 44, 45, 56
...
push      dword L
call      func
add       esp, 4
call      print_int
...
func:
push      ebp
mov       ebp, esp
push      [ebp+8]
push      8
call      reference
add       esp, 8
add       eax, 10
pop        ebp
ret
reference:
push      ebp
mov       ebp, esp
mov       eax, [ebp+12]
add       eax, [ebp+8]
mov       eax, [eax]
pop        ebp
ret
prints “54”
Practice

What things are wrong with the following program?

```
push    ebx
push    30
call    func
add     esp, 4
call    print_int
call    print_nl
.
.
func:   push    ebp
        mov     ebp, esp
        mov     eax, [ebp+8]
        add     eax, [ebp+4]
        ret
```
Practice (Solution)

What 5 things are wrong with the following program?

```assembly
push ebx
push dword 30
call func
add esp, 8
call print_int
call print_nl

func:
push ebp
mov ebp, esp
mov eax, [ebp+12]
add eax, [ebp+8]
pop ebp
ret
```
Practice

What does the stack look like?

```assembly
push    ebx
push    dword 30
call    func
        <----------------------------- HERE?
add     esp, 8
call    print_int
call    print_nl
...

func:   push    ebp
mov     ebp, esp
        <----------------------------- HERE?
mov     eax, [ebp+12]
add     eax, [ebp+8]
pop     ebp
ret
```
Practice (Solution)

What does the stack look like?

```assembly
push ebx
push dword 30
call func
<---------------------------
add esp, 8
call print_int
call print_nl

func: push ebp
mov ebp, esp
<---------------------------
mov eax, [ebp+12]
add eax, [ebp+8]
pop ebp
ret
```

```
xxxxxx
EBX
30

xxxxxx
EBX
30
Return @
EBP
```
Local Variables Example

- Inside the body of the subprogram, parameters are referenced as:
  - [EBP+8]: 1st parameter
  - [EBP+12]: 2nd parameter

- Inside the body of the subprogram, local variables are referenced as:
  - [EBP-4]: 1st local variable
  - [EBP-8]: 2nd local variable
  - [EBP-12]: 3rd local variable

Very important you have this picture in mind; you should be able to redraw it.
A Full Example

- Let’s write the assembly code equivalent to the following C/Java function

```c
int f(int num) { // computes Fibonacci numbers
    int x, sum;
    if (num == 0) return 0;
    if (num == 1) return 1;
    x = f(num-1);
    sum = x + f(num-2);
    return sum;
}
```

- Let’s write a “straight” translation, without optimizing variables away, just for demonstration purposes
A Full Example (main program)

%include "asm_io.inc"

segment .data
    msg1    db   "Enter n: ", 0
    msg2    db   "The result is: ", 0

    ; declaration of asm_main and setup

    mov     eax, msg1    ; eax = address of msg1
    call    print_string ; print msg1
    call    read_int    ; get an integer from the keyboard (in EAX)
    push    eax          ; put the integer on the stack (parameter #1)
    call    f            ; call f
    pop     ebx          ; remove the parameter from the stack
    mov     ebx, eax     ; save the value returned by f
    mov     eax, msg2    ; eax = address of msg2
    call    print_string ; print msg2
    mov     eax, ebx     ; eax = sum
    call    print_int    ; print the sum
    call    print_nl     ; print a new line

    ; clean up
A Full Example (function f)

; FUNCTION: f
; Takes one parameter: an integer
; eax = return value

segment .text
f:      enter  8,0          ; num in [ebp+8]
        ; local var x in [ebp-4],
        ; local var sum in [ebp-8]
push    ebx        ; save ebx
push    ecx        ; save ecx
push    edx        ; save edx

mov    eax, [ebp+8]     ; eax = num
sub     eax, 2               ; eax -= 2
jns     next     ; if not <0, goto next
add     eax, 2               ; eax += 2
jmp     end

next:
mov     eax, [ebp+8]    ; eax = num
add     eax, -1              ; eax -= 1
push    eax           ; put (num -1) on stack
call     f                ; call f (recursively)
add     esp, 4        ; remove (num-1) from stack
mov     [ebp-4], eax    ; put the returned value in x
mov     eax, [ebp+8]     ; eax = num
add     eax, -2        ; eax -= 2
push   eax            ; put (num -2) on stack
call     f               ; call f (recursively),
                      ; the return value is in eax
add     esp, 4      ; remove (num-1) from stack
add     eax, [ebp-4]    ; eax += x

mov     eax, [ebp+8]     ; eax = num
sub     eax, 2               ; eax -= 2
jmp     end

end:

pop     edx            ; restore ebx
pop     ecx           ; restore ecx
pop     ebx           ; restore edx
leave                   ; clean up the stack
ret                      ; return