Recap Relative encoding

08048357 : 72 e7 jb 08048340

08048359 :

- 00000019

(1) 10 01 11

↓

-25₁₀

↓

-19₁₆

```
  1 1 1
  08048359
+ ffffffe7
  ---------
  00000340
```

804 8200 : 02 00 00 00 je

804 8205 :

804 8207

Little Endian (Least significant byte comes first)

0x 00 00 00 02

02 87 92 00

↓

0x 00 92 87 02

③ 080482bf : eq eo ff ff ff jmp ——————

080482c4 : — — — 080482a4

+ 0x ff ff ff e0

a 4

3.7 Procedures

Transfer of control (and data) from the caller to the callee and back.

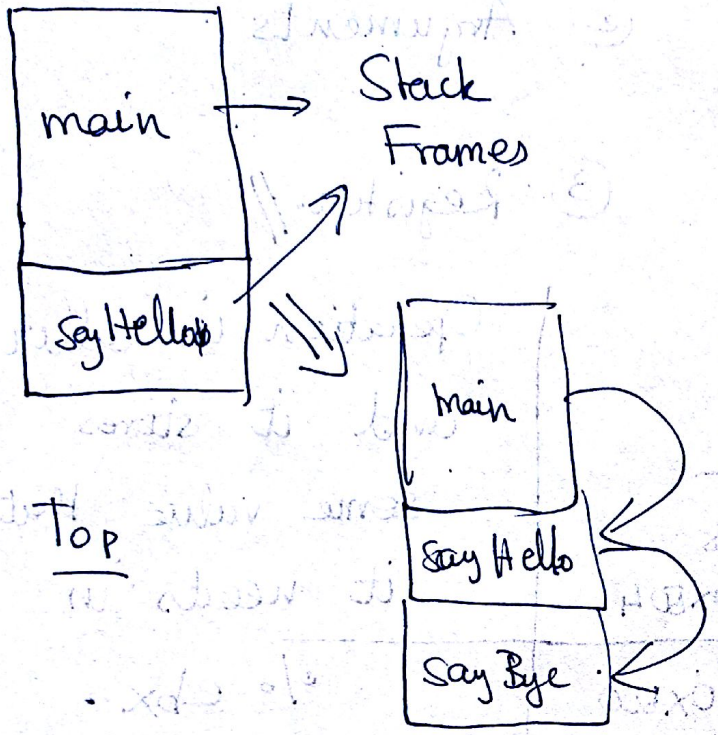
We use stacks for this

→ To pass arguments
→ To store return address & values
→ To save registers for later use.
→ For local storage

} all data.

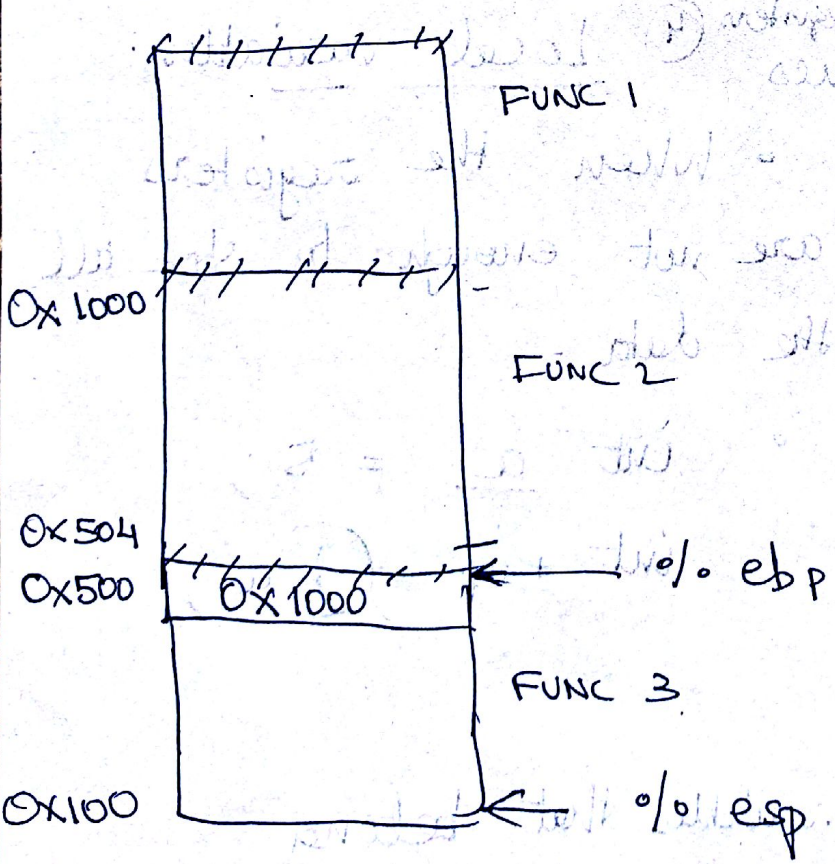
Stacks

Bottom



Text Area

```
<main> :
    movl . . —
    incl . . .
    call ———
<say_Hello> :
    ———
    ———
<say_Bye>
    ———
```



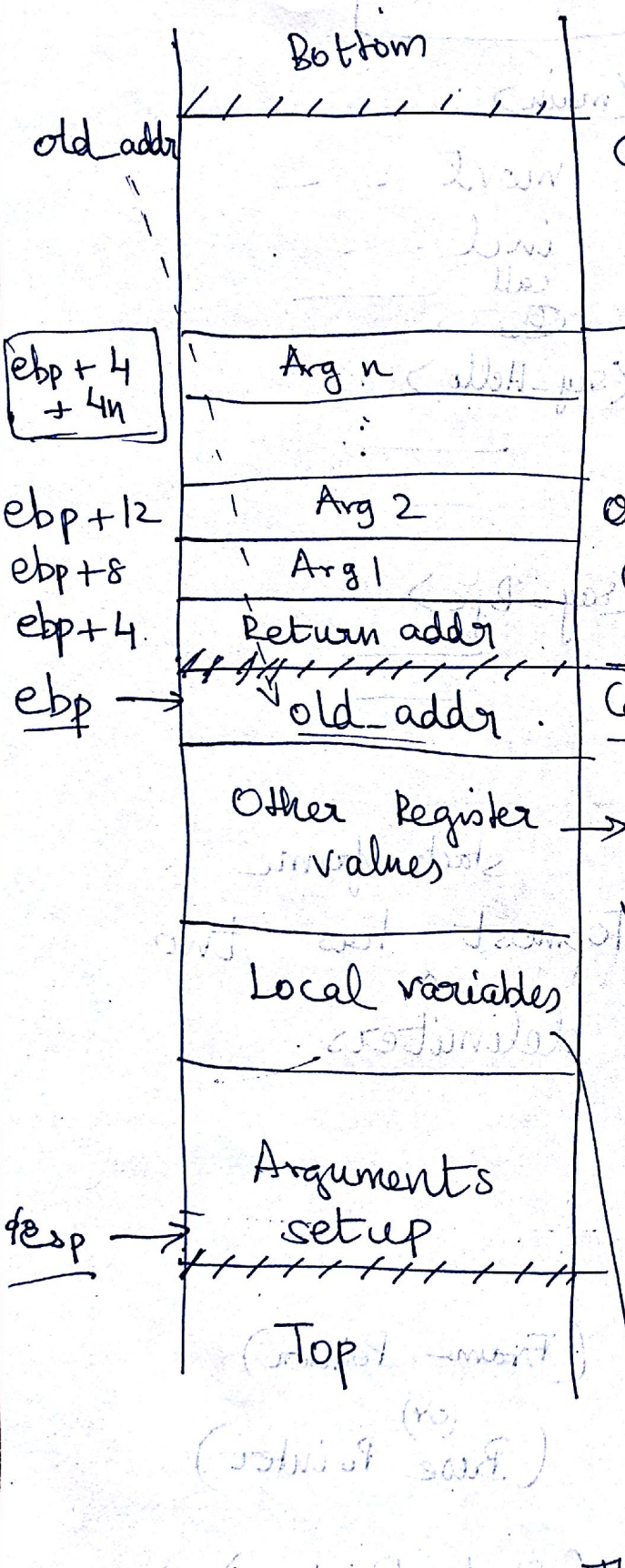
stack frame

Topmost has two delimiters.

(Frame Pointer)
(or)
(Base Pointer).

(Stack Pointer)

%ebp 0x500 %esp 0x100



① Return Address

② Arguments

③ Registers //

Operation in caller and it stores some value that it needs in % ebx.

④ Local variables.

- When the registers are not enough to store all the data.
- int a = 5;
- int *ap = &a;

The variables that belong to the callee.

Transfer Control

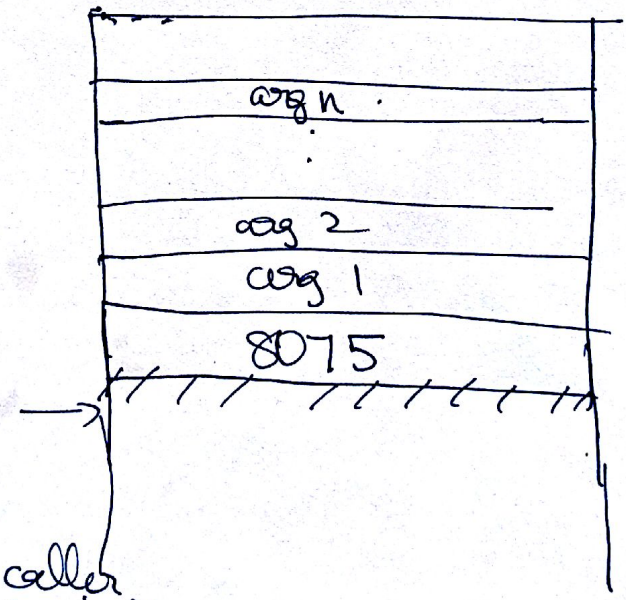
Call

Syntax → call Label

→ call *operand.

It jumps to the target specified and it continues execution there.

Also pushes the return address onto the stack before jumping.



<callee>
:
;
ret

caller <main> :
8071 : call <callee>
8075 : add1
8078 : mov1
!

Return

It pops the return address off the stack and it jumps to that location.

Syntax → ret

Before we do a return, make sure that esp points to that location.
