Last Class

→ Jumps

→ If / For

→ Encoded → Relative

→ Absolute.

① 8048286 : 74 05 je 8048296

8048288 :

(+) 05 0000 0101

② 8048357 72 e7 jb 8048340

8048359

(+) -19

1110 0111

= -25 10

= -19 16

16 | 25

1 - 9 ↑

③ 8048200 : 74 02 00 00 00 je 8048207

8048205 :

0x 00 00 00 02 → 2

IA32 uses little endian (Little end comes first)

④ 80482bf : e9 e0 ff ff ff jmp

80482c4 :

>> e9 e0 (equivalent)

0x ff ff ff e0

1111

e0

1110 0000

e9 e0 00 00 00

0x 00 0000 e0

0000

$= -1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6$

$= -128 + 64 + 32$

$= -32_{10}$

$= -20_{16} = -0x20$

16 | 32

2 — 0

If / Else (Example the book):

goto statement

jmp Label

goto Label ;

Eg:

```
int a = 5 ;
goto end_of_x ; //
.
.
.
end_of_x :
    int b = 5 ;
.
.
.
```

Reading assignment (Monday)

3.6.5 // End of 3.6

3.7 Procedures

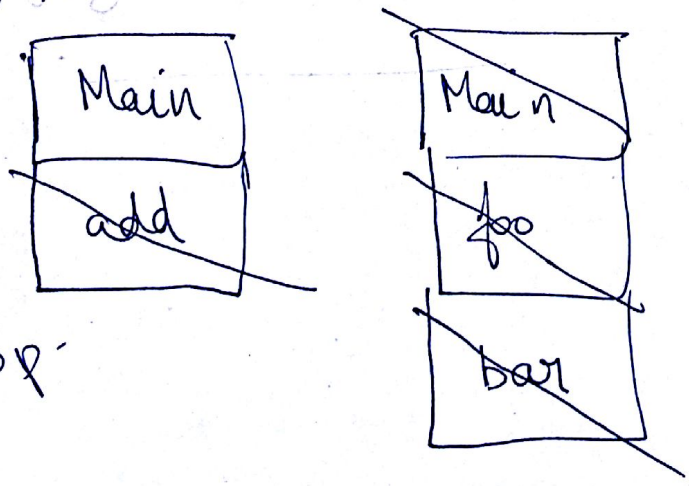
~~Proc~~

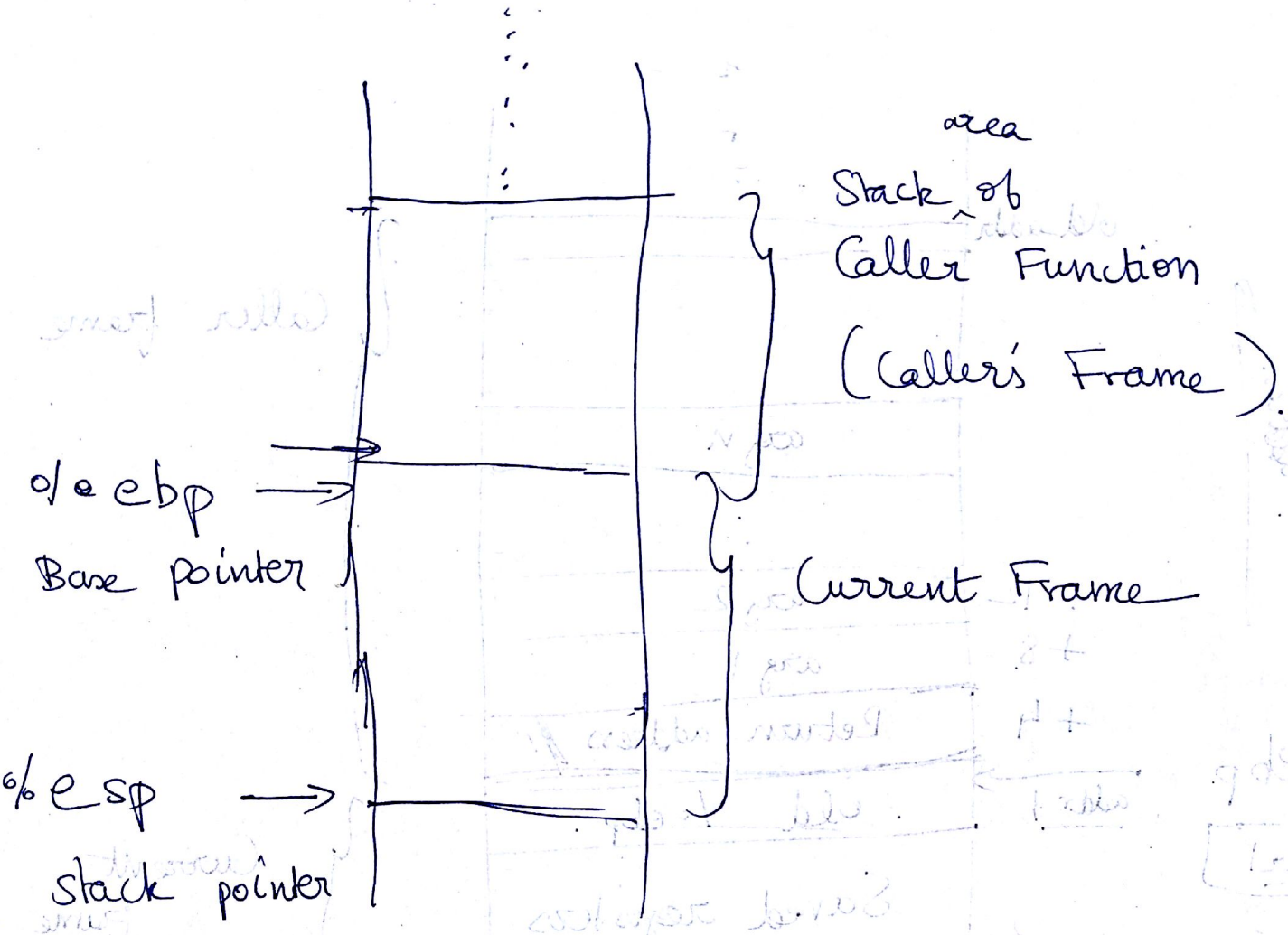
How does assembly handle / support functions?

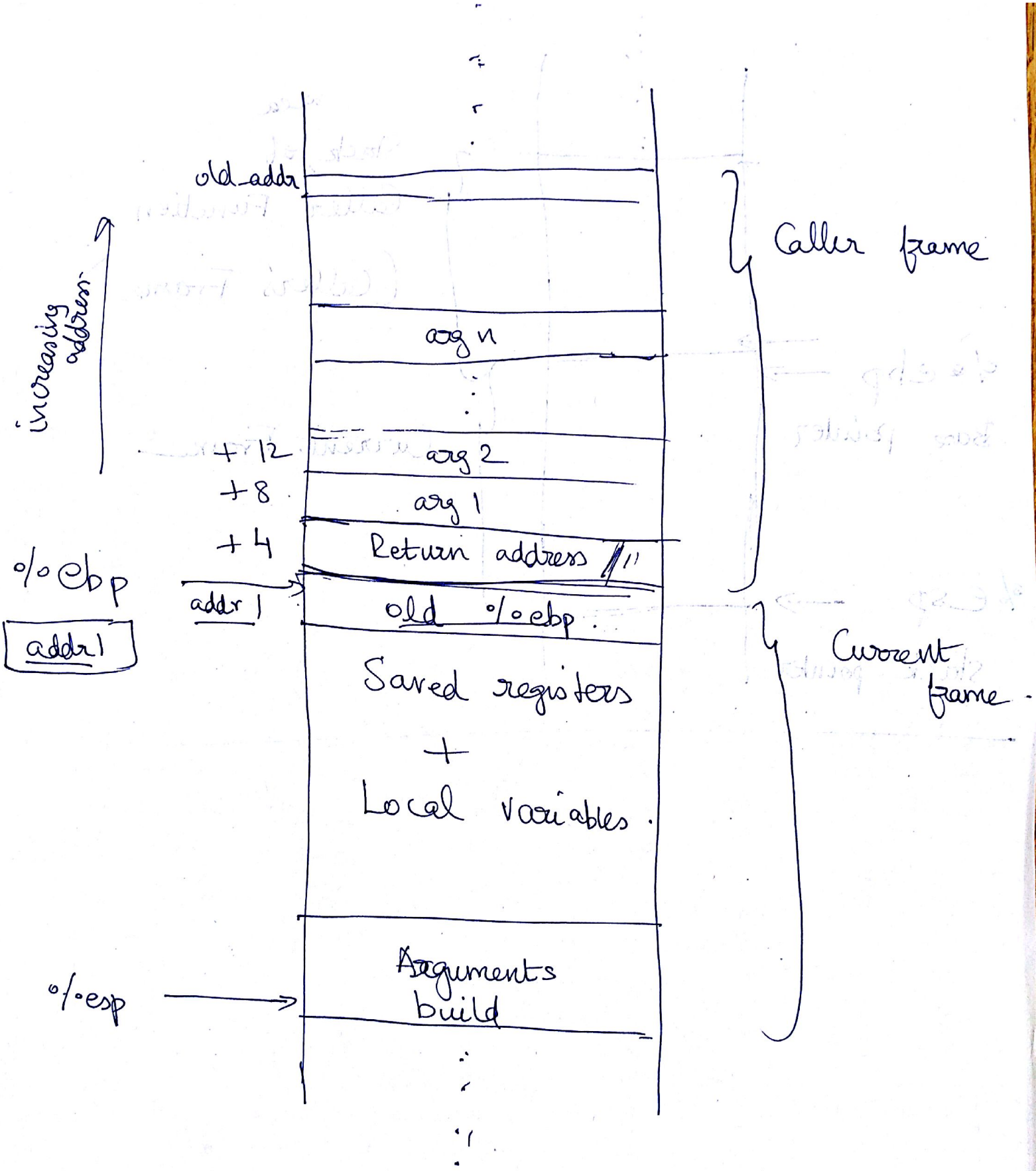
```
int add (int a, int b)
{
    int x = a + b;
    return x;
}

main ()
{
    sum = add (a, b);
}
```

Transfer of data & Transfer of control

Bottom







Return address

Where the program should resume execution when the called function returns.

Arguments

the values that need to be sent / passed to the callee function.

First arg → %ebp + 8

to get the n^{th} argument → %ebp + 4 + 4n

ebp stores the value of the old ebp so that it can point to it again when it returns.

registers

Local Variables

When ? (Local Variables)

→ not enough registers to hold all the local data.

→ When there are references.

&a → address of the first value in a.

How do we transfer control?

call instruction

Syntax

call Label
call * operand.

} Jump to the target specified and continue execution there.

But before jumping, it pushes the return address on to the stack.

↓

Address of the instruction after the call instruction.

call <add>

① mov %eax, %ebx

Return.

(syntax → ret)

The return instruction pops this address off the stack and transfers execution to that address.