```
int multiply (int a, int b) {
    return a*b;
}

int caller ()
    int i = 12;
    int j = 15;
        ;
    multiply()
```

Pass params return values

Save/restore registers used by caller

Room in case we run out of registers

STACK

Every function or procedure has a

Stack frame

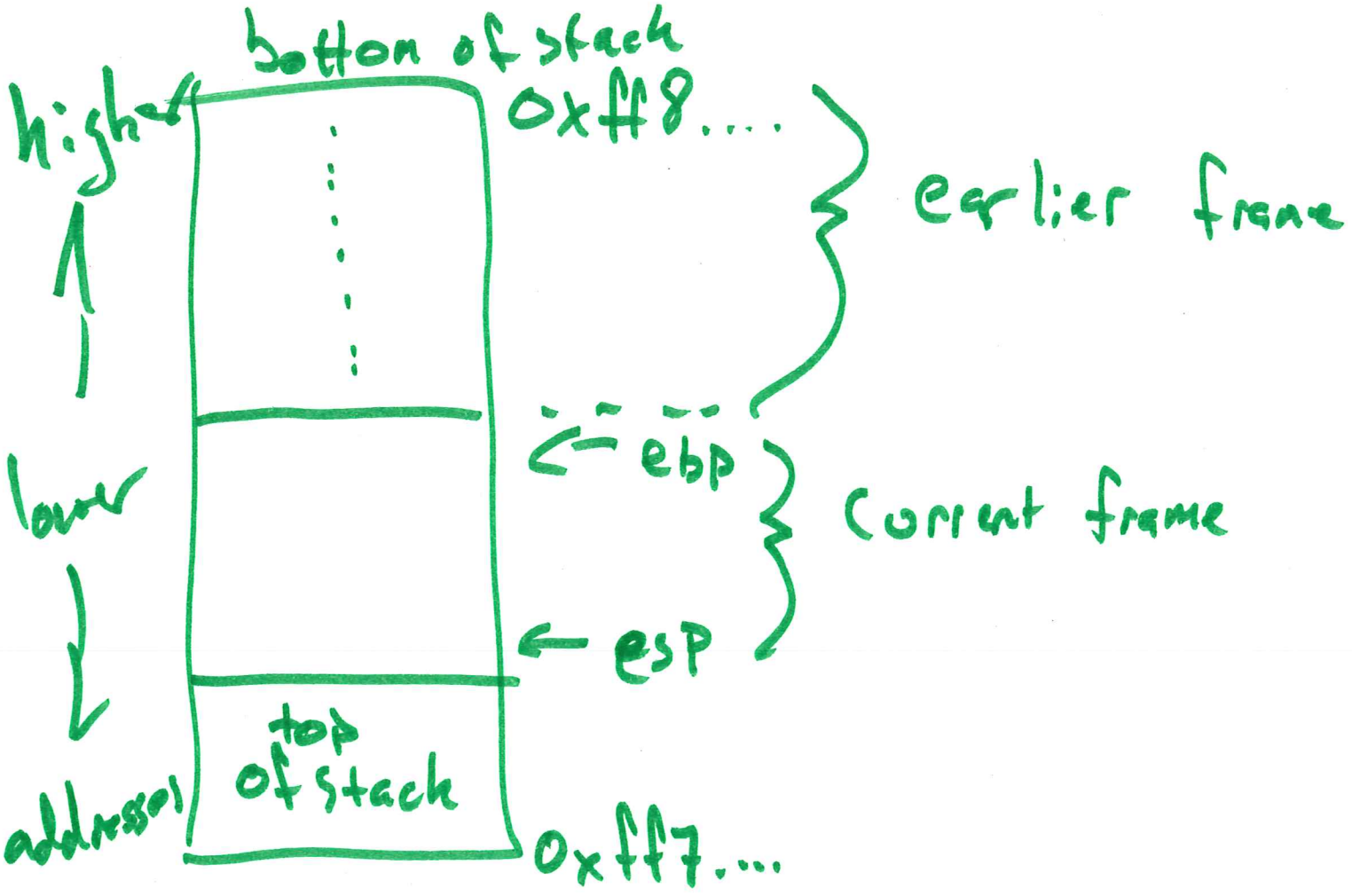
Top of the stack hold the most recent value

↳ confusing!
actually lowest address

Stack is deliminated by 2 pointes

%ebp → base of current frame

%esp → top of current frame



---

new instructions

know push + pop

subl $4, %esp
movl op, (%esp)

movl (%esp), op
addl $4, %esp

Call label

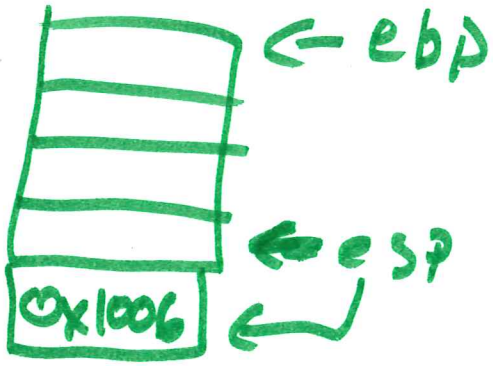
2 things 1) pushes return address onto the stack

2) jumps to label

0x1000 ~~~

0x1002 → call func

0x1006 movl - -



---

ret → returns

opposite of call

pops the top item on stack

jumps to that address

leave called before return

-> fixes up the stack back to original value

```
movl %ebp, %esp
popl %ebp
```

---

return value always in %eax

What to do with reused registers?

```
movl $12, %edx        cool_func:
movl $20, %esi            :
call cool_func        movl $5000, %edx
     :                movl $0, %esi
```

2 options:

Caller saves registers

Callee saves

Some saved by caller → Volatile

%eax, %edx, %ecx

Some are callee saved → non-volatile

%ebx, %esi, %edi