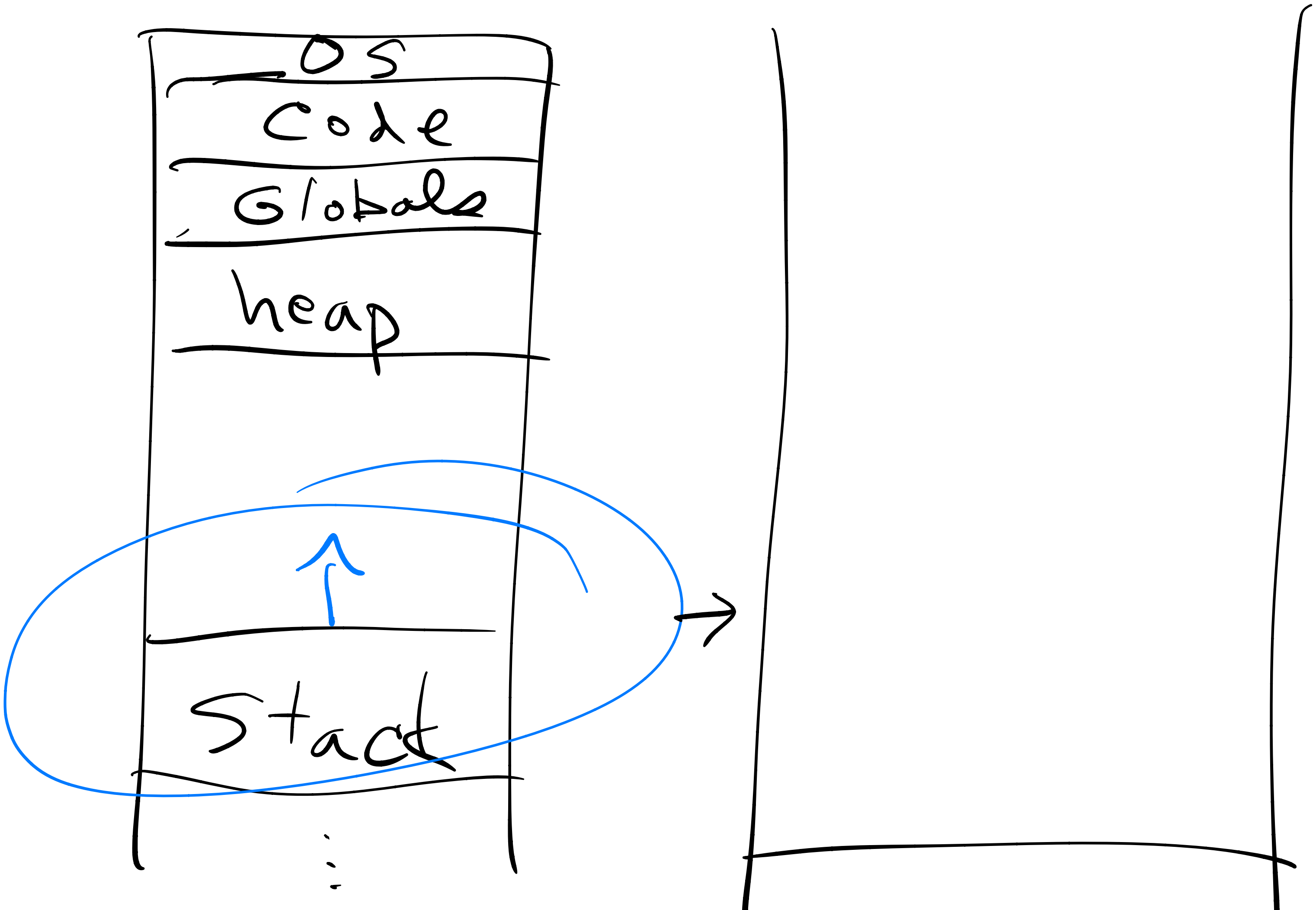
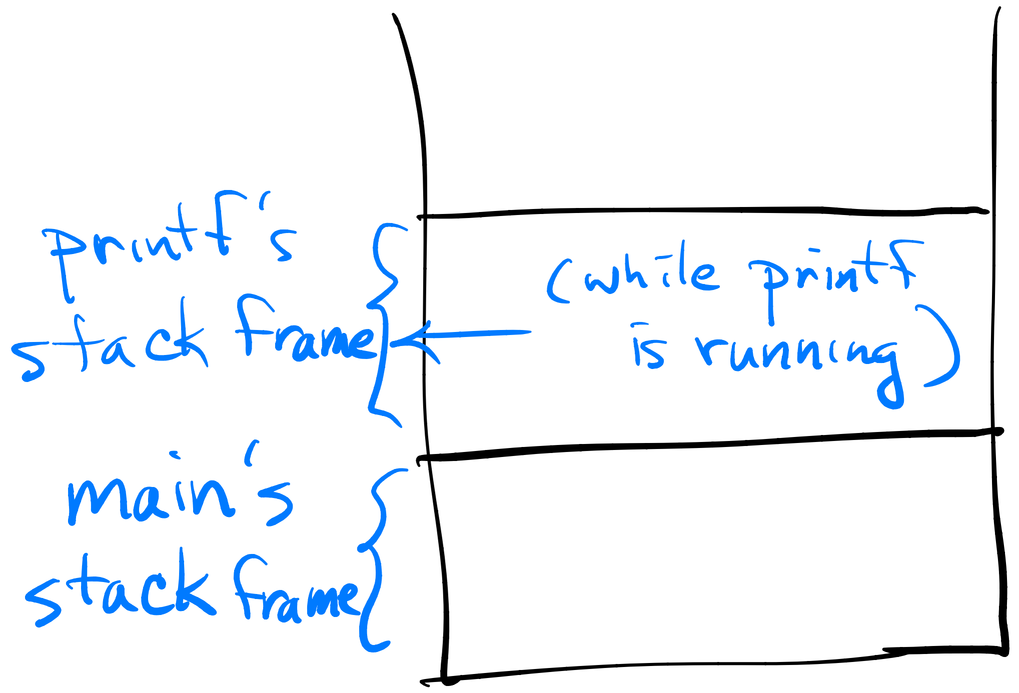
CS208

F, 24 Oct 2025

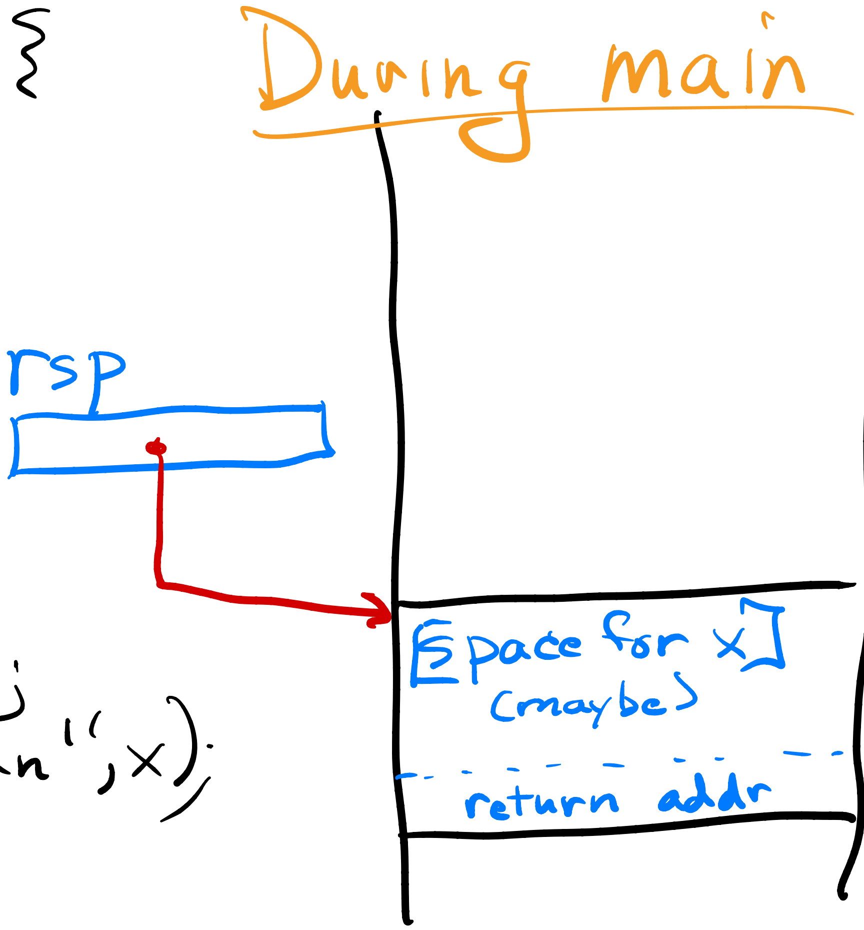

OS
Code
Globals
heap
Stack

When you call a function, it gets a chunk of memory on top of the stack



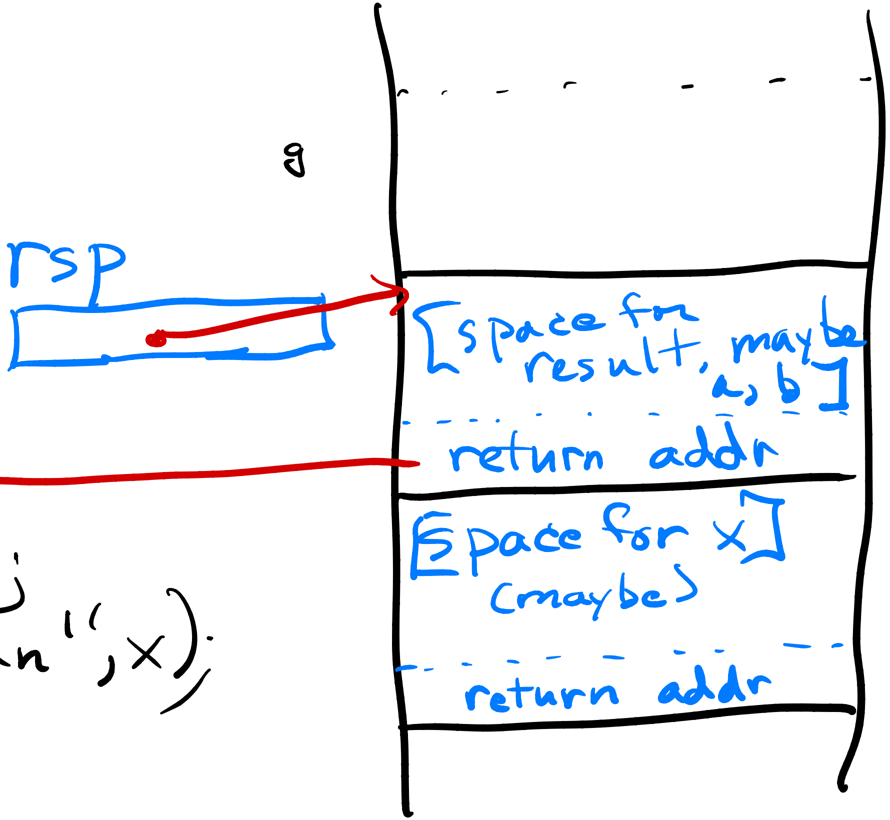
```
int main() {
    .
    .
    .
    printf(...)

}
```

```
int f(int a, int b) {
    int result;
       :
    return ____
}

int main() {
    printf("hi\n");
    int x = f(3, 7);
    printf("hello %d\n", x);
    return 0;
}
```



During f

```
int f(int a, int b) {
    int result;
  ... g(a+b) ...
    return ___
}

int main() {
  printf("hi\n");
  int x = f(3,7);
  printf("hello %d\n", x);
  return 0;
}
```



Relevant registers

rsp — addr of current top of stack

rip — "instruction pointer" address of the next instr. to be executed

(rbp — "block ptr" — sometimes shows up)

What does call f do?

① push rip

ie. puts the addr of the next instruction after <u>call</u> onto the stack

Has the effect of rsp = rsp - 8

② jmp f

ie. rip = addr of f

What does ret do?

① pops top 8 bytes of stack into rip
(i.e. copies 8 bytes wherever rsp points into rip
& rsp = rsp + 8)