CS 354 - Machine Organization & Programming
Thursday, November 7, 2019

Midterm Exam (~18%): TONIGHT Thursday, November 7th, 7:15 - 9:15 pm

- **Lec 1 (2:30 pm):** room 3650 of Humanities
- **Lec 2 (4:00 pm):** room B10 of Ingraham Hall
- UW ID required
- #2 pencils required
- closed book, no notes, no electronic devices (e.g., calculators, phones, watches)
- see “Midterm Exam 2” on course site Assignments for topics

Project p4b (~4%): DUE at 10 pm on Wednesday, November 13th

Last Time

Instructions - Arithmetic and Shift
Instructions - CMP and TEST, Condition Codes
Instructions - SET
Instructions - Jumps
Encoding Targets
Converting Loops
---------- END of Exam 2 Material -----------

Today

The Stack from a Programmer’s Perspective
The Stack and Stack Frames
Instructions - Transferring Control
Exam Mechanics

Next Time

More Stack Frames
Read: B&O 3.7.3 - 3.7.5

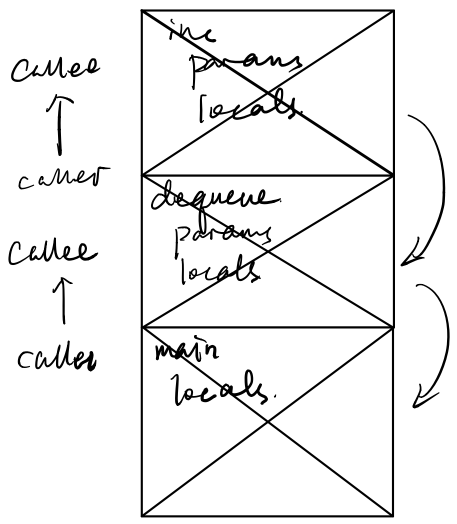
The Stack from a Programmer's Perspective

Consider the following code:

```
int inc(int index, int size) {
   int incindex = index + 1;
   if (incindex == size) return 0;
   return incindex;
}

int dequeue(int *queue, int *front,
         int rear, int *numitems, int size) {
   if (*numitem == 0) return -1;
   int dqitem = queue[*front];
   *front = inc(*front, size);
   *numitems -= 1;
   return dqitem;
}

int main(void) {
   int queue[5] = {11,22,33};
   int front = 0;
   int rear = 2;
   int numitems = 3;
   int qitem = dequeue(queue, &front, rear,
        &numitems, 5);
   ...
```



What does the compiler need to do to make function calls work?

- transfer control to callee (store returen addr) and then transfer back to caller
- handle arg passing
- alloc free stack frame.
- alloc free local variable.
- handle return value.
- other details

The Stack and Stack Frames

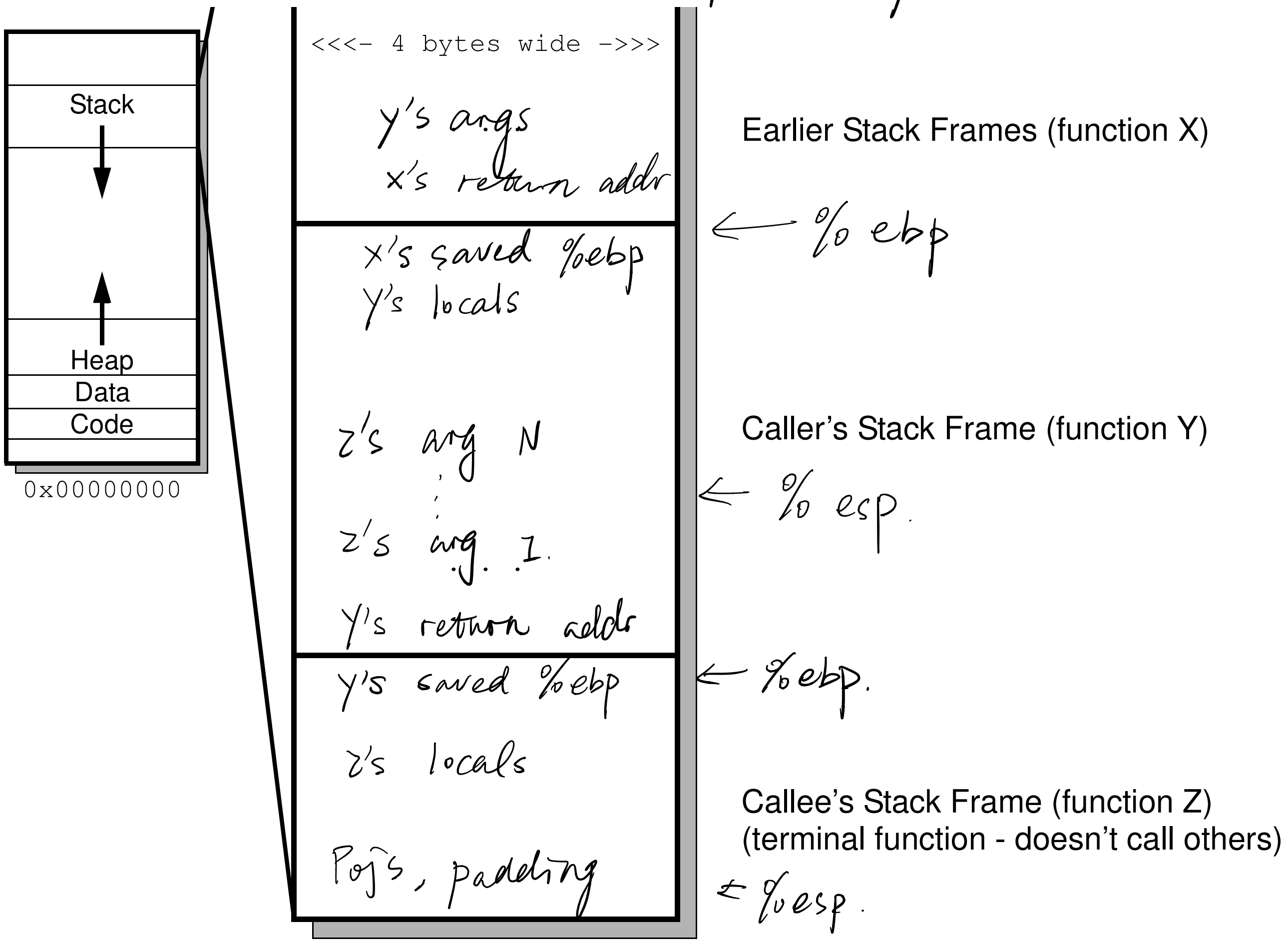
Stack Frame is a block of stack mem. used by a single function call.

IA-32: a complete cycling frame must be multiple of 16B.

<u>%ebp</u> base ptr reg. points to bottom 4 bytes of stack frame.

<u>%esp</u> stack ptr register. points to top 4 bytes of top stack frame.

Stack Layout



❋ *A Callee's args* are actually in its caller's stack frame.

→ What is the offset from the %ebp to get to a callee's first argument?

+8

→ When are local variables allocated on the stack?

1. not enough regs
2. local is a composite.
3. code uses "&" address of on local.

Instructions - Transferring Control

Flow Control

function call:

`call *Operand` indirect

`call Label` direct.

steps (for both forms of call)

1. push ret address onto stack.
   equivalent : pushl %eip

2. jump to start of callee.
   equivalent : jmp *operand.
   jmp label.

function return:

`ret`

step

1. jump to the return address that is popped off stack.
   equivalent : popl %eip.

Stack Frames

allocate stack frame:

no special instruction.
use : subl $x , %esp. where x is size of new stack frame.

free stack frame:

`leave`

steps

1. remove all of callee's frame except for caller's saved %ebp
   equivalent : movl %ebp , %esp.
2. restore caller's frame
   equivalent : popl %ebp