CS 354 - Machine Organization & Programming
Tuesday Nov 5th, and Thursday Nov 7th,2024

Midterm Exam - Thurs Nov 7th, 7:30 - 9:30 pm

- **UW ID and #2 required, room information sent via email (bring copy to exam)**
- **closed book, no notes, no electronic devices (e.g., calculators, phones, watches) see "Midterm Exam 2" on course site Assignments for topics**

A10 e2_cheatsheet.pdf

Homework hw4: DUE on or before Monday, Nov 4

Homework hw5: will be DUE on or before Monday, Nov 11

Project p4B: DUE on or before Sunday, Nov 10

Project p5: DUE on or before Friday Nov 22

Learning Objectives

- identify and describe conventions for IA-32 registers and cond codes ZF, SF, OF, CF
- trace and describe how conditional assembly instructions and execution
- trace and describe how repetition is achieved in ASM and Mach Code
- trace and describe how control is transferred to a function call
- trace and describe how control is returned from a function call

This Week

Finish W09 Outline Instructions - LEAL, Arithmetic and Shift Instructions - CMP and TEST, C.C.s Instructions - SET & Jumps Encoding Targets & Converting Loops	The Stack from a Programmer's Perspective The Stack and Stack Frames Instructions - Transferring Control Register Usage Conventions Function Call-Return Example
Next Week: Finish Stack Frames B&O 3.7 Intro - 3.7.5 3.8 Array Allocation and Access 3.9 Heterogeneous Data Structures	

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?

◆

◆

◆

◆

◆

◆

The Stack and Stack Frames

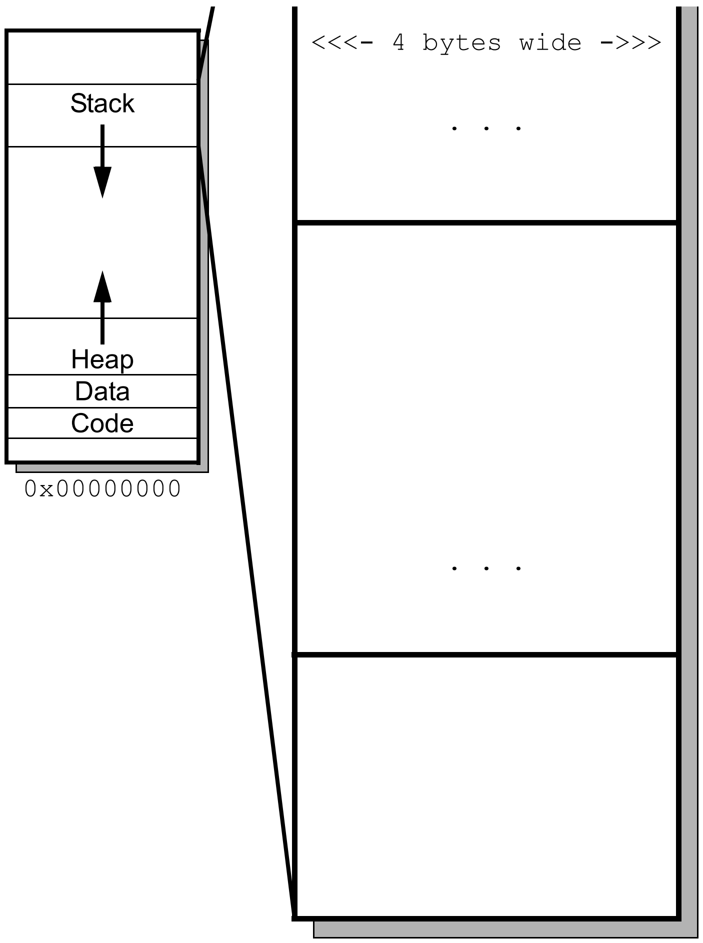
Stack Frame

IA-32:

%ebp

%esp

Stack Layout



Stack
Heap
Data
Code
0x00000000
<<<- 4 bytes wide ->>>
. . .
Earlier Stack Frames (function X)
Caller's Stack Frame (function Y)
. . .
Callee's Stack Frame (function Z)
(terminal function - doesn't call others)

✵ *A Callee's args*

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

→ When are local variables allocated on the stack?

Instructions - Transferring Control

Flow Control

function call:

```
call *Operand
```

```
call Label
```

steps (for both forms of call)

1.

2.

function return:

```
ret
```

step

1.

Stack Frames

allocate stack frame:

free stack frame:

```
leave
```

steps

1.

2.

Register Usage Conventions

Return Value

Frame Base Pointer `%ebp`

callee uses to

Stack Pointer `%esp`

caller uses to

callee uses to

Registers and Local Variables

→ Why use registers?

→ Potential problem with multiple functions using registers?

IA-32

caller-save:

callee-save:

Function Call-Return Example

```
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);     1ab setup calleE's args
                                   2  call the calleE function
                                    a  save caller's return address
                                    b  transfer control to calleE
                                   7  caller resumes, assigns return value
   *numitems -= 1;
   return dqitem;
}
int inc(int index, int size) {     3  allocate callee's stack frame
                                    a  save calleR's frame base
                                    b  set callee's frame base
                                    c  set callee's top of stack
   int incindex = index + 1;       4  callee executes ...
   if (incindex == size) return 0;
   return incindex;                5  free callee's stack frame
}                                   a  restore calleR's top of stack
                                    b  restore calleR's frame base
                                   6  transfer control back to calleR
```

CALL code in `dequeue`

```
1a 0x0_2C  movl index,(%esp)

 b 0x0_2E  movl size,4(%esp)

2  0x0_30  call inc
 a

 b
```

RETURN code in `dequeue`

```
7  0x0_55 movl %eax,(%ebx)
```

CALL code in `inc`

```
3a 0x0_F0  pushl %ebp

 b 0x0_F2  movl %esp,%ebp

 c 0x0_F4  subl $12,%esp

4  0x0_F6  execute inc function's body
```

RETURN code in `inc`

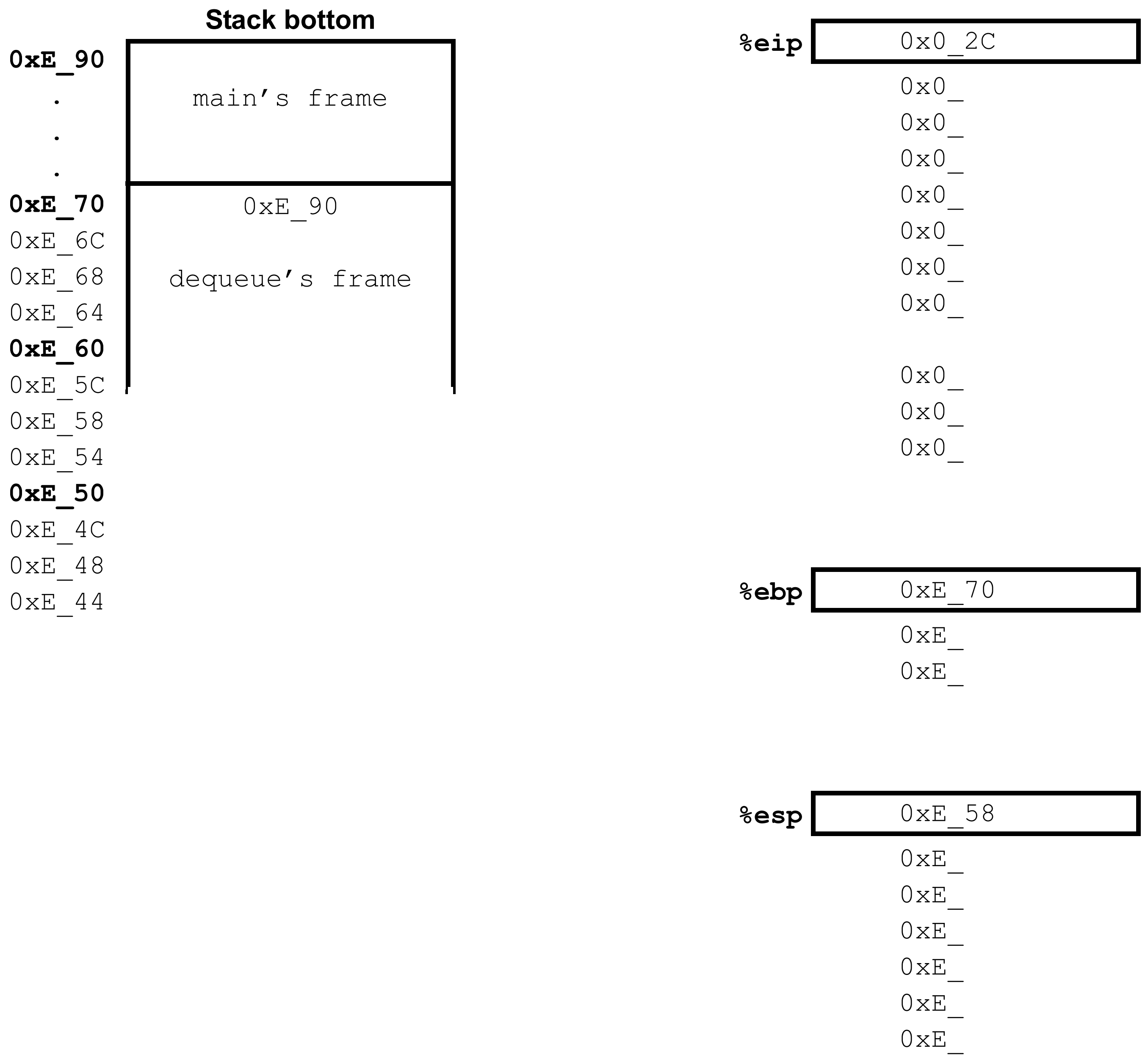
```
5  0x0_FA  leave
 a

 b

6  0x0_FB  ret
```

Function Call-Return Example

Execution Trace of Stack and Registers



Stack bottom
0xE_90
.
.
.
0xE_70
0xE_6C
0xE_68
0xE_64
0xE_60
0xE_5C
0xE_58
0xE_54
0xE_50
0xE_4C
0xE_48
0xE_44
main's frame
0xE_90
dequeue's frame
%eip
0x0_2C
0x0_
0x0_
0x0_
0x0_
0x0_
0x0_
0x0_
0x0_
0x0_
0x0_
%ebp
0xE_70
0xE_
0xE_
%esp
0xE_58
0xE_
0xE_
0xE_
0xE_
0xE_
0xE_