# Tuesday April 9th, and Thursday April 11th, 2023

## Last Week

Operand/Instruction Caveats Instruction - LEAL

Instructions - Arithmetic and Shift

---------- END of Exam 2 Material -----------

Instructions - CMP and TEST, Condition Codes

C, Assembly, & Machine Code Low-level View of Data Registers

Operand Specifiers & Practice L18-7 Instructions - MOV, PUSH, POP

|  |  |
| --- | --- |
| This Week:  From L18: 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**: Stack Frames B&O 3.7 Intro - 3.7.5  3.8 Array Allocation and Access  3.9 Heterogeneous Data Structures | |

A white paper with writing on it

Description automatically generated

***Stack Frame***

IA-32:

*%ebp*

*%esp*

## Stack Layout

Earlier Stack Frames (function X)



<<<- 4 bytes wide ->>>

. . .

0x00000000

. . .

Code

Data

Heap

Stack

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?

A hand holding a piece of paper

Description automatically generated

A paper with writing on it

Description automatically generated

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**

1. **save caller’s return address**
2. **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**

1. **save calleR’s frame base**
2. **set callee’s frame base**
3. **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**

1. **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

1. 0x0\_55 movl %eax,(%ebx)

## CALL code in inc

**3a** 0x0\_F0 pushl %ebp

* 1. 0x0\_F2 movl %esp,%ebp
  2. 0x0\_F4 subl $12,%esp

1. 0x0\_F6 execute inc function’s body

## RETURN code in inc

1. 0x0\_FA leave

**a b**

1. 0x0\_FB ret

## Execution Trace of Stack and Registers

**0xE\_90**

0xE\_90 dequeue’s frame

main’s frame

.

.

.

## 0xE\_70

0xE\_6C

0xE\_68

0xE\_64

## 0xE\_60

0xE\_5C

0xE\_58

0xE\_54

## 0xE\_50

0xE\_4C

0xE\_48

0xE\_44

## Stack bottom

## %eip

**%ebp**

0x0\_

0x0\_2C

0x0\_

0x0\_

0x0\_

0x0\_

0x0\_

0x0\_

0x0\_

0x0\_

0x0\_

0xE\_

0xE\_70

0xE\_

## %esp

0xE\_58

0xE\_

0xE\_

0xE\_

0xE\_

0xE\_

0xE\_