PRINCIPLES OF COMPUTER ORGANIZATION

Pre-requisites: CMSC B 151 or H106 or H107 and CMSC B231
ADMINISTRIVIA

Instructor: Deepak Kumar  
Office: Room 202 PSB  
Office Hours: TBA  

Lectures: Tue-Thur 12:55p to 2:15p  
Lecture Room: 245 Park Science Building  

Labs: Tuesdays from 2:25p to 3:45p (starts next week)  
Lab Room: 231 Park Science Building  

Class Website: https://cs.brynmawr.edu/Courses/cs240/Spring2024/

TEXT & SOFTWARE

Main Text  

Software  
LC3 Simulator (Windows/MacOS/Linux)
LEARNING OUTCOMES

- Describe the major components of a modern computer (CPU, Memory, I/O) and how they are implemented and interact in hardware.
- Understand how programs and data are represented in hardware, how instructions are executed, and how data is stored in and retrieved from memory.
- Design circuits to implement Boolean functions and basic storage/memory constructs using digital logic.
- The von Neumann Model
- Relate the behavior of high-level languages like C or Java to the underlying low-level assembly language.

TOPICS

- What is a Computer?
- Instruction Set Architecture (ISA)
- Bits, Data Types, and Operations
- The von Neumann Model
- The LC3 ISA
- Programming in Assembly using LC3 ISA
- Subroutines and the Stack
- I/O Operations: Service Routines, Traps and Interrupts
- Memory Hierarchy & Caching
- Non-von Neumann Architectures
MY ROLE

• Create educational opportunities for you to achieve the learning outcomes.
• Assess your progress and provide timely feedback
• Provide support so you can successfully complete the course.

YOUR RESPONSIBILITIES

• Put in the effort
• Attend all classes and labs (let me know if you are going to miss any)
• Follow the rules and guidelines in the class website
• Stay connected
  • Check the class website for schedule, updates, readings, and assignments
  • Do the readings as and when they are assigned. This is a key to success.
  • Be familiar with the main concepts before every class meeting
  • Ask for help when you need it!
# IMPORTANT DATES & EVALUATION

<table>
<thead>
<tr>
<th>Important Dates</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 23</td>
<td>First Class Meeting</td>
</tr>
<tr>
<td>February 20</td>
<td>Exam 1</td>
</tr>
<tr>
<td>April 2</td>
<td>Exam 2</td>
</tr>
<tr>
<td>May 2</td>
<td>Exam 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assessments</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exam 1</td>
<td>20%</td>
</tr>
<tr>
<td>Exam 2</td>
<td>20%</td>
</tr>
<tr>
<td>Exam 3</td>
<td>20%</td>
</tr>
<tr>
<td>Assignments</td>
<td>30%</td>
</tr>
<tr>
<td>Labs</td>
<td>10%</td>
</tr>
</tbody>
</table>

**WHAT IS A COMPUTER?**
WHAT IS A COMPUTER?

A device with

- A processor (CPU – Central Processing Unit)
- Storage/Memory (RAM, Disk, USB Stick)
- A keyboard
- A mouse
- A Monitor
- A printer

THESE ARE ALL COMPUTERS
FRONTIER
World's Fastest Computer (2023)
Oak Ridge National Labs, TN
~50,000 Processors
Size of two tennis courts
Costs $600 million
Can execute ~1 ExaFLOP ($10^{18}$ FLOPS)

WHAT WAS THE FIRST COMPUTER?
BABBAGE’S DIFFERENCE ENGINE#2 (1832, 2002)
See video at: https://youtu.be/XSKgY6LchJs?si=GzVuDHh9yPtsW9DBL

ENIAC FEBRUARY 16, 1946 (U. PENN)
See video at: https://youtu.be/XSKgY6LchJs?si=GzVuDHh9yPtsW9DBL
WHAT IS THE DIFFERENCE BETWEEN THE ENIAC, IPHONE, AND FRONTIER?

NOTHING!
Besides the size, cost, the speed of the processor(s), and amount of memory.
BUT REALLY, WHAT IS A COMPUTER?

TURING MACHINE: AN IDEALIZED COMPUTER

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>0,R,s</td>
<td>1,R,a</td>
</tr>
<tr>
<td>a</td>
<td>0,R,b</td>
<td>1,R,a</td>
</tr>
<tr>
<td>b</td>
<td>0,R,s</td>
<td>1,R,c</td>
</tr>
</tbody>
</table>
TURING MACHINES FOR ADDING & MULTIPLYING

\[ T_{\text{ADD}} \quad T_{\text{MUL}} \]

\[ a, b \rightarrow a + b \quad a, b \rightarrow a \times b \]

(Turing machine that adds)
(Turing machine that multiplies)

Figure 1.7 Black box models of Turing Machines

CHURCH-TURING THESIS* (1936)

EVERY COMPUTATION CAN BE PERFORMED BY SOME TURING MACHINE.
UNIVERSAL TURING MACHINES

$T_{\text{ADD}}, T_{\text{MUL}}$ → $U$ → $g \times (e + f)$

$e, f, g$ → (Universal Turing machine)

Figure 1.8 Black box model of a universal Turing Machine

UNIVERSAL TURING MACHINE = PROGRAMMABLE COMPUTER

$T_{\text{ADD}}, T_{\text{MUL}}$ → $U$

Program → $g \times (e + f)$

$e, f, g$ → (Universal Turing machine)

Figure 1.8 Black box model of a universal Turing Machine
KEY IDEA

A COMPUTER IS ESSENTIALLY A UNIVERSAL TURING MACHINE.

PUT THIS ON YOUR WATCH LIST

“THE BEST BRITISH FILM OF THE YEAR”

THE IMITATION GAME

BASED ON THE INCREDIBLE TRUE STORY OF ALAN TURING
OK, SO...

HOW DO WE ACTUALLY BUILD A UNIVERSAL TURING MACHINE (COMPUTER)???

VON NEUMAN ARCHITECTURE (1945)
Technically, most of today's computers are non-von Neumann computers.

I what way?

To understand/answer this, we will first need to learn what von Neumann computers are.
HOW DO WE GET THE COMPUTER TO WORK???

LEVELS OF TRANSFORMATION

Problems

Algorithms

Language

Machine (ISA) Architecture

Microarchitecture

Circuits

Devices
LEVELS OF TRANSFORMATION

Problems

Algorithms

Language

Machine (ISA) Architecture

Microarchitecture

Circuits

Devices

LEVELS OF TRANSFORMATION: EXAMPLE

PROBLEM

Compute the square root of a given number, \( a \).
LEVELS OF TRANSFORMATION

PROBLEM

Compute the square root of a given number, $a$.

A SOLUTION

To compute the square root $x = \sqrt{a}$ do the following:
1. Start with some guess $x_1 > 0$
2. Compute a sequence of guesses $x_1, x_2, ..., x_n$ using the equation

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right)$$

until the numbers produced converge.

ALGORITHM

1. To compute $\sqrt{a}$
2. Start with some guess $x_i = 1$. This is our initial guess.
3. Compute the next guess $x_{i+1} = \frac{1}{2} \left( x_i + \frac{a}{x_i} \right)$
4. If $x_{i+1} \neq x_i$
   Set $x_i$ to be same as $x_{i+1}$
   And then repeat from Step 3.

   Otherwise, because $x_{i+1} = x_i$, they have converged.
   Therefore, $\sqrt{a} = x_{i+1}$
LEVELS OF TRANSFORMATION

ALGORITHM

1. To compute $\sqrt{a}$
2. Start with some guess $x_i = 1$. This is our initial guess.
3. Compute the next guess $x_{i+1} = \frac{1}{2} \left( x_i + \frac{a}{x_i} \right)$
4. If $x_{i+1} \neq x_i$
   Set $x_i$ to be same as $x_{i+1}$
   And then repeat from Step 3.

Otherwise because $x_{i+1} = x_i$, they have converged.
Therefore, $\sqrt{a} = x_{i+1}$

Definition:
An algorithm is a precise, unambiguous, and effective procedure.

PROGRAM

```c
double sqrt (double a) {
    if (a <= 0) return 0;
    double x0 = 1;
    double x1 = (x0 + a/x0)/2.0;
    while (x0 != x1) {
        x0 = x1;
        x1 = (x1 + (a/x1))/2.0;
    }
    return x1;
} // sqrt()
```
LEVELS OF TRANSFORMATION

ASSEMBLY PROGRAM (ISA)

.globl sqrt
sqrt:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsi, %rbp
.cfi_offset 6, -24
movsd %xmm0, -24(%rbp)
pxor %xmm0, %xmm0
comisd -24(%rbp), %xmm0
jb .L8
pxor %xmm0, %xmm0
jmp .L4
.L8:
.L5:
movsd .LC1(%rip), %xmm0
movsd %xmm0, -16(%rbp)
movsd -24(%rbp), %xmm0
divsd -16(%rbp), %xmm0
addsd -16(%rbp), %xmm0
movsd .LC2(%rip), %xmm1
divsd %xmm1, %xmm0
movsd %xmm0, -8(%rbp)
jmp .L5
.L6:
movsd -8(%rbp), %xmm0
movsd %xmm0, -16(%rbp)
movsd -24(%rbp), %xmm0
divsd -8(%rbp), %xmm0
addsd -8(%rbp), %xmm0
movsd .LC2(%rip), %xmm1
divsd %xmm1, %xmm0
movsd %xmm0, -8(%rbp)
jmp .L5
.L7:
movsd .LC1(%rip), %xmm0
movsd %xmm0, -16(%rbp)
movsd -24(%rbp), %xmm0
divsd -16(%rbp), %xmm0
addsd -16(%rbp), %xmm0
movsd .LC2(%rip), %xmm1
divsd %xmm1, %xmm0
movsd %xmm0, -8(%rbp)
jmp .L5
.L8:
movsd -8(%rbp), %xmm0
movsd %xmm0, -16(%rbp)
movsd -24(%rbp), %xmm0
divsd -8(%rbp), %xmm0
addsd -8(%rbp), %xmm0
movsd .LC2(%rip), %xmm1
divsd %xmm1, %xmm0
movsd %xmm0, -8(%rbp)
jmp .L5
.L9:
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size sqrt, .-sqrt
.section .rodata
.align 8
.LC1:
.long 0
.long 1072693248
.align 8
.LC2:
.long 0
.long 1073741824

LEVELS OF TRANSFORMATION

ASSEMBLY PROGRAM (ISA)

.globl sqrt
sqrt:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsi, %rbp
.cfi_offset 6, -24
movsd %xmm0, -24(%rbp)
pxor %xmm0, %xmm0
comisd -24(%rbp), %xmm0
jb .L8
pxor %xmm0, %xmm0
jmp .L4
.L8:
.L5:
movsd .LC1(%rip), %xmm0
divsd -16(%rbp), %xmm0
addsd -16(%rbp), %xmm0
movsd .LC2(%rip), %xmm1
divsd %xmm1, %xmm0
movsd %xmm0, -8(%rbp)
jmp .L5
.L6:
movsd -8(%rbp), %xmm0
movsd %xmm0, -16(%rbp)
movsd -24(%rbp), %xmm0
comisd -8(%rbp), %xmm0
addsd -8(%rbp), %xmm0
movsd .LC2(%rip), %xmm1
movsd %xmm0, -8(%rbp)
jmp .L5
.L7:
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size sqrt, .-sqrt
.section .rodata
.align 8
.LC1:
.long 0
.long 1072693248
.align 8
.LC2:
.long 0
.long 1073741824
LEVELS OF TRANSFORMATION

MACHINE LANGUAGE PROGRAM

0101000000100000 0001010010000000 01100100000000
0011000000100001 0000100000000101 00000011111111
0101001001100000 0001001001111011 01010000010000
0010100000010001 0010000010000000 01010000010000
0110101000000000 0101000001000000 010101101110101
0011000000100000 0100000100000000 01010000001000
0001010010000001 0100000010000000 01010000001000
0000010000000101 0100000001000000 01010000001000
0101000000100000 0010000000100000 00110000001000
0001010010000001 0100000010000000 01010000001000
0000010000000101 0010000000000000 01010000001000
0001010001111111 0100000001000000 01010000001000
0111000000101010 0010000000100000 01010000001000
0011000000100000 0100000001000000 01010000001000

Assembly Program

Assembler

Machine Language Program

What the CPU does:
do forever
fetch the next instruction
decode it
carry it out

Binary representation of assembly program
LEVELS OF TRANSFORMATION

Problems


1.
2.
3.
...

Algorithms


Language


Python Program


Machine (ISA) Architecture


Machine Code (LC3)


Microarchitecture


Circuits


Devices


We will start here!

PUT THIS ON YOUR READING LIST
FOR NEXT CLASS...

Read
Chapter 1 and Sections 2.1 and 2.2 from your text.

THANK YOU