Learning Objectives

After completing this unit, you should be able to:

- Describe fetch-decode-execute in terms of CPU, Bus, and RAM.
- Encode/decode 16-bit instructions from the Brookshear machine.
- Define the instructions LOAD, STORE, MOVE, ADD, and JUMP.
- Explain the difference (none) between code and data in memory.
- Trace the execution of machine code in registers and in memory.
- Write a simple program (6-10 instructions) in machine language.
- Perform bit masking logic operations using AND, OR, and XOR.

Textbook Sections

2.1 Computer Architecture
2.2 Machine Language
2.3 Program Execution
2.4 Arithmetic/Logic

Video Lectures

- Machine Language
- What IS a Dot Diva?
- CS is Changing Everything

Assignments

Act03 Hardware; Chapter 2 Problems

Lab03 Brookshear Machine; CPU and RAM Simulator

Unit 3 Checklist: Sep 09 – Sep 15

Before Wednesday	**Date Completed**
FINISH models 1 and 2 of the Hardware activity	
WATCH video lecture: Machine Language (take notes)	
READ textbook 2.2 Machine Language (take notes) ANSWER questions 3, 6, and 7 in your notes	
READ textbook 2.3 Program Execution (take notes) ANSWER questions 1 and 2 in your notes	
DO tutorial: Brookshear Machine	
START Lab03: Machine language simulator	(10 pts)
Before Friday	**Date Completed**
READ textbook 2.1 Computer Architecture (take notes) ANSWER questions 1, 2, and 3 in your notes	
READ textbook 2.4 Arithmetic/Logic (take notes) ANSWER questions 1, 2, and 3 in your notes	
WATCH video: What IS a Dot Diva? (take notes)	
WATCH video: CS is Changing Everything (take notes)	
START Act03 exercises (complete at least 75%)	(15 pts)
Before Monday	**Date Completed**
COMPARE your Lab03 and Act03 with the solutions in Canvas	
SUBMIT Quiz03 – 1st attempt closed: see what you don't know	
STUDY your notes, ask questions on Piazza, meet with the TAs	
SUBMIT Quiz03 – 2nd attempt open: try to get the full 10 points	(10 pts)

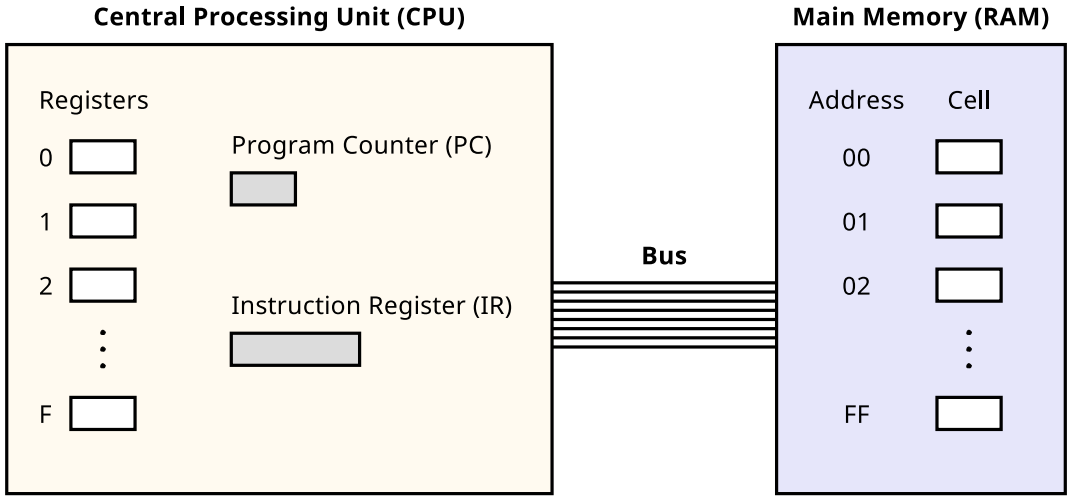
TAKE Exam03	(40 pts)

Activity 3: Hardware

Have you ever wondered what goes on inside your smartphone? Not just how the apps run, but how the hardware actually works? Last week you designed a simple circuit that adds two 4-bit numbers. With enough time and energy, you could learn to build an entire computer! But that's not the goal of this course, so we'll abstract those details. For now, let's take a look at how computers store and process information.

Model 1 Computer Architecture

Here is the 8-bit machine described in Appendix C of Brookshear and Brylow (2015):



Questions (12 min) **Start time:** _____

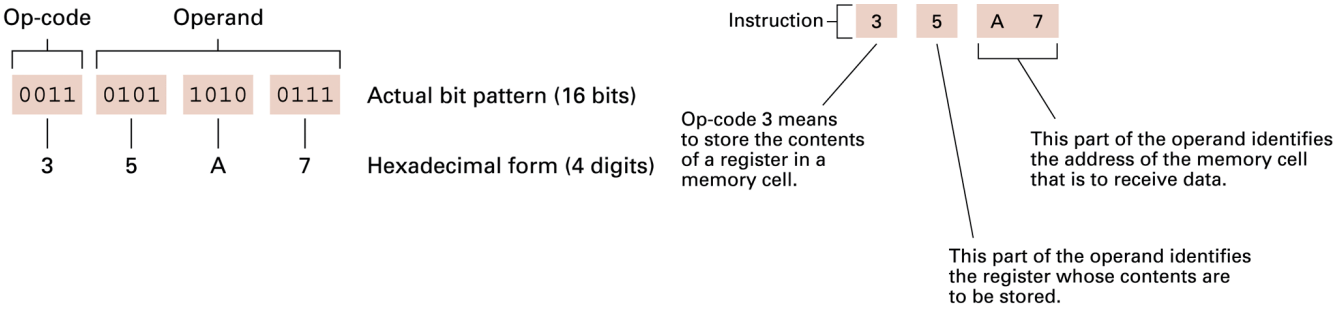
1. What are the three main components in the diagram? Based on its name, what do you think each hardware component does?

2. How many registers does the CPU have? How many memory cells are in RAM?

3. The CPU has circuits for adding the contents of two registers; think of registers as the "pins" in last week's lab. What is the largest number this machine can add without overflowing?

4. Common tasks the CPU performs include ***loading*** data from a memory cell into a register and ***storing*** data from a register into a memory cell. Describe what the CPU would need to do in order to add twenty numbers stored in memory.

Model 2 Machine Instructions



Questions (8 min) Start time: _____

5. How many bits is the op-code? How many possible op-codes can the machine support?

6. The op-code for loading data from memory into a register is 1. Write an instruction (in hex) for loading data from address 3D into register 4.

7. Why is the instruction register in Model 1 twice as large as the other registers? How many memory cells are needed to store a single instruction?

Chapter 2: Data Manipulation

Complete the following Chapter Review Problems on pages 119–124.

#3 (range of memory cells)

#4 (value of program counter)

#5 (end of each fetch phase)

Program Counter	Instruction Register	Memory Cell 02

#7 (machine language to English) – *parts a, b, and e only*

#9 (English to machine language)

#23 (write a short program) – *parts a and b only*

#34 (logical operations) – *each student should do one of each type*

#35 (bit masking) – *parts a, b, and d only*