The Central Processing Unit: CPU

CSC 103

September 24, 2007

Overview for Today

- Paper topics
  - No AI – class discussion
  - Outline and references *next* Wednesday
- First view of programming
  - Addition in ‘machine language’
- The CPU – central processing unit
  - Elements of the CPU
  - Fetch-execute cycle
  - The Pippin simulator

Chapter 3 Objectives

- To see how a mechanical/electrical device can automatically execute any list of instructions, written in binary
  - *i.e.*, it can execute programs
- To see how a simple machine that stores and executes programs (Pippin)
  - All computers do is execute instructions
  - Organized into programs: a precise list of simple instructions

The CPU & ‘Fetch-Execute’

Computer Operation

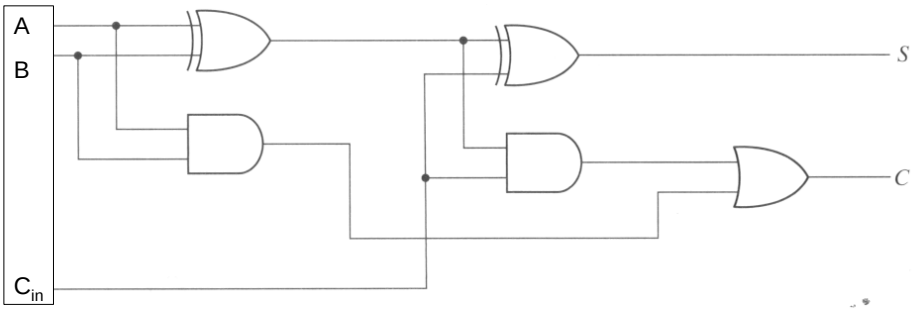
- Computers compute – manipulate 1s & 0s
  - Questions from quiz 1
    - Examples of computation and data types
- Computers *only* execute programs
  - Programs are sequences of instructions
- To execute programs the computer
  - **Fetches** the next instruction, and any data needed
  - **Executes** the instruction
- This is the ‘fetch-execute’ cycle

Class To Date: Hardware

- Addition
  - The three basic logic gates
  - Subtraction: A + (-B) ...
- Memory
  - Logic circuits with feedback
  - Memory circuits and RAM array
  - Data bus and control lines

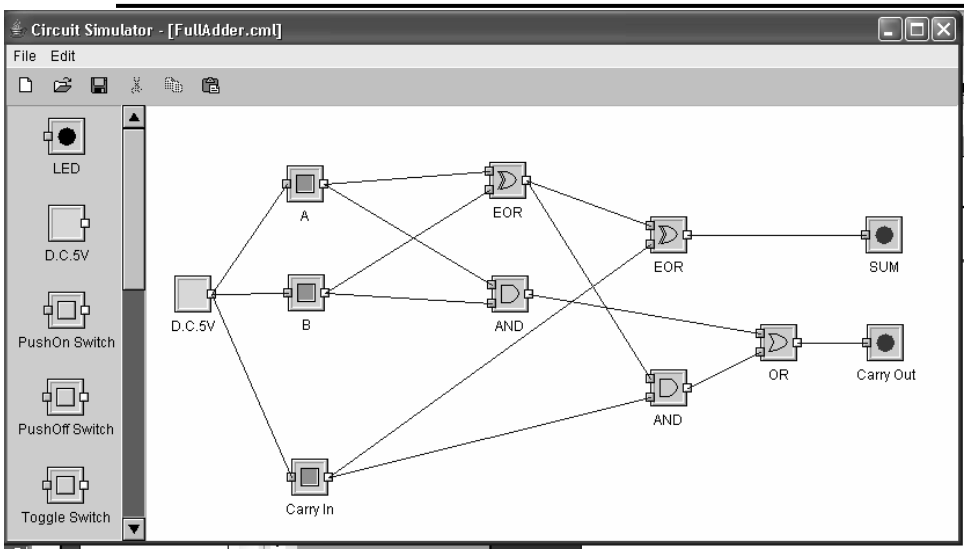
The Arithmetic-Logic-Unit

- Inside the CPU is the Arithmetic-Logic-Unit (ALU), which performs addition, subtraction...

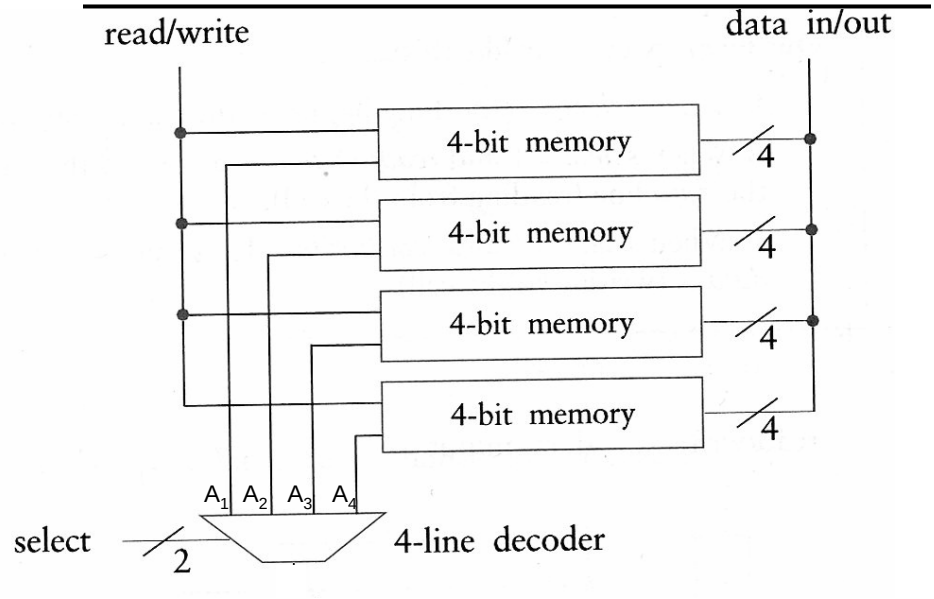


The Full-Adder in SimCir

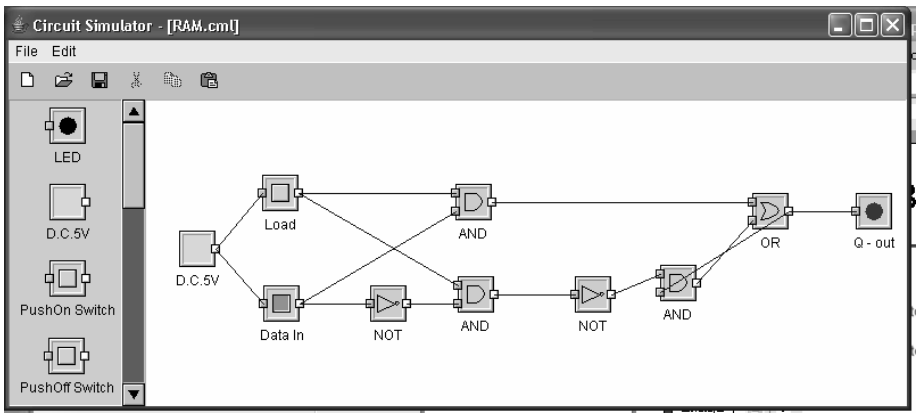
Circuit Simulator link



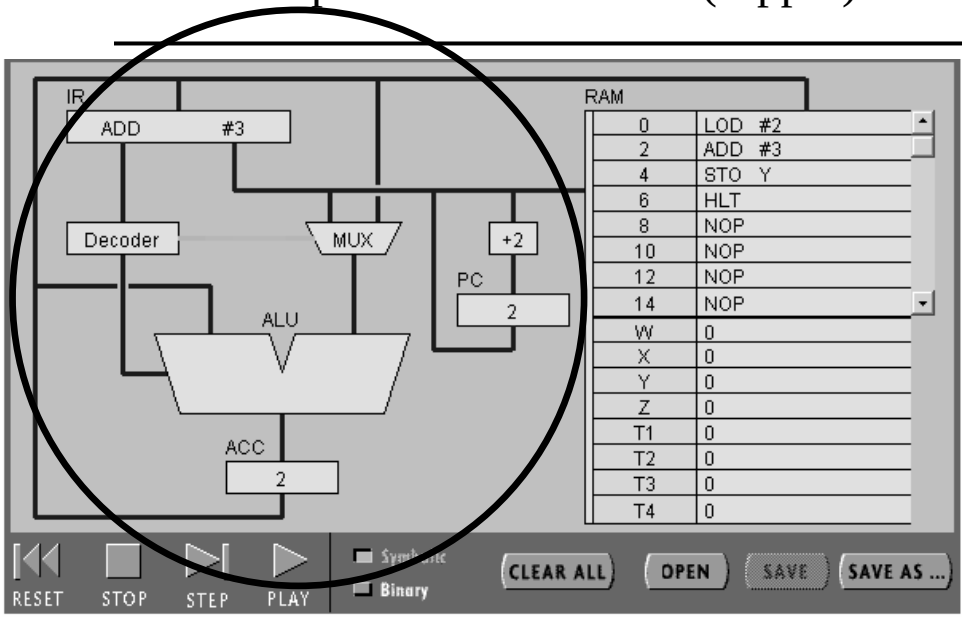
Memory: RAM & Registers



A RAM Circuit in SimCir



Components of the CPU (Pippin)



CPU Examples: Text

- Sample CPU in text is composed of
  - ALU = arithmetic logic unit
  - Registers – memory, temporary staging areas
  - Main memory
  - Control circuit
  - Clock
- PIPPIN has similar components / circuits
- *Read the text chapter! (chapter 3)*

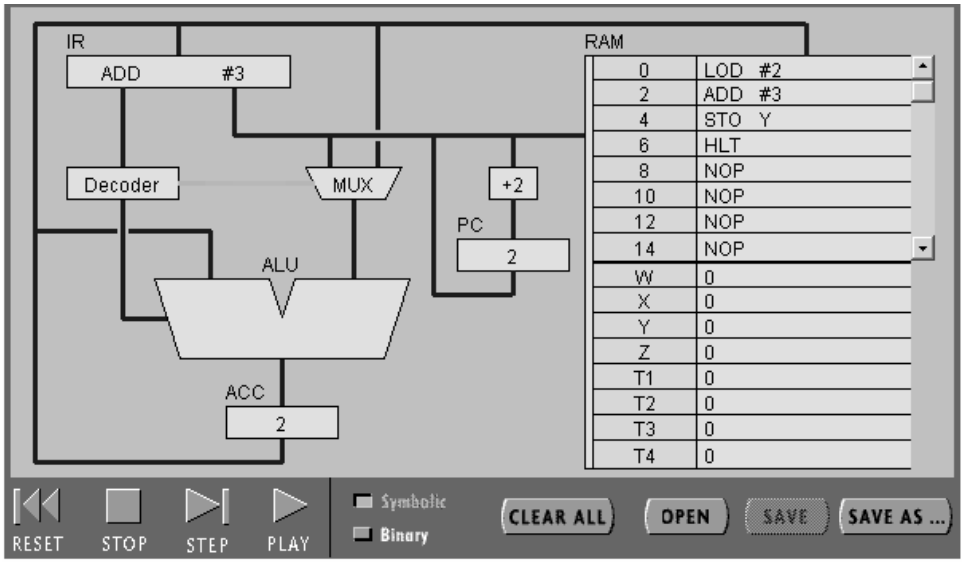
Addition Example

To perform 2 + 3 = 5 :

1. Load ‘2’ into the CPU → in the ACC
2. Add ‘3’ to ‘2’ and temporarily store the result, ‘5’ → in the ACC
3. (=) Store the result to main memory

‘Fetch’ each instruction and data, and then do what it directs (‘execute’)

Components of Pippin



Components of Pippin

- Pippin handout
  - ALU = arithmetic logic unit
  - Registers
    - PC, the program counter
    - Instruction register
    - Accumulator (the computer's scratch pad)
  - Decoder
  - MUX = multiplexor
  - RAM = random access memory

The Pippin Simulator

http://www.science.smith.edu/~jcardell/Courses/CSC103/CPUsim/cpusim.html

Addition Example Con’t

- Keeping everything organized
  - The CPU must know where to find everything
    - Data and instructions
  - The CPU must know where to store the result

Addition Example Con’t

- Keeping everything organized
  - The CPU must know where to find everything
    - Data and instructions
  - The CPU must know where to store the result
- Performing the example in *binary*!
  - The operations (*e.g.*, ‘ADD’) need a binary number assigned to them
  - The operands (data) must be in binary

Exercise at Computers

Type in the Pippin code for 2 + 3 = 5 :

1. Load ‘2’ into the CPU
2. Add ‘3’ to ‘2’ and temporarily store the result, ‘5’
3. (=) Store the result to main memory

Assembly Language

- Our example must be written in simple steps
  - LOD #2 = 0001 0100 0000 0010
  - ADD #3 = 0001 0000 0000 0011
  - STO Y = 0000 0101 1000 0010
  - HLT = 0000 1111 0000 0000

Summary

- Programming in
  - Machine language
  - Assembly language
- The CPU
  - The fetch-execute cycle
  - The Pippin CPU simulator