CS1001

Lecture 3

Overview

- Computer Science; Algorithms
- Multidisciplinary Heritage
- Evolution of Machine Architecture
- Modern Machine Architecture
- Some Simulators
- Homework 1

Goals

- Define Computer Science; what things are part of computer science?
- Define "Algorithm"
- Understand origins of Mechanical Computing
- How did mechanical computing influence modern computer architecture?

Goals (2)

- Understand the components of modern machine architecture
- Examine some basic assembly languages
- Learn how to codify solutions using a given set of actions

Assignments

- Brookshear, Ch 2 (Read)
- Read linked documents on these slides (slides will be posted in courseworks)
- Check your email and courseworks (http://courseworks.columbia.edu)

What is an Algorithm?

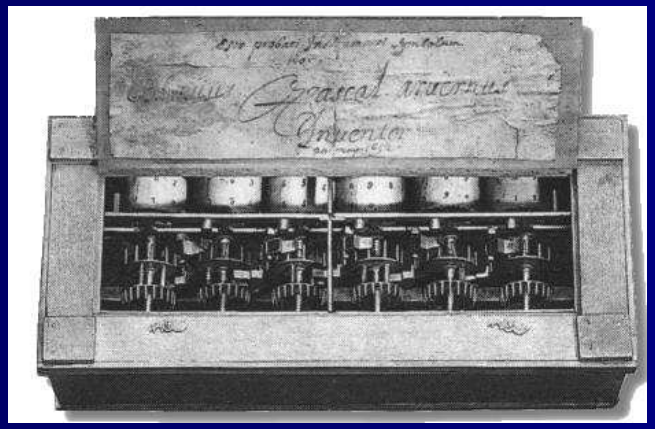
- "An ordered set of unambiguous, executable steps that define a *terminating* activity"

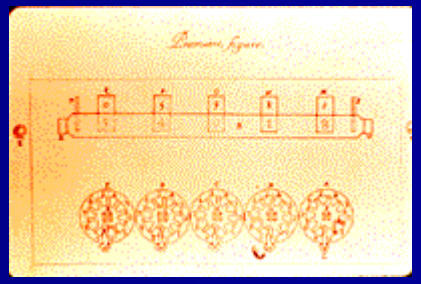
Pythagoras

- The natural world and whole number ratios
- Credited with "discovering" irrational numbers, but that's up for debate
- http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Pythagoras.html

Pascal

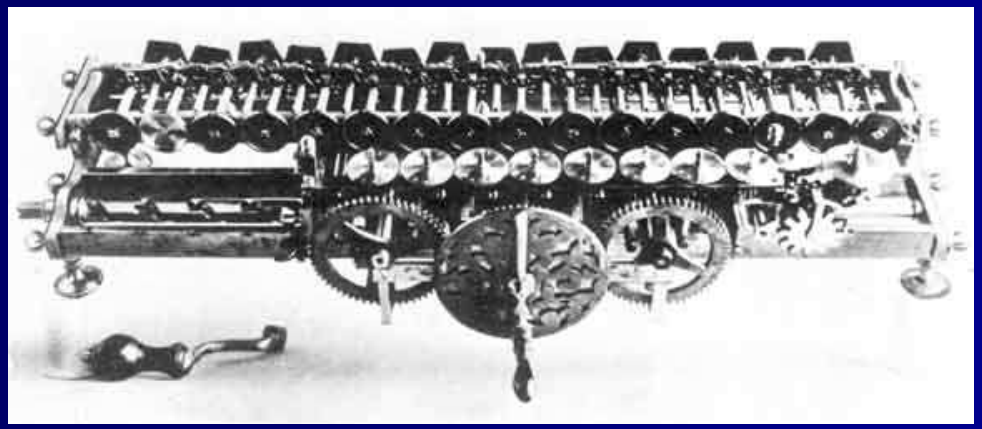
- Many contributions to mathematics
- A primitive adder, like an odomoter, called the "Pascaline"
- Economics limited its appeal (too expensive)
- http://lecture.eingang.org/pascal.html





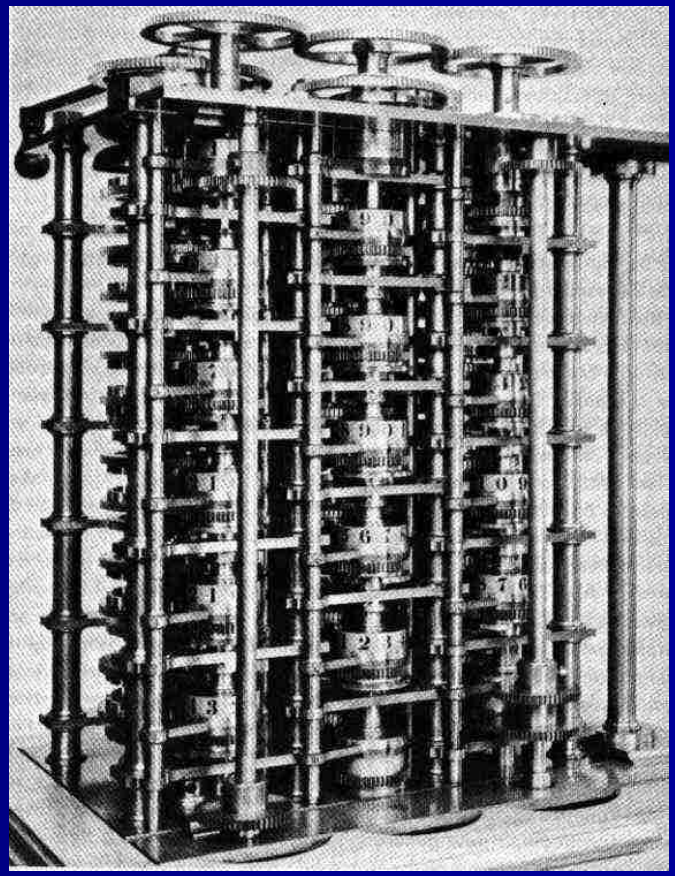
Gottfried Wilhelm von Leibniz

- 1666 as *Dissertatio de arte combinatoria* (Dissertation on the combinatorial art)
- (Mozart/Dice Games) http://www.worldvillage.com/jchuang/Music/Dice/dice.cgi
- Machine below: business success; add, subtract, mult, divide. There was a primitive "language" for programming this device (setting wheel positions)
- http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Leibniz.html



Charles Babbage

- Designed a machine (Difference Engine) capable of performing a calculation many times over (useful for limits, logs, etc)
- Steam Powered; only capable of performing one type of operation without changing the gears on the machine



Charles Babbage (2)

- Analytical Engine – A generic machine (never built) that was capable of reading instructions from punched cards.
- *This is the basis of the modern instruction set/execution architecture*
- Ada Lovelace is credited with designing the language for the analytical engine

Foundations of CS (To be continued)

- David Hilbert – Infinity, is mathematics consistent?
- Kurt Goedel – No formal system is consistent
- Alan Turing – Demonstrated a real problem in a formal system that no machine could solve
- Turing Machines

The Von Neumann Architecture

- The "Shared Program Technique"
  - Hardware need not be fixed to solve any problem. Given a simple hardware design, one can program that hardware to accomplish any task
- "Conditional Control Transfer"
  - Any program can be written with simple IF/THEN decisions plus Jumps (GOTOs)

Modern Systems

- Memory (a program *is* data) to store volatile information
- Hardware that reads the program and manipulates associated data (or itself, like a virus)
- Critical idea: A program *is* data

Figure 2.8: The machine cycle

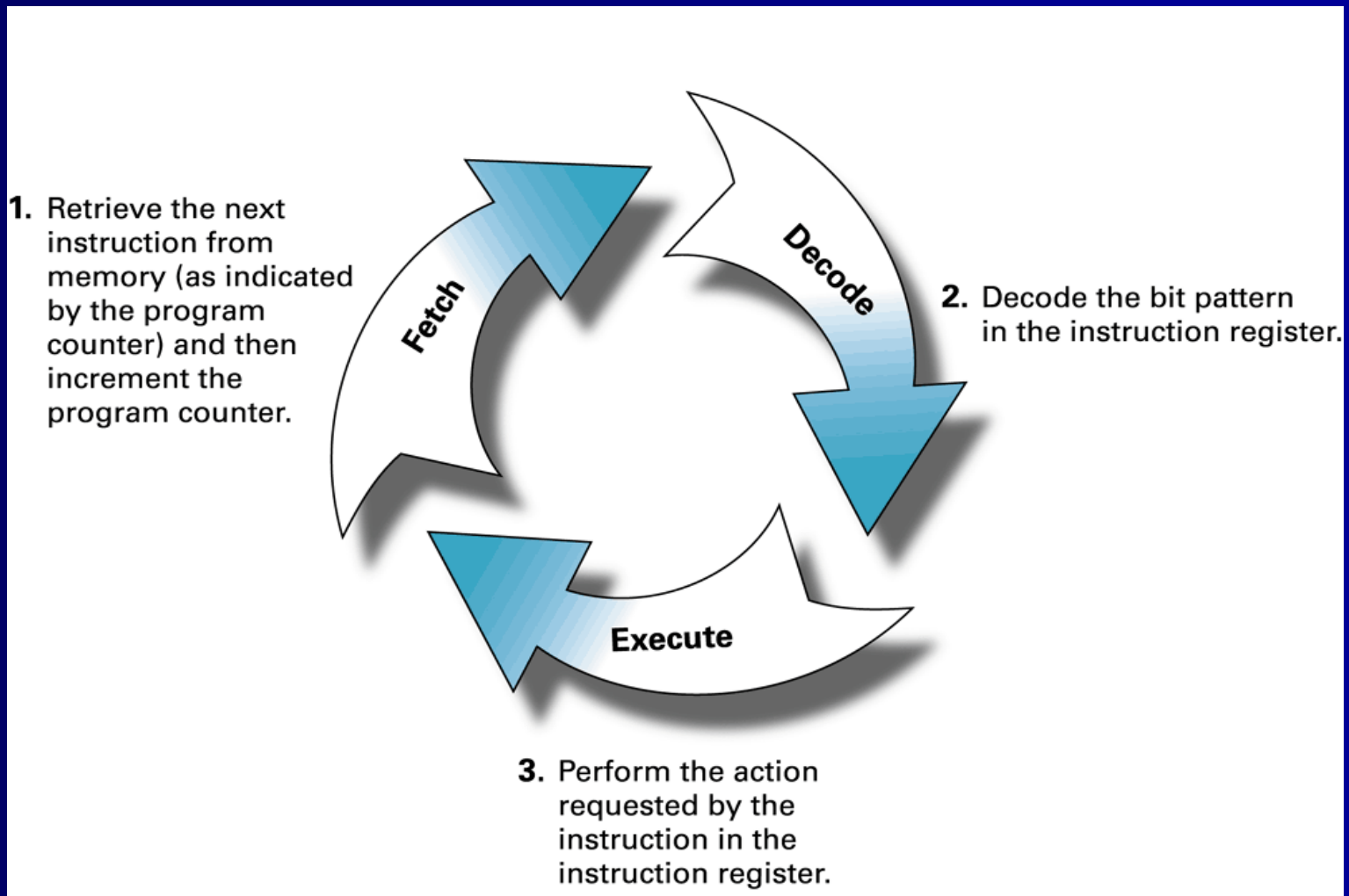


Figure 2.1: CPU and main memory connected via a bus

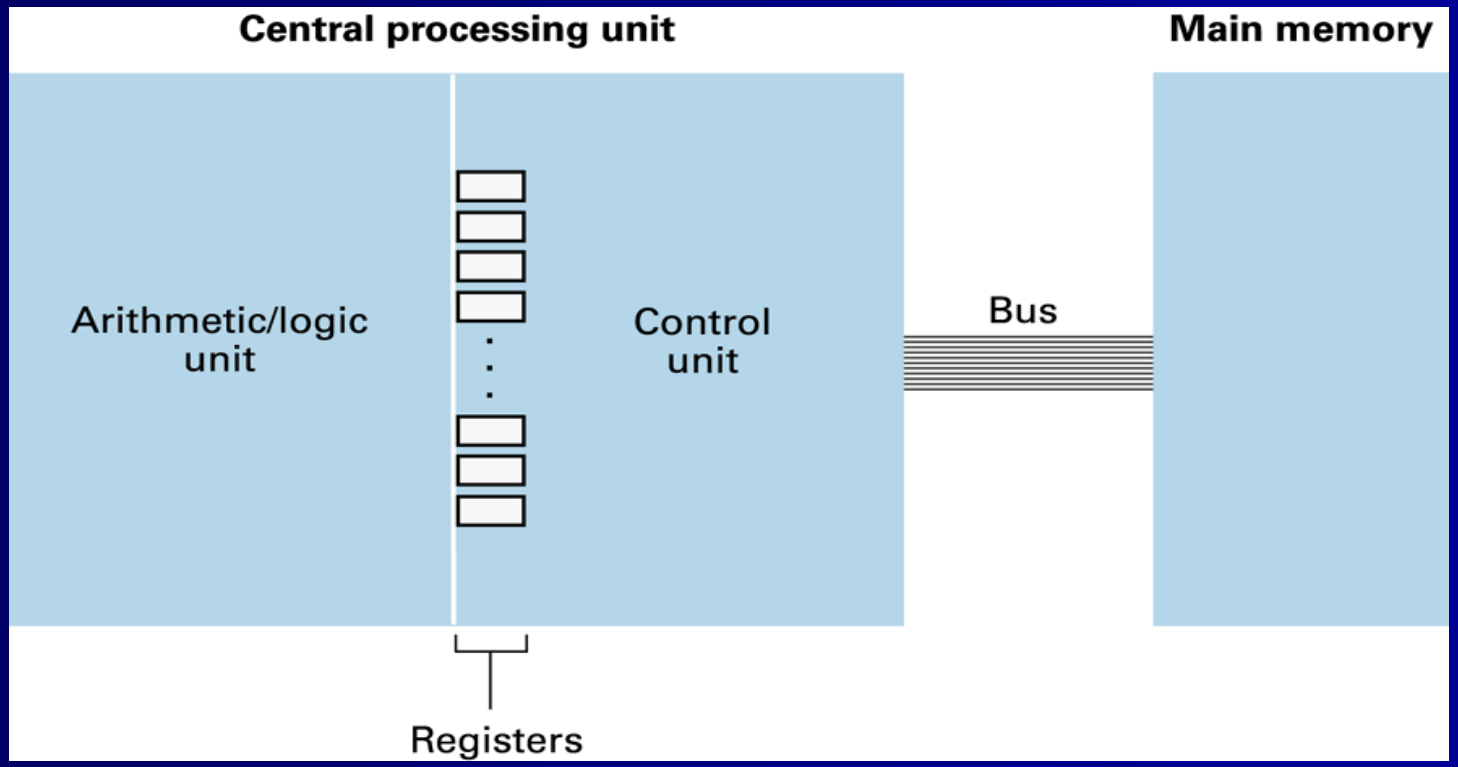


Figure 2.4: The architecture of the machine described in Appendix C

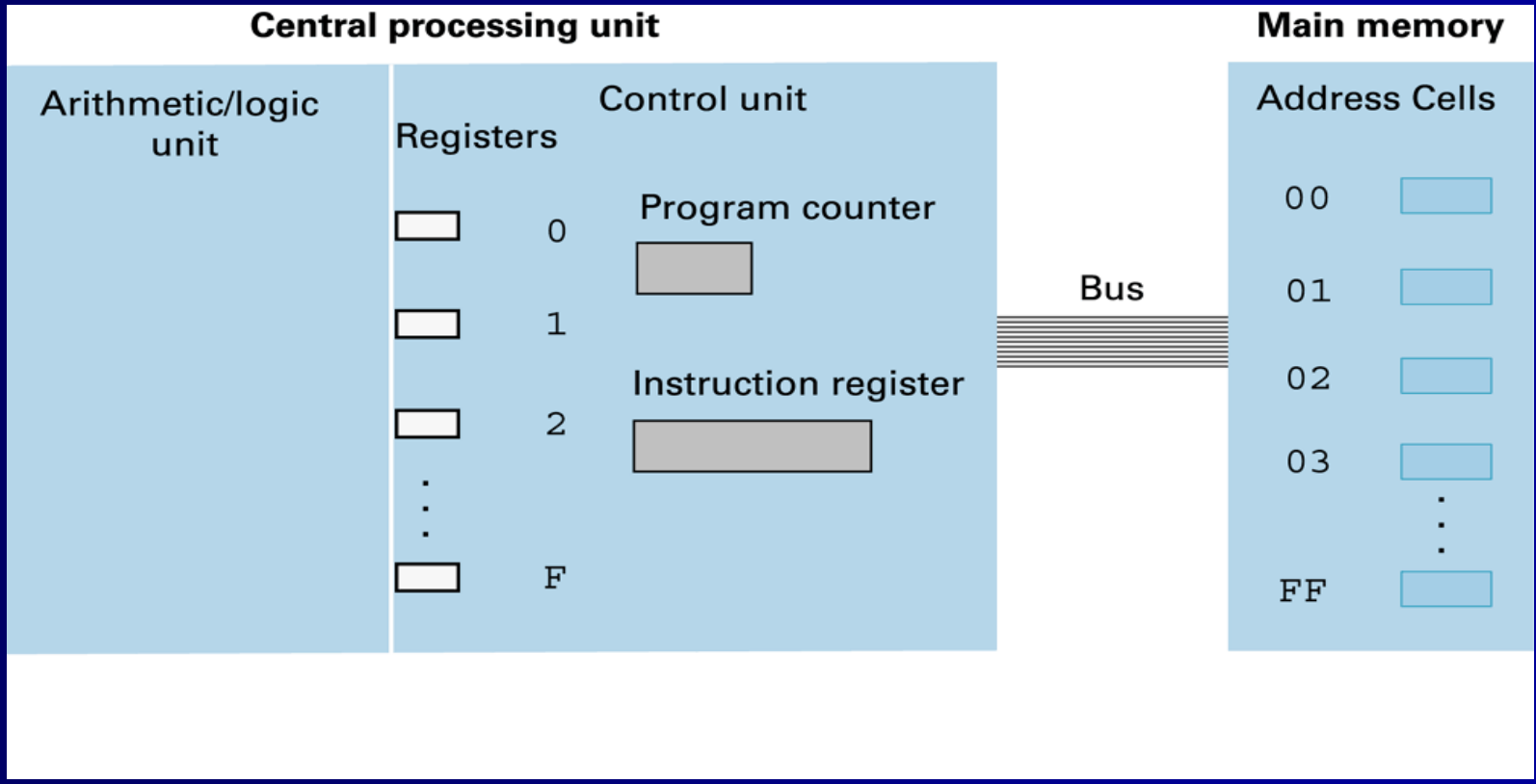


Figure 2.5: The composition of an instruction for the machine in Appendix C

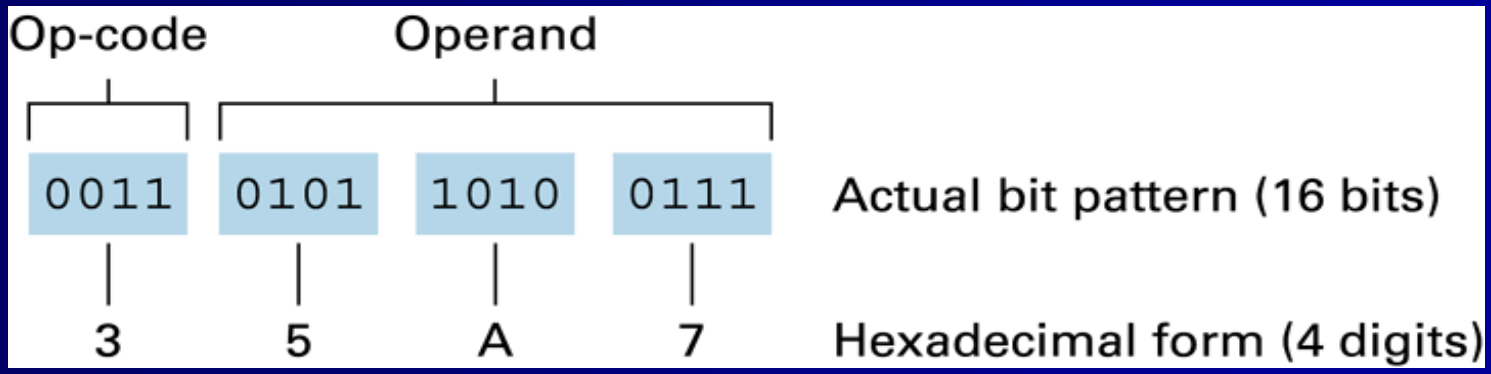


Figure 2.6: Decoding the instruction 35A7

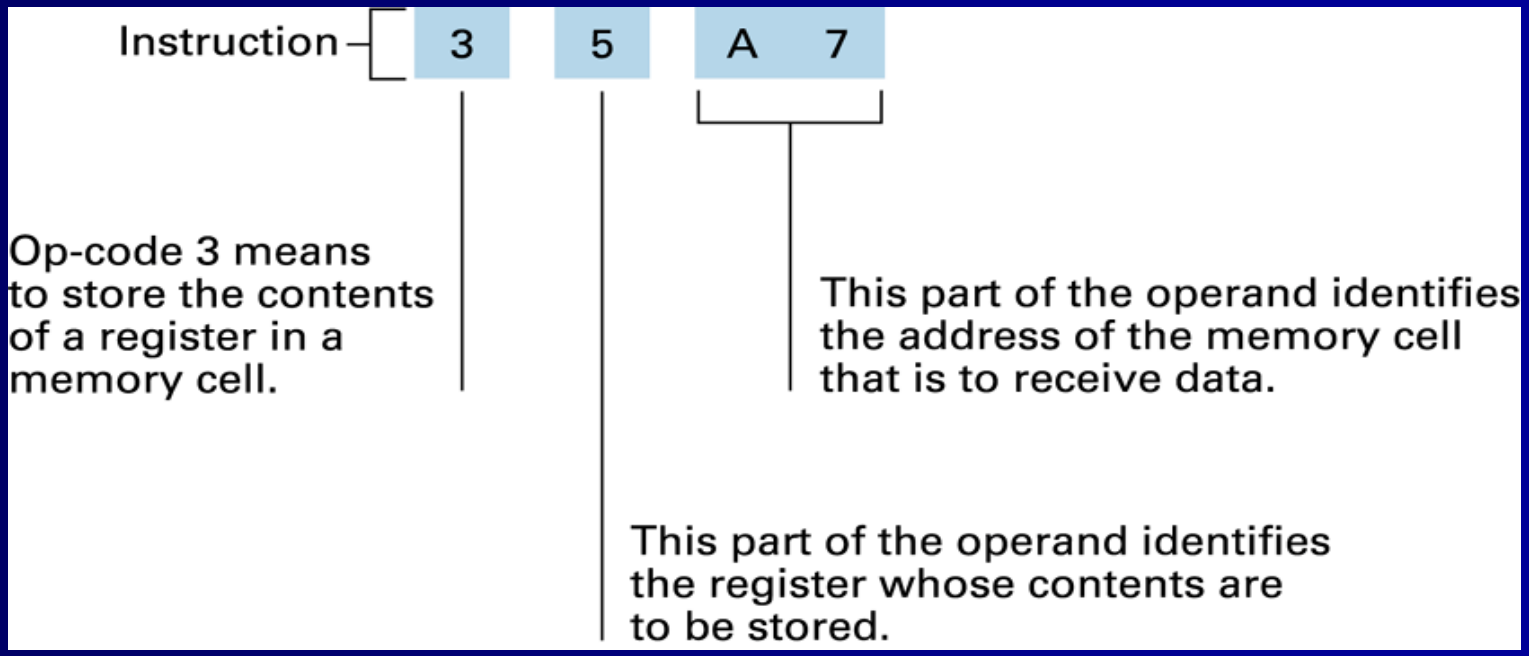


Figure 2.10: The program from Figure 2.7 stored in main memory ready for execution

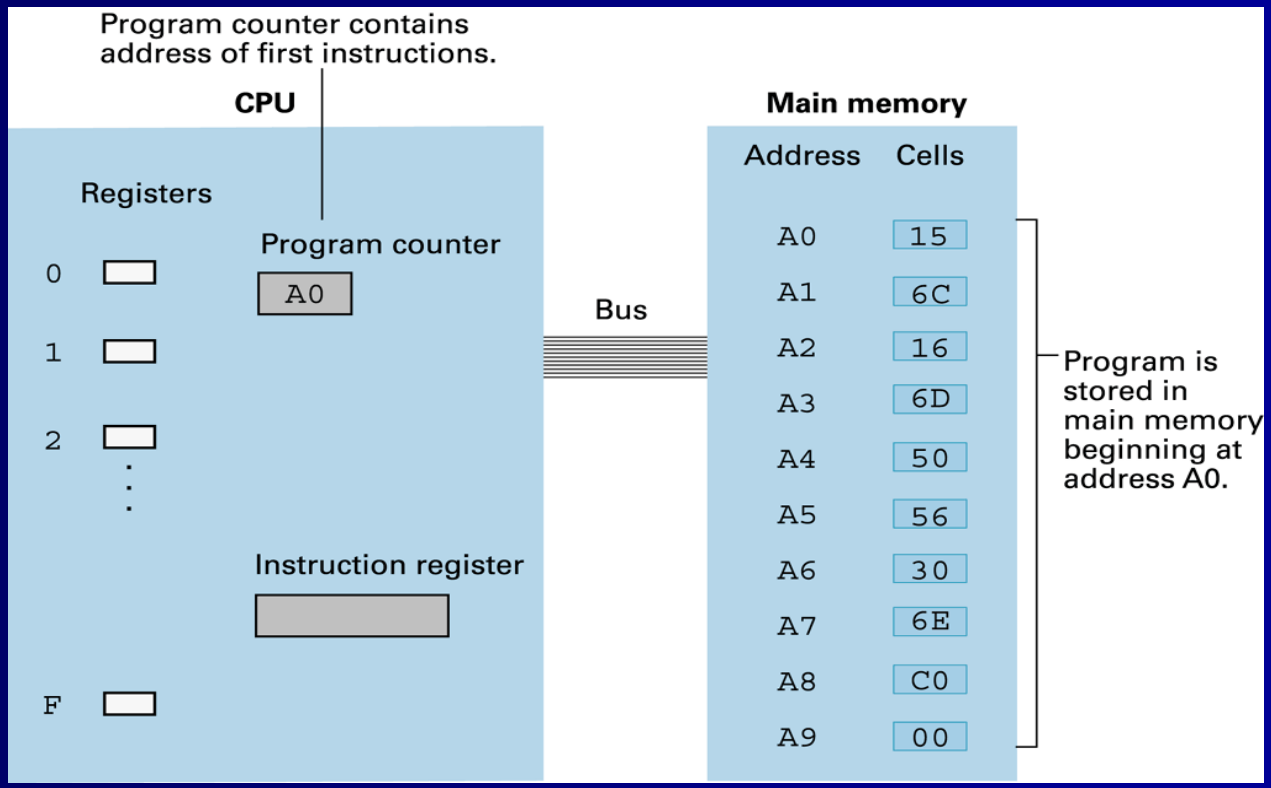
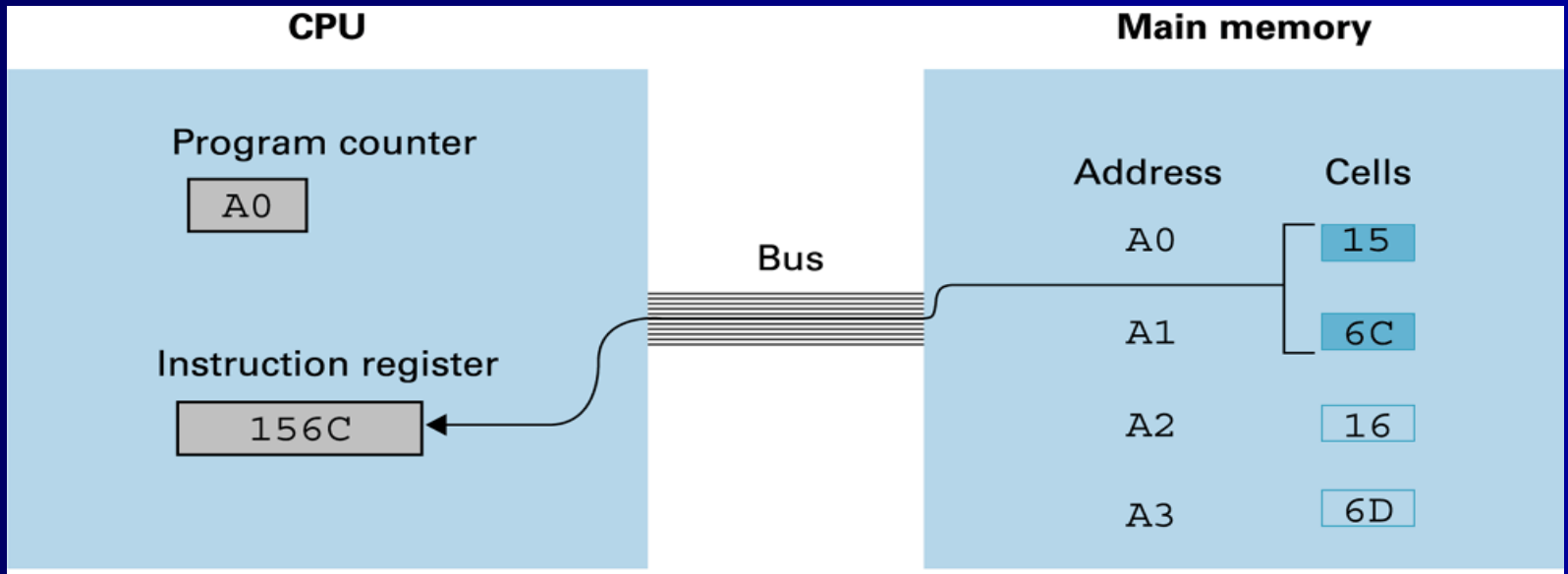
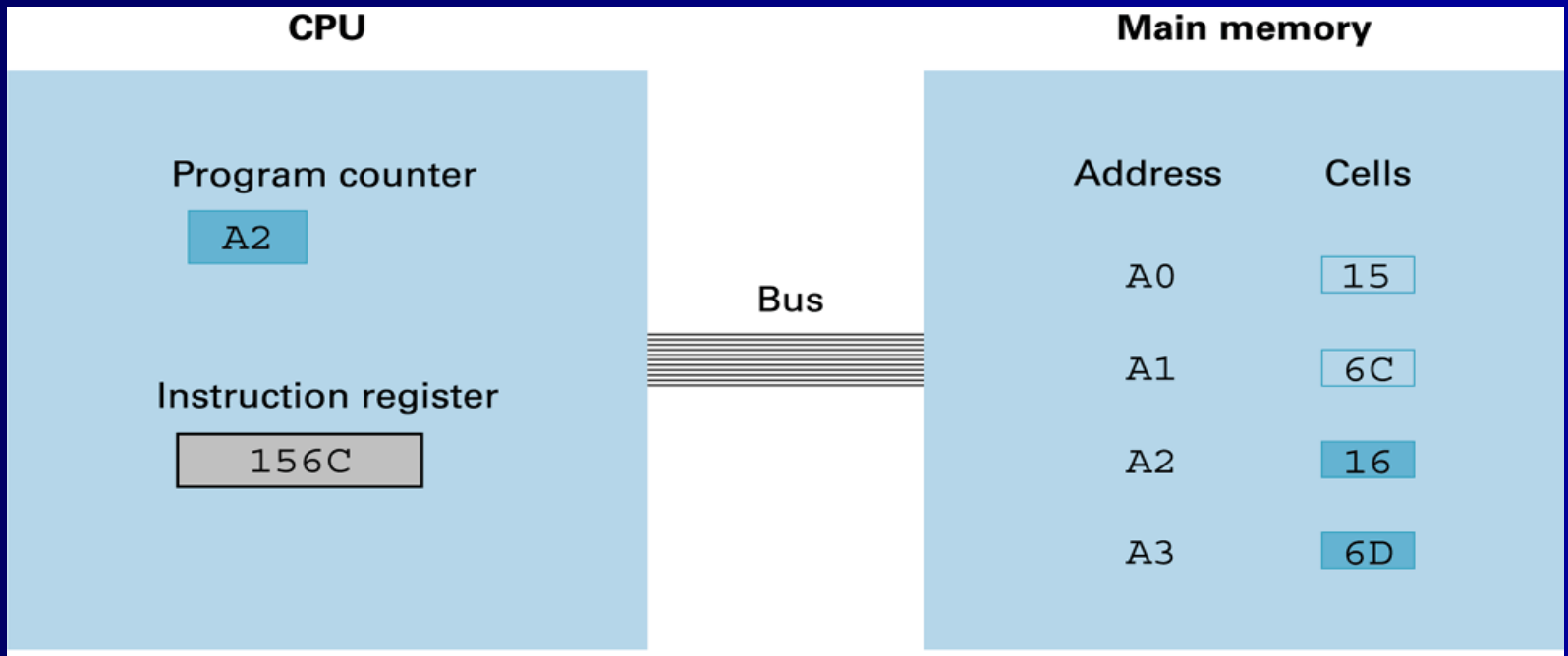


Figure 2.11: Performing the fetch step of the machine cycle (continued)



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

Figure 2.11: Performing the fetch step of the machine cycle



b. Then the program counter is incremented so that it points to the next instruction.

A simple instruction set

- MOVR <id>
- MOVL <id>
- MOVU <id>
- LABEL <labelid>
- GOTO <labelid>

Simulators

- Modern Architectures
  - Intel
  - Apple/Motorola Power
  - ARM
  - Simulation/Emulation

Homework

- The challenge – expressing a process using a fixed number of operations
- The problem:

Two creatures (C1 and C2) are to be parachuted onto random locations on an infinite line. When they land, their parachutes detach and remain where they are. The robots may be programmed from the following instruction set:

Go left one unit (MOVL <C1 or C2>)

Go right one unit (MOVR <C1 or C2>)

Label (LABEL <labelid>)

Skip next instruction unless there is a parachute here (SKIPPAR)

Go to label (GOTO <labelid>)

Each instruction takes one cycle to execute.

Program the robots to collide.