Chapter 2
Data Manipulation

Dr. Farzana Rahman

Assistant Professor

Department of Computer Science

James Madison University

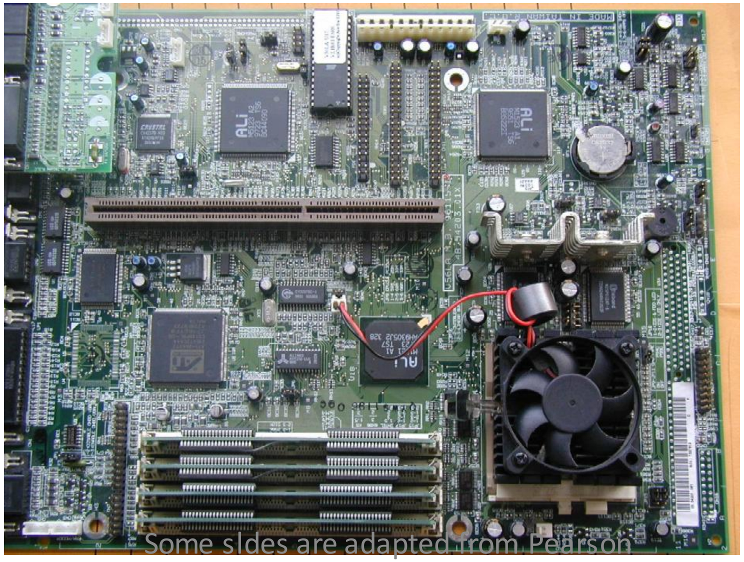
Some slides are adapted from Pearson Education

What the chapter is about?

- 2.1 Computer Architecture
- 2.2 Machine Language
- 2.3 Program Execution
- 2.4 Arithmetic/Logic Instructions
- 2.5 Communicating with Other Devices

Big ideas

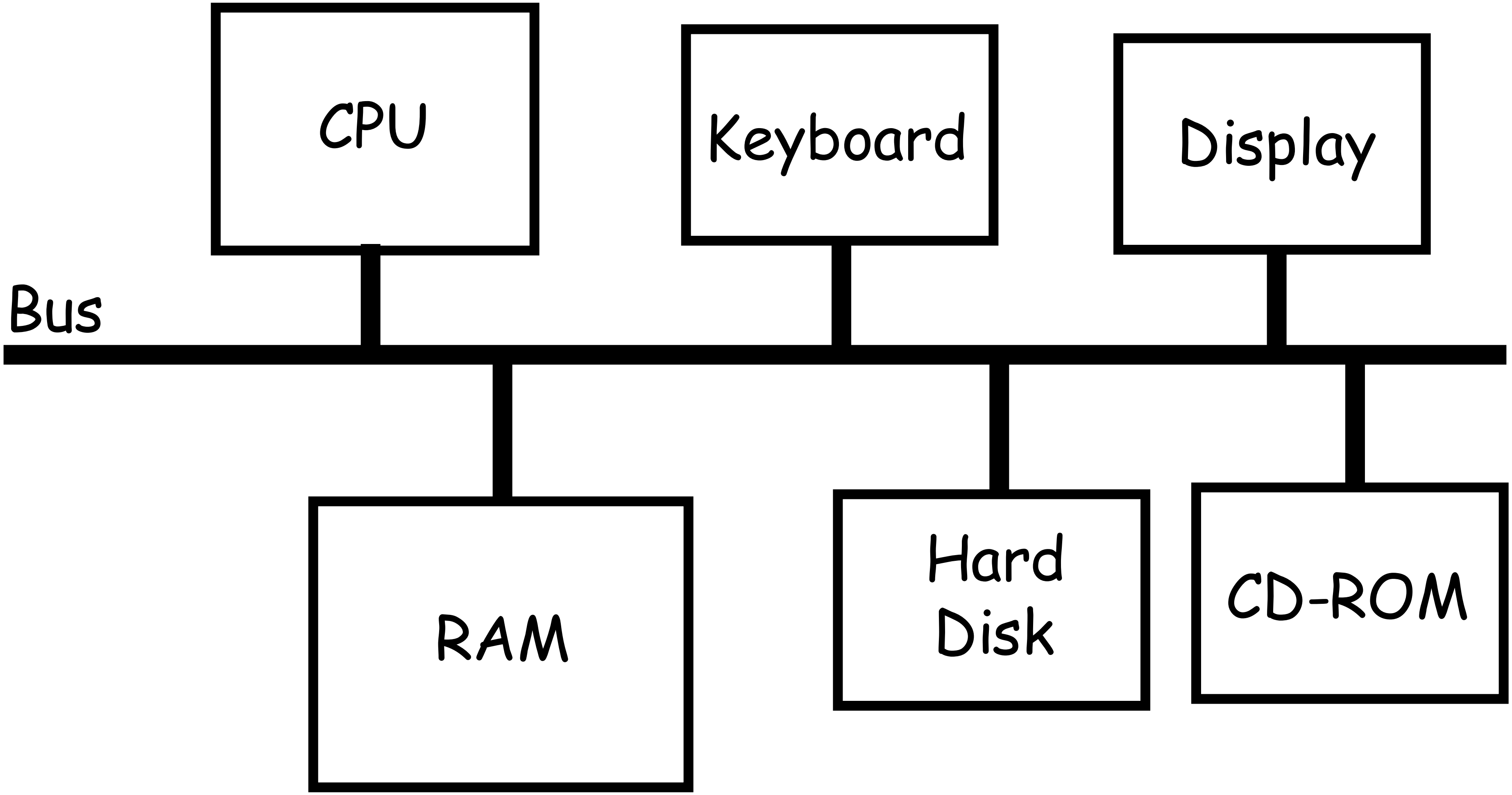
- How computer manipulates data
- What is the basic architecture of computer
- How computer is programmed by means of encoded instructions, i.e. machine language



Computer Architecture

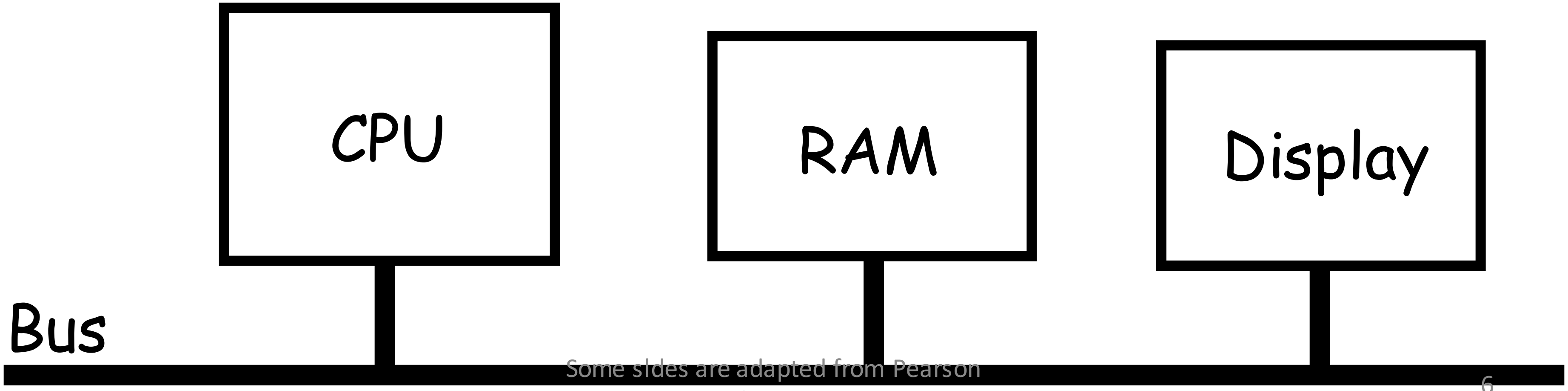
- Central Processing Unit (CPU) or processor
- Bus
- Motherboard
- RAM
- Peripheral devices

Computer Architecture

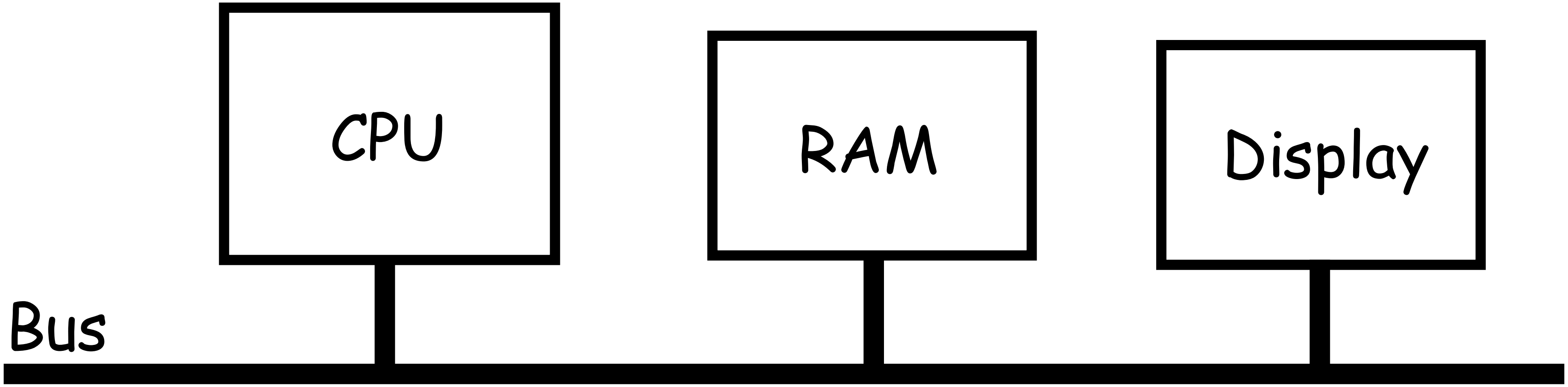


The Bus

- What is a bus?
- It is a simplified way for many devices to communicate to each other.
- Looks like a “highway” for information.
- Actually, more like a “basket” that they all share.

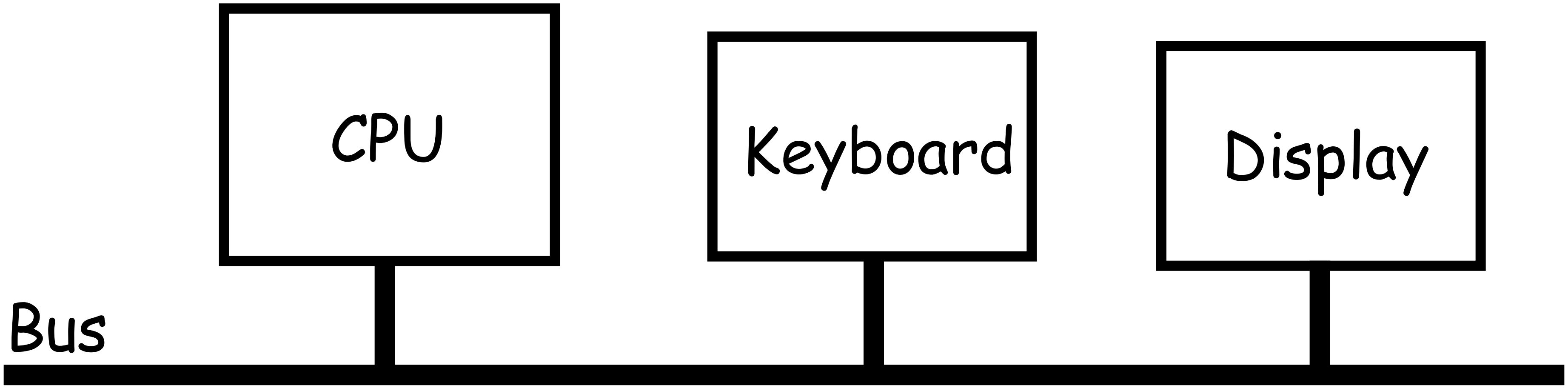


The Bus



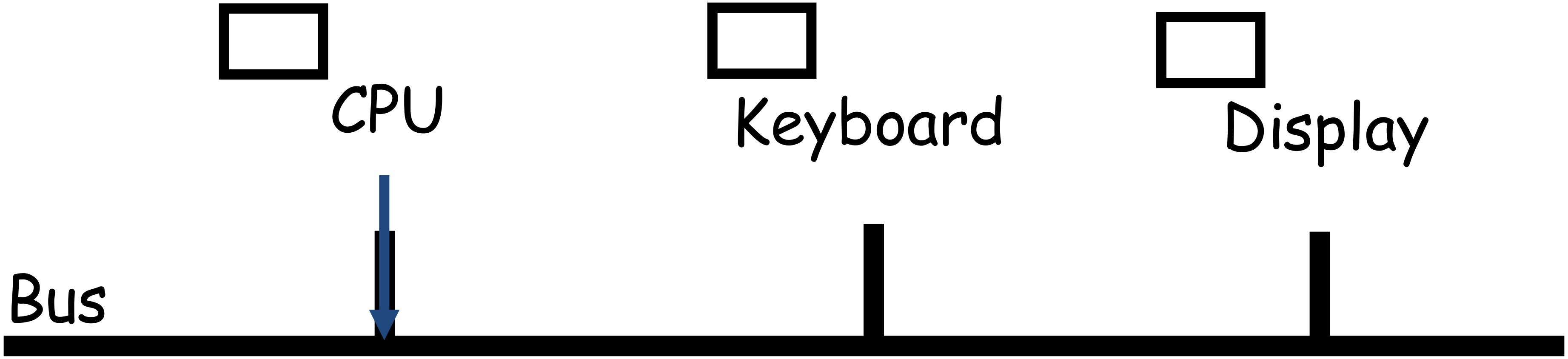
The Bus

- Suppose CPU needs to check to see if the user typed anything.



The Bus

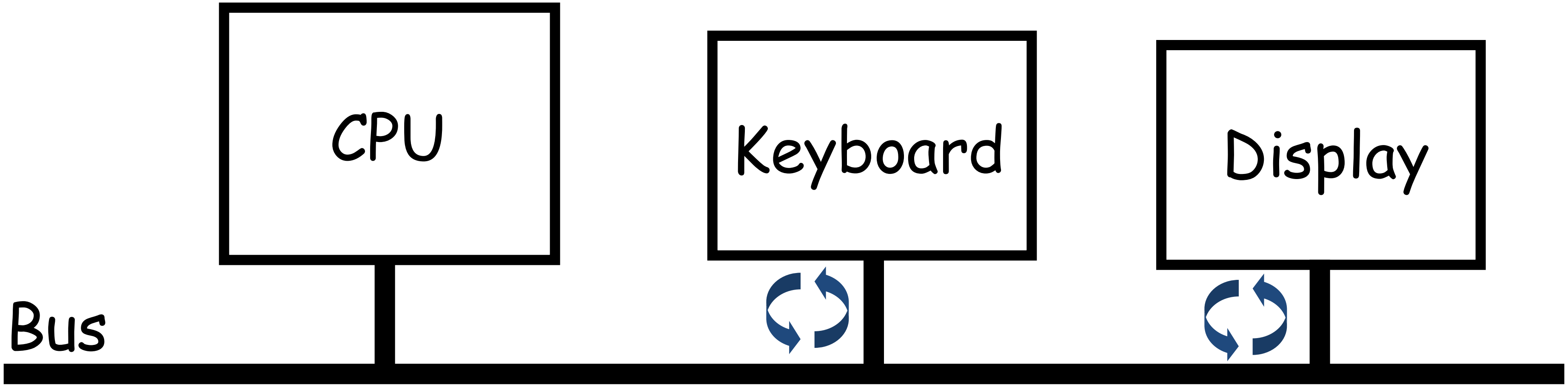
- CPU puts "Keyboard, did the user type anything?" (represented in some way) on the Bus.



"Keyboard, did the user type anything?"

The Bus

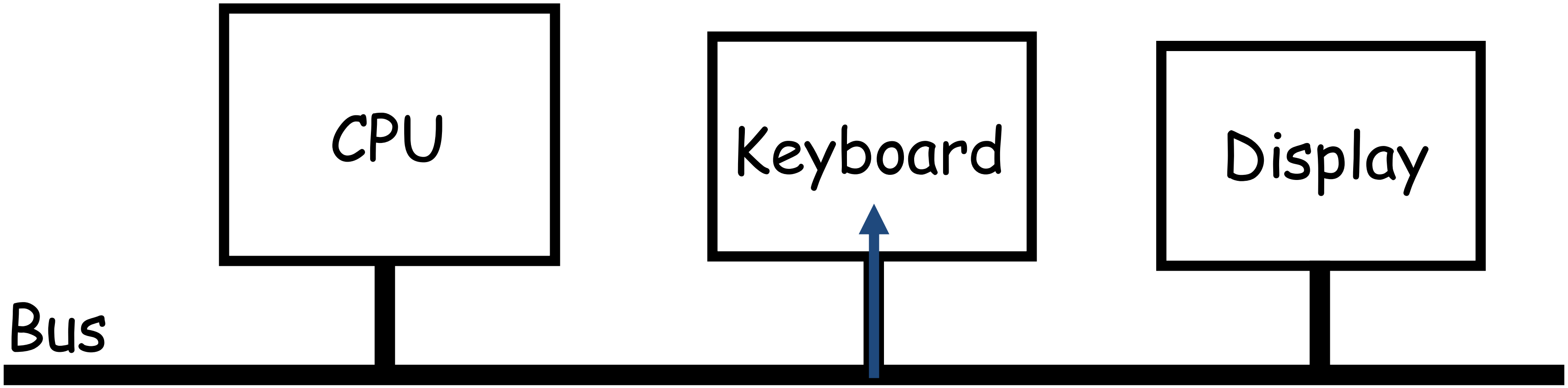
- Each device (except CPU) is a State Machine that constantly checks to see what's on the Bus.



"Keyboard, did the user type anything?"

The Bus

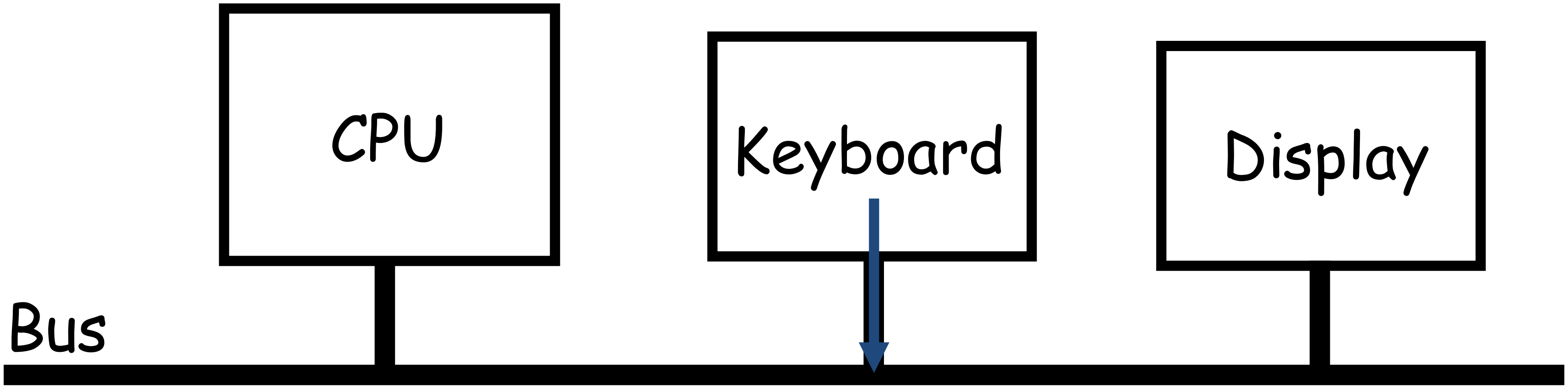
- Keyboard notices that its name is on the Bus, and reads info. Other devices ignore the info.



"Keyboard, did the user type anything?"

The Bus

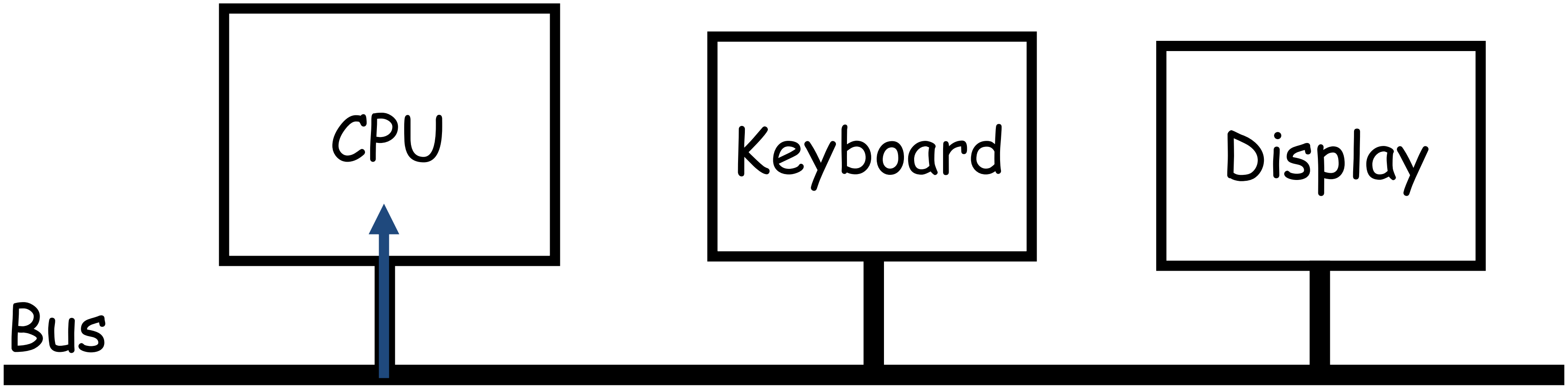
- Keyboard then writes “CPU: Yes, user typed ‘a’.” to Bus.



“CPU: Yes, user typed ‘a’.”

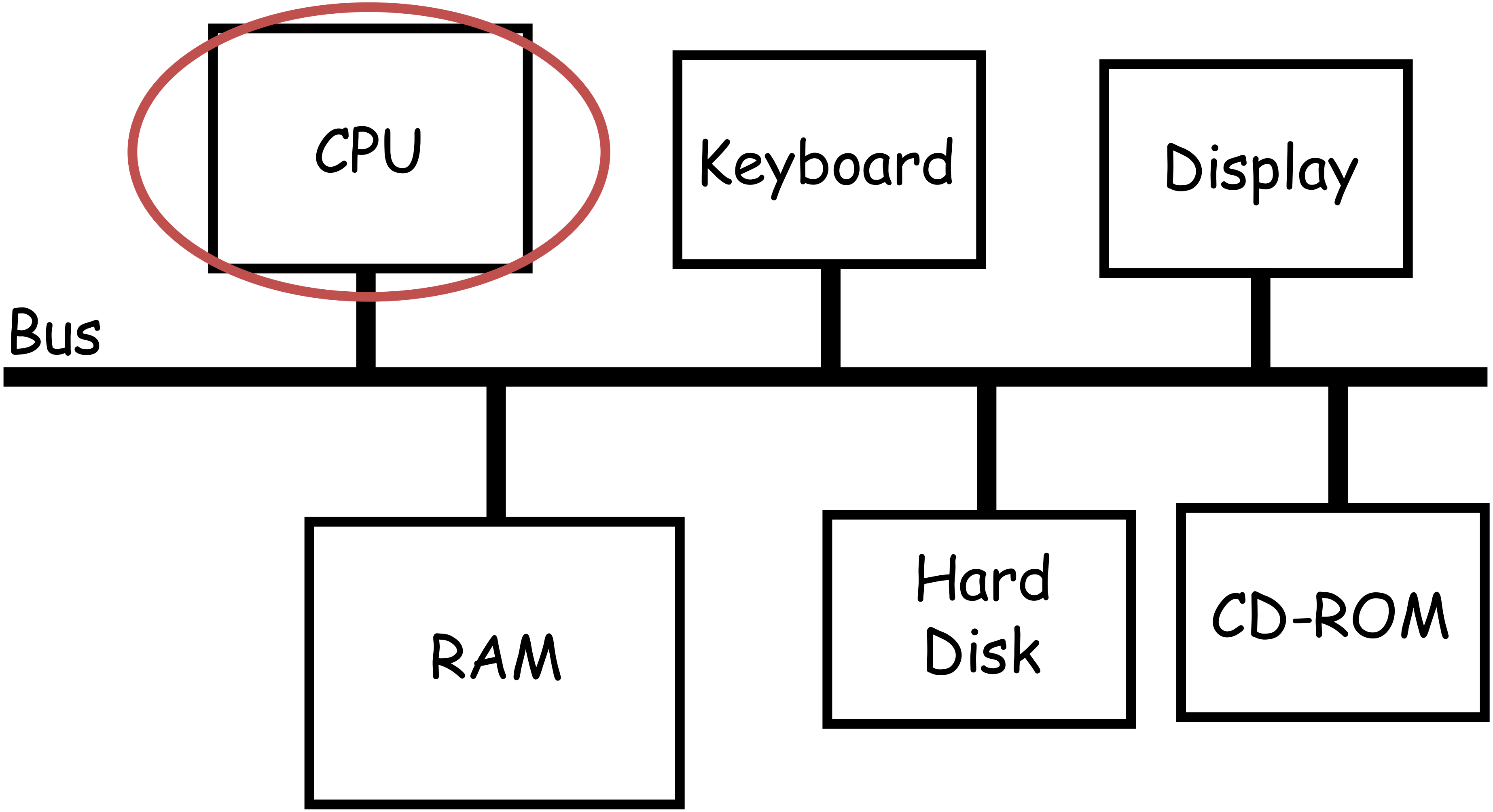
The Bus

- At some point, CPU reads the Bus, and gets the Keyboard’s response.



“CPU: Yes, user typed ‘a’.”

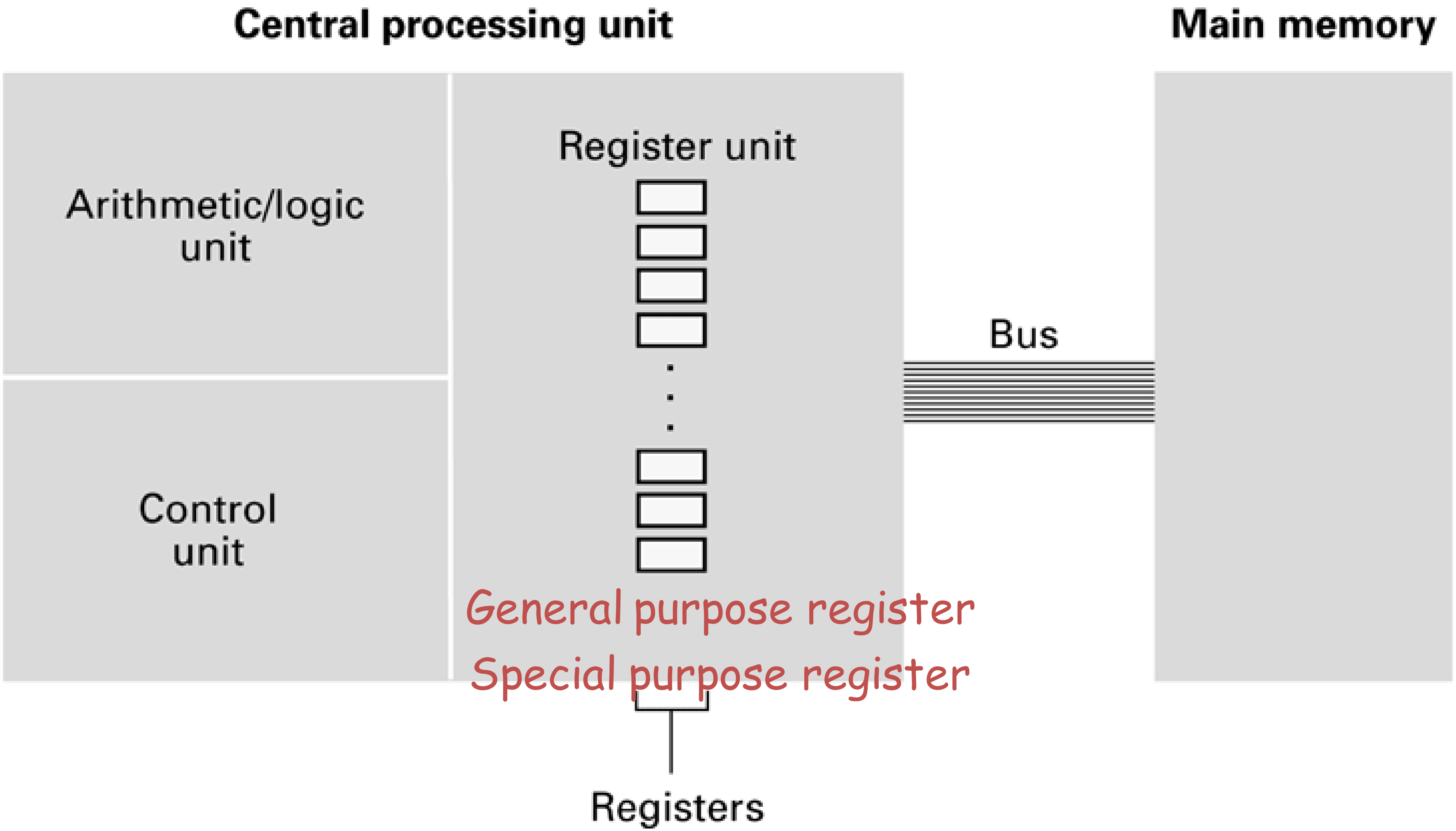
Computer Architecture



Inside the CPU

- The CPU is the brain of the computer.
- It is the part that actually executes the instructions.
- Let's take a look inside.

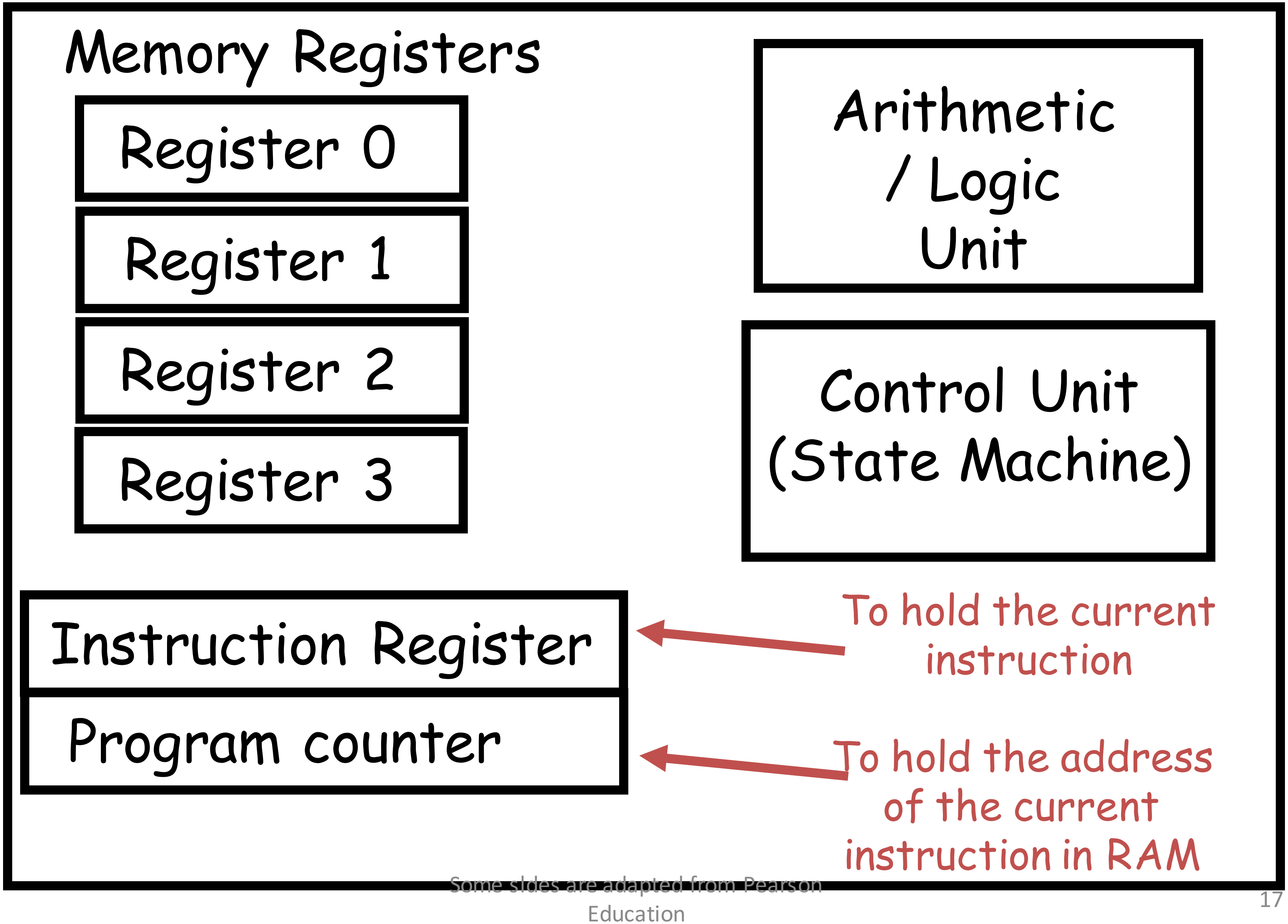
CPU and main memory connected via a bus



General purpose register

Special purpose register

Inside the CPU (cont.)



Memory Registers

Register 0

Register 1

Register 2

Register 3

Arithmetic / Logic Unit

Control Unit (State Machine)

Instruction Register

Program counter

To hold the current instruction

To hold the address of the current instruction in RAM

Stored Program Concept

- A program can be encoded as bit patterns and stored in main memory.
- From there, the CPU can then extract the instructions and execute them.
- In turn, the program to be executed can be altered easily.

Terminology

- **Machine instruction:** An instruction (or command) encoded as a bit pattern recognizable by the CPU

- **Machine language:** The set of all instructions recognized by a machine

Machine Language Philosophies

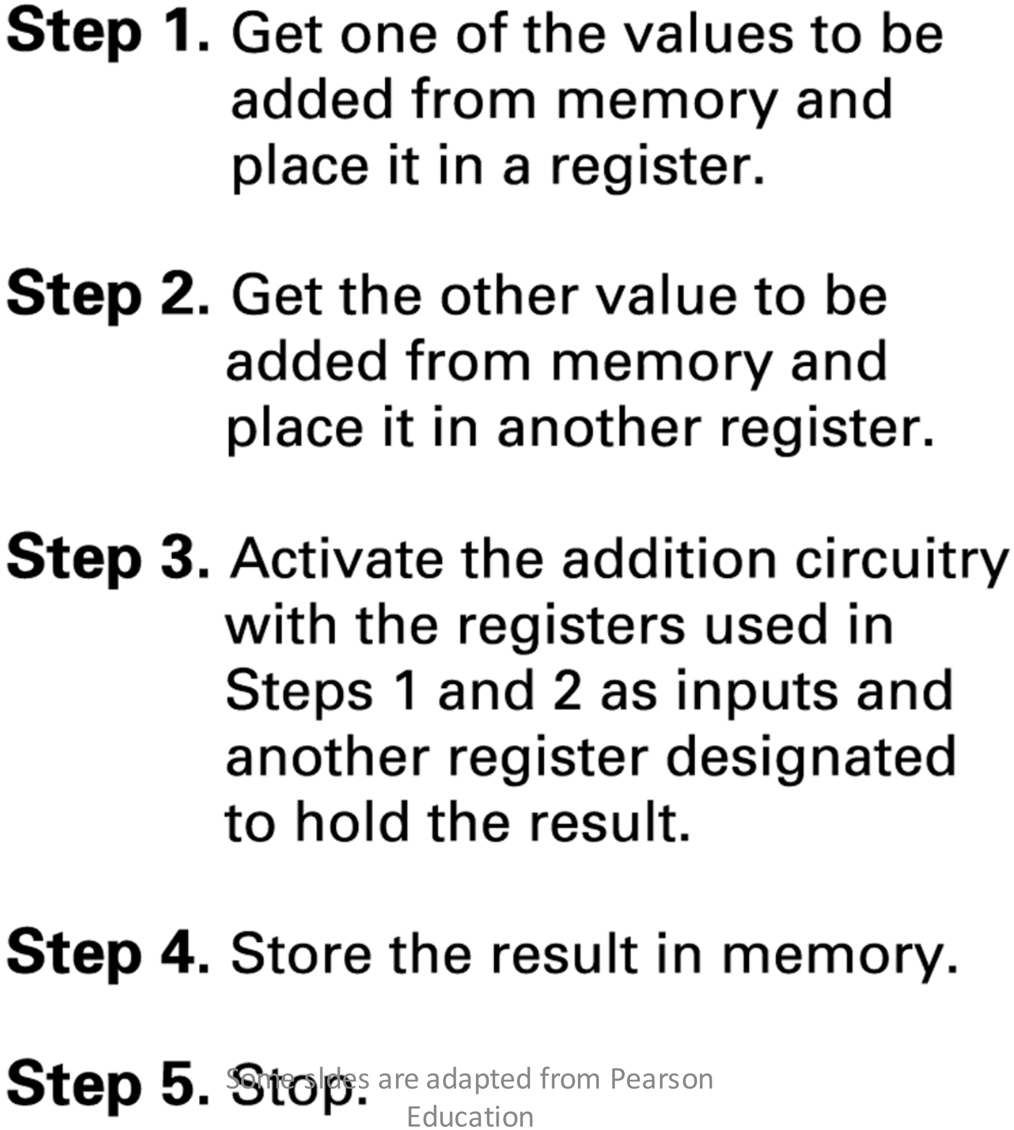
- Reduced Instruction Set Computing (RISC)
  - Few, simple, efficient, and fast instructions
  - Examples: PowerPC from Apple/IBM/Motorola and ARM

- Complex Instruction Set Computing (CISC)
  - Many, convenient, and powerful instructions
  - Example: Intel

Machine Instruction Types

- **Data Transfer:** copy data from one location to another
- **Arithmetic/Logic:** use existing bit patterns to compute a new bit patterns
- **Control:** direct the execution of the program

Adding values stored in memory



Step 1. Get one of the values to be added from memory and place it in a register.

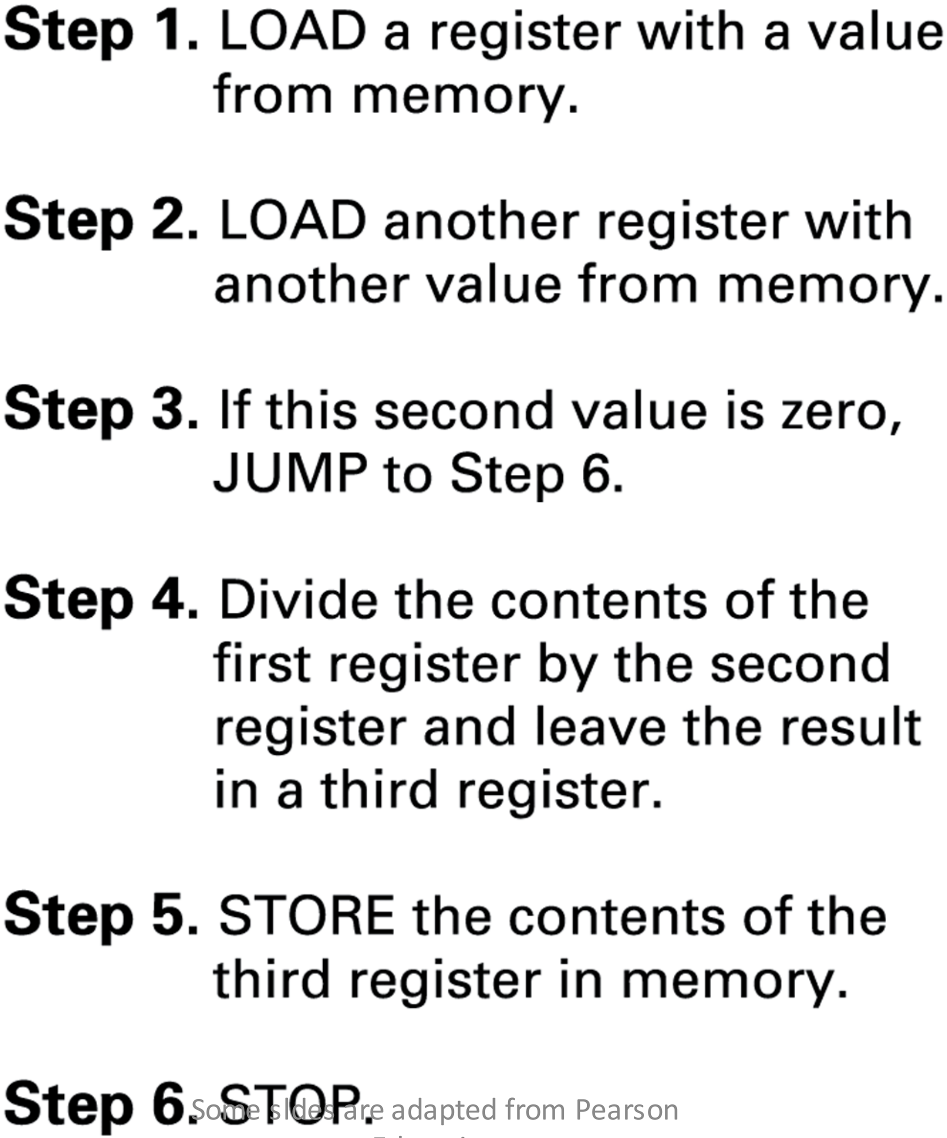
Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

Step 5. Stop.

Dividing values stored in memory



Step 1. LOAD a register with a value from memory.

Step 2. LOAD another register with another value from memory.

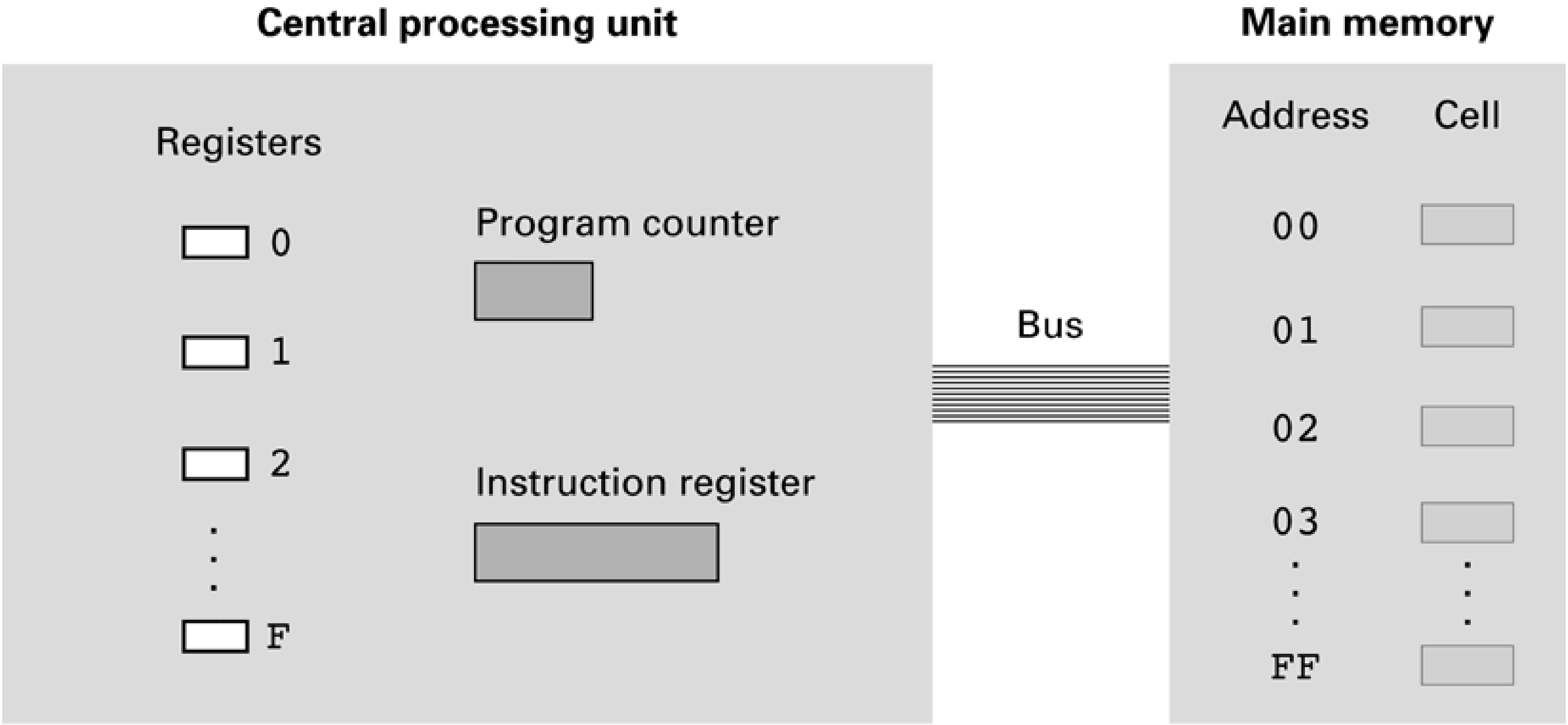
Step 3. If this second value is zero, JUMP to Step 6.

Step 4. Divide the contents of the first register by the second register and leave the result in a third register.

Step 5. STORE the contents of the third register in memory.

Step 6. STOP.

The architecture of the machine described in Appendix C



Parts of a Machine Instruction

- **Op-code:** Specifies which operation to execute
- **Operand:** Gives more detailed information about the operation
  - Interpretation of operand varies depending on op-code

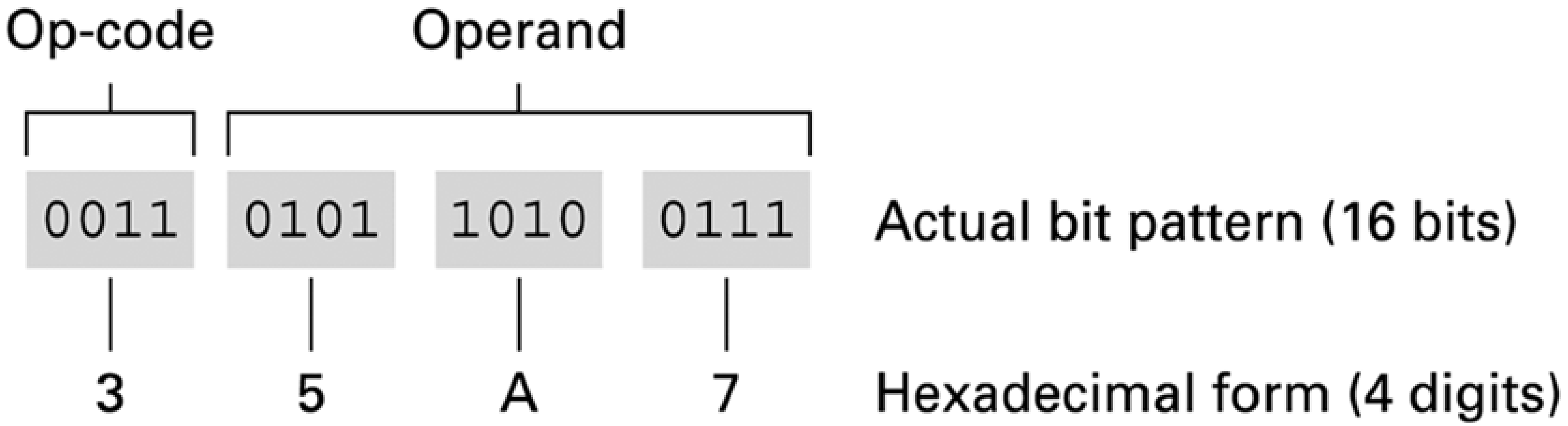
Some sldes are adapted from Pearson Education

Typical Assembly Instructions

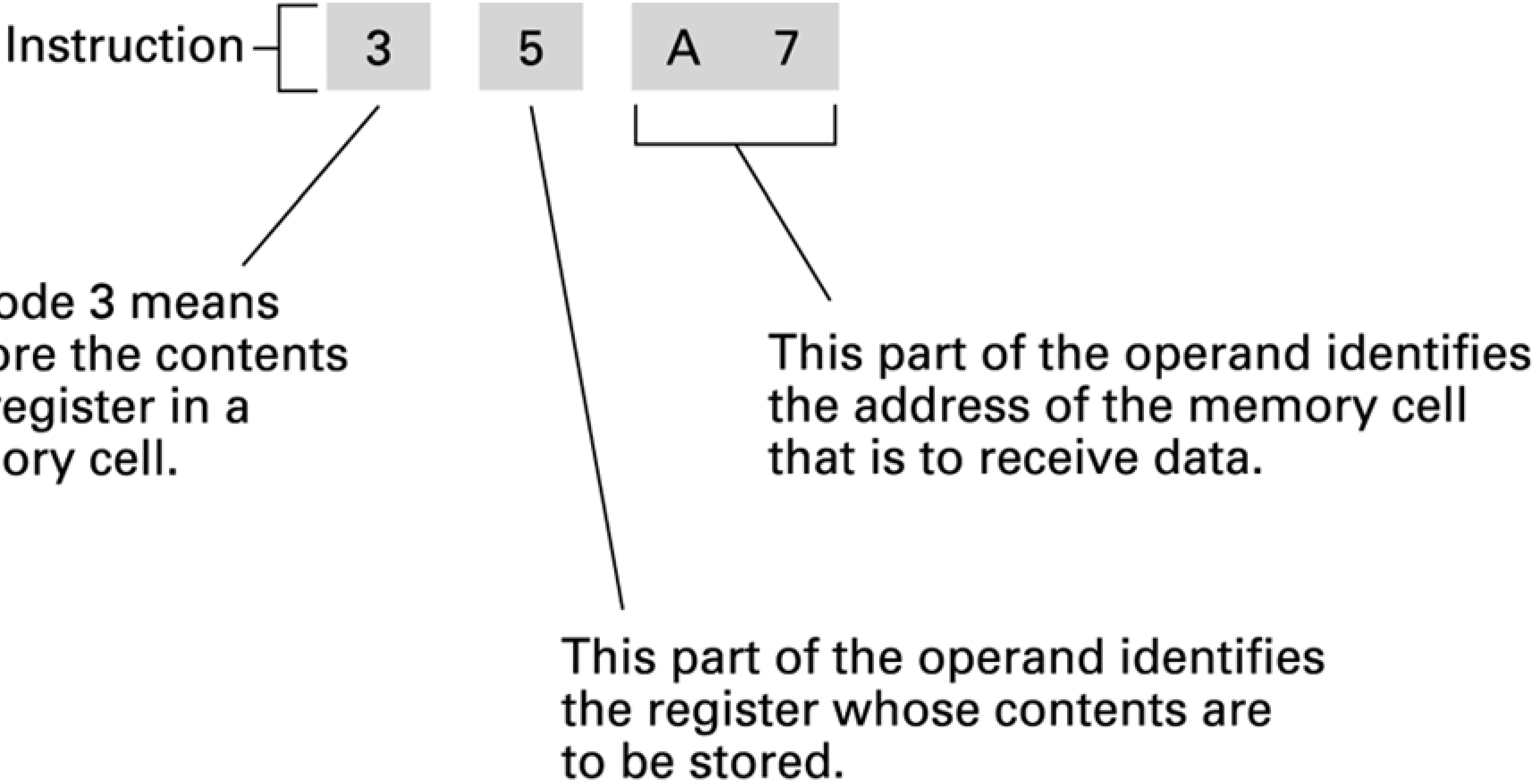
1. LOAD
2. LOAD directly
3. STORE
4. MOVE
5. ADD
6. ADD (floating point)
7. OR
8. AND
9. XOR
10. ROTATE
11. JMUP
12. HALT

Some sldes are adapted from Pearson Education

The composition of an instruction for the machine in Appendix C



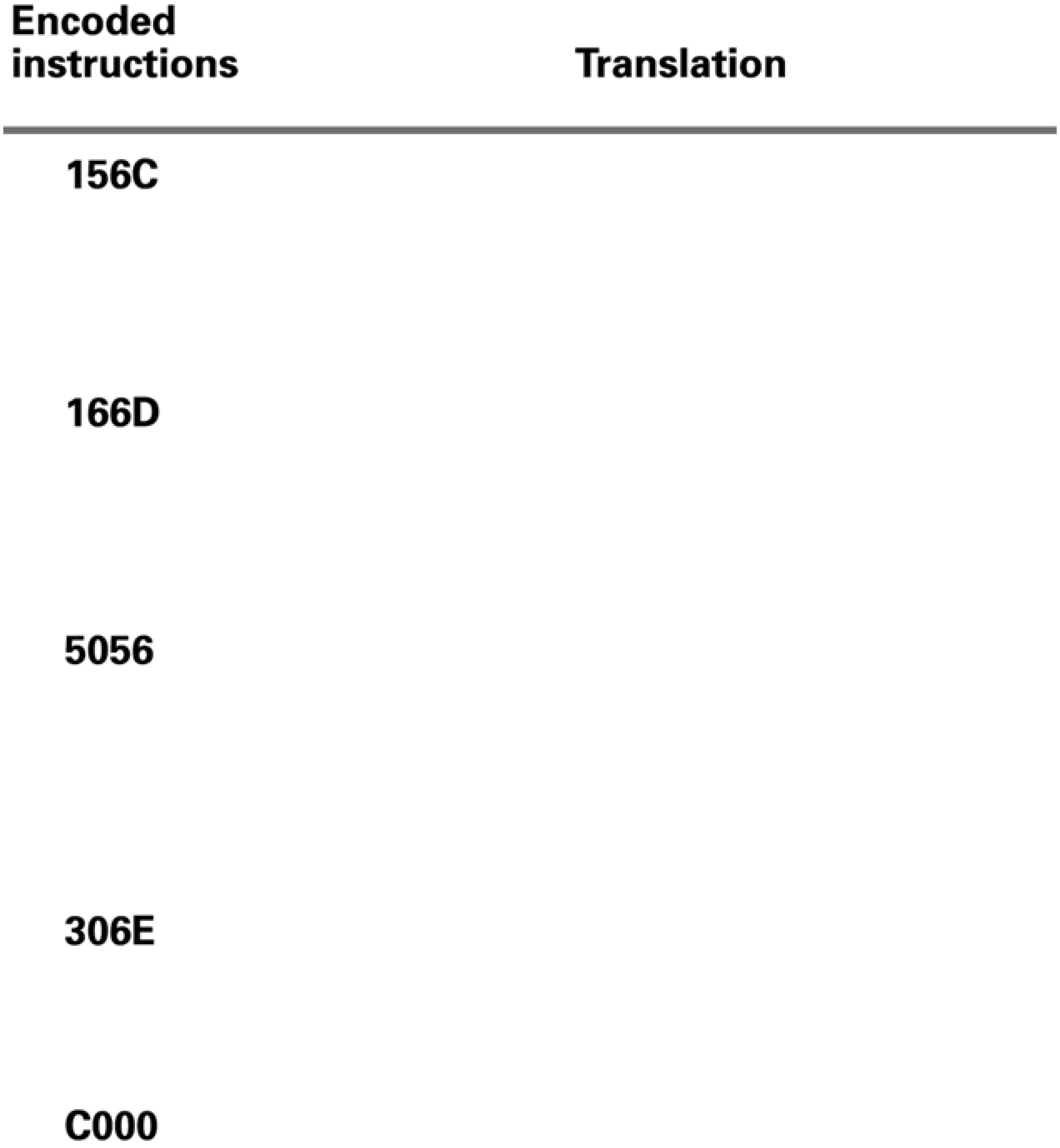
Decoding the instruction 35A7



Op-code 3 means to store the contents of a register in a memory cell.

This part of the operand identifies the address of the memory cell that is to receive data.

This part of the operand identifies the register whose contents are to be stored.



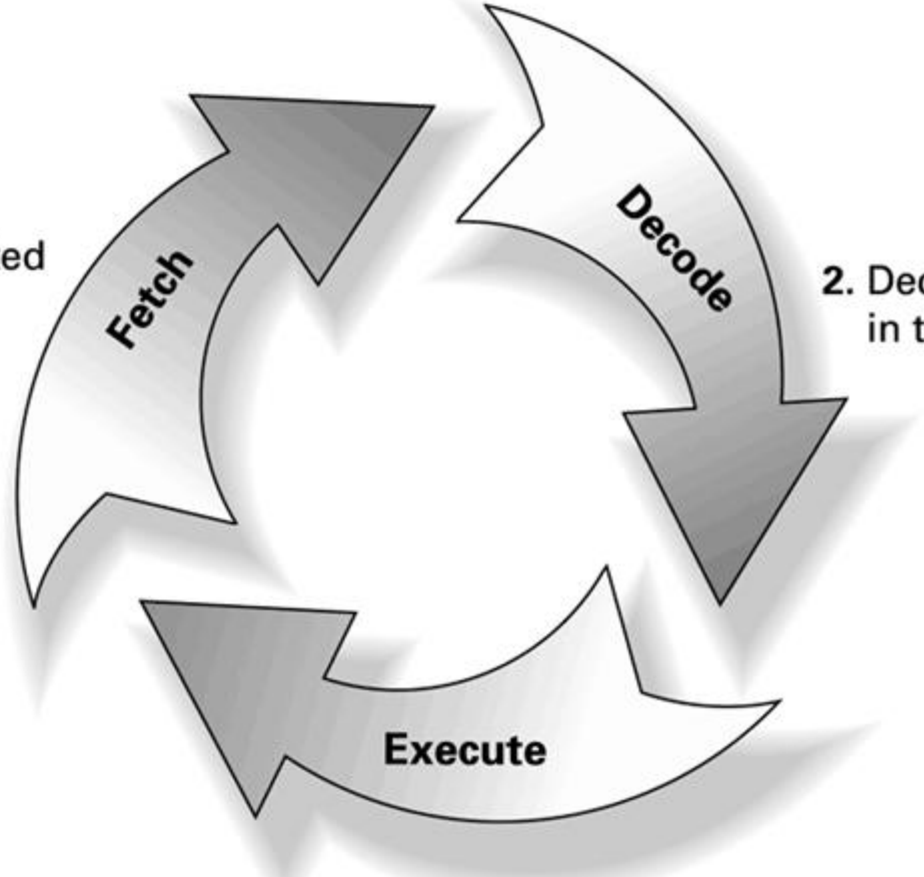
Encoded instructions	Translation
156C	
166D	
5056	
306E	
C000	

Program Execution

- Controlled by two special-purpose registers
  - **Program counter:** address of next instruction
  - **Instruction register:** current instruction

- Machine Cycle
  - Fetch
  - Decode
  - Execute

The machine cycle

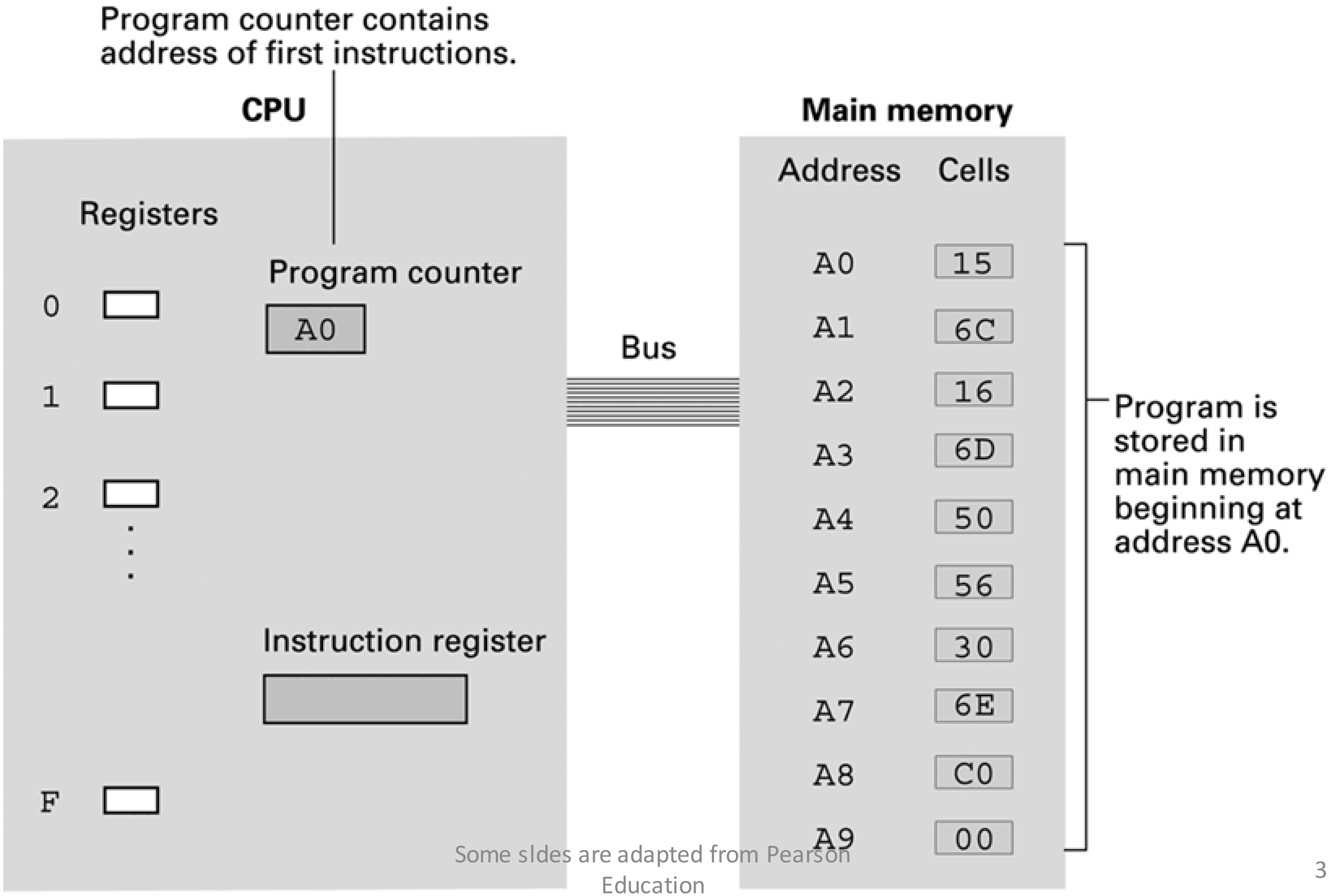


1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.

2. Decode the bit pattern in the instruction register.

3. Perform the action required by the instruction in the instruction register.

The program stored in main memory ready for execution



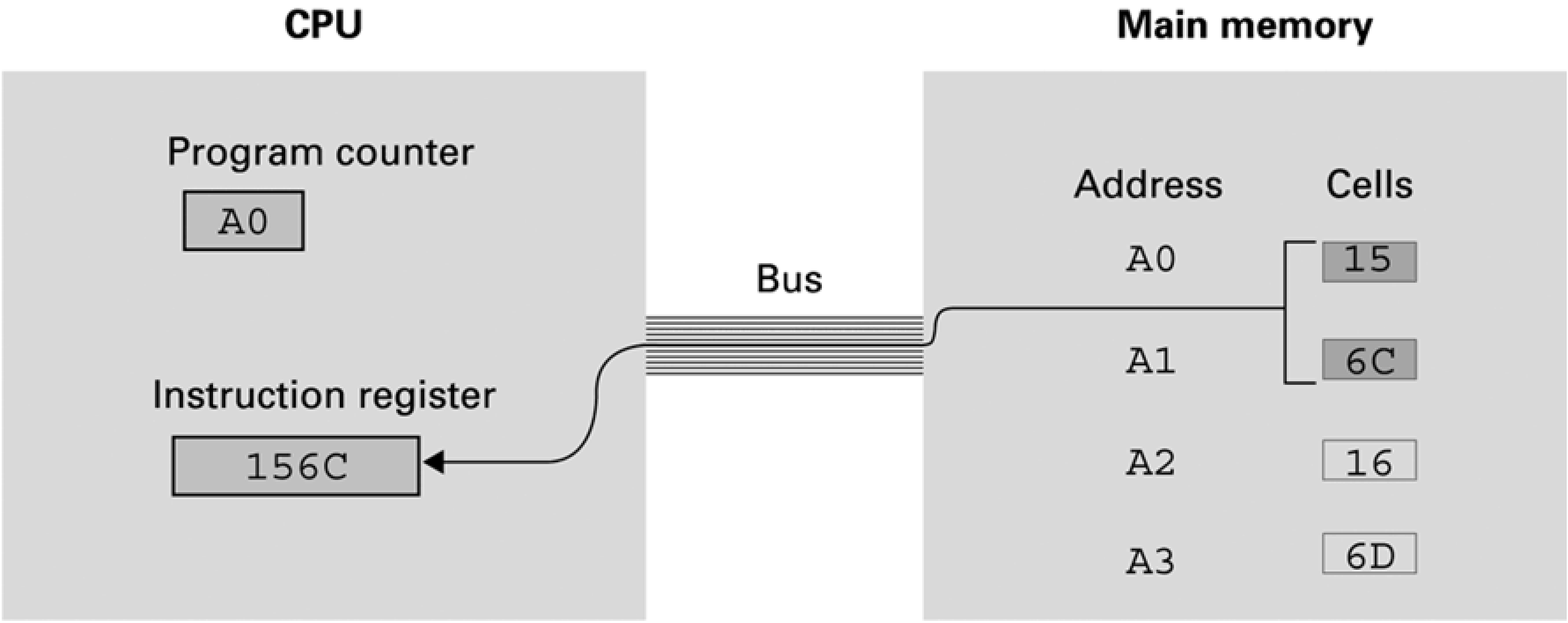
Program counter contains address of first instructions.

Address	Cells
A0	15
A1	6C
A2	16
A3	6D
A4	50
A5	56
A6	30
A7	6E
A8	C0
A9	00

Program is stored in main memory beginning at address A0.

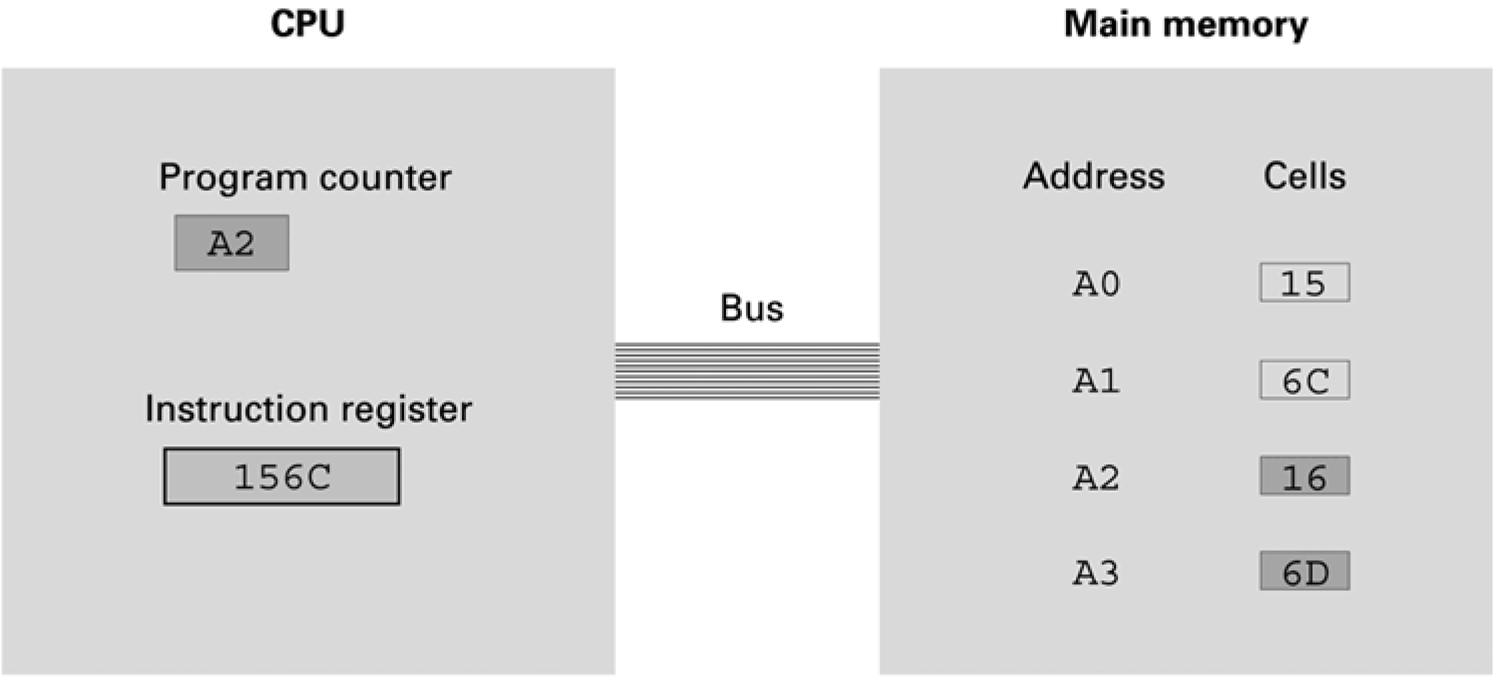
Some sldes are adapted from Pearson Education

Performing the fetch step of the machine cycle



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

Performing the fetch step of the machine cycle (cont'd)



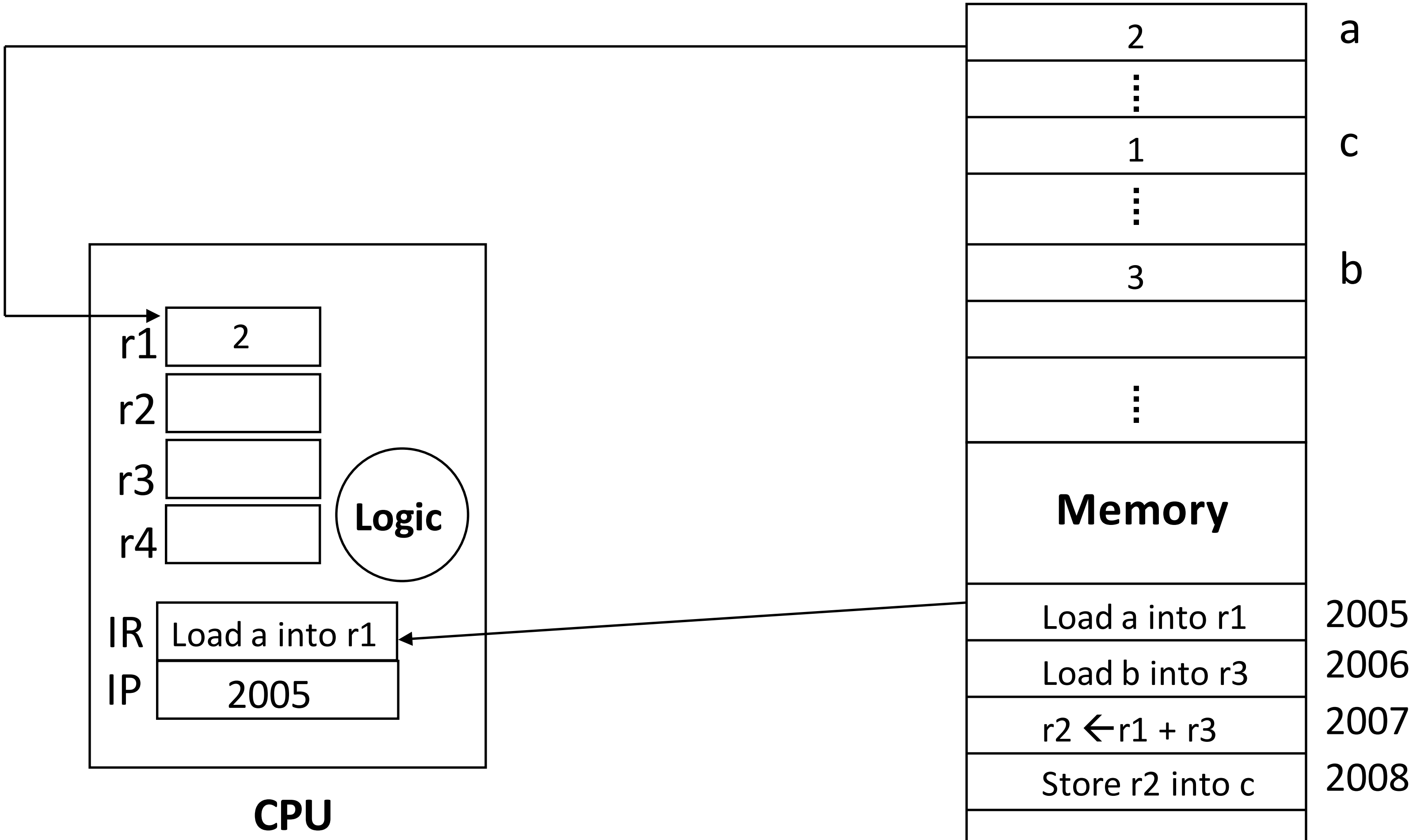
b. Then the program counter is incremented so that it points to the next instruction.

A Simple Program

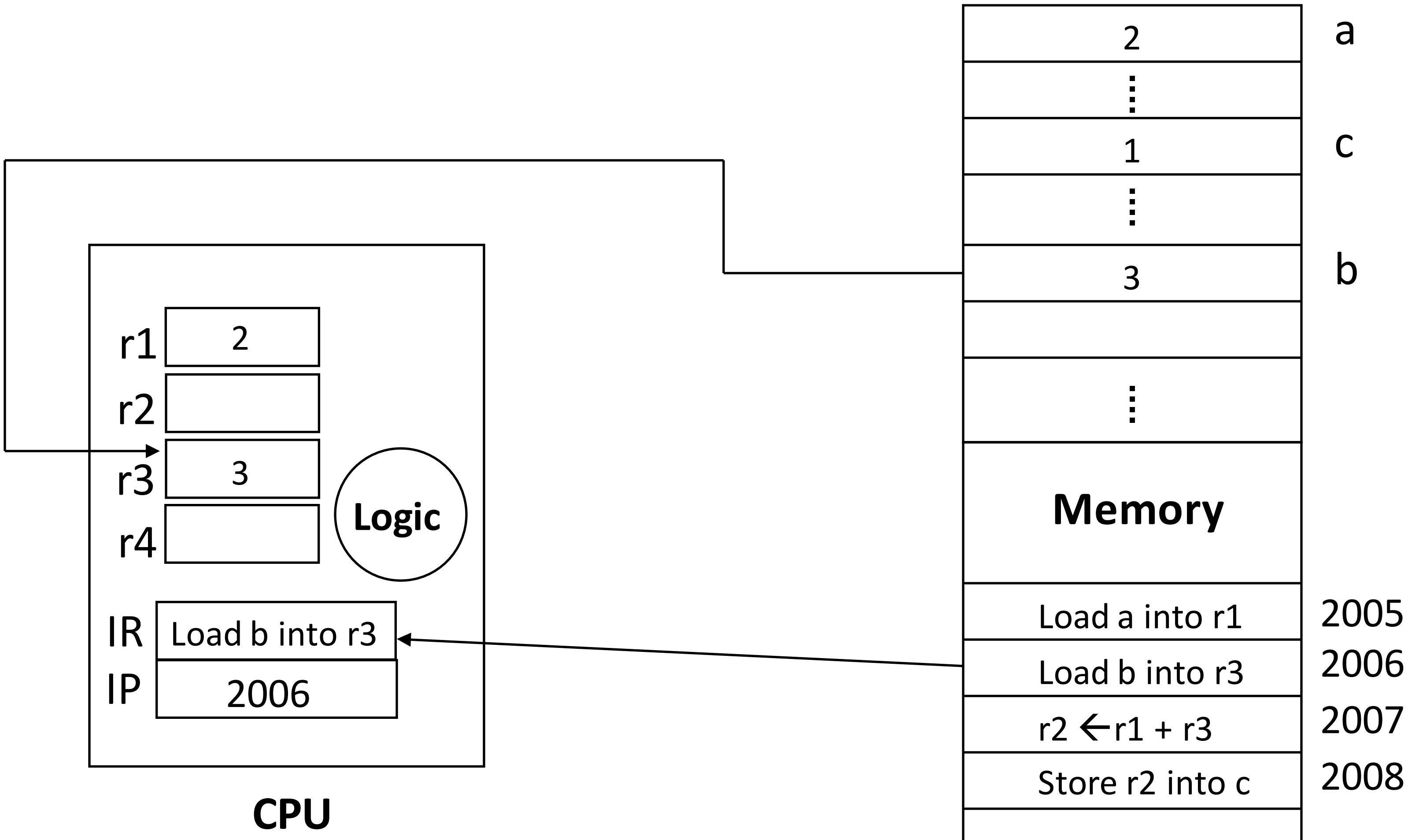
- Want to add values of variables a and b (assumed to be in memory), and put the result in variable c in memory, I.e. c ← a+b

- Instructions in program
  - Load a into register r1
  - Load b into register r3
  - r2 ← r1 + r3
  - Store r2 in c

Running the Program

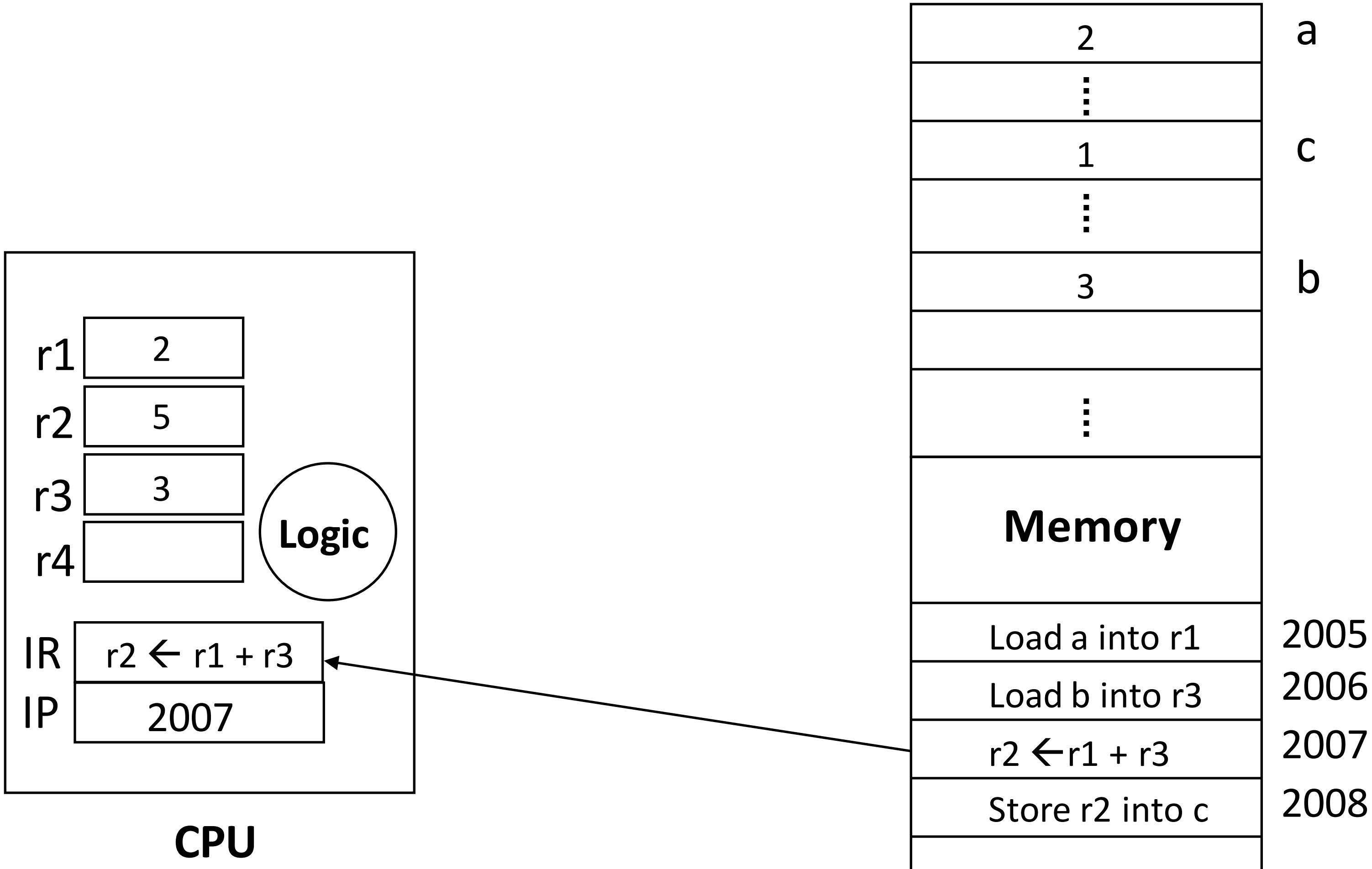


Running the Program

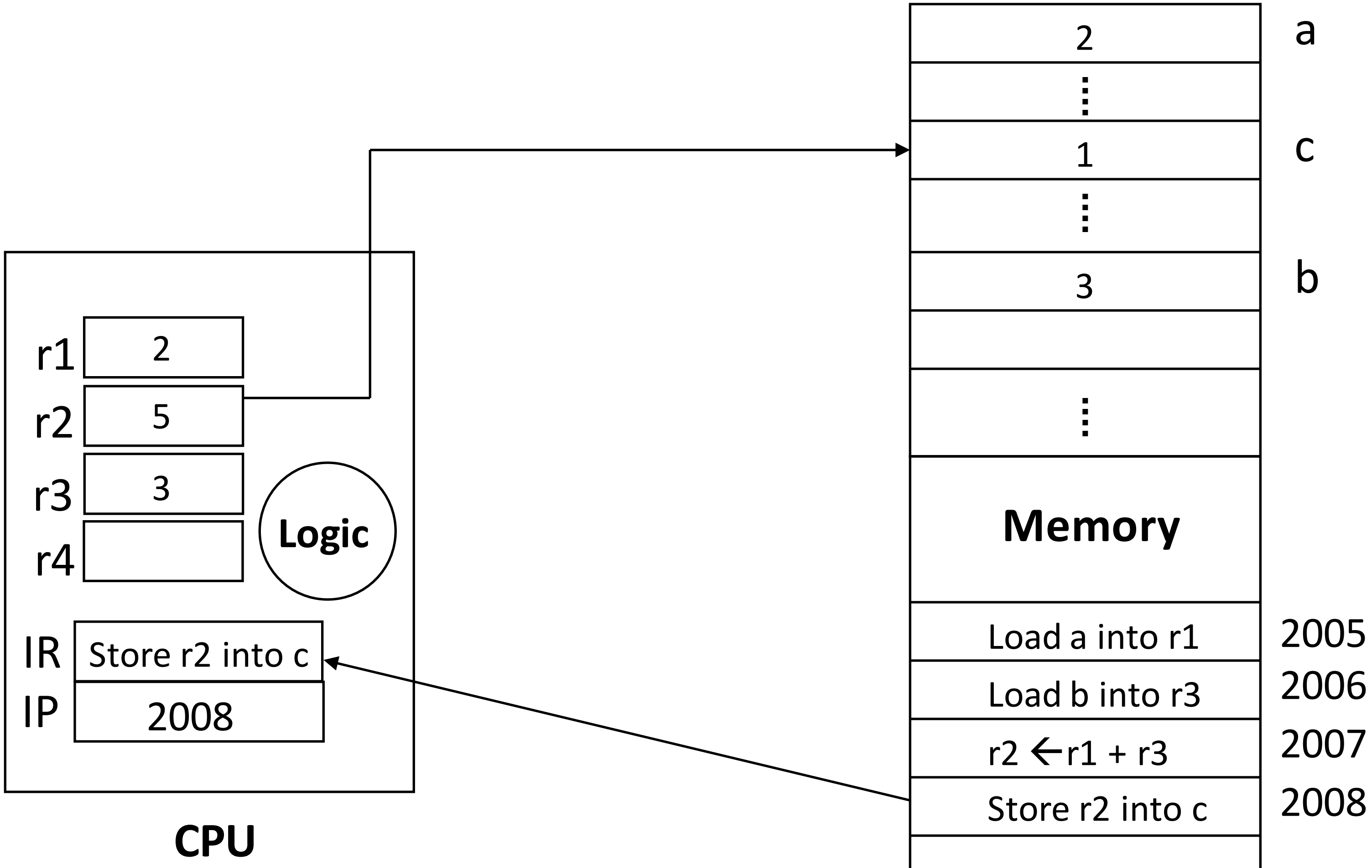


Some slDes are adapted from Pearson Education

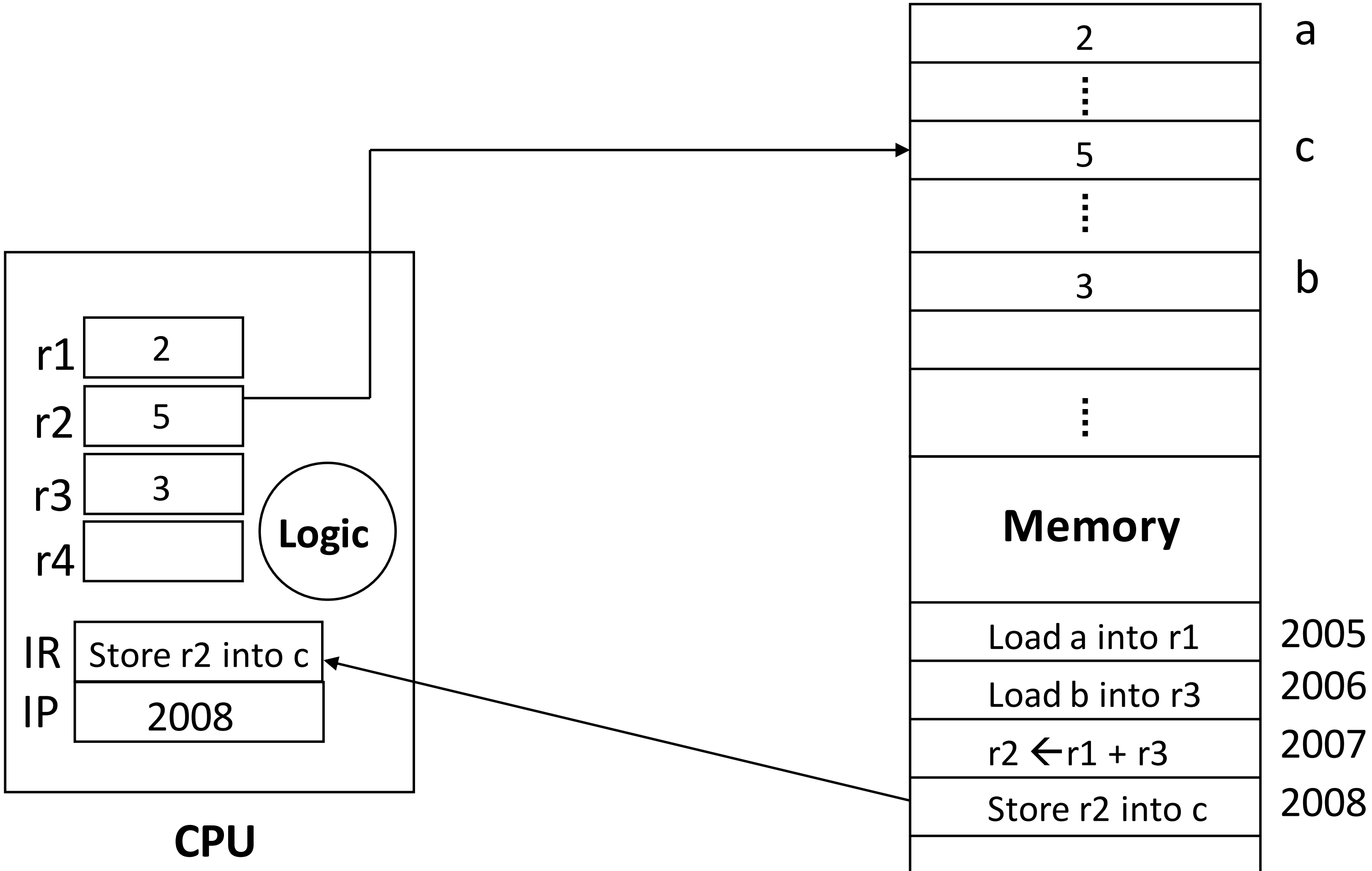
Running the Program



Running the Program



Running the Program

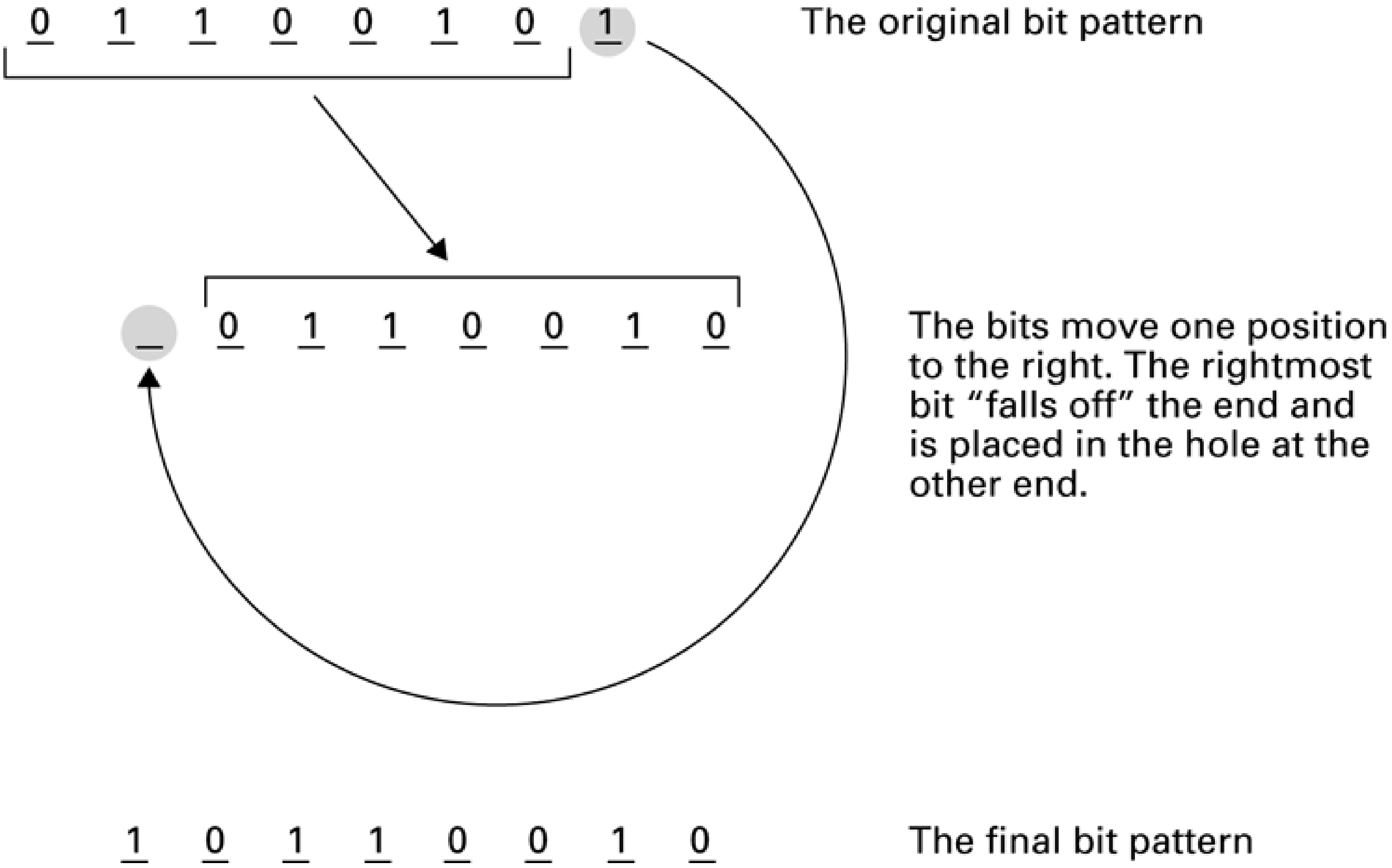


Some slldes are adapted from Pearson Education

Arithmetic/Logic Operations

- Logic: AND, OR, XOR
  - Masking
- Rotate and Shift
- Arithmetic: add, subtract, multiply, divide

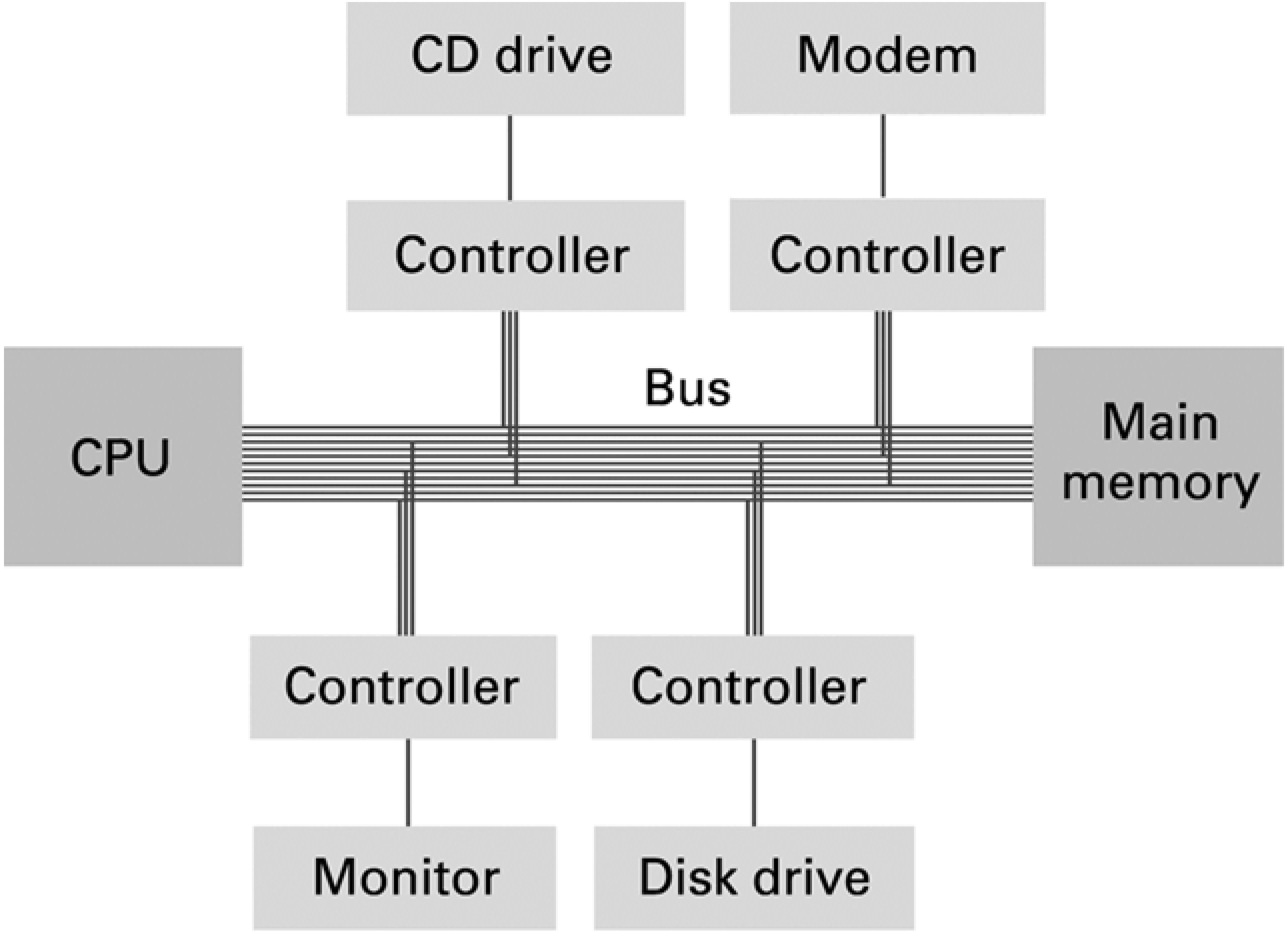
Rotating the bit pattern 65 (hexadecimal) one bit to the right


0 1 1 0 0 1 0 1
The original bit pattern
_ 0 1 1 0 0 1 0
The bits move one position to the right. The rightmost bit "falls off" the end and is placed in the hole at the other end.
1 0 1 1 0 0 1 0
The final bit pattern

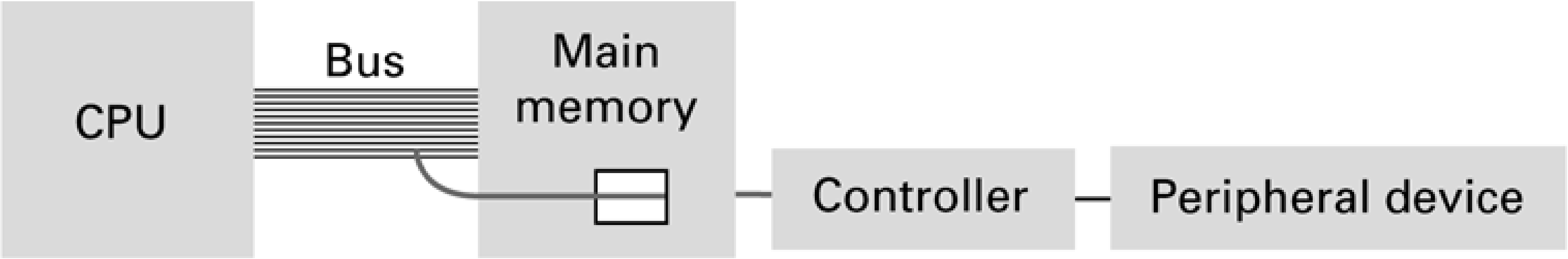
Communicating with Other Devices

- **Controller:** An intermediary apparatus that handles communication between the computer and a device
  - Specialized controllers for each type of device
  - General purpose controllers (USB and FireWire)
- **Port:** The point at which a device connects to a computer
- **Memory-mapped I/O:** CPU communicates with peripheral devices as though they were memory cells

Figure 2.13 Controllers attached to a machine's bus



A conceptual representation of memory-mapped I/O



Communicating with Other Devices (continued)

- **Direct memory access (DMA):** Controller access main memory directly over the bus

- **Von Neumann Bottleneck:** Insufficient bus speed impedes performance
  - CPU and controller compete the bus

- **Handshaking:** The process of coordinating the transfer of data between components
  - Computer and device exchange information about the device’s status and coordinate their activities

Putting it all together

- We learned about
  - Architecture of computer
  - Components of CPU
  - Program counter and instruction register
  - Machine language using encoded bit patterns
  - Machine cycle
  - Op code and operand info
  - Communication with other peripherals

Some slde are adapted from Pearson Education

Question

Direct to rahma2fx@jmu.edu