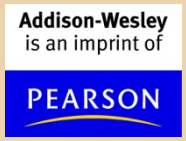
Data Manipulation

Slides derived from those available on the web site of the book:

Computer Science: An Overview, 11th Edition, by J. Glenn Brookshear



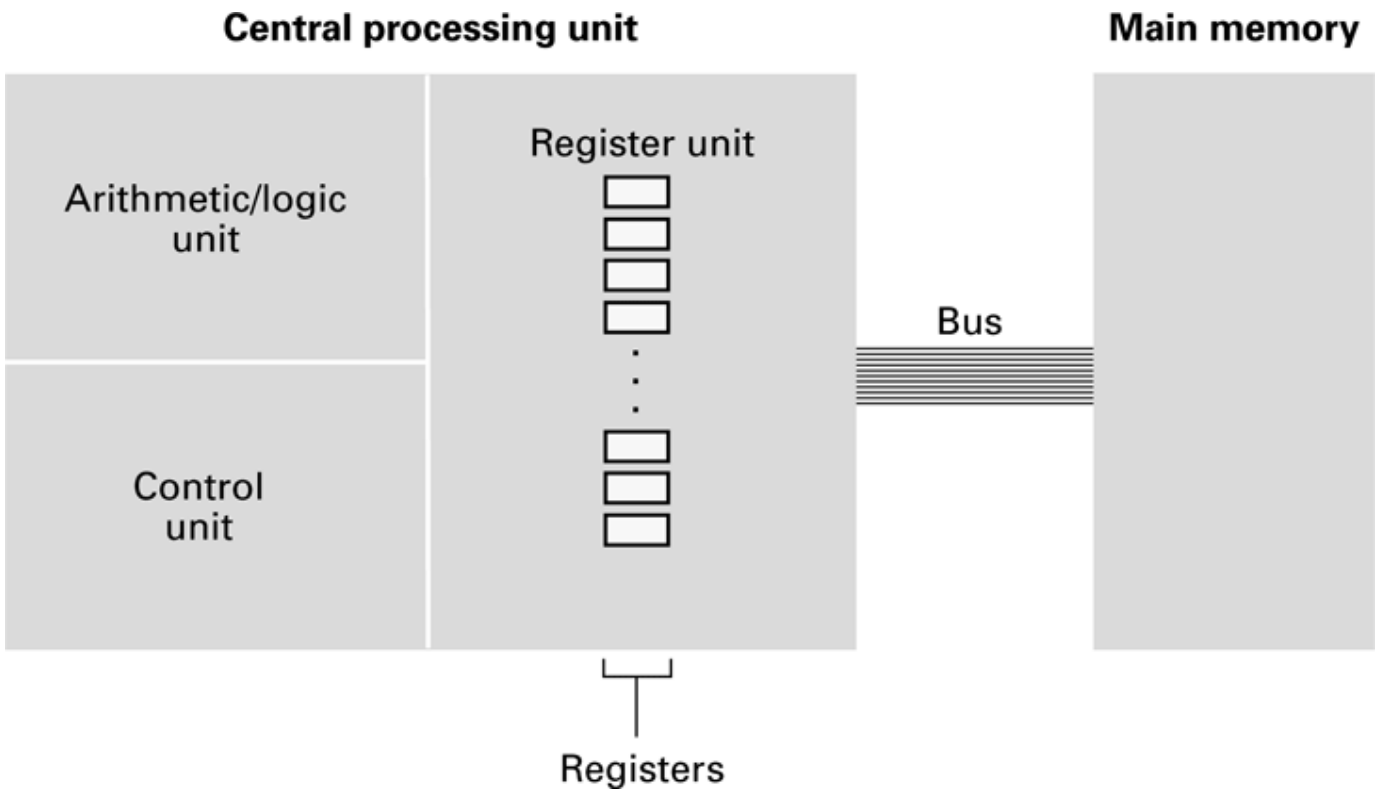
Data Manipulation

- Computer Architecture
- Machine Language
- Program Execution

Computer Architecture

- Central Processing Unit (CPU) or processor
  - Arithmetic/Logic unit versus Control unit
  - Registers
    - General purpose
    - Special purpose
- Bus
- Motherboard

CPU and main memory connected via a bus



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.

Data Manipulation

- Computer Architecture
- Machine Language
- Program Execution

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.

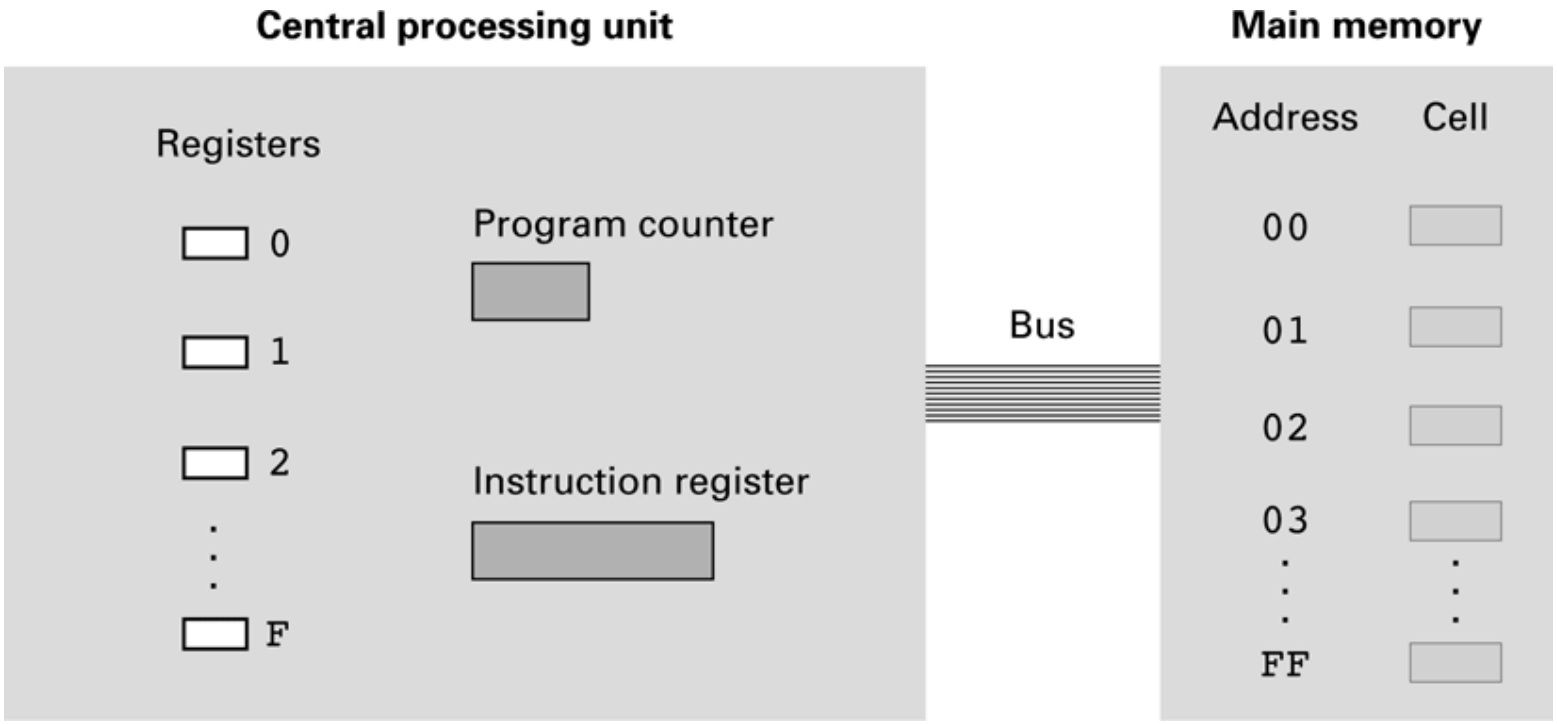
Step 3. If this second value is zero, JUMP to Step 6.

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.

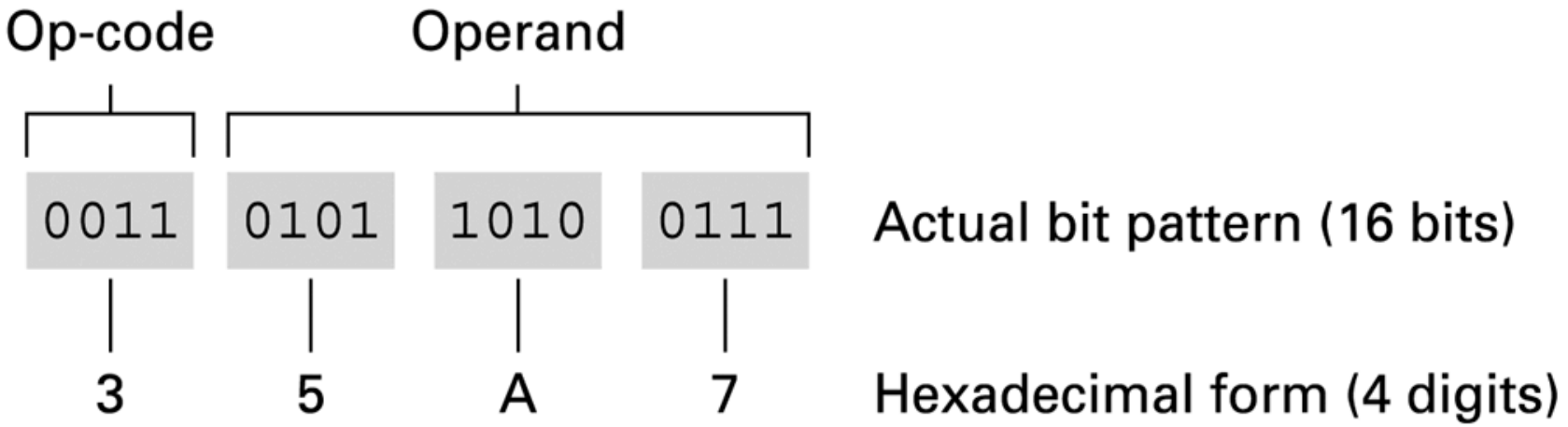
The architecture of the machine: An example



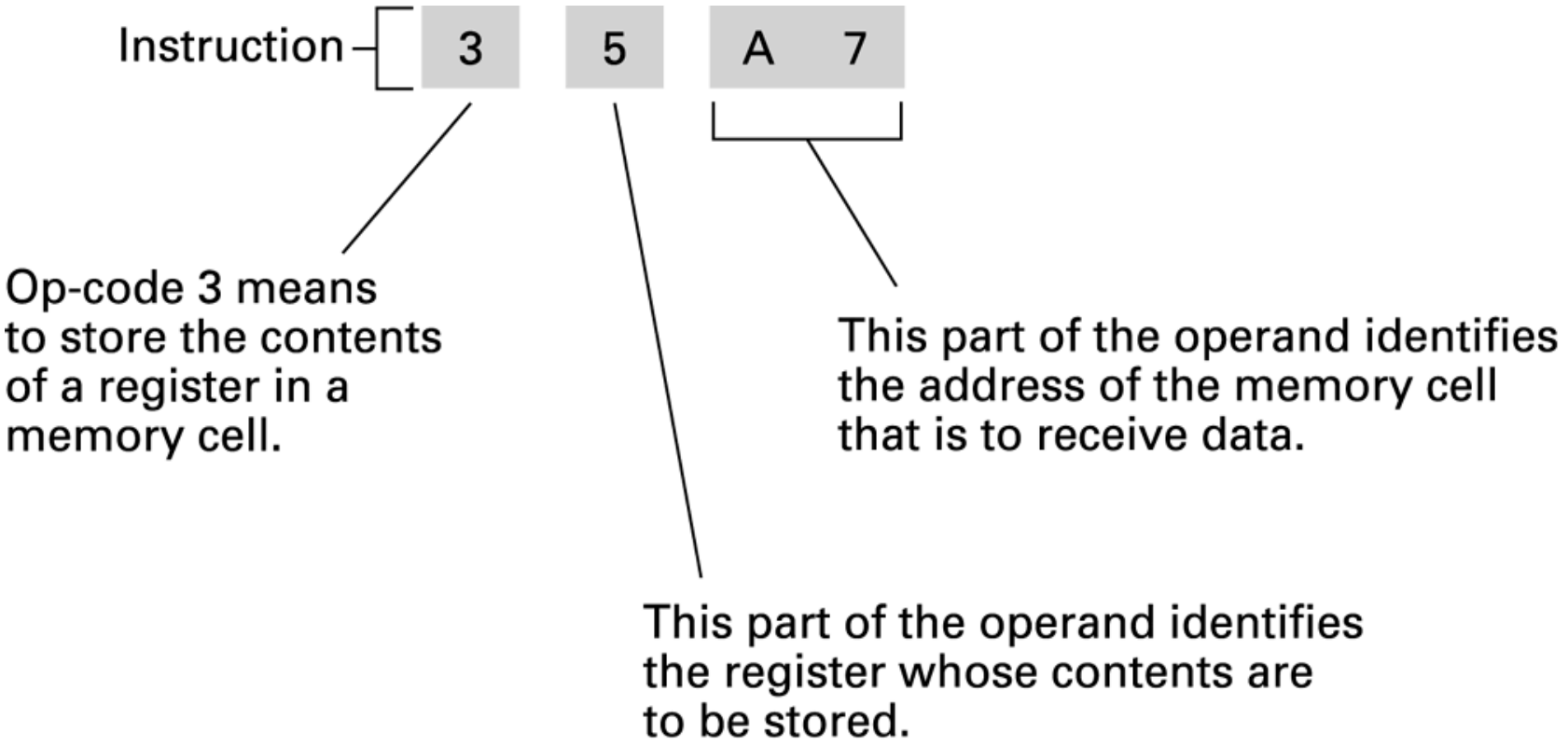
Parts of a Machine Instruction

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

The composition of an instruction: An example



Decoding the instruction 35A7



A Simple Machine Language

Op-code	Operand	Description
1	RXY	LOAD reg. R from cell XY.
2	RXY	LOAD reg. R with XY.
3	RXY	STORE reg. R at XY.
4	0RS	MOVE R to S.
5	RST	ADD S and T into R. (2's comp.)
6	RST	ADD S and T into R. (floating pt.)

A Simple Machine Language (continued)

Op-code	Operand	Description
7	RST	OR S and T into R.
8	RST	AND S and T into R.
9	RST	XOR S and T into R.
A	R0X	ROTATE reg. R X times.
B	RXY	JUMP to XY if R = reg. 0.
C	000	HALT.

An encoded version of the instructions

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

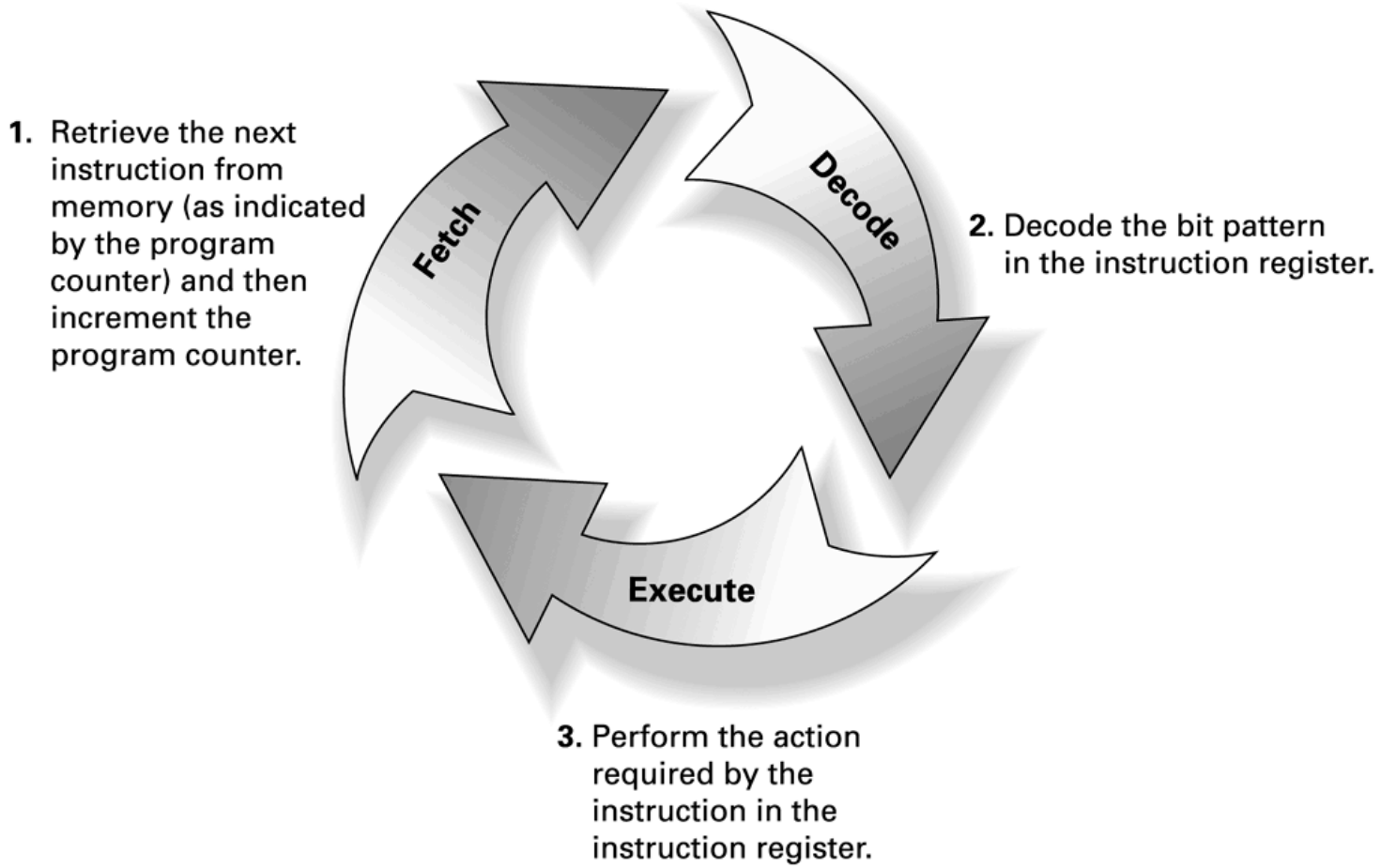
Data Manipulation

- Computer Architecture
- Machine Language
- Program Execution

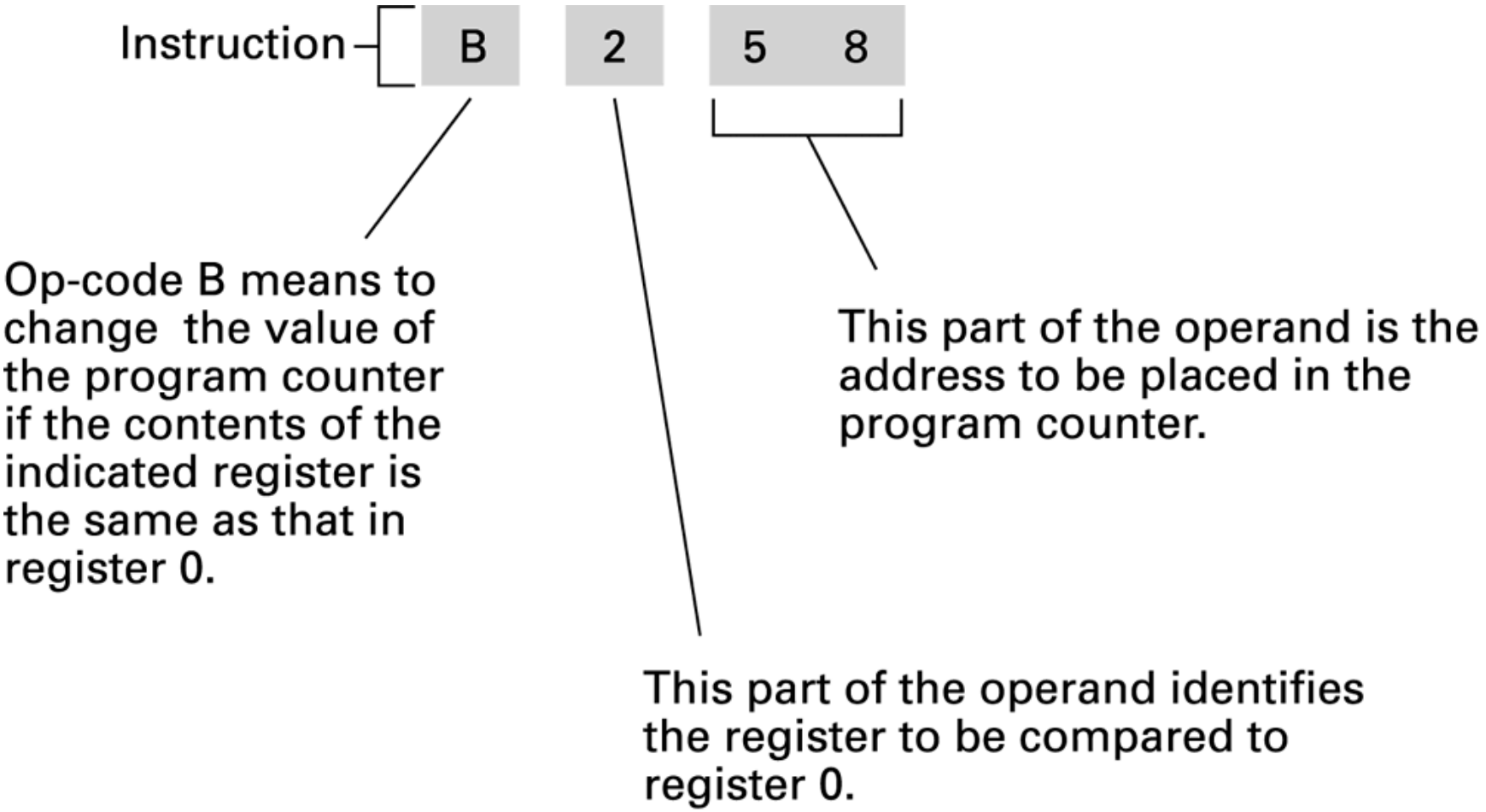
Program Execution

- Controlled by two special-purpose registers
  - Program counter: address of next instruction
  - Instruction register: current instruction
- Machine Cycle
  - Fetch
  - Decode
  - Execute

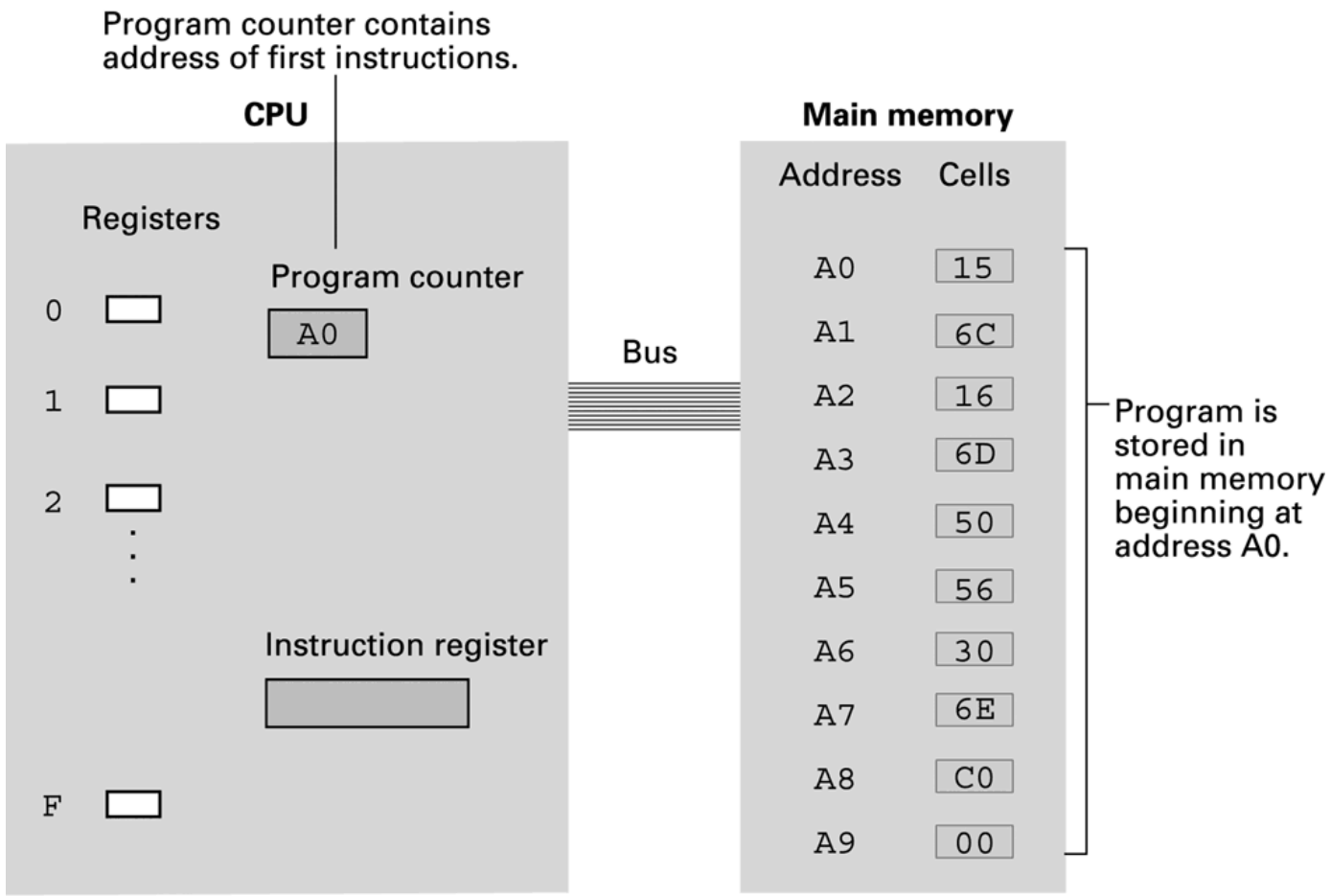
The machine cycle



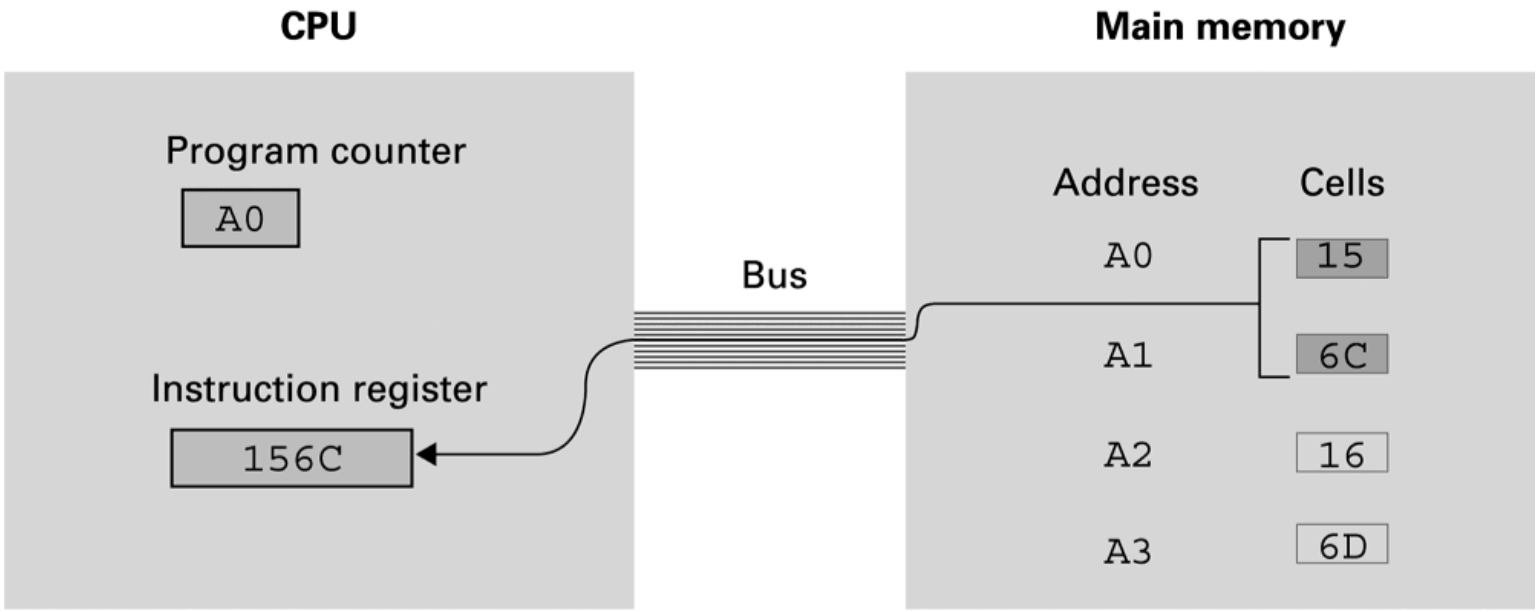
Decoding the instruction B258



The program of adding stored in main memory ready for execution

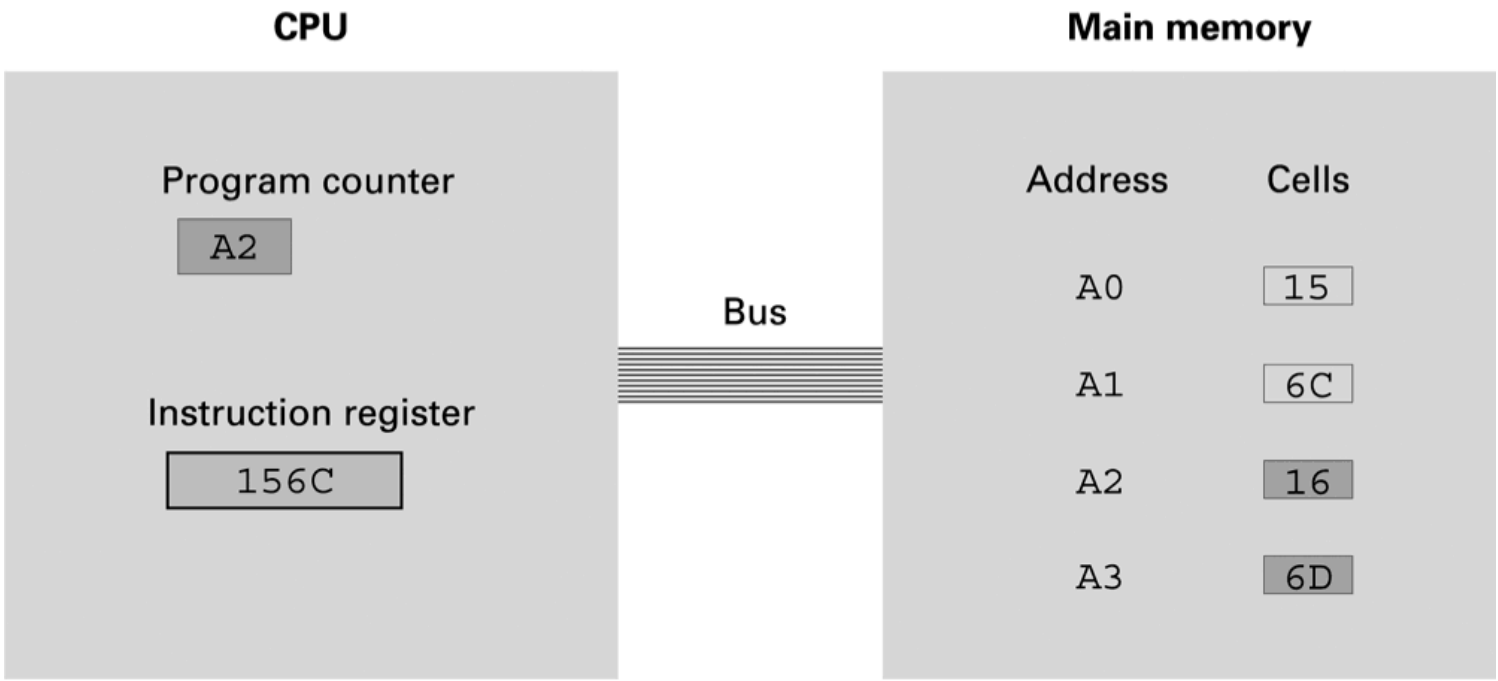


Performing the fetch step of the machine cycle



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.