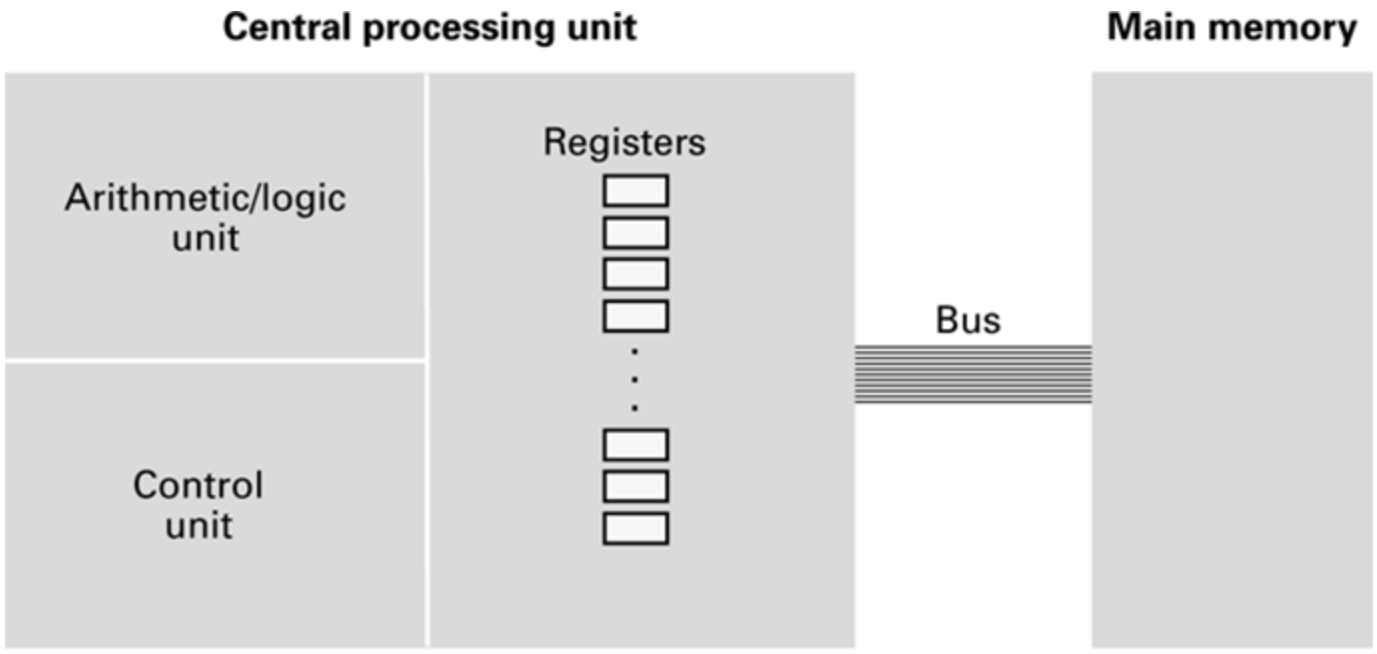
Chapter 2: Data Manipulation

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

Computer Architecture

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

Figure 2.1 **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.

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

Figure 2.2 **Adding values stored in memory**

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

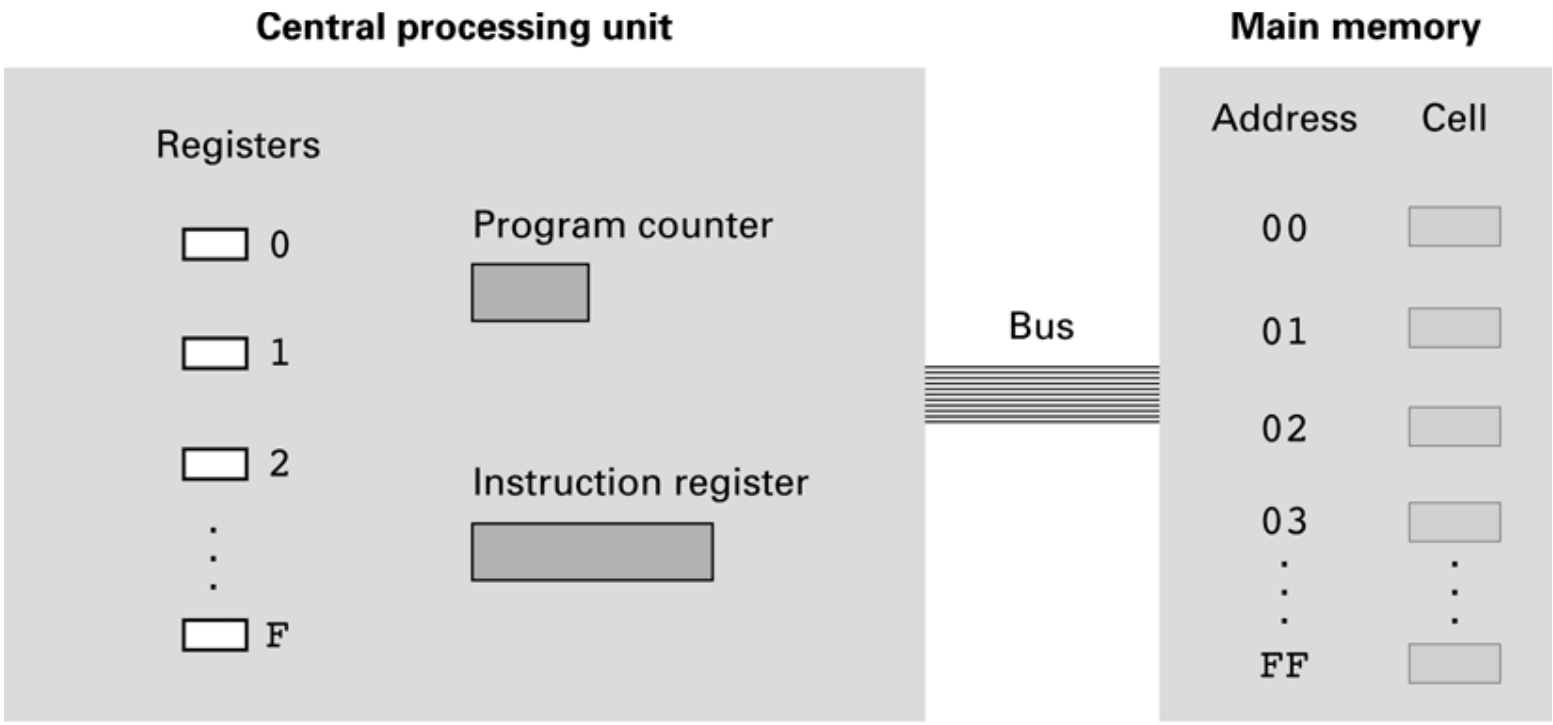
Step 2. Get the other value to be added from memory and place it in another register.

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.

Figure 2.4 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

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

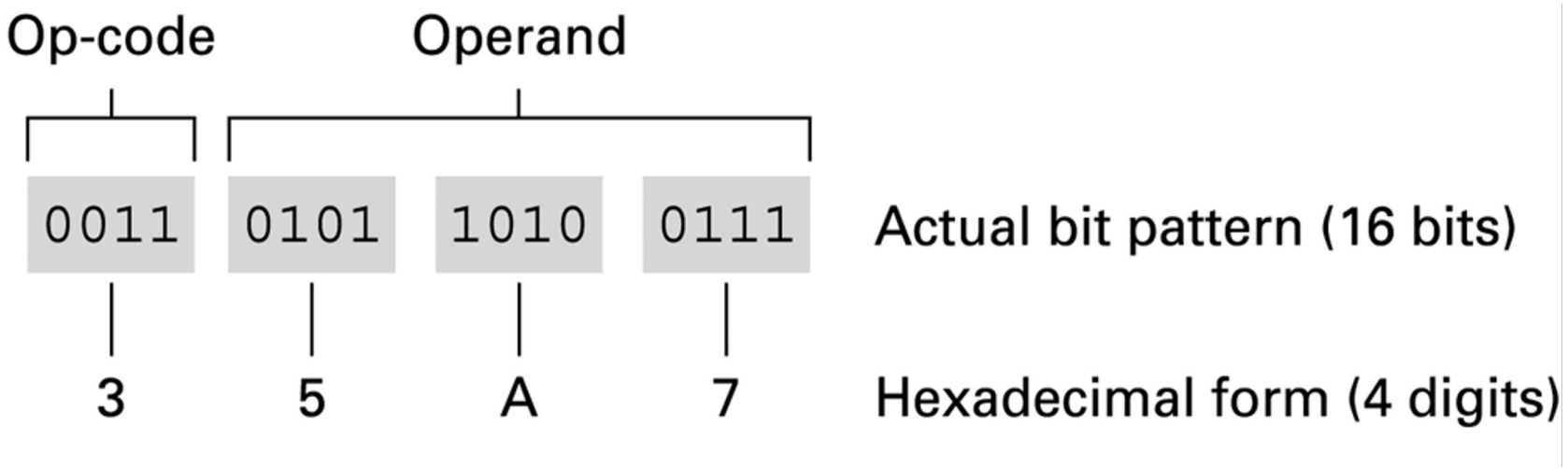


Figure 2.6 **Decoding the instruction 35A7**

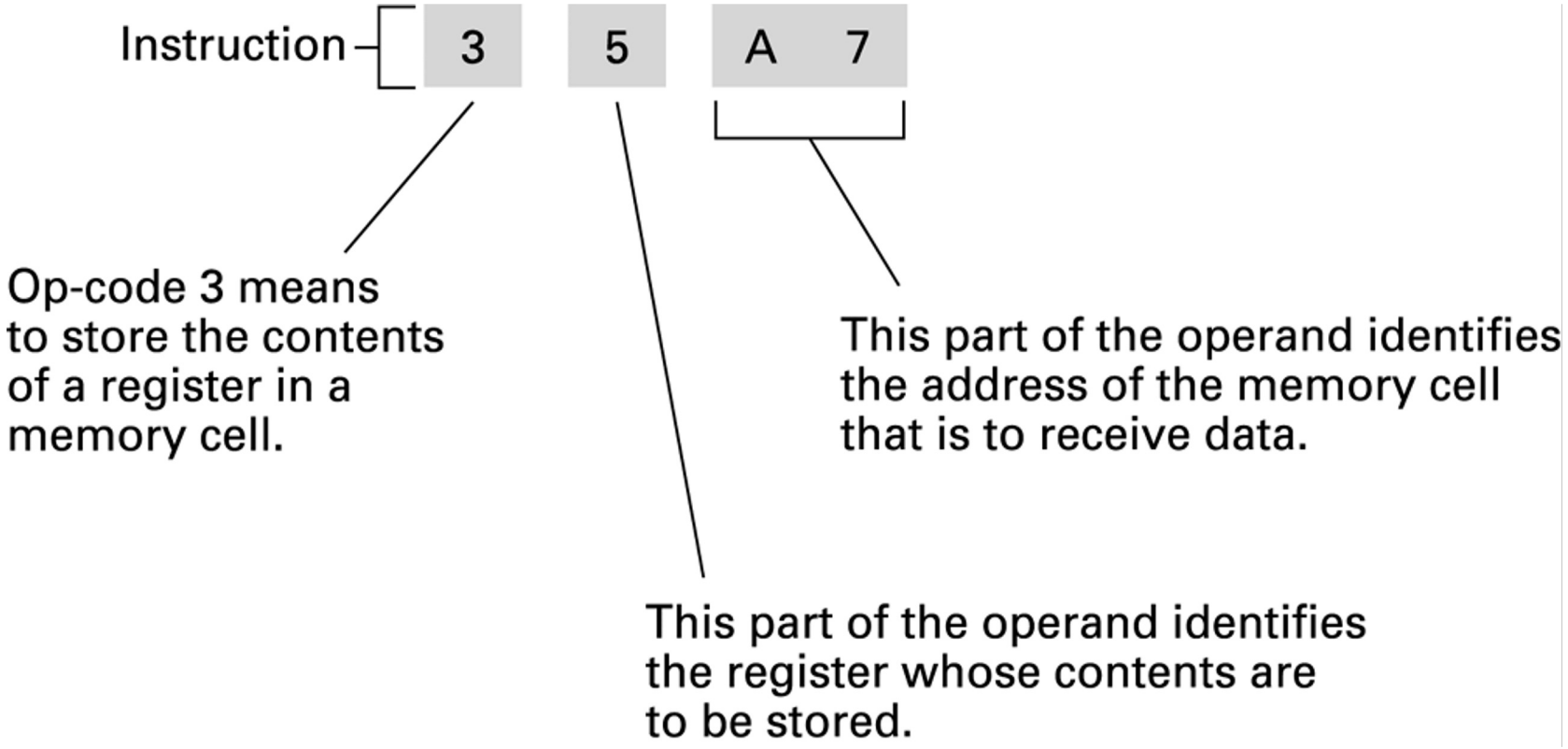


Figure 2.7 An encoded version of the instructions in Figure 2.2 Adding Values stored in memory

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.

Program Execution

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

Figure 2.8 **The machine cycle**

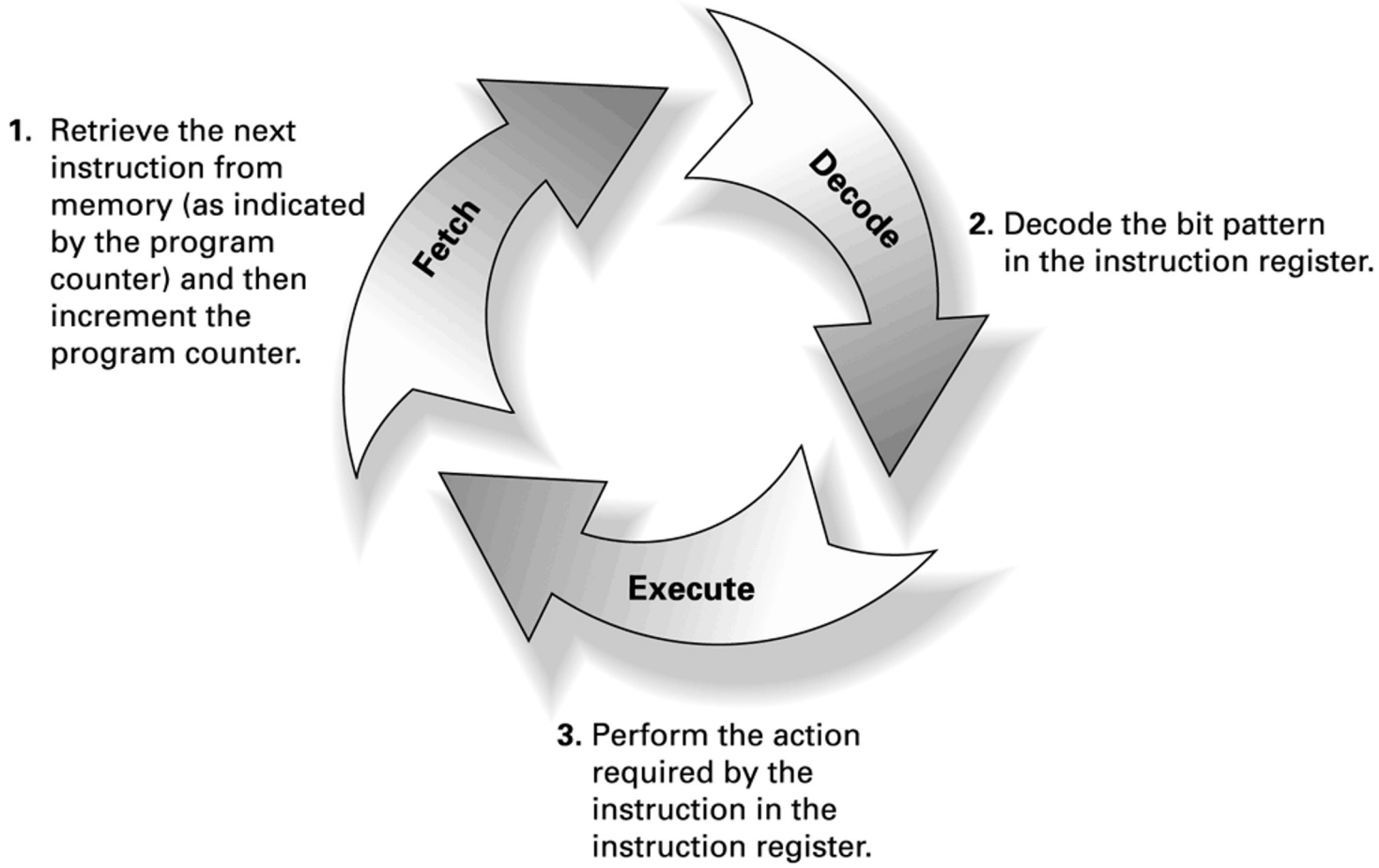


Figure 2.9 **Decoding the instruction B258**

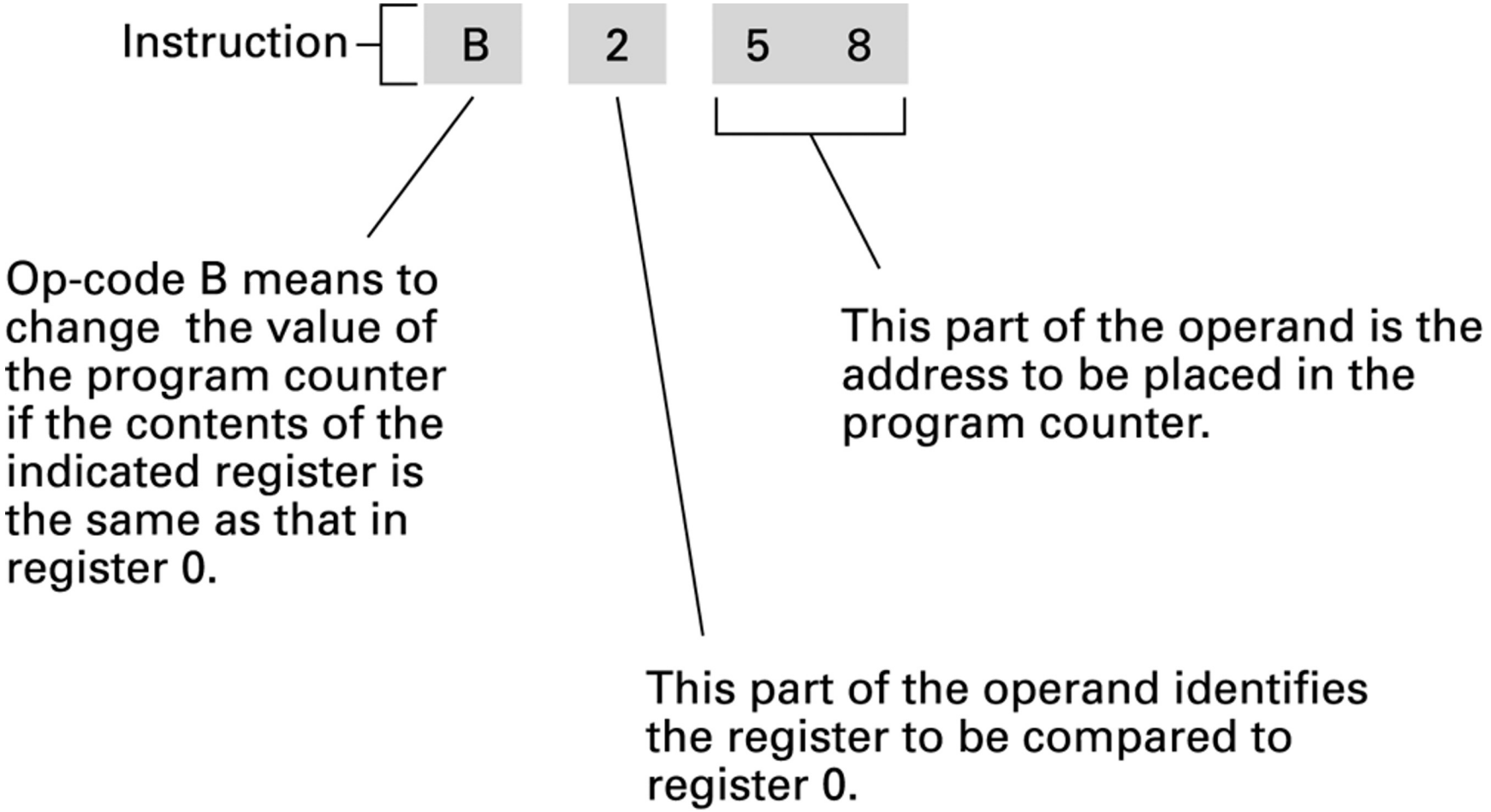


Figure 2.10 The program from Figure 2.7(adding) stored in main memory ready for execution

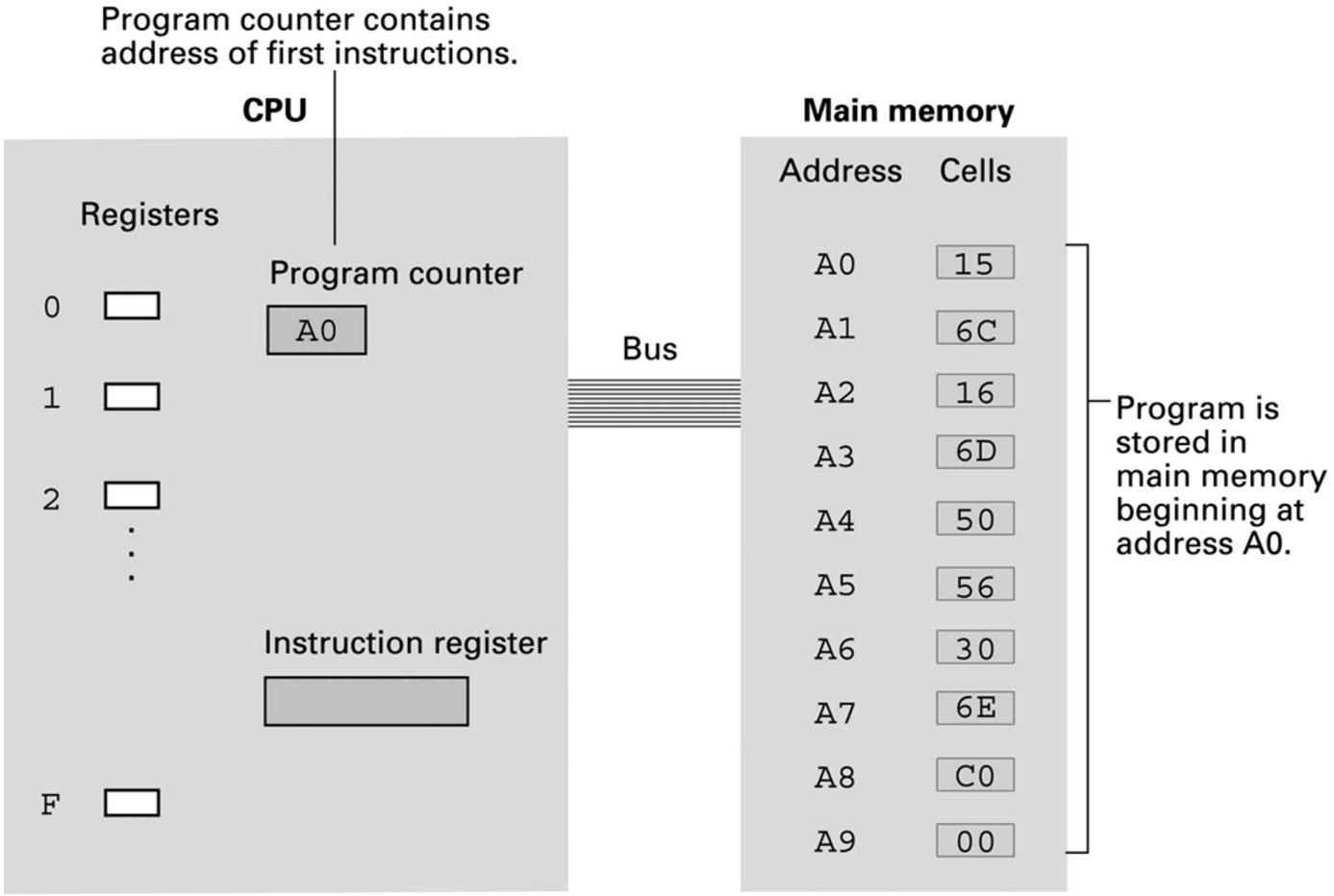
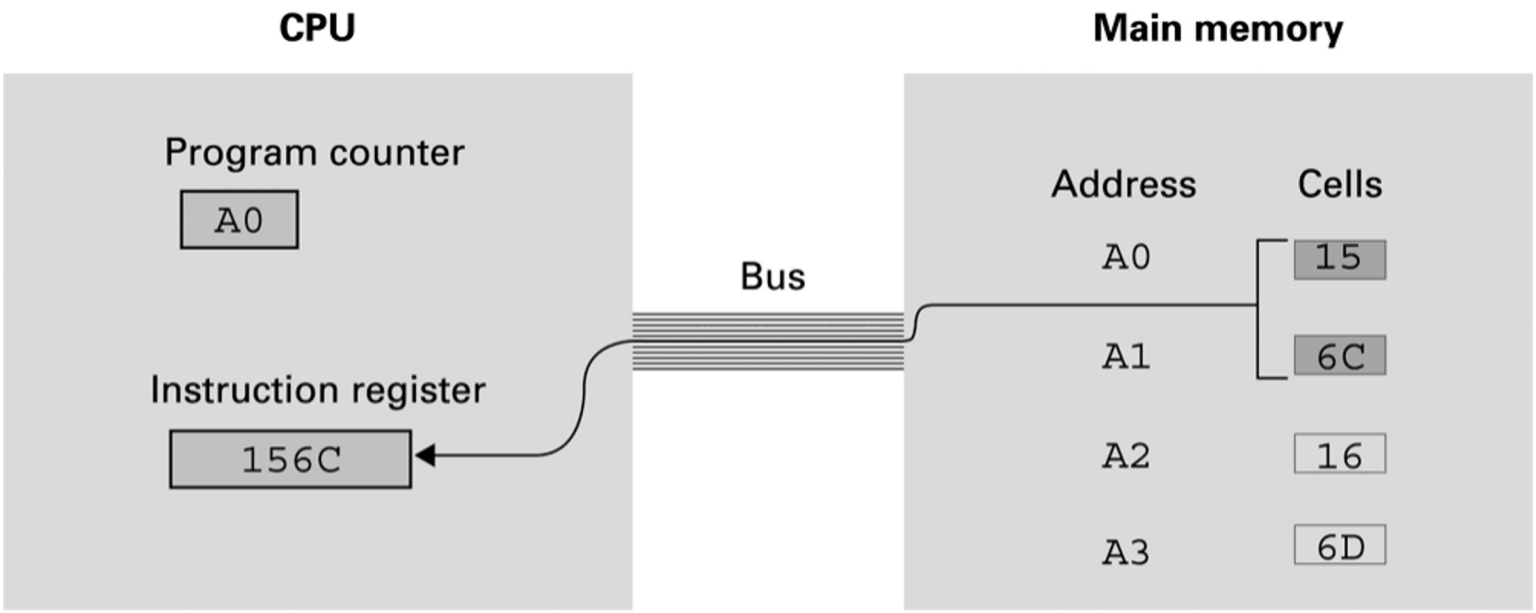
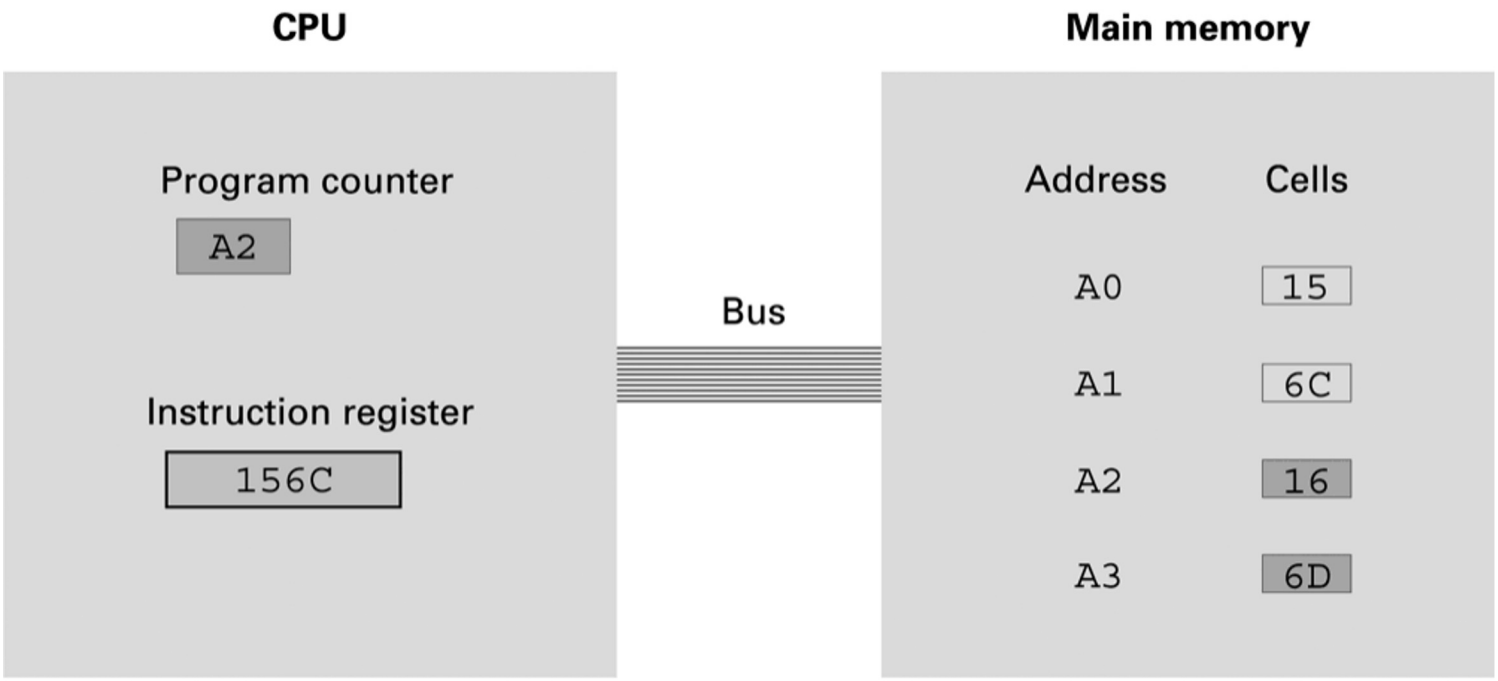


Figure 2.11 **Performing the fetch step of the machine cycle**



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

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



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

Arithmetic/Logic Operations

- Logic: AND, OR, XOR
  - Masking
- Rotate and Shift: circular shift, logical shift, arithmetic shift
- Arithmetic: add, subtract, multiply, divide
  - Precise action depends on how the values are encoded (two's complement versus floating-point).

Figure 2.12 **Rotating the bit pattern 65 (hexadecimal) one bit to the right**