

Chapter 2: Data Manipulation

**Computer Science: An Overview
Eleventh Edition**

**by
J. Glenn Brookshear**

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2012 Pearson Education, Inc.

Chapter 2: Data Manipulation

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

Copyright © 2012 Pearson Education, Inc.

0-2

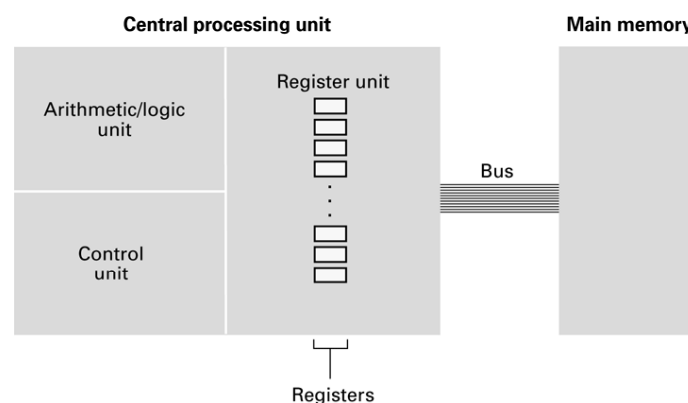
Computer Architecture

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

Copyright © 2012 Pearson Education, Inc.

0-3

Figure 2.1 CPU and main memory connected via a bus



Copyright © 2012 Pearson Education, Inc.

0-4

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

Copyright © 2012 Pearson Education, Inc.

0-7

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

Copyright © 2012 Pearson Education, Inc.

0-8

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.

Copyright © 2012 Pearson Education, Inc.

0-9

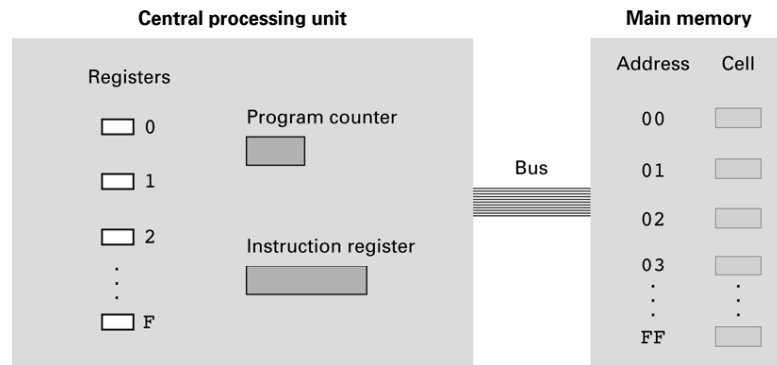
Figure 2.3 Dividing values stored in memory

- 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.

Copyright © 2012 Pearson Education, Inc.

0-10

Figure 2.4 The architecture of the machine described in Appendix C



Copyright © 2012 Pearson Education, Inc.

0-11

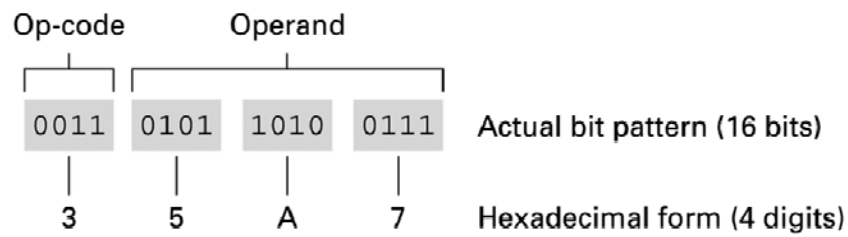
Parts of a Machine Instruction

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

Copyright © 2012 Pearson Education, Inc.

0-12

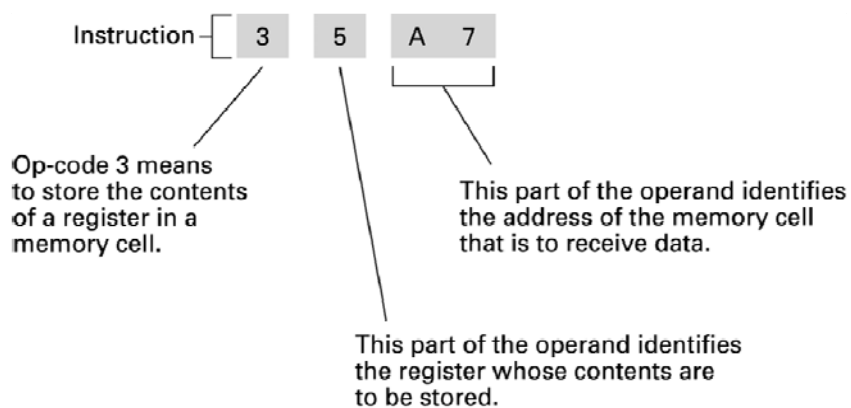
Figure 2.5 The composition of an instruction for the machine in Appendix C



Copyright © 2012 Pearson Education, Inc.

0-13

Figure 2.6 Decoding the instruction 35A7



Copyright © 2012 Pearson Education, Inc.

0-14

Figure 2.7 An encoded version of the instructions in Figure 2.2

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.

Copyright © 2012 Pearson Education, Inc.

0-15

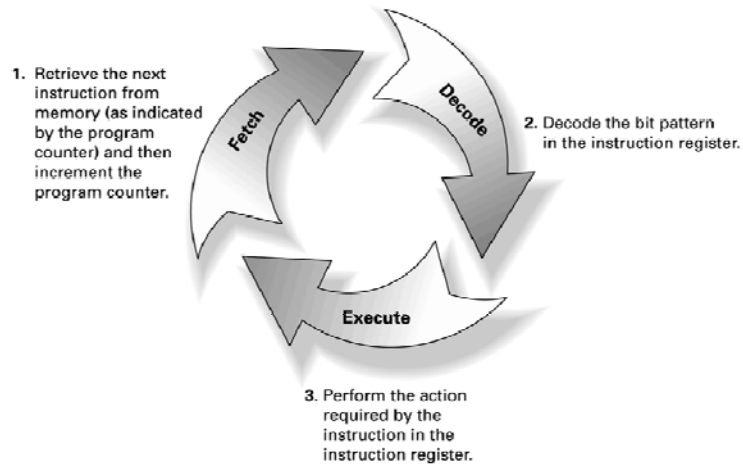
Program Execution

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

Copyright © 2012 Pearson Education, Inc.

0-16

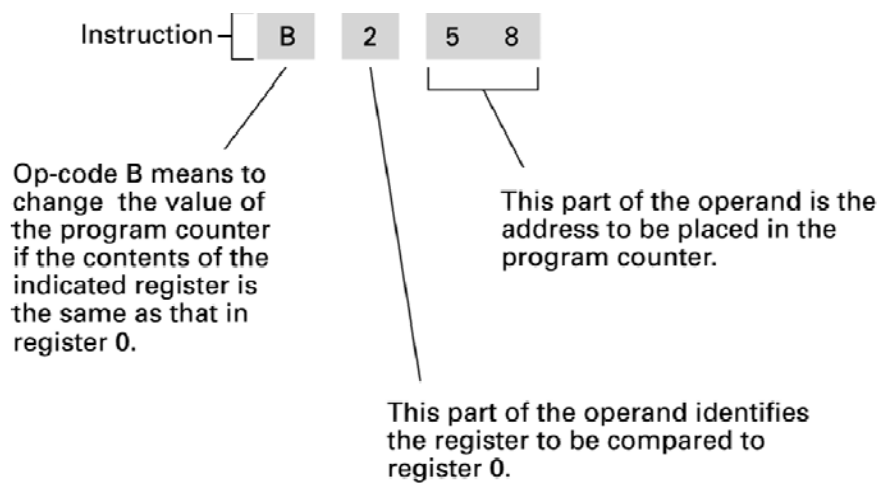
Figure 2.8 The machine cycle



Copyright © 2012 Pearson Education, Inc.

0-17

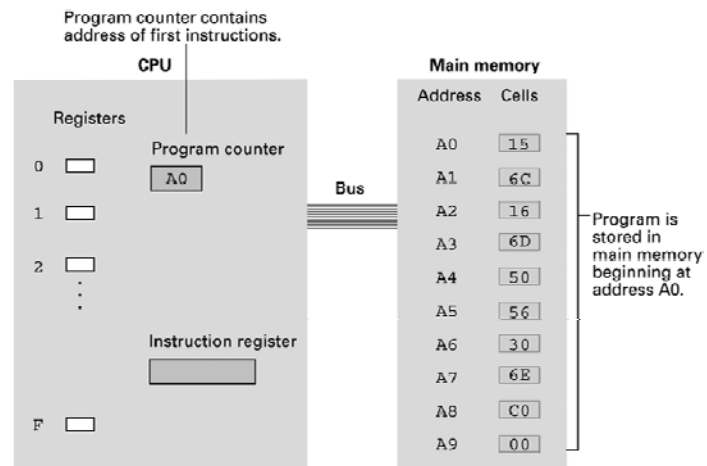
Figure 2.9 Decoding the instruction B258



Copyright © 2012 Pearson Education, Inc.

0-18

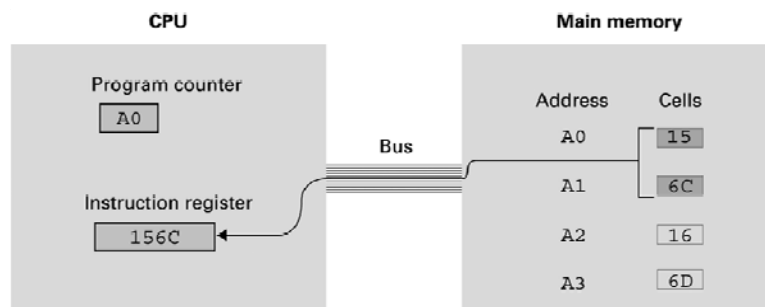
Figure 2.10 The program from Figure 2.7 stored in main memory ready for execution



Copyright © 2012 Pearson Education, Inc.

0-19

Figure 2.11 Performing the fetch step of the machine cycle

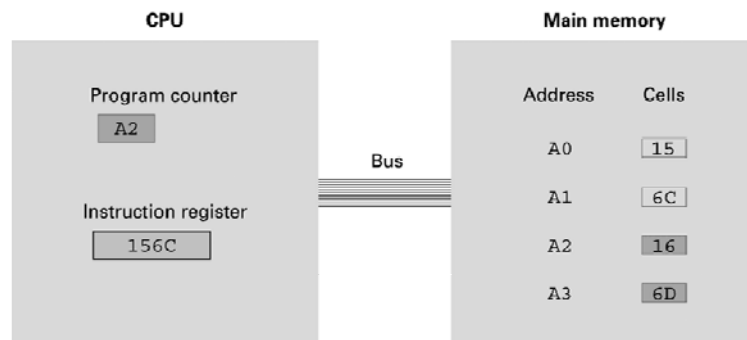


a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

Copyright © 2012 Pearson Education, Inc.

0-20

Figure 2.11 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.

Copyright © 2012 Pearson Education, Inc.

0-21

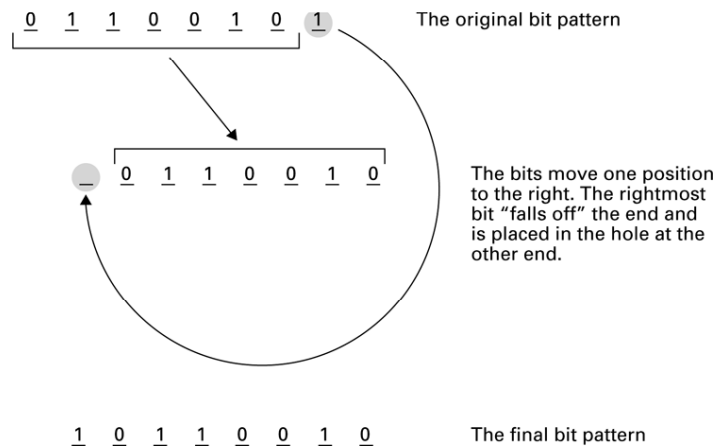
Arithmetic/Logic Operations

- 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).

Copyright © 2012 Pearson Education, Inc.

0-22

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



Copyright © 2012 Pearson Education, Inc.

0-23

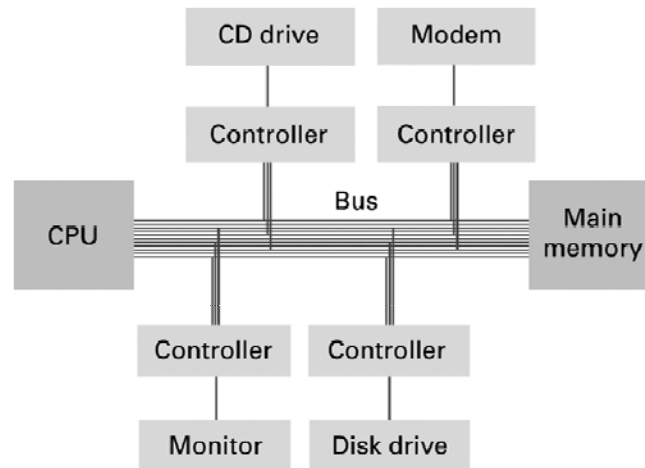
Communicating with Other Devices

- **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

Copyright © 2012 Pearson Education, Inc.

0-24

Figure 2.13 Controllers attached to a machine's bus



Copyright © 2012 Pearson Education, Inc.

0-25

Figure 2.14 A conceptual representation of memory-mapped I/O



Copyright © 2012 Pearson Education, Inc.

0-26

Communicating with Other Devices (continued)

- **Direct memory access (DMA):** Main memory access by a controller over the bus
- **Von Neumann Bottleneck:** Insufficient bus speed impedes performance
- **Handshaking:** The process of coordinating the transfer of data between components

Copyright © 2012 Pearson Education, Inc.

0-27

Communicating with Other Devices (continued)

- **Parallel Communication:** Several communication paths transfer bits simultaneously.
- **Serial Communication:** Bits are transferred one after the other over a single communication path.

Copyright © 2012 Pearson Education, Inc.

0-28

Data Communication Rates

- Measurement units
 - Bps: Bits per second
 - Kbps: Kilo-bps (1,000 bps)
 - Mbps: Mega-bps (1,000,000 bps)
 - Gbps: Giga-bps (1,000,000,000 bps)
- Bandwidth: Maximum available rate

Copyright © 2012 Pearson Education, Inc.

0-29

Other Architectures

- Technologies to increase throughput:
 - Pipelining: Overlap steps of the machine cycle
 - Parallel Processing: Use multiple processors simultaneously
 - SISD: No parallel processing
 - MIMD: Different programs, different data
 - SIMD: Same program, different data

Copyright © 2012 Pearson Education, Inc.

0-30