CMPS1134
Fundamentals of Computing

Data Manipulation 2

Computer Science: An Overview
Eleventh Edition
J. Glenn Brookshear
Chapter 2

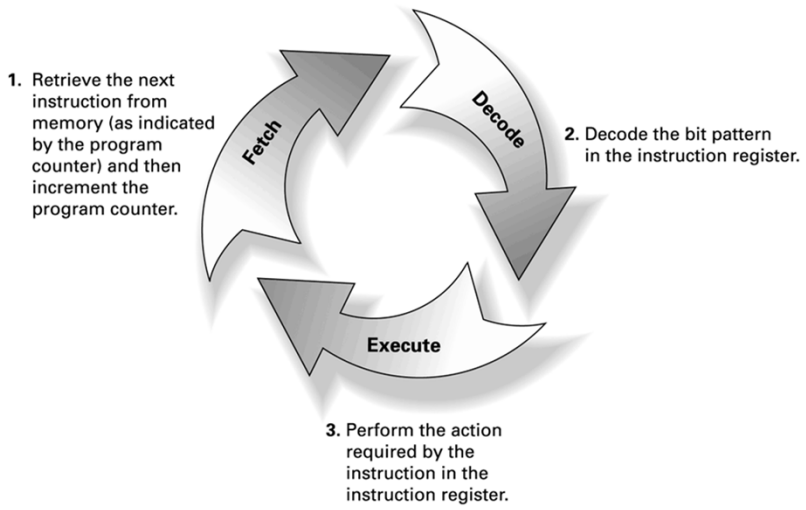
Chapter 2: Data Manipulation

- Program Execution
- Arithmetic/Logic Instructions
- Communicating with Other Devices
- Other Architectures

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**



Appendix C: **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.)
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

Figure 2.9 **Decoding the instruction B258**

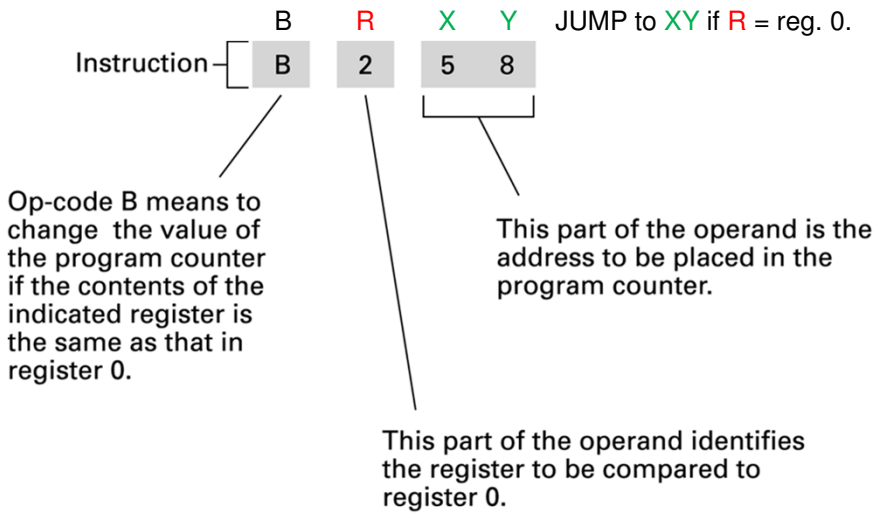
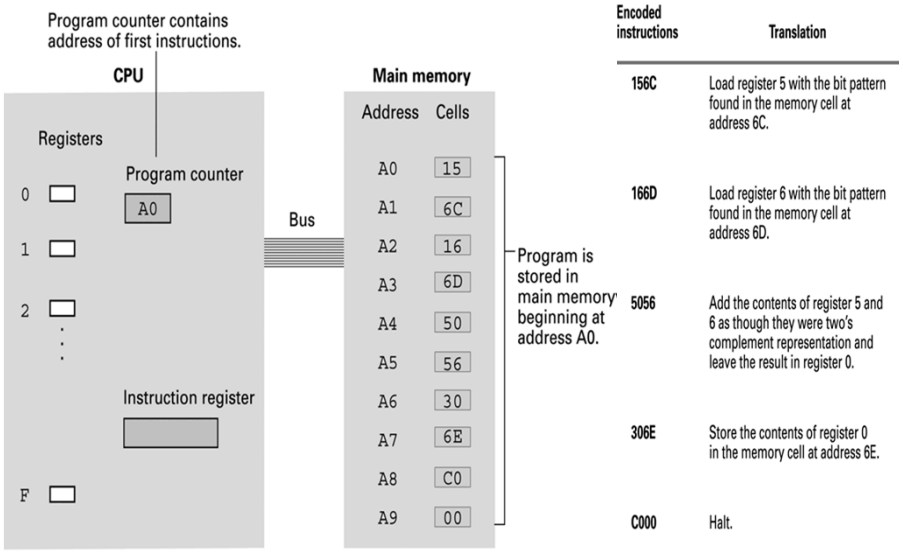


Figure 2.10 **Program from Fig 2.7 stored in main memory ready for execution**

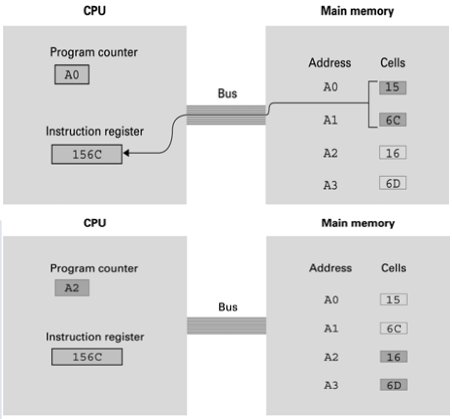


 7

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.

b. Then the program counter is incremented to **A2** so it points to the next instruction.



The instruction is then **decoded** and **executed** and then the cycle repeats until a **HALT** instruction is executed.

 8

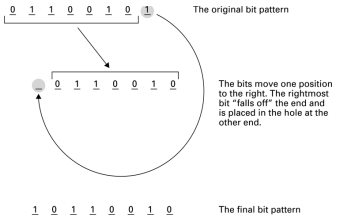
Arithmetic/Logic Operations

- Logic: AND, OR, XOR
  - Masking

00001111 AND 10101010 00001010	Used to duplicate part of a string (specified by 1s) while placing 0s in the non duplicated part (specified by 0s)
00001111 OR 10101010 10101111	Used to duplicate part of a string (specified by 0s) while placing 1s in the non duplicated part (specified by 1s)
11111111 XOR 10101010 01010101	Major use is in forming the complement of a bit string by using a mask of all 1s

Arithmetic/Logic Operations (con't)

- Rotate and Shift:
  - **Circular shift** (Figure 2.12 Rotating the bit pattern 65 (hexadecimal) one bit to the right)



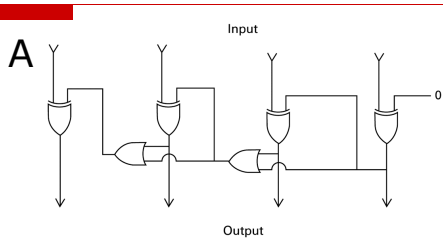
  - **Logical shift**
    - Left (multiplication by 2) or right (division by 2)
    - care must be taken to preserve the sign bit
  - A shift that leaves the sign bit unchanged is sometimes called an **Arithmetic shift**

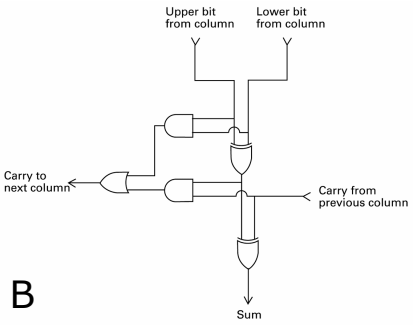
Arithmetic/Logic Operations (con't)

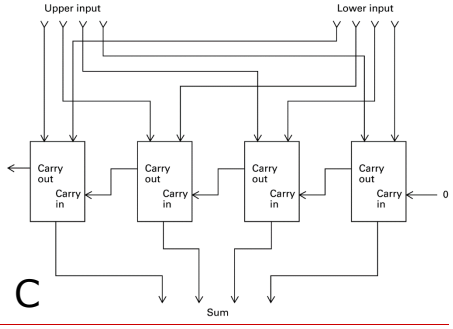
- Arithmetic:
  - Addition
  - Subtraction: addition and negation
  - Multiplication: repeated addition
  - Division: repeated subtraction
- Precise action depends on how the values are encoded:
  - Two's complement: straightforward column by column addition
  - Floating-point: translate from floating-point notation –> carry out operation –> translate back to floating-point notation

Appendix B: **Circuits to manipulate two's complement representations**

A. A circuit that negates a two's complement pattern
B. A circuit to add a single column
C. A circuit for adding two's complement values







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**

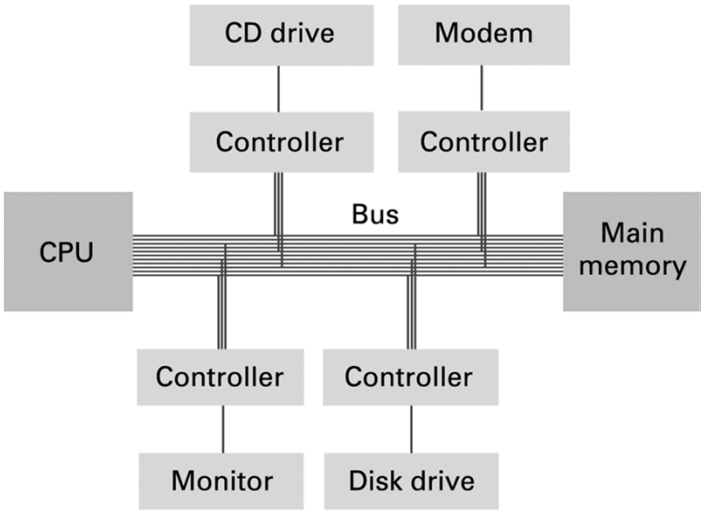
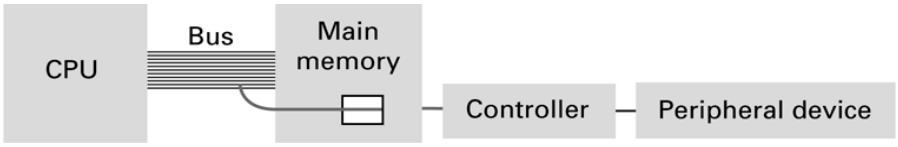


Figure 2.14 **A conceptual representation of memory-mapped I/O**



- Memory-mapped I/O uses a section of memory for I/O.
- The idea is simple. Instead of having "real" memory (i.e., RAM) at that address, place an I/O device.
- Thus, communicating to an I/O device can be the same as reading and writing to memory addresses devoted to the I/O device.
- The I/O device merely has to use the same protocol to communicate with the CPU as memory uses.

 15

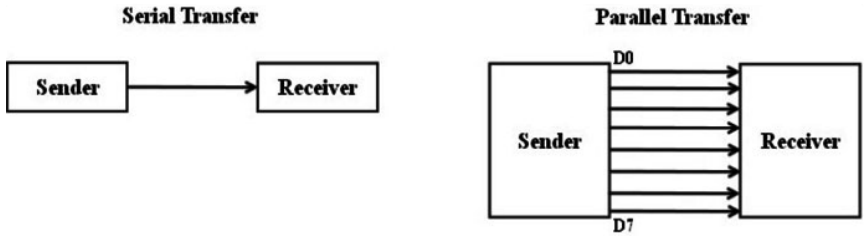
Communicating with Other Devices (continued)

- **Direct memory access (DMA):**
  - Main memory access by a controller over the bus during the nanoseconds in which the CPU is not using the bus
  - A significant asset to a computer's performance
  - Has a detrimental effect of complicating the communication on the bus, making the bus an impediment to the CPU
- **Von Neumann Bottleneck:** Insufficient bus speed impedes performance
- **Handshaking:** The process of coordinating the transfer of data between components

 16

Communicating with Other Devices (continued)

- **Serial Communication:** Bits are transferred one after the other over a single communication path
  - Example: USB, FireWire, modems
- **Parallel Communication:** Several communication paths transfer bits simultaneously
  - Example: Computer's internal bus



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)
- **Multiplexing** encodes or interweaves data so a single communication path serves the purpose of multiple communication paths
- **Bandwidth** is the maximum transfer rate available in a particular setting

Other Architectures

- Traditional machine architecture:
  - **SISD** (Single Instruction, Single Data)
- Technology alternatives to the to increase throughput:
  - **Pipelining**: Overlap steps of the machine cycle – while one instruction executes, the next can be fetched and placed in the "pipe"
  - **Parallel Processing**: Use multiple processors simultaneously
    - **SIMD** (Single Instruction, Multiple Data): Same program, different data
    - **MIMD** (Multiple Instruction, Multiple Data): Different programs, different data