PREVIOUS LESSONS

Week 1: Historyof Computing

Week 2: Data Storage

Chapter 2:
Data Manipulation

Computer Science: An Overview
Twelfth Edition

by
J. Glenn Brookshear

- Previous Week:
- How data is **stored** inside a computer.

- This Week:
- How a computer **manipulates** this data.

(moving data, arithmetic calculations, image manipulation, …)

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

2.1. Computer Architecture

- The circuitry in a computer that controls the manipulation of data is called the
- **central processing unit** (CPU) or processor
- Central Processing Unit (CPU) or processor
  - Arithmetic/Logic unit versus Control unit
  - Registers
    - General purpose
    - Special purpose
- Bus
- Motherboard

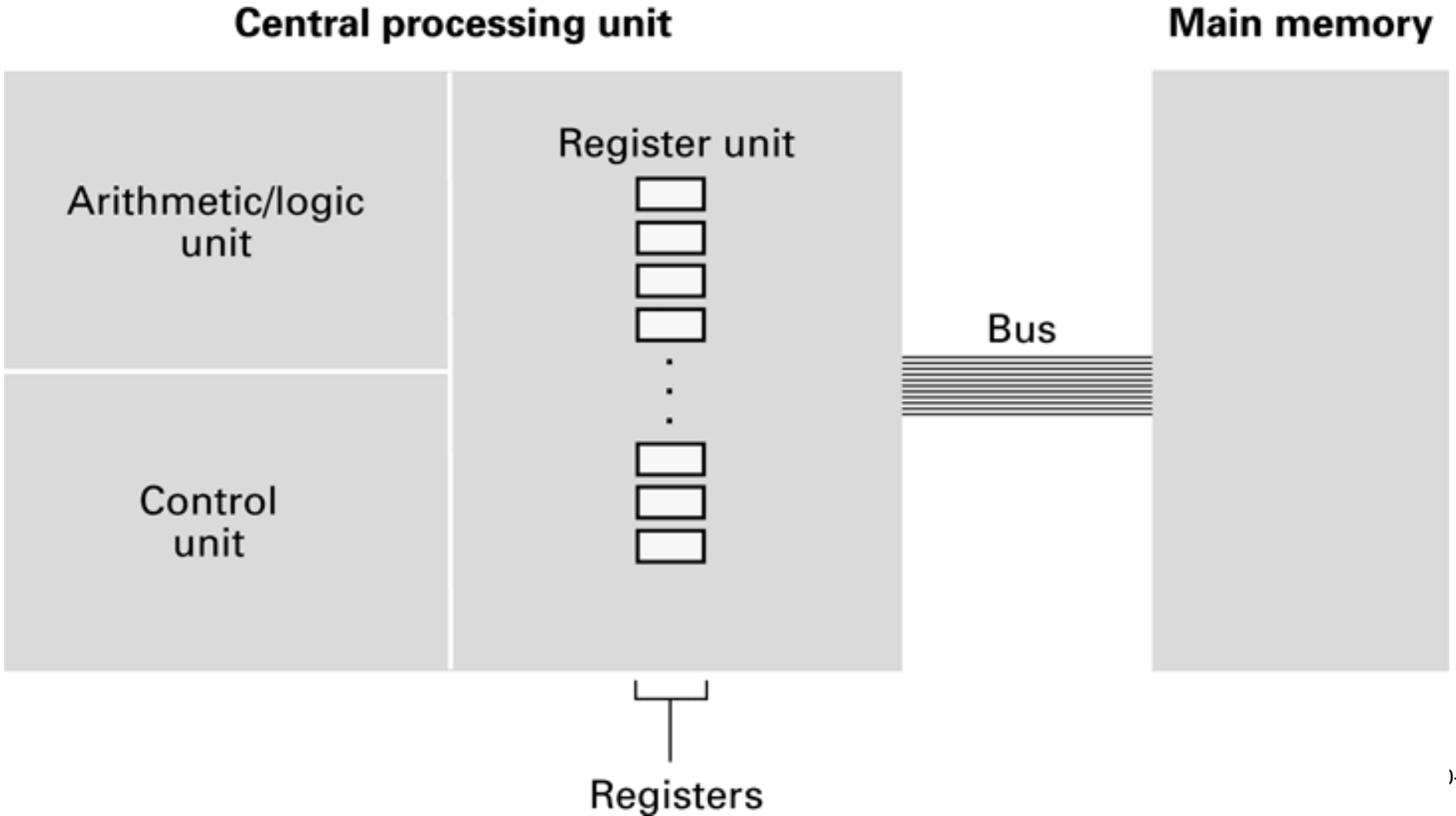
- CPU consists of 3 parts:
- 1) Arithmetic/Logic Unit circuitry that performs operations on data (such as addition and subtraction);

- 2) Control unit circuitry for coordinating machine's activities

- 3) Registers temporary storage of information within the CPU
  - **A)** General purpose (temporary holding places for data being)
  - **B)** Special purpose (for holding program state)

Figure 2.1 **CPU and main memory connected via a bus**

Bus (collection of wires for transfer of data or adress)
Motherboard



Stored Program Concept

A program can be encoded as bit patterns and stored in main memory.

- The idea of storing a computer's program in its main memory is called the **stored-program concept (John von Neumann)**

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 instruction **set** 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

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

Figure 2.2 Adding values stored in memory

Task of adding 2 values stored in main memory involves more than mere execution of the addition operation. Data must be transferred from main memory to registers within the CPU, the values must be added with result being placed in a register, and result must then be stored in a memory cell.

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 2.3 **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.

Appendix C: **A Simple Machine Language (Instruction Set)**

Op-code: Specifies which operation to execute

Operand: Gives more detailed information about operation

Interpretation of operand varies depending on op-code

Op-code	Operand	Description
1	**RXY**	LOAD reg. R from cell XY.
2	**Rxy**	LOAD reg. R with data "xy".
3	RXY	STORE reg. R at cell 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.)

Appendix C: **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 cell XY if R = reg. 0.
C	000	HALT.

Figure 2.4 The architecture of the machine described in Appendix C

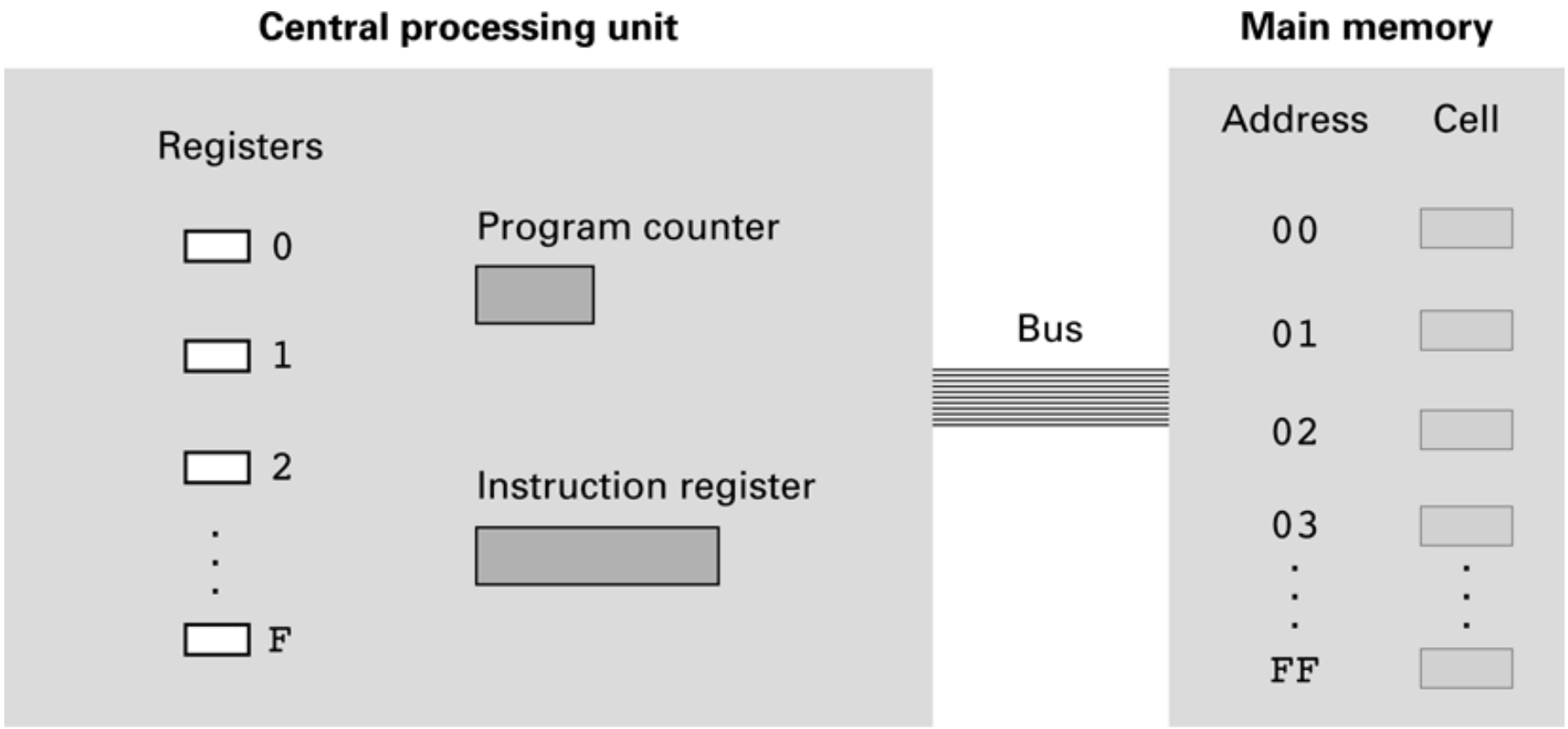


Figure 2.5 The composition of an instruction for the machine in Appendix C

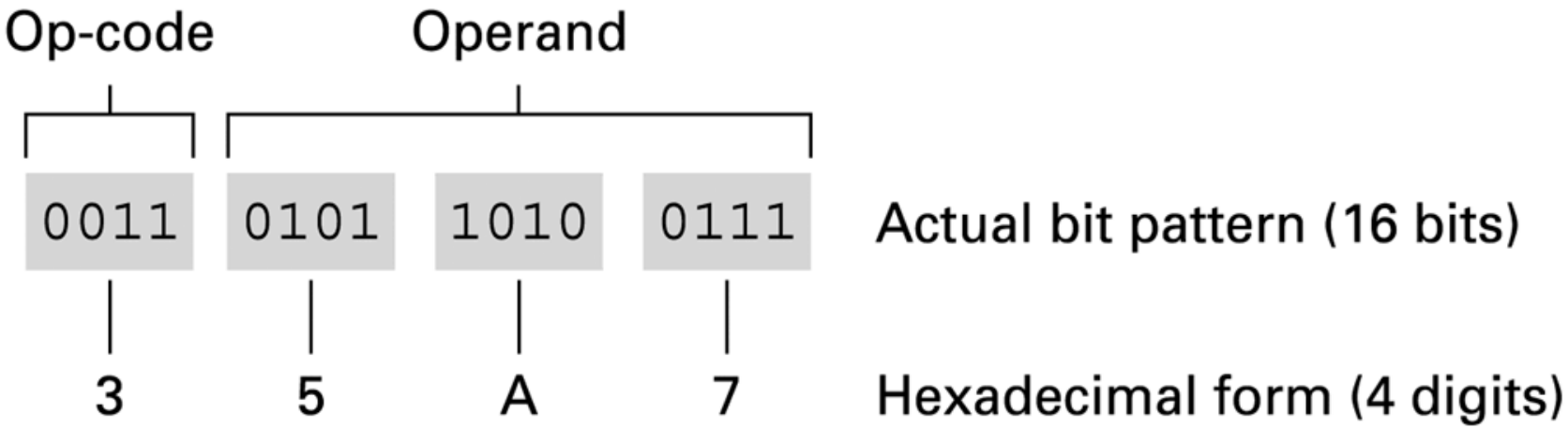
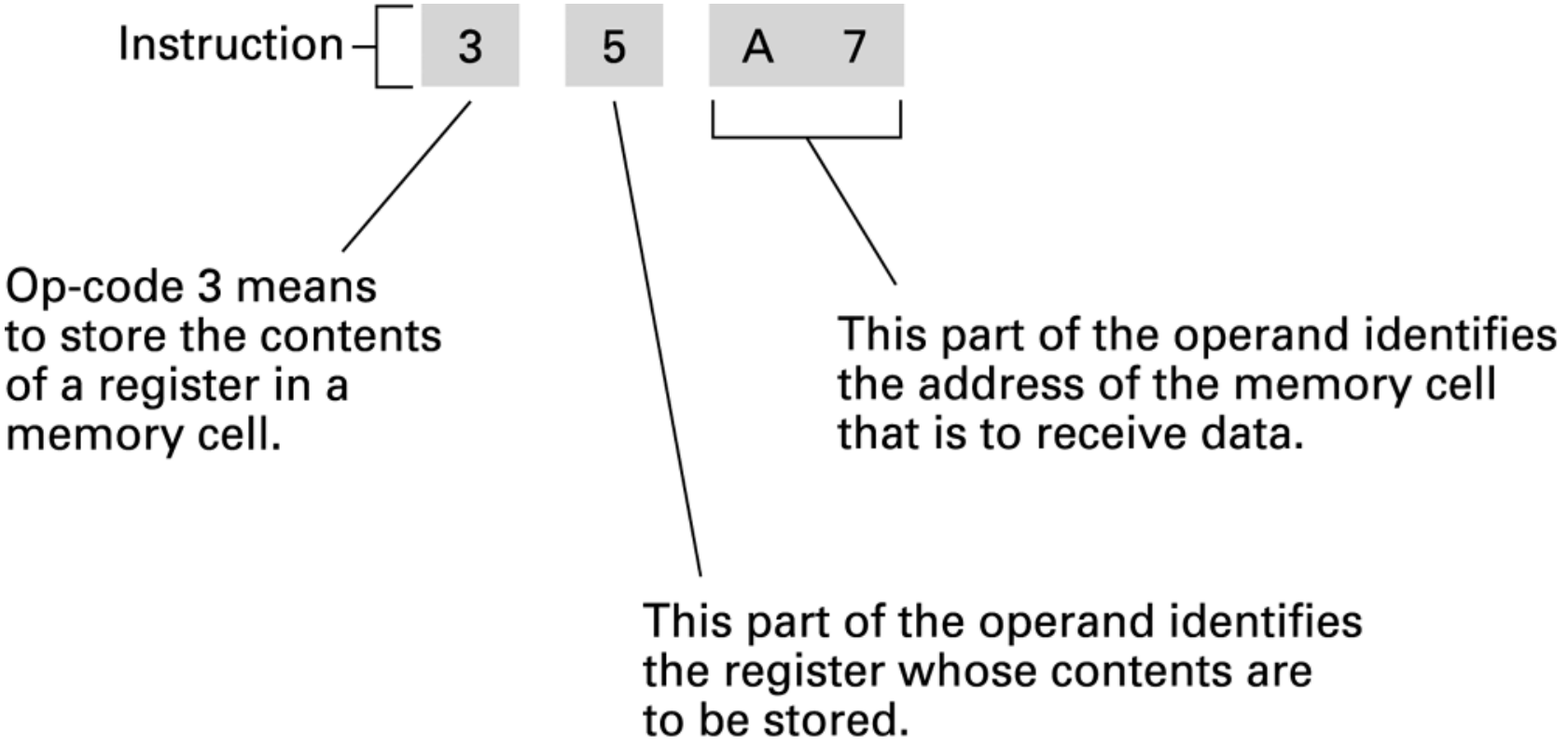


Figure 2.6 Decoding the instruction 35A7



3 RXY STORE reg. R at cell XY.

Program Execution

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

- Machine Cycle
  - Fetch (alma çevrimi)
  - Decode (çözme çevrimi)
  - Execute (yürütme çevrimi)

Figure 2.8 The machine cycle

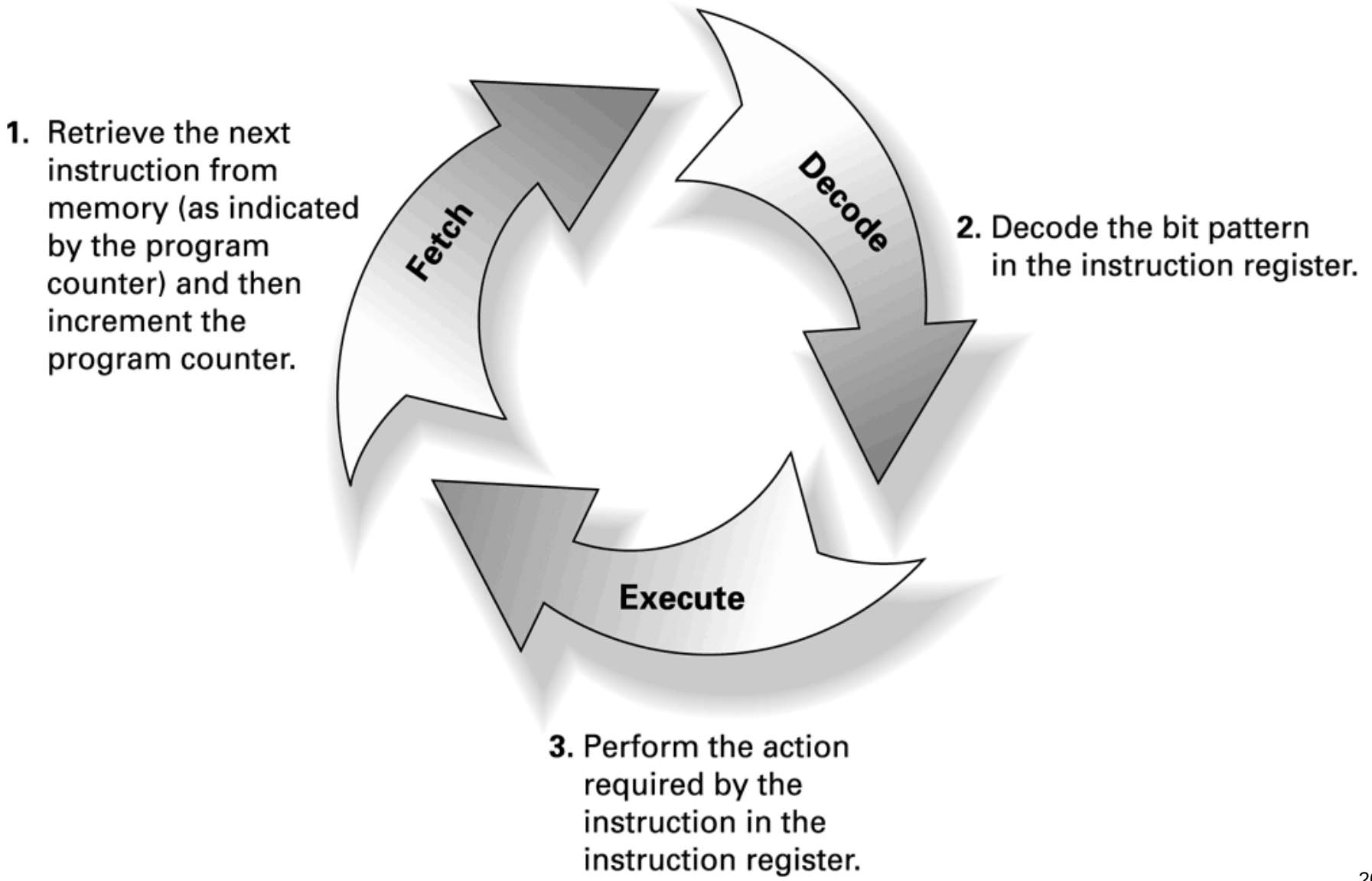


Figure 2.9 **Decoding the instruction B258**

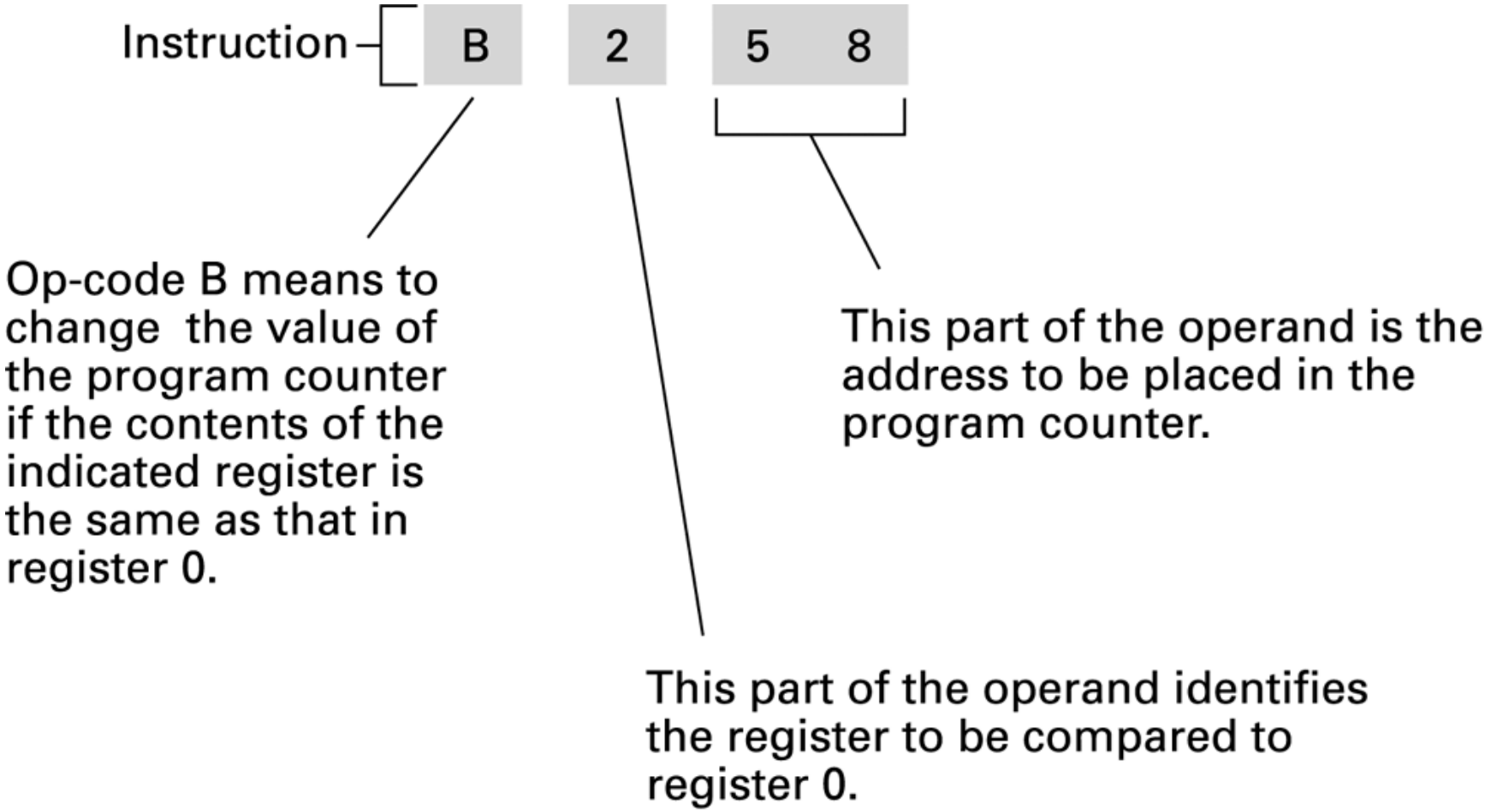


Figure 2.7 An encoded version of the instructions in Figure 2.2 (addition example)

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.

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

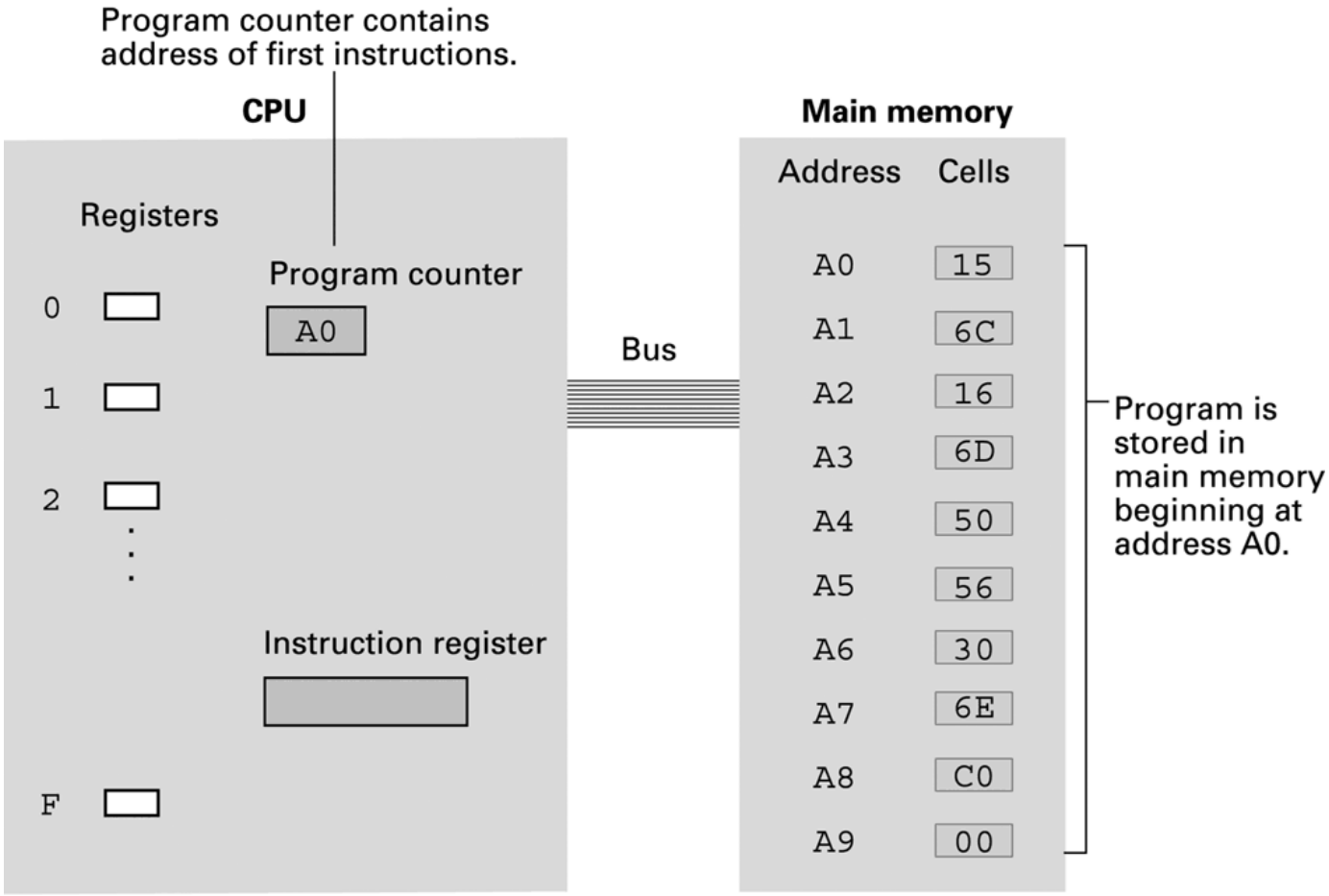
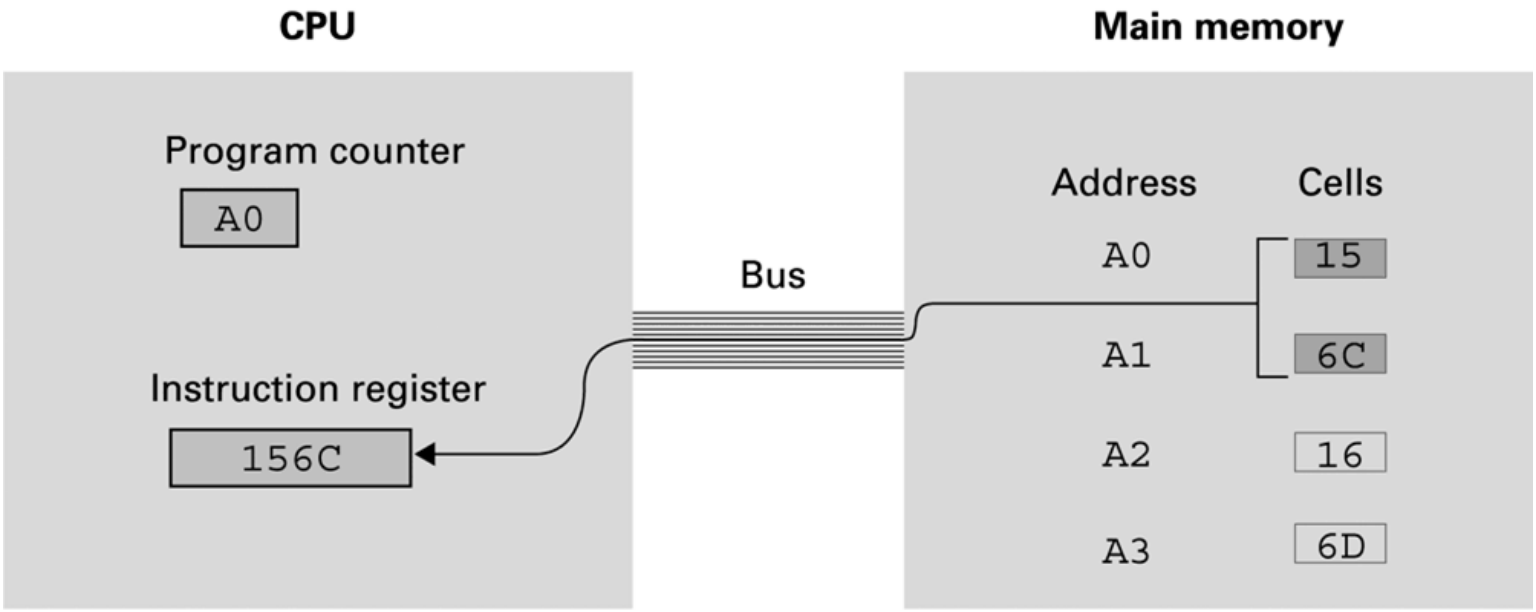
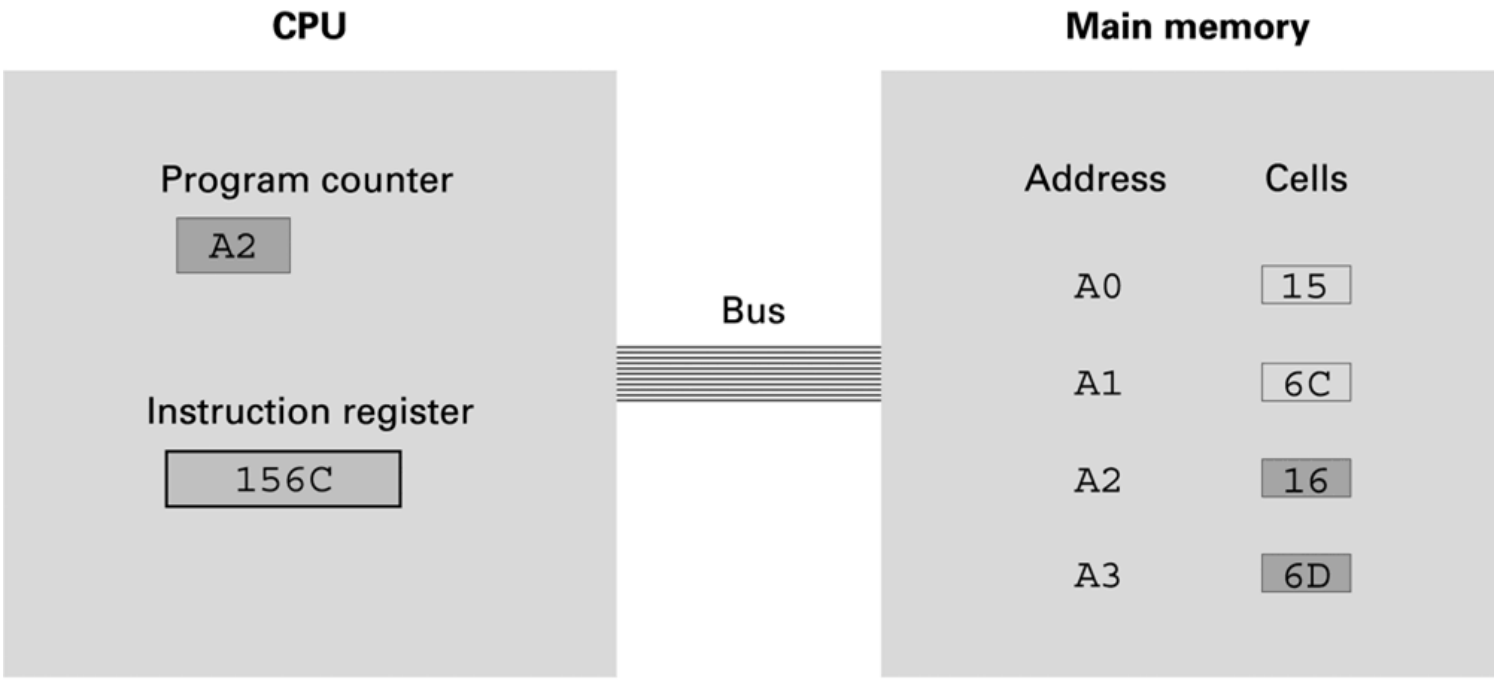


Figure 2.11 Performing the fetch step of the machine cycle



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 (cont'd)



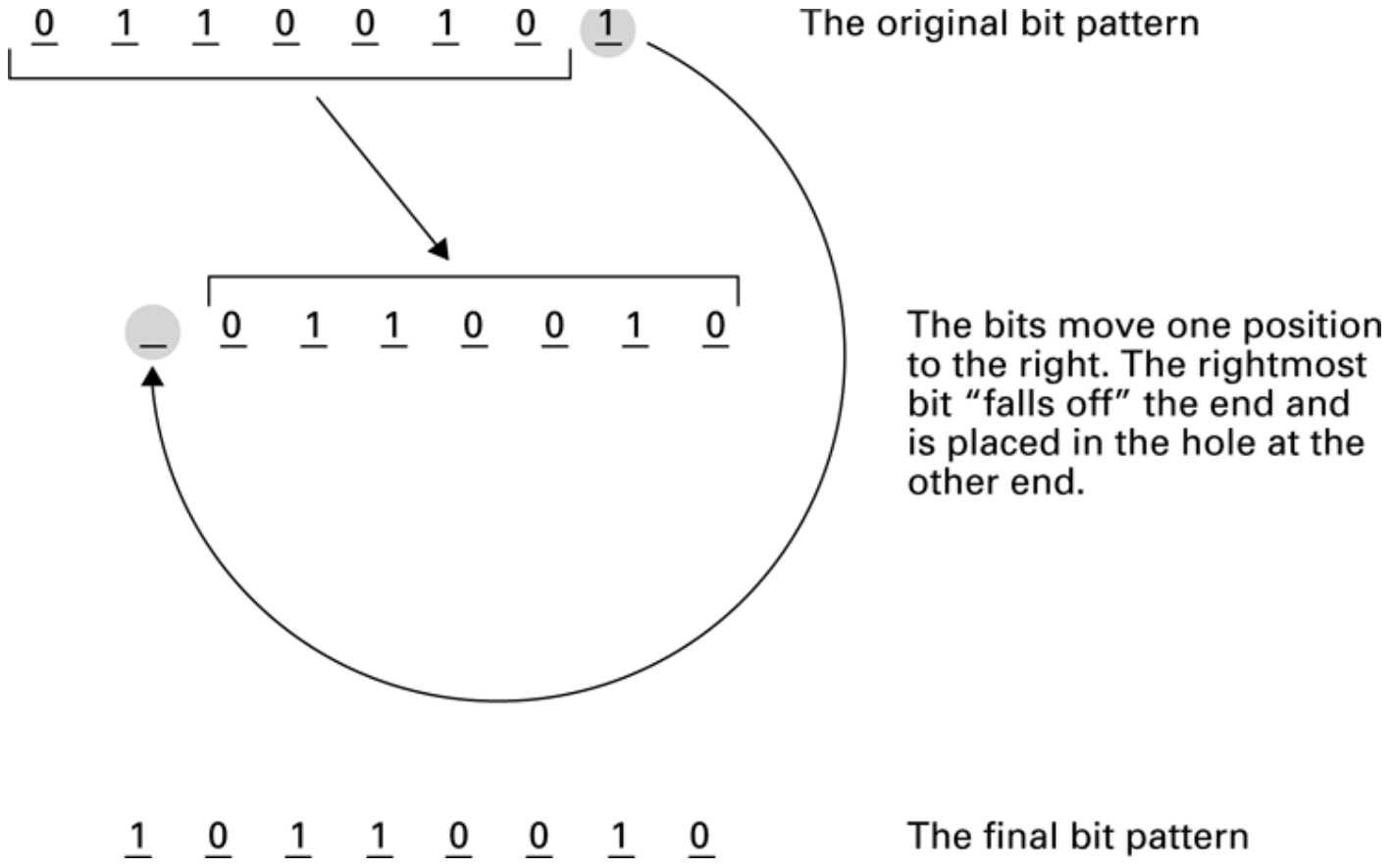
b. Then the program counter is incremented so that it points to the next instruction.

Programs versus Data

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**

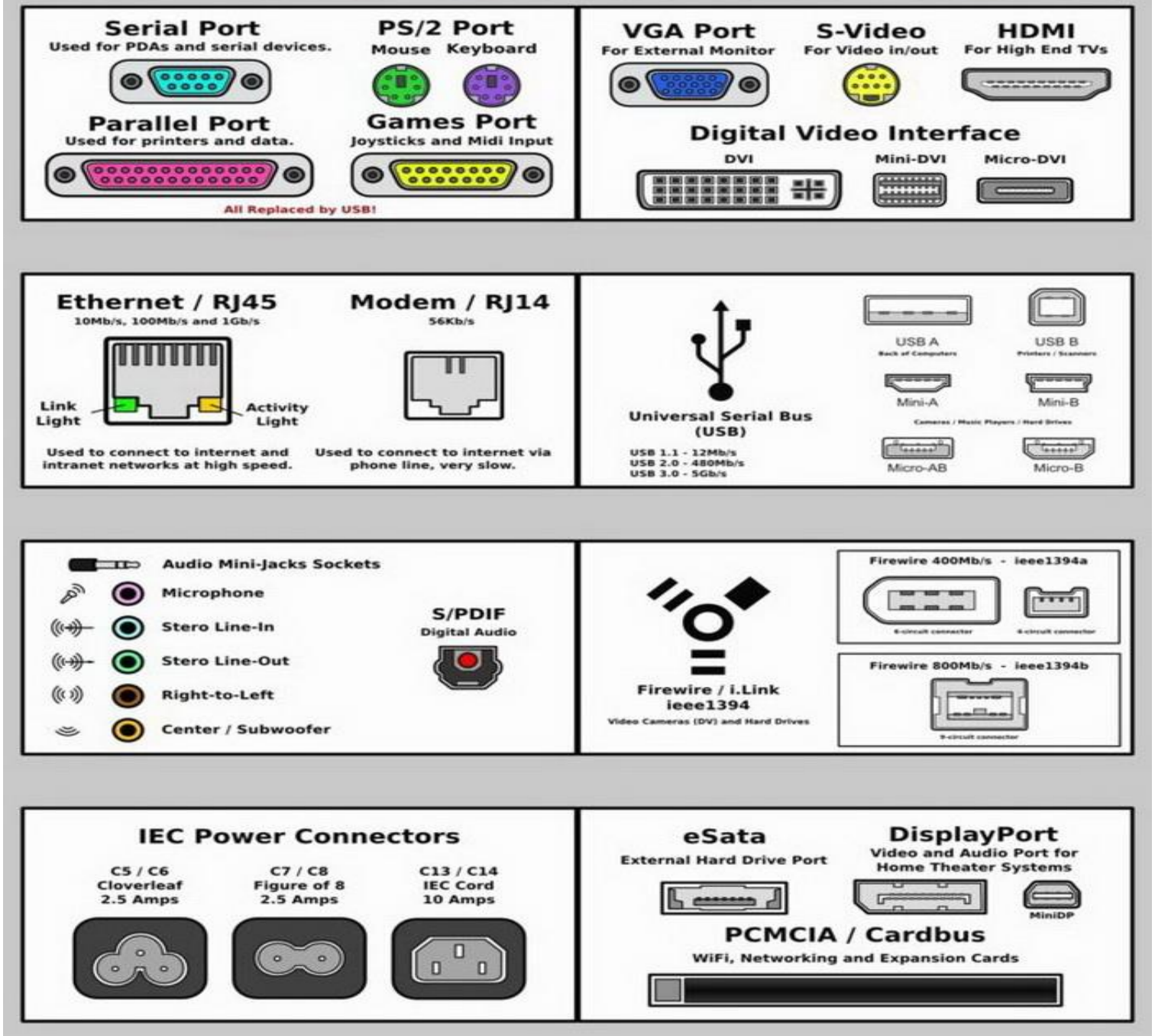


Communicating with Other Devices

- **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

COMPUTER PORTS IDENTIFICATION



- In computer networking, a *port* is an application-specific or process-specific software construct serving as a communications endpoint in a computer's host operating system.
- The purpose of ports is to uniquely identify different applications or processes running on a single computer and thereby enable them to share a single physical connection to a packet-switched network like the Internet. In the context of the Internet Protocol, a port is associated with an IP address of the host, as well as the type of protocol used for communication.

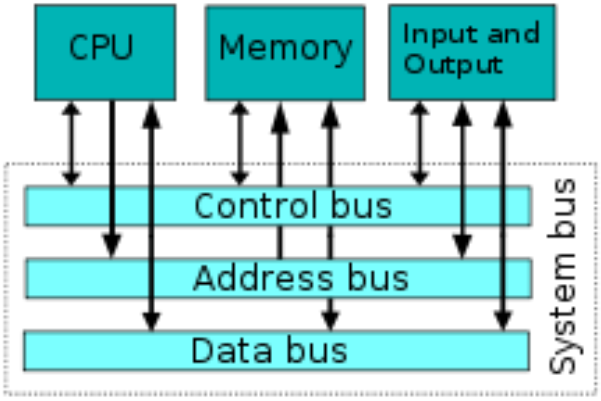
Memory-mapped I/O:

- CPU communicates with peripheral devices **as though they were memory cells**
- Uses the **same address bus and same instructions** to address both memory and I/O devices

Alternative: Programmed (Isolated) I/O:

There are **special instructions** for I/O operations

Figure 2.13 Controllers attached to a machine's bus



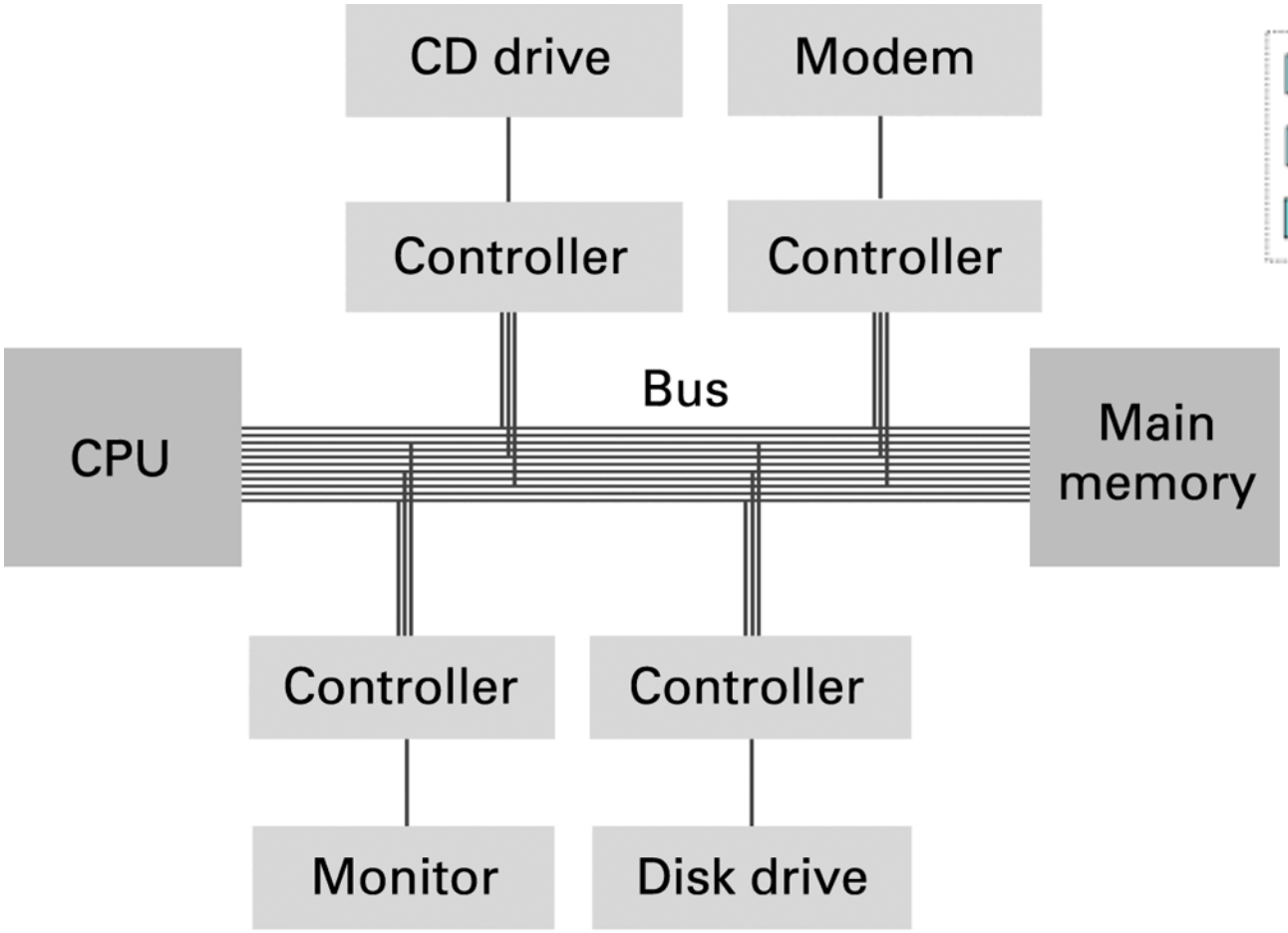
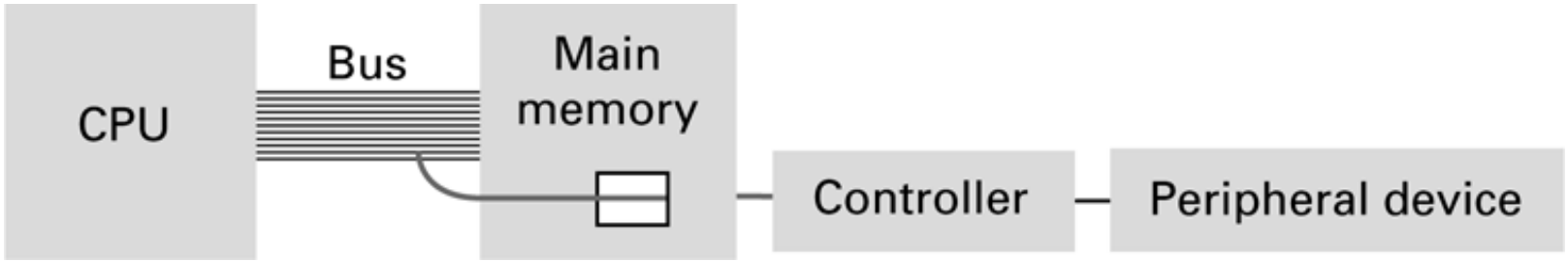


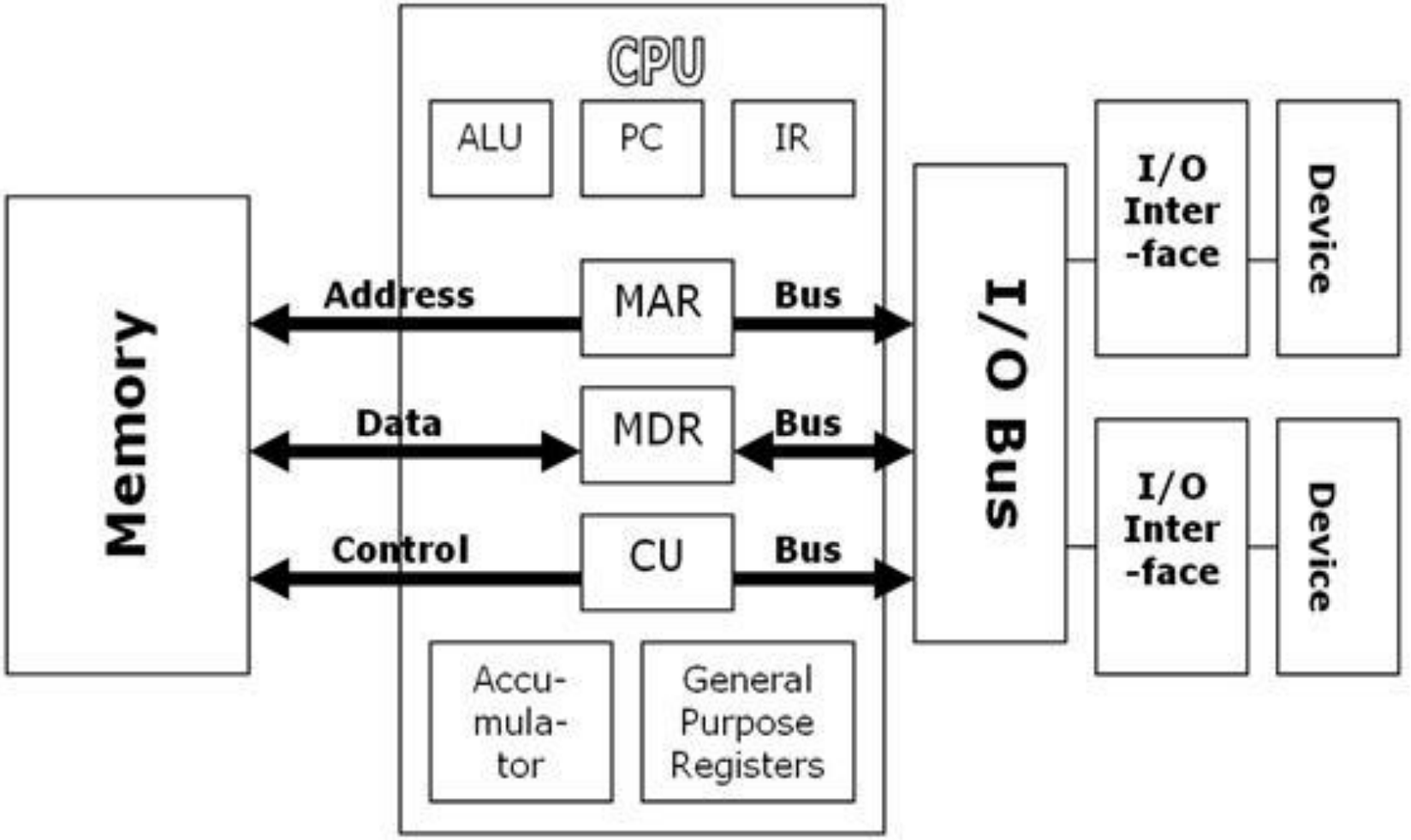
Figure 2.14 **A conceptual representation of memory-mapped I/O**



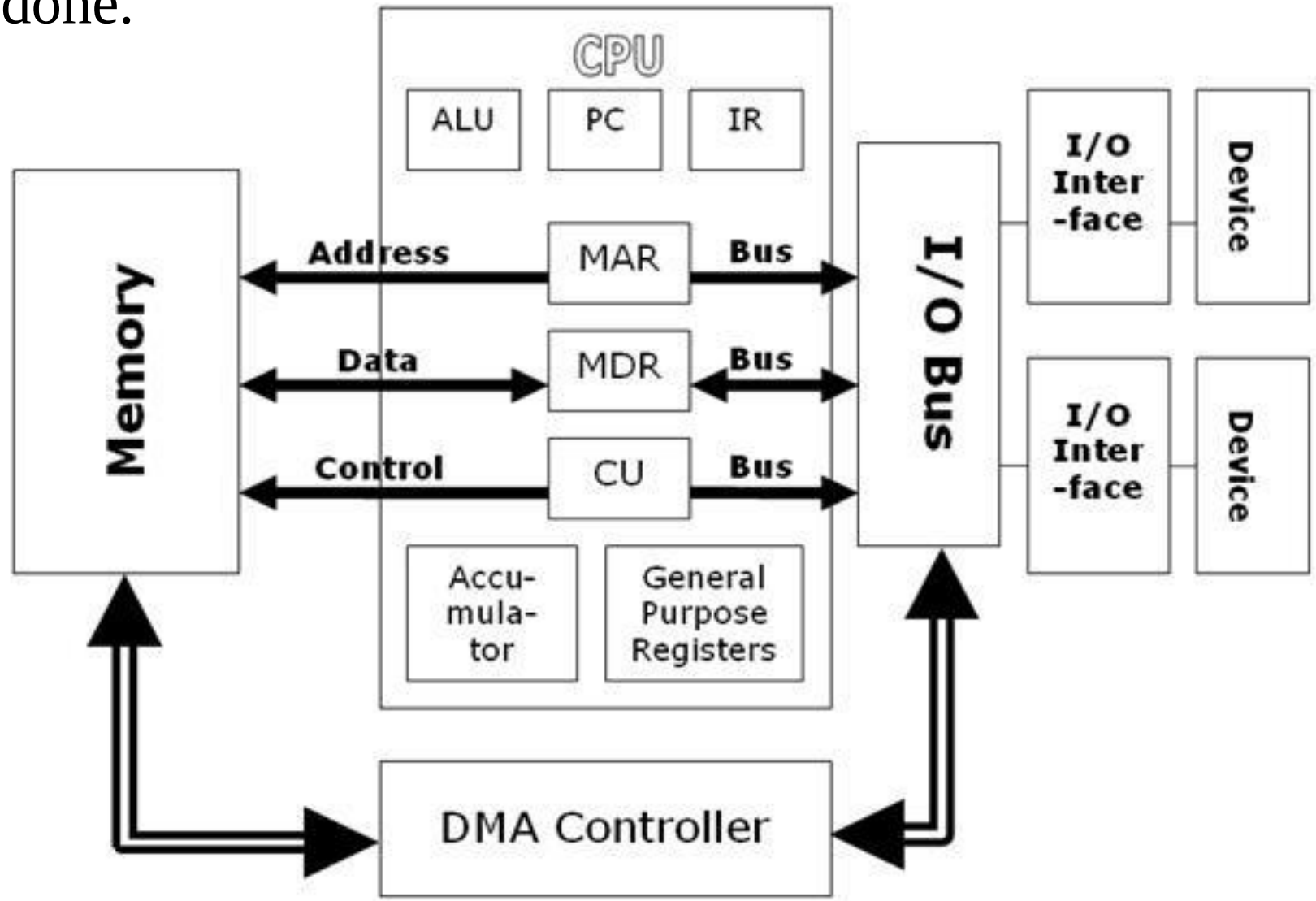
Communicating with Other Devices (continued)

- **Direct memory access (DMA): Main memory access by a controller over the bus**
- is a feature of modern computers that allows certain hardware subsystems within the computer to access system memory independently of CPU.

Without DMA, when CPU is using programmed I/O, it is typically fully occupied for the entire duration of R/W operation, and is thus unavailable to perform other work.



With DMA, the CPU **initiates** the transfer, does other operations while the transfer is in progress, and receives an **interrupt from the DMA controller** when the operation is done.



DMA feature is useful any time CPU cannot keep up with the rate of data transfer, or where CPU needs to perform useful work while waiting for a relatively slow I/O data transfer.

Many hardware systems use DMA
- disk drive controllers,
- graphics cards,
- network cards and sound cards,
- also used for intra-chip data transfer in multi-core processors.

Von Neumann Bottleneck

- The shared bus between the program memory and data memory leads to the ***Von Neumann bottleneck***,
- It is the limited throughput (data transfer rate) between the CPU and memory compared to the amount of memory.
- Because program memory and data memory cannot be accessed at the same time, throughput is much smaller than the rate at which the CPU can work.

- This seriously limits the effective processing speed when the CPU is required to perform minimal processing on large amounts of data.
- The CPU is continuously forced to wait for needed data to be transferred to or from memory. Since CPU speed and memory size have increased much faster than the throughput between them, the bottleneck has become more of a problem, a problem whose severity increases with every newer generation of CPU.

Other Architectures (than DMA)

- Technologies to increase throughput (the total amount of work the machine can accomplish in a given amount of time).
  - Cache memory
  - 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

Characteristics of Communication Systems

- must be a Sender and Receiver

- A protocol is a set of rules which governs the transfer of data between computers. Protocols allow communication between computers and networks.

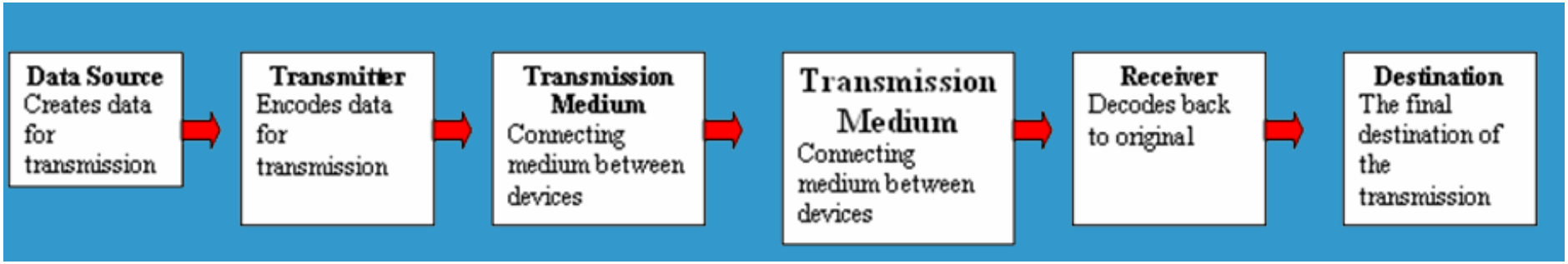
- protocols will determine the speed of transmission, error checking method, size of bytes, and whether synchronous or asynchronous

- Examples of protocols are: token ring, CSMA/CD, X.25, TCP/IP

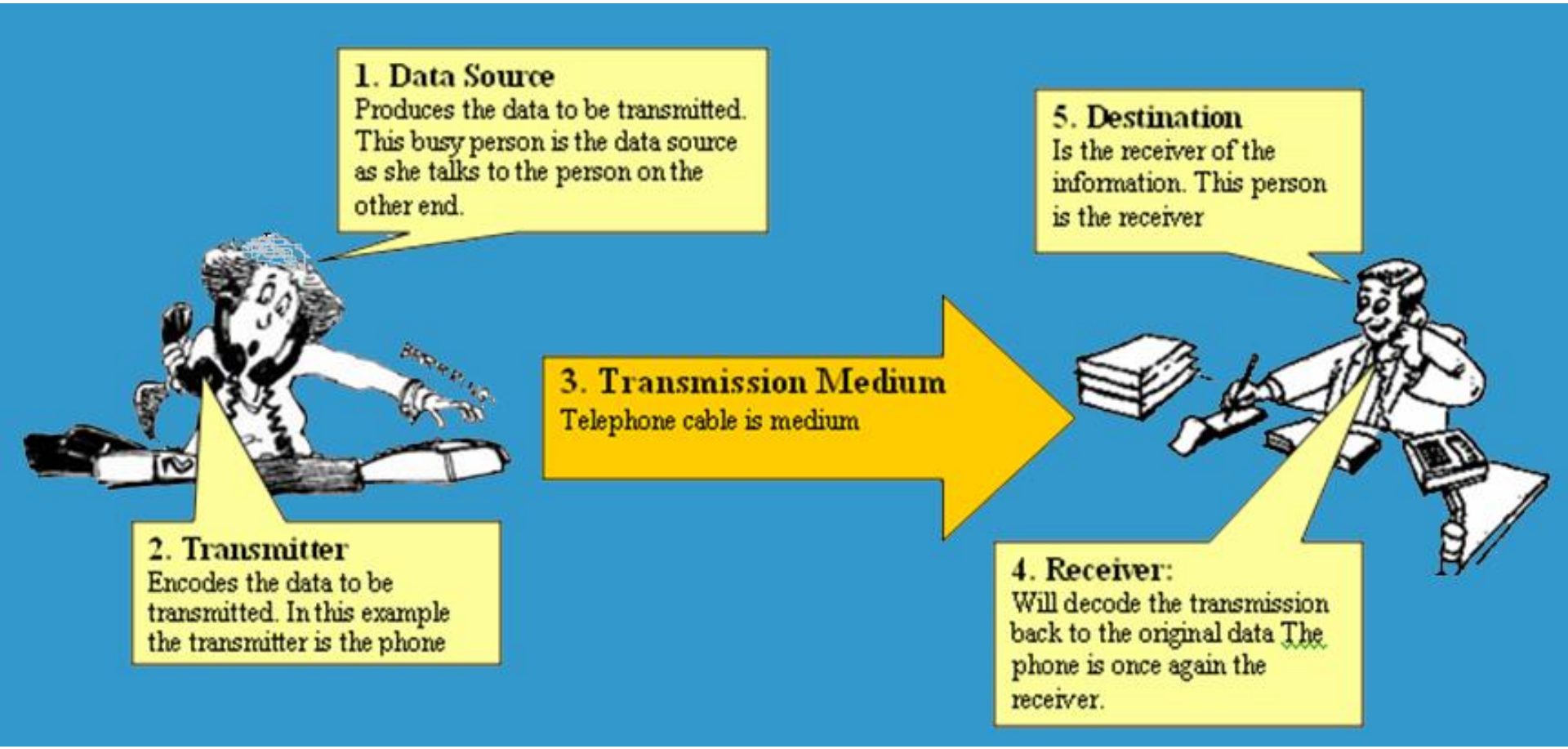
5 Basic Components

Every communication system has 5 basic requirements

- **Data Source** (where the data originates)
- **Transmitter** (device used to transmit data)
- **Transmission Medium** (cables or non cable)
- **Receiver** (device used to receive data)
- **Destination** (where the data will be placed)



5 Basic Components



Graham Betts

3. HAFTA Communication Concepts

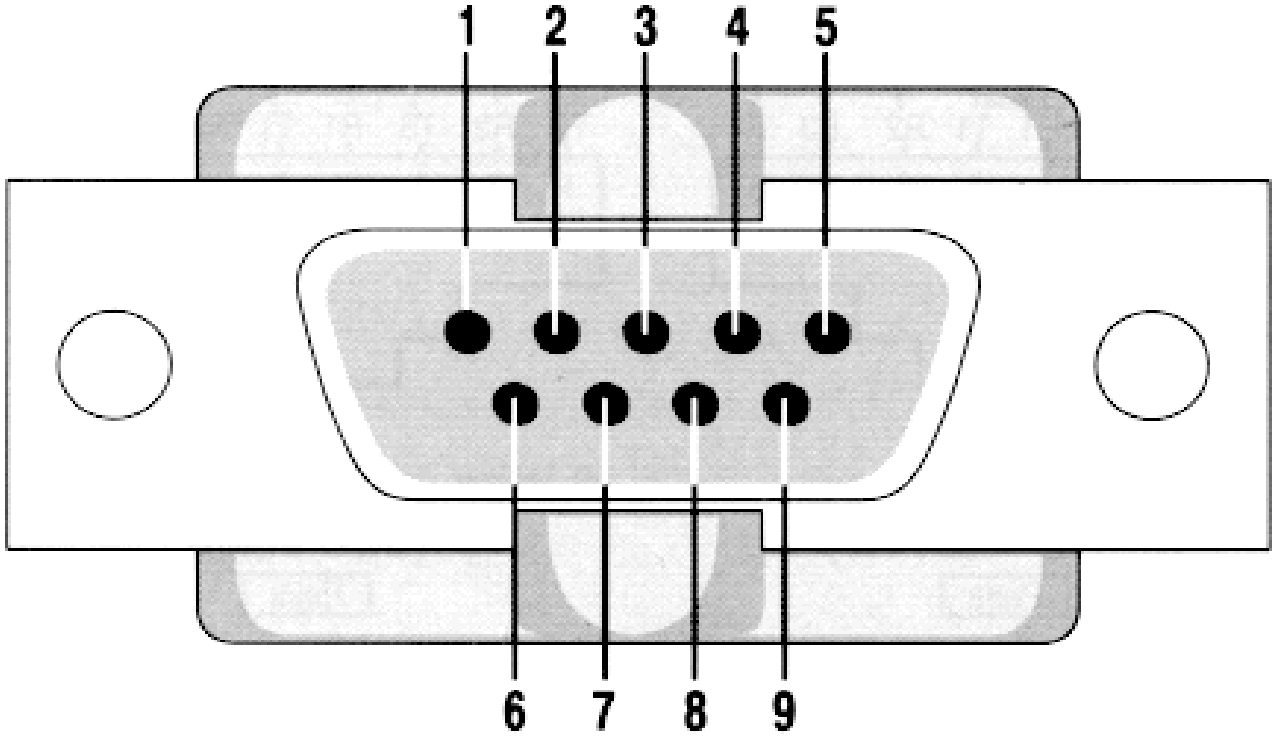
Any transmission May be:

- analog or digital
- Serial or parallel

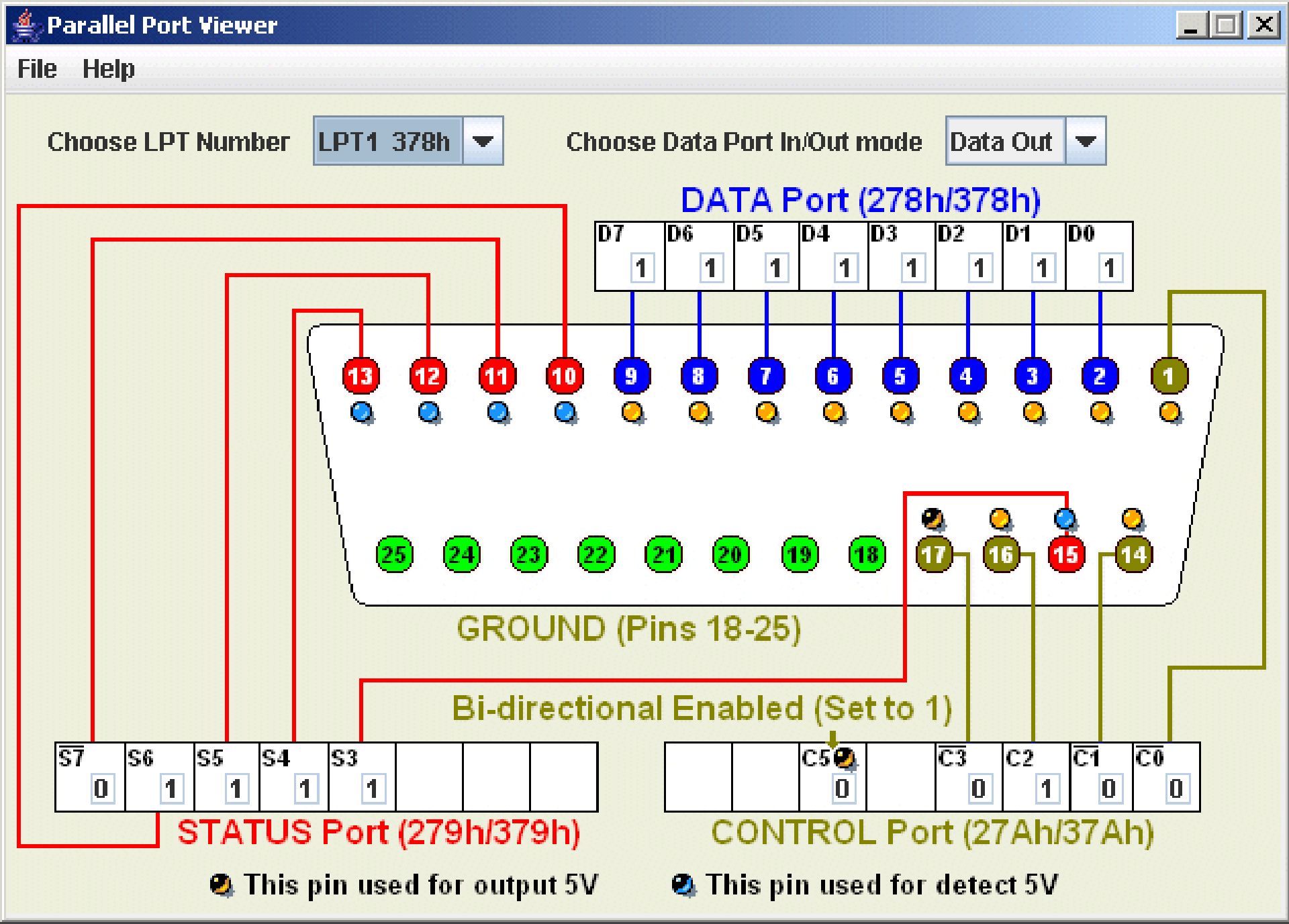
Communicating with Other Devices (continued)

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

Serial Port:



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		


Parallel Port Viewer
File
Help
Choose LPT Number
LPT1 378h
Choose Data Port In/Out mode
Data Out
DATA Port (278h/378h)
D7 1
D6 1
D5 1
D4 1
D3 1
D2 1
D1 1
D0 1
13 12 11 10 9 8 7 6 5 4 3 2 1
25 24 23 22 21 20 19 18 17 16 15 14
GROUND (Pins 18-25)
Bi-directional Enabled (Set to 1)
S7 0
S6 1
S5 1
S4 1
S3 1
STATUS Port (279h/379h)
C5 0
C3 0
C2 1
C1 0
C0 0
CONTROL Port (27Ah/37Ah)
This pin used for output 5V
This pin used for detect 5V

Handshaking:

- Handshaking is used to establish which protocols to use.
- Handshaking controls the **flow of data** between computers. It is the process of **coordinating the transfer of data** (even one way: printer) between components

Handshaking:

- computer and the peripheral device **exchange information** about the device's status and coordinate their activities
- Handshaking often involves a **status word**, **which is a bit pattern that is generated** by peripheral device and sent to the controller. Status word is a bit map in which the bits reflect the conditions of the device.

Transmission Media Speed

- **Bandwidth**:The maximum amount of data which can be transmitted on a medium over a fixed amount of time (second). It is measured on Bits per Second or Baud

- **Bits per Second (bps):** A measure of transmission speed. The number of bits (0 0r 1) which can be transmitted in a second

- **Baud Rate**: Is a measure of how fast a change of state occurs (i.e. a change from 0 to 1)

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)

56 kbit/s	Modem / Dialup
1.5 Mbit/s	ADSL Lite
1.544 Mbit/s	T1/DS1
2.048 Mbit/s	E1 / E-carrier
10 Mbit/s	Ethernet
11 Mbit/s	Wireless 802.11b
44.736 Mbit/s	T3/DS3
54 Mbit/s	Wireless 802.11g
100 Mbit/s	Fast Ethernet
155 Mbit/s	OC3
600 Mbit/s	Wireless 802.11n
622 Mbit/s	OC12
1 Gbit/s	Gigabit Ethernet
2.5 Gbit/s	OC48
9.6 Gbit/s	OC192
10 Gbit/s	10 Gigabit Ethernet
100 Gbit/s	100 Gigabit Ethernet

Examples of Communication Systems

- E-mail
- Voice Mail
- Smart Phone
- Telecommuting
- Groupware
- E-Commerce
- Bulletin board system
- Global positioning system
- Fax
- Instant Messaging
- Video-conferencing
- Telephony
- The Internet
- The Web