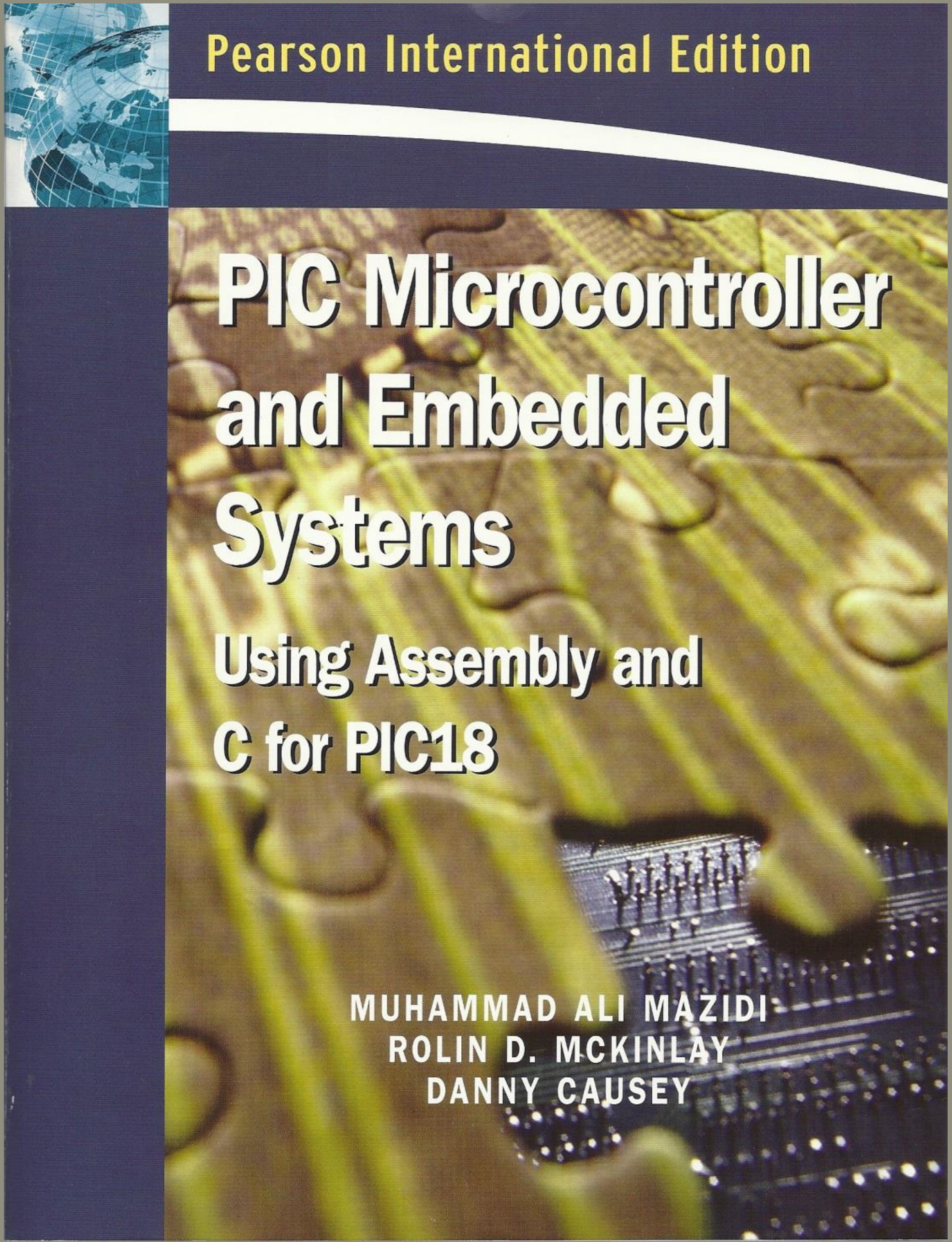
Chapter 0-1

Introduction to PIC Microcontroller


Pearson International Edition
PIC Microcontroller and Embedded Systems
Using Assembly and C for PIC18
MUHAMMAD ALI MAZIDI
ROLIN D. MCKINLAY
DANNY CAUSEY

Numbering and Coding systems

Human beings use base 10 (*decimal*) arithmetic.

Computers use the base 2 (*binary*) system.

The Binary system is based on powers of two.

The decimal system (base 10)

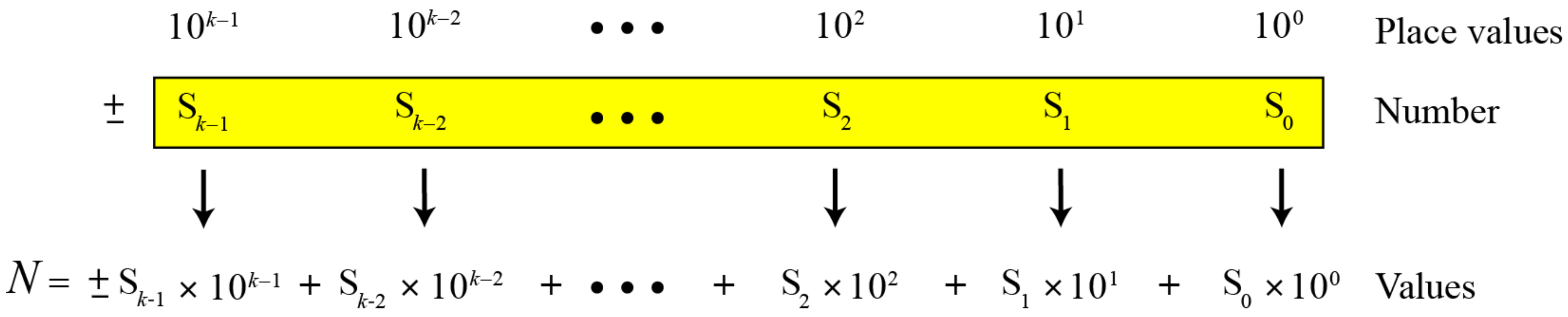
The word decimal is derived from the Latin root **decem** (ten). In this system the **base b = 10** and we use ten symbols

$$\boxed{S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}}$$

The symbols in this system are often referred to as **decimal digits** or just **digits**.

Integers

$$N = \pm \quad S_{k-1} \times 10^{k-1} + S_{k-2} \times 10^{k-2} + \ldots + S_2 \times 10^2 + S_1 \times 10^1 + S_0 \times 10^0$$



10^{k-1} 10^{k-2} • • • 10^2 10^1 10^0 Place values
$\pm$ S_{k-1} S_{k-2} • • • S_2 S_1 S_0 Number
$N = \pm S_{k-1} \times 10^{k-1} + S_{k-2} \times 10^{k-2} + \bullet\bullet\bullet + S_2 \times 10^2 + S_1 \times 10^1 + S_0 \times 10^0$ Values

Figure 2.1 **Place values for an integer in the decimal system**

The binary system (base 2)

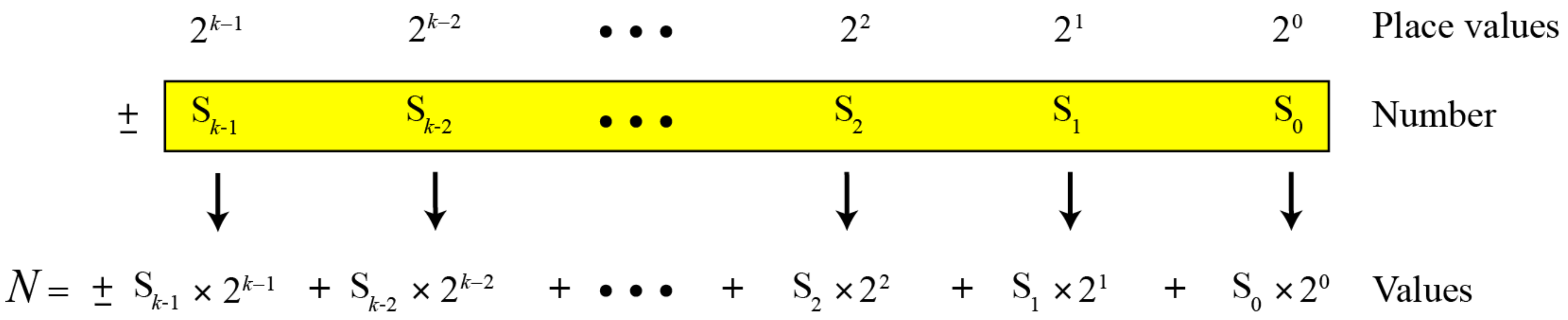
The word binary is derived from the Latin root **bini** (or two by two). In this system the **base b = 2** and we use only two symbols,

$$\boxed{\mathbf{S = \{0, 1\}}}$$

The symbols in this system are often referred to as **binary digits** or **bits** (binary digit).

Integers

$$N = \pm\; S_{k-1} \times 2^{k-1} + S_{k-2} \times 2^{k-2} + \ldots + S_2 \times 2^2 + S_1 \times 2^1 + S_0 \times 2^0$$



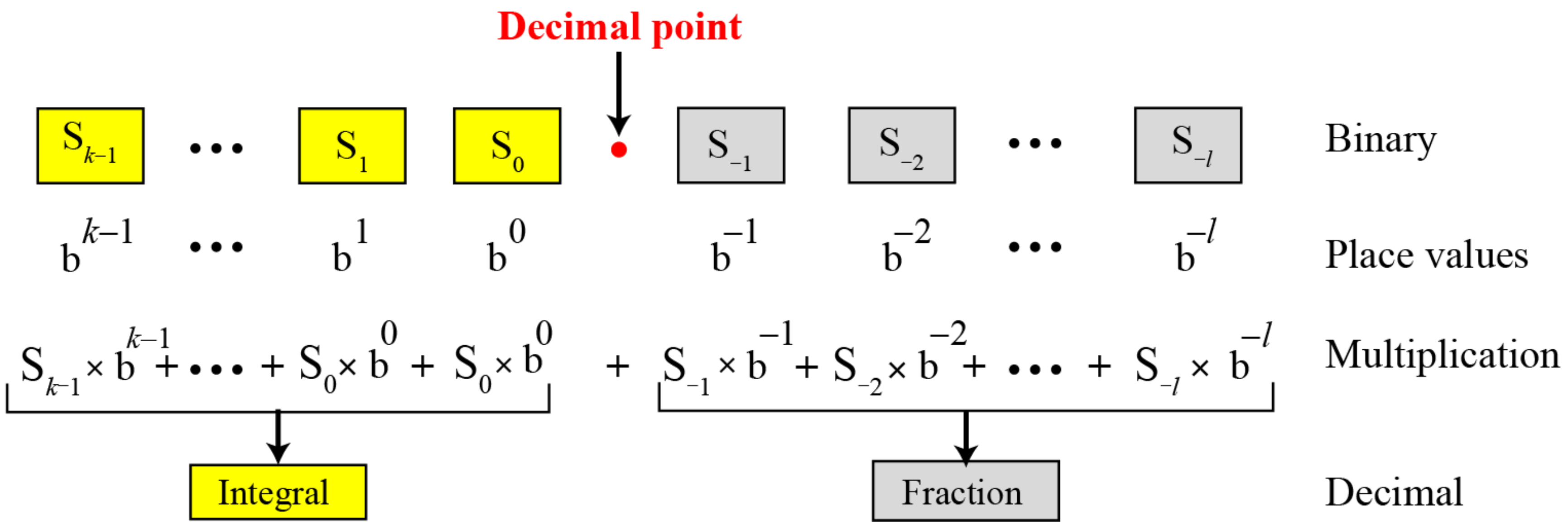
2^{k-1} 2^{k-2} • • • 2^2 2^1 2^0 Place values
$\pm$ S_{k-1} S_{k-2} • • • S_2 S_1 S_0 Number
$N = \pm\; S_{k-1} \times 2^{k-1} + S_{k-2} \times 2^{k-2} + \bullet\bullet\bullet + S_2 \times 2^2 + S_1 \times 2^1 + S_0 \times 2^0$ Values

Figure 2.2 **Place values for an integer in the binary system**

Conversion

We need to know how to convert a number in one system to the equivalent number in another system.

Any base to decimal conversion



Decimal point
S_{k-1} ••• S_1 S_0 • S_{-1} S_{-2} ••• S_{-l}
Binary
b^{k-1} ••• b^1 b^0 b^{-1} b^{-2} ••• b^{-l}
Place values
$S_{k-1} \times b^{k-1} + \cdots + S_0 \times b^0 + S_0 \times b^0 + S_{-1} \times b^{-1} + S_{-2} \times b^{-2} + \cdots + S_{-l} \times b^{-l}$
Multiplication
Integral
Fraction
Decimal

Figure 2.5 **Converting other bases to decimal**

Example 2.8

The following shows how to convert the binary number $(110.11)_2$ to decimal: $(110.11)_2 = 6.75$.

Binary	1		1		0	•	1		1
Place values	2^2		2^1		2^0		2^{-1}		2^{-2}
Partial results	4	+	2	+	0	+	0.5	+	0.25

Decimal: 6.75

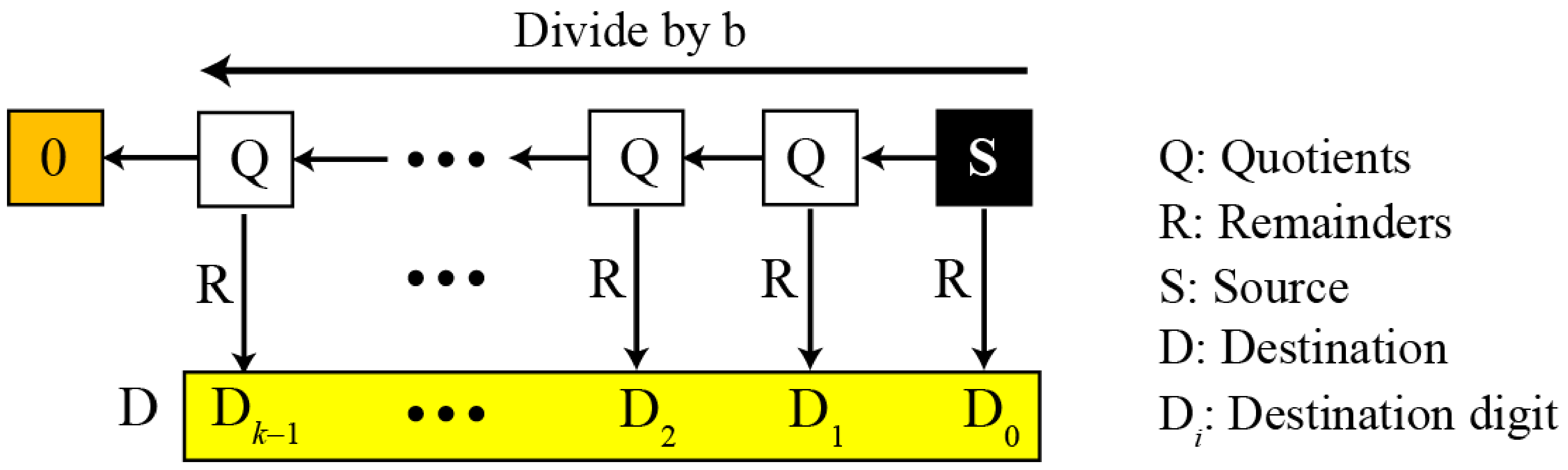
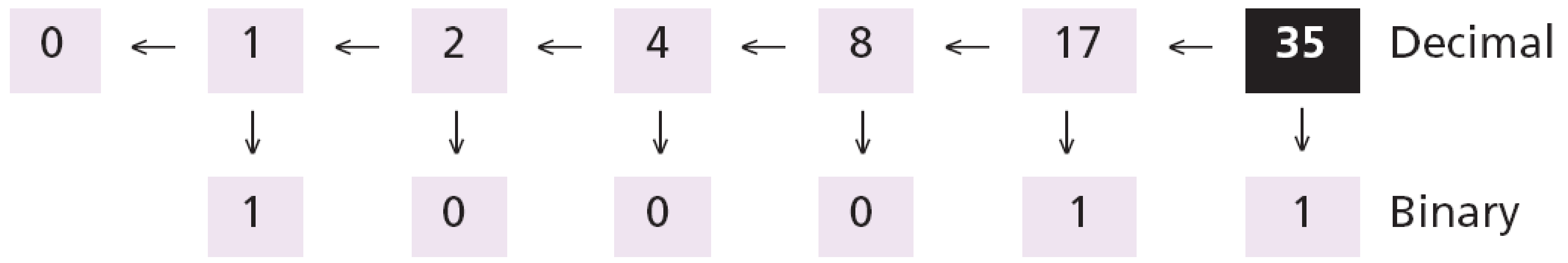
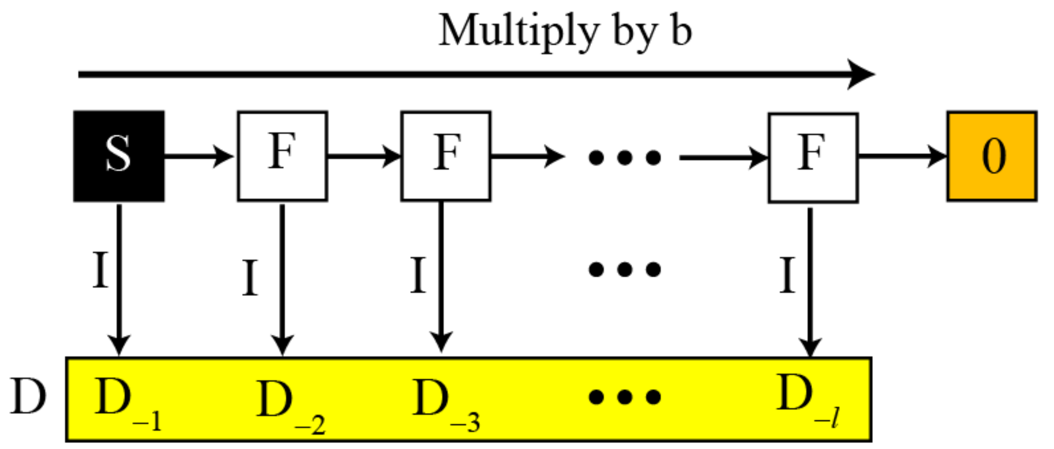


Figure 2.7 **Converting the integral part of a number in decimal to other bases**

Example 2.11

The following shows how to convert 35 in decimal to binary. We start with the number in decimal, we move to the left while continuously finding the quotients and the remainder of division by 2. The result is 35 = $(100011)_2$.





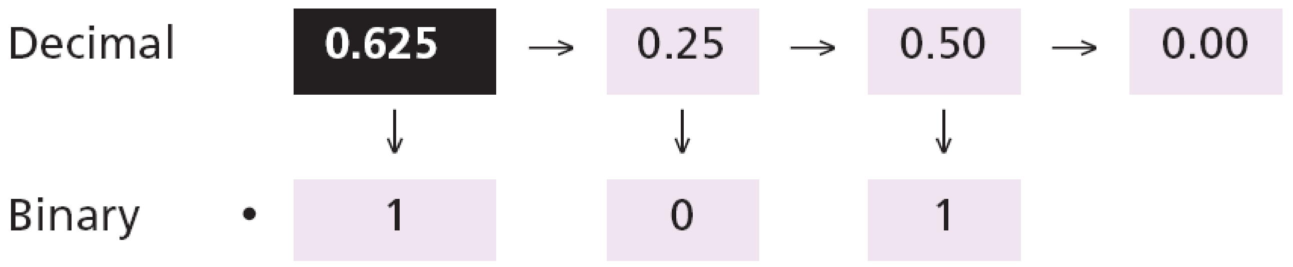
I: Integral part
F: Fractional part
S: Source
D: Destination
D_i: Destination digit

Note:
The fraction may never become zero.
Stop when enough digits have been created.

Figure 2.9 **Converting the fractional part of a number in decimal to other bases**

Example 2.14

Convert the decimal number 0.625 to binary.



Since the number $0.625 = (0.101)_2$ has no integral part, the example shows how the fractional part is calculated.

Binary-hexadecimal conversion

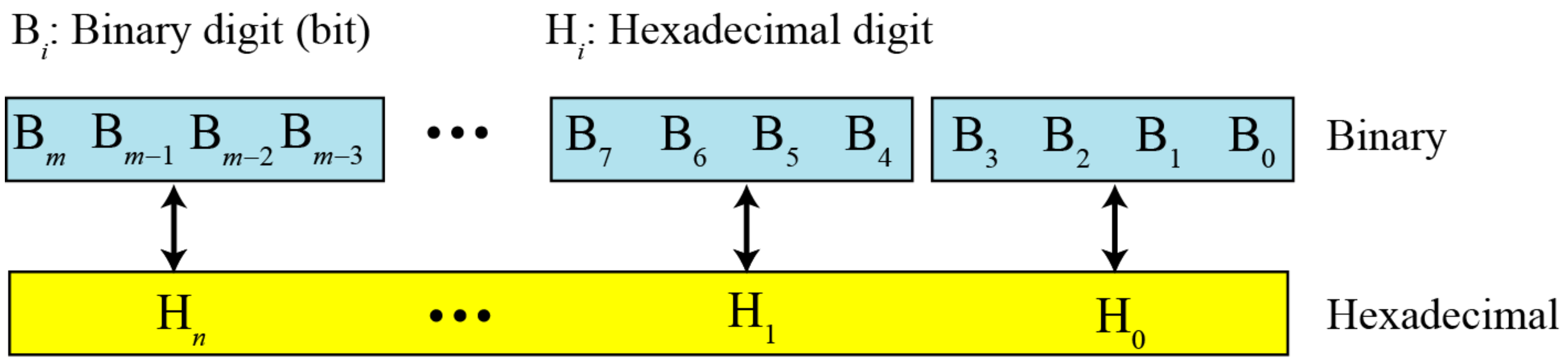


Figure 2.10 **Binary to hexadecimal and hexadecimal to binary conversion**

Example 2.19

Show the hexadecimal equivalent of the binary number $(10011100010)_2$.

Solution

We first arrange the binary number in 4-bit patterns:

100 1110 0010

Note that the leftmost pattern can have one to four bits. We then use the equivalent of each pattern shown in Table 2.2 on page 25 to change the number to hexadecimal: (4E2)16.

Example 2.20

What is the binary equivalent of $(24C)_{16}$?

Solution

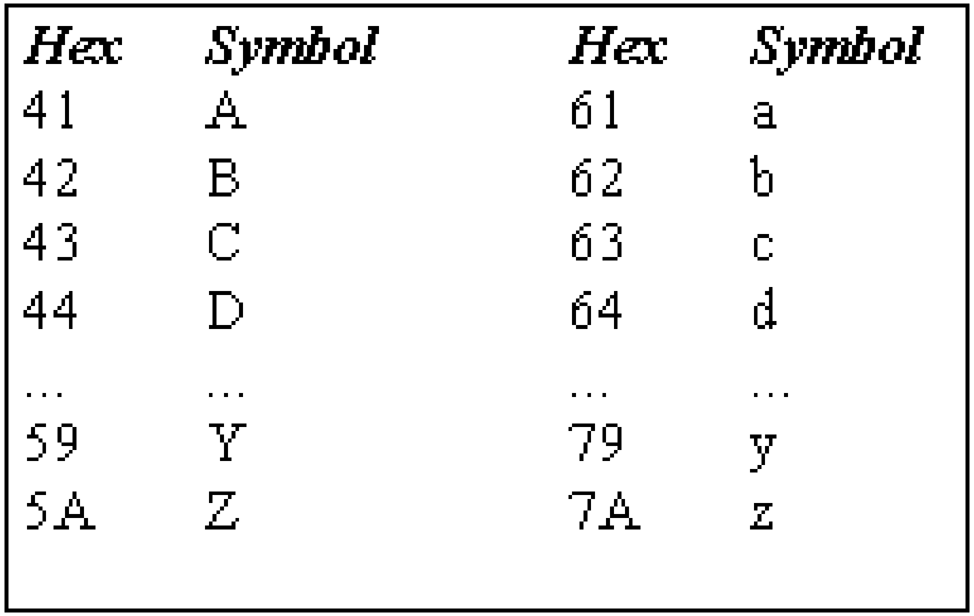
Each hexadecimal digit is converted to 4-bit patterns:

$$\mathbf{2 \rightarrow 0010,\ 4 \rightarrow 0100,\ and\ C \rightarrow 1100}$$

The result is $(001001001100)_2$.

ASCII Codes

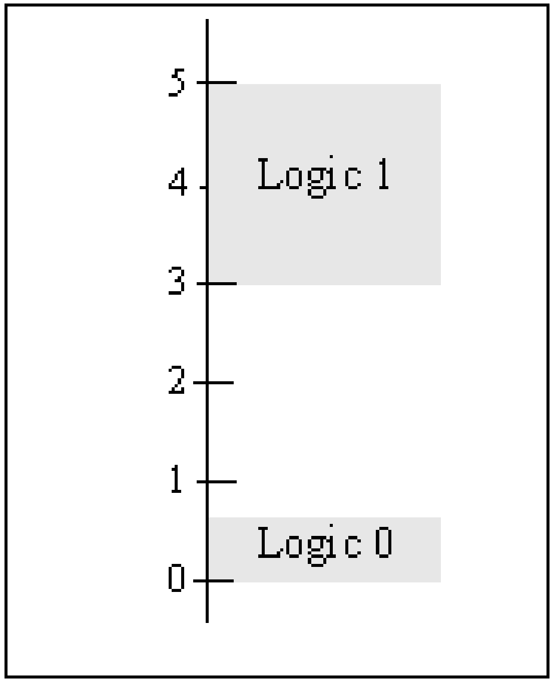
Figure 0-1. Selected ASCII Codes



Hex	Symbol	Hex	Symbol
41	A	61	a
42	B	62	b
43	C	63	c
44	D	64	d
…	…	…	…
59	Y	79	y
5A	Z	7A	z

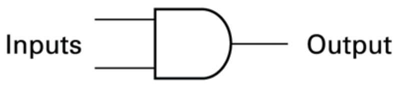
Digital Primer

Figure 0-2. Binary Signals



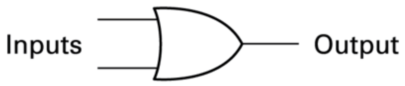
A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values

AND



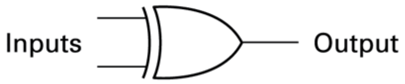
Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR



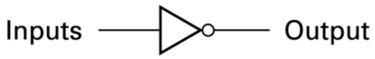
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR



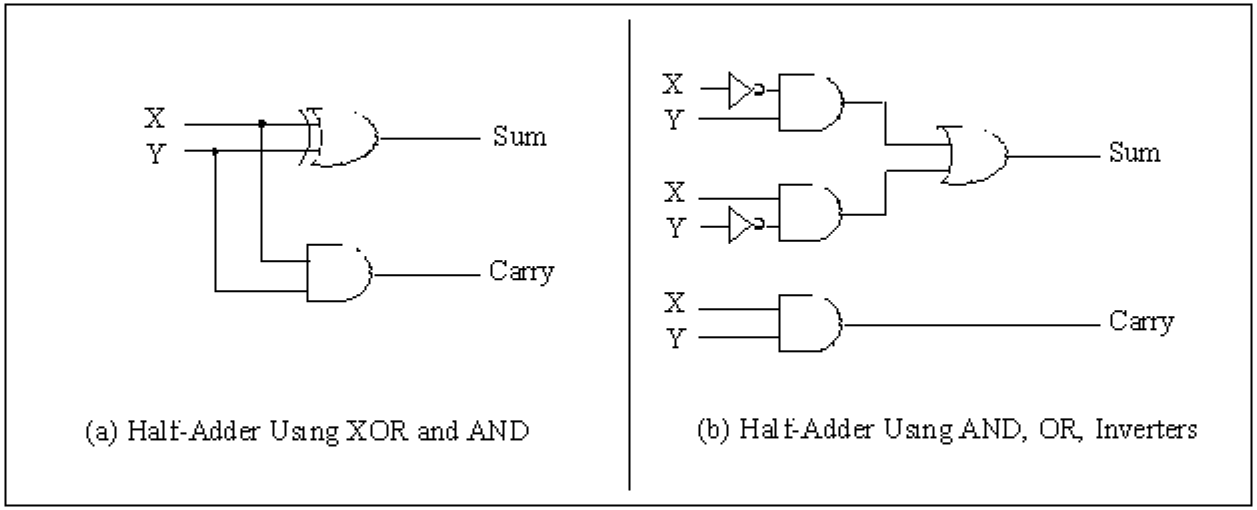
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

NOT

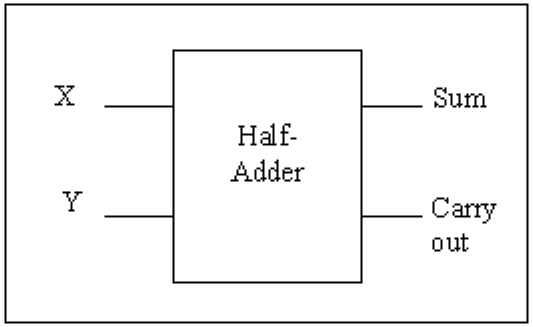


Inputs	Output
0	1
1	0

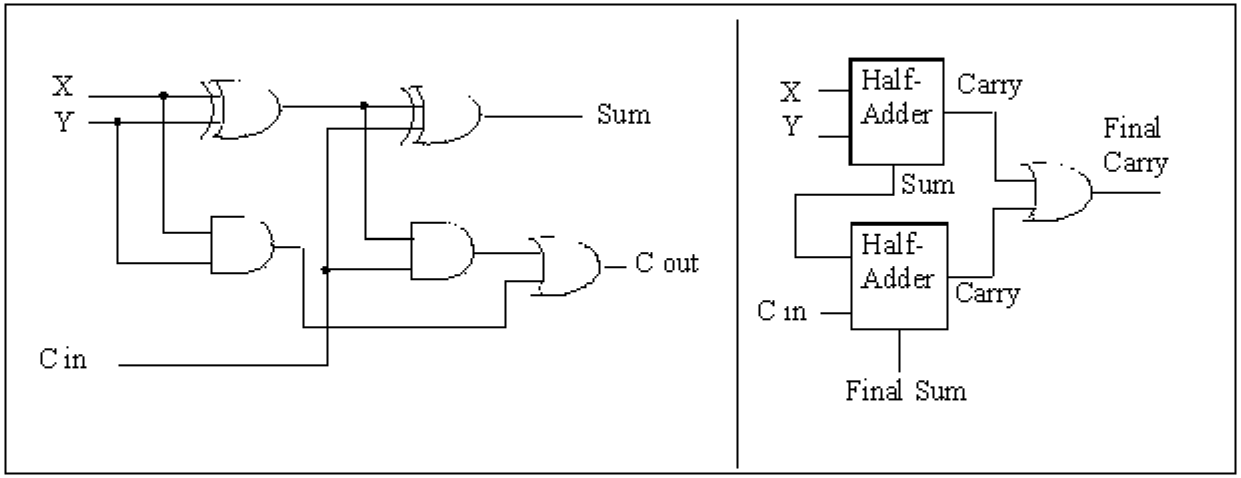
Two Implementations of a Half-Adder


X
Y
Sum
Carry
(a) Half-Adder Using XOR and AND
X
Y
X
Y
X
Y
Sum
Carry
(b) Half-Adder Using AND, OR, Inverters

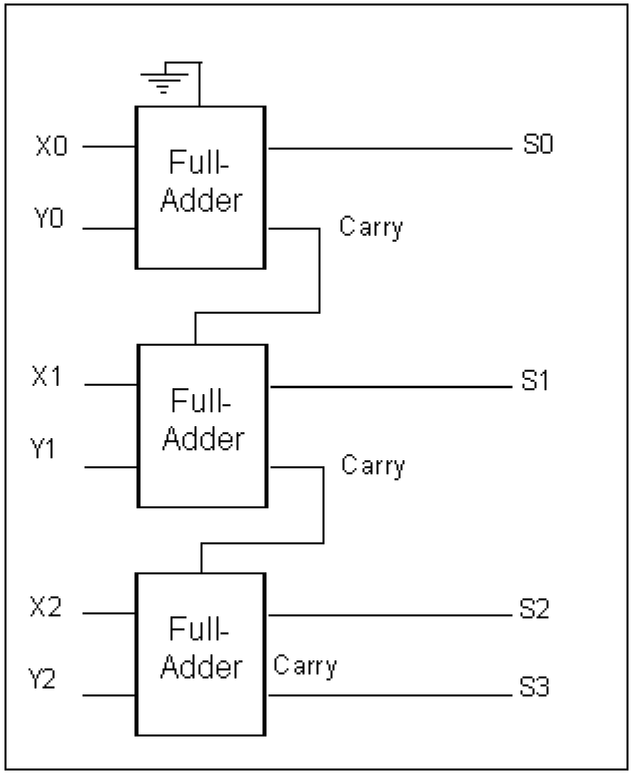
Block Diagram of a Half-Adder



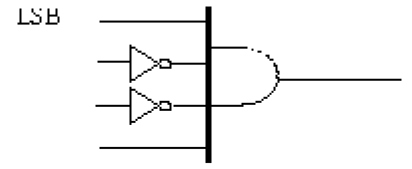
Full-Adder Built From a Half-Adder



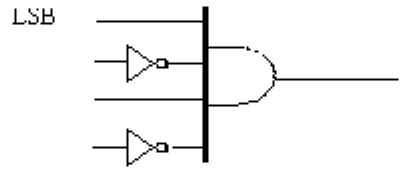
3-Bit Adder Using Three Full-Adders



Address Decoders

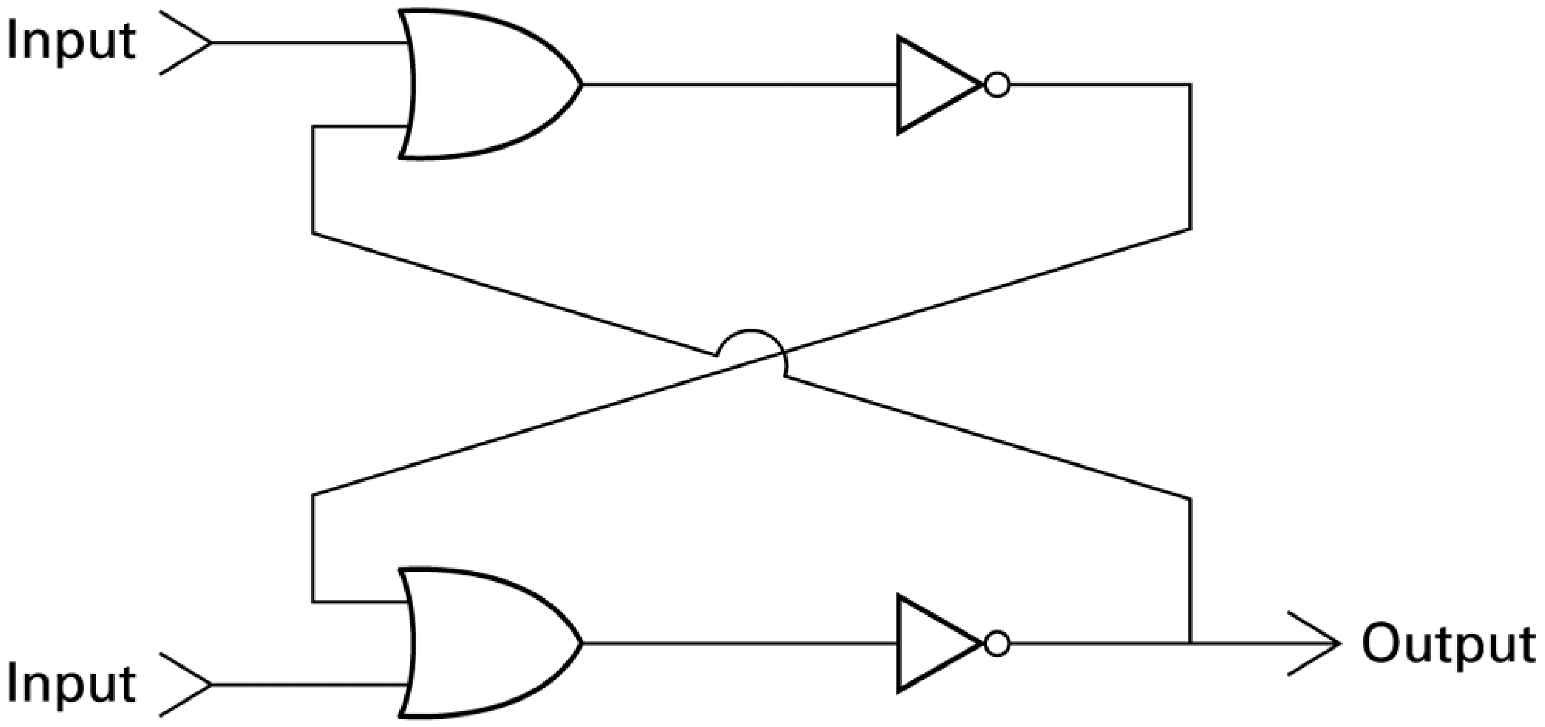


(a) Address decoder for 9 (binary 1001)
The output of the AND gate will be 1
if and only if the input is binary 1001.

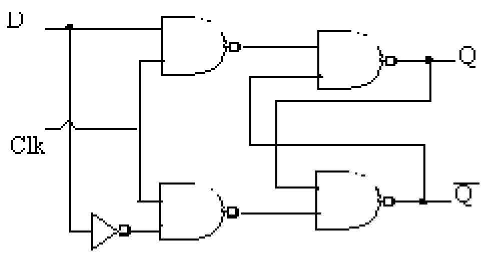


(b) Address decoder for 5 (binary 0101)
The output of the AND gate will be 1
if and only if the input is binary 0101.

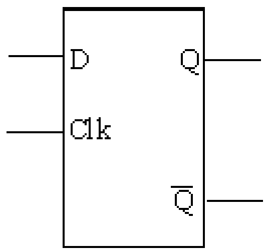
Constructing a flip-flop



D Flip-Flops



(a) Circuit diagram



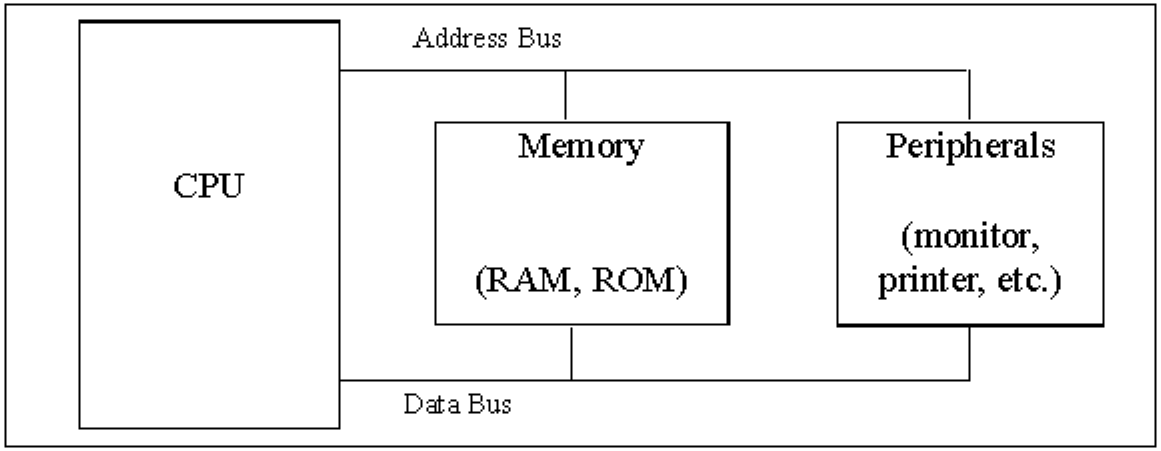
(b) Block diagram

Clk	D	Q
No	x	no change
↓	0	0
↓	1	1

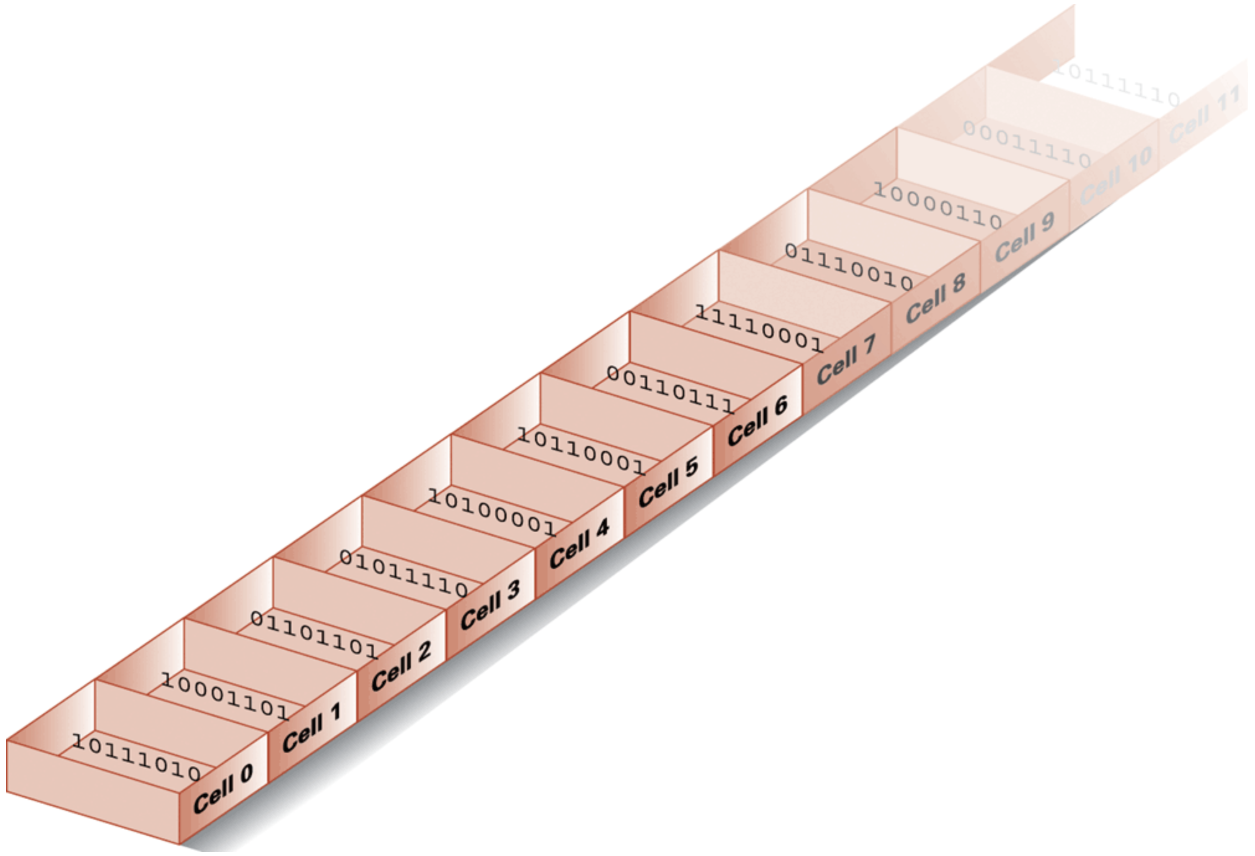
x = don't care

(c) Truth table

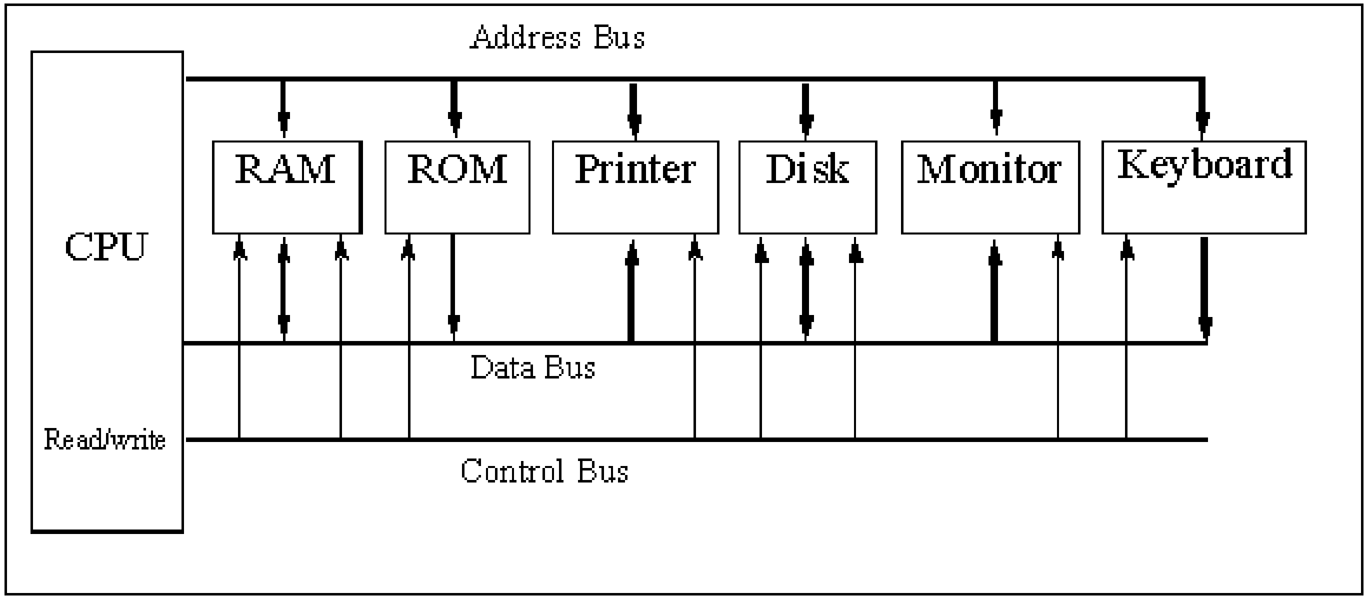
Inside the Computer



Memory cells arranged by address



Internal Organization of a Computer



Internal Block Diagram of a CPU

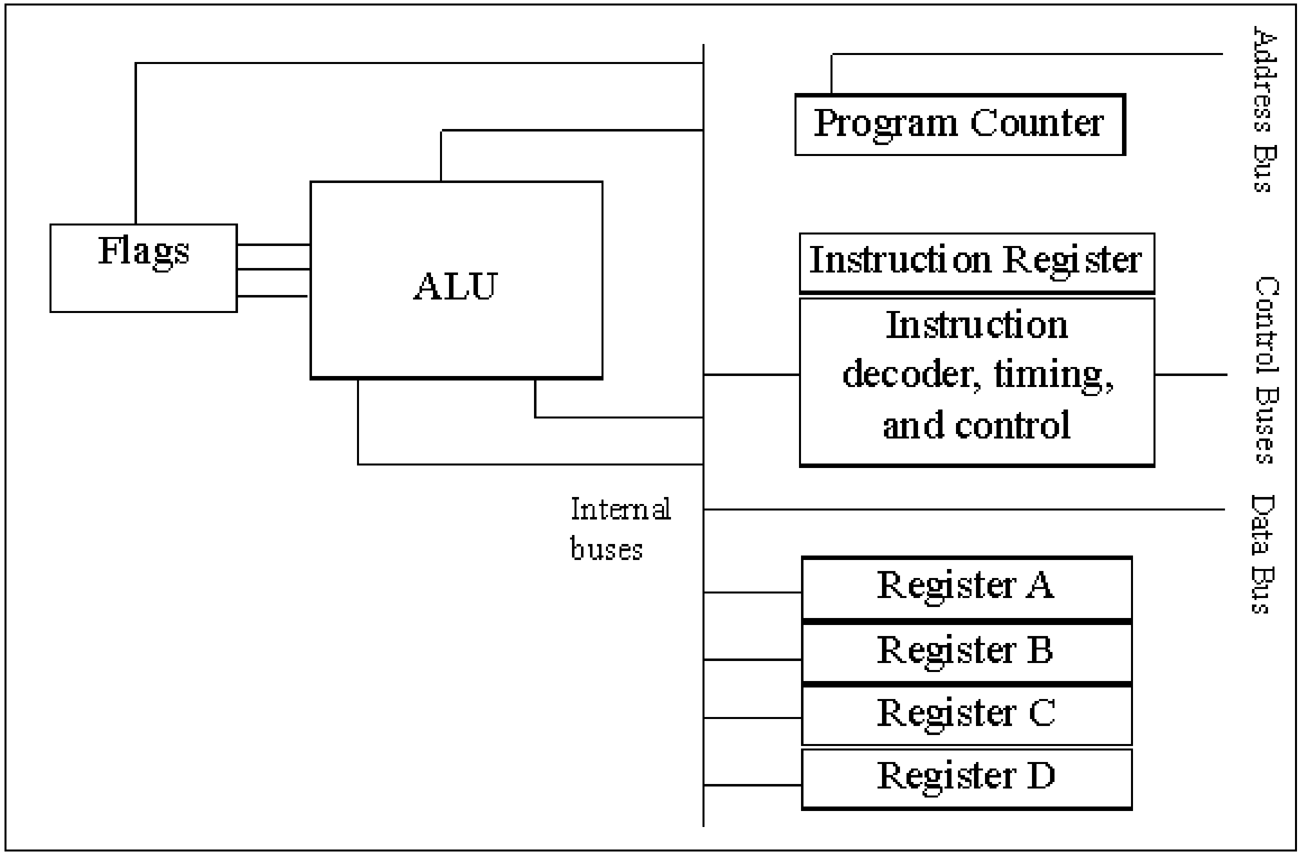
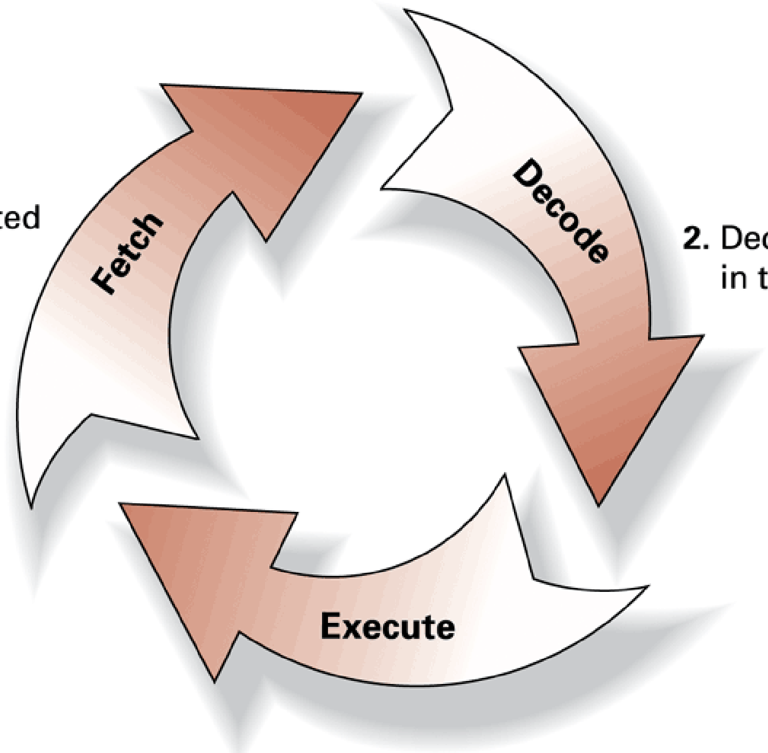


Figure 2.8 The machine cycle

1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.



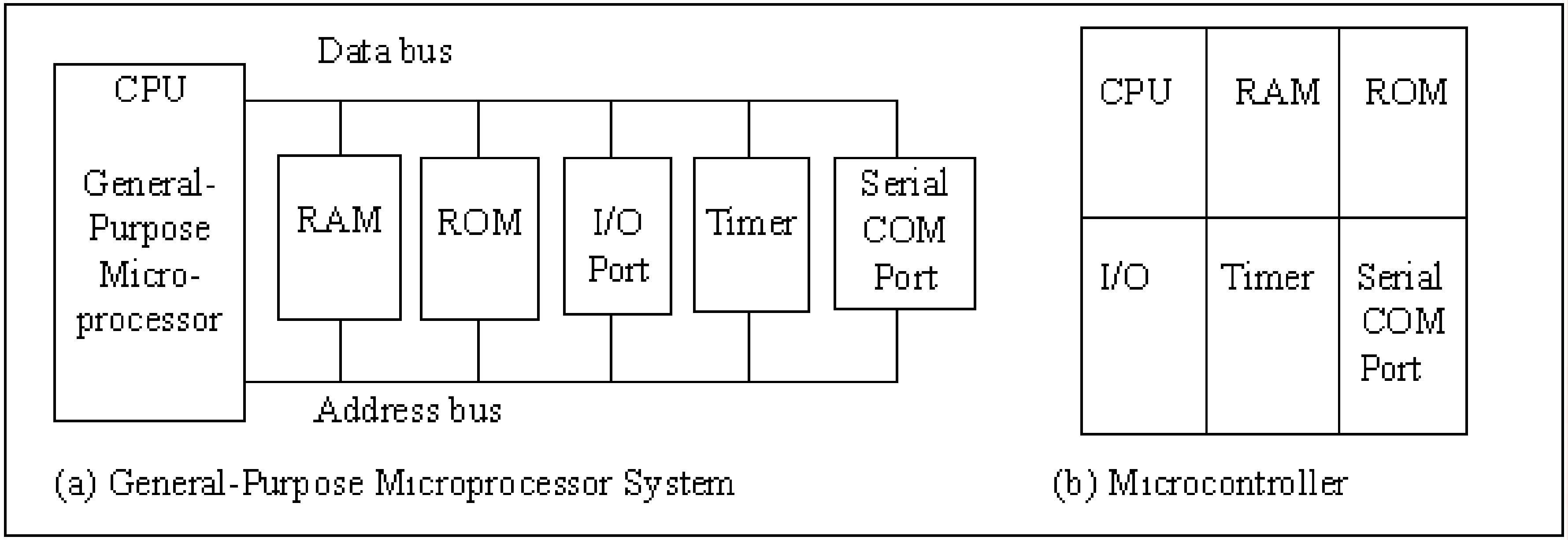
2. Decode the bit pattern in the instruction register.

3. Perform the action required by the instruction in the instruction register.

Microprocessor vs. microcontroller (MCU)

- Microprocessor
  - A processor implemented on a very large scale integration (VLSI) chip
  - Peripheral chips are needed to construct a product
- Microcontroller
  - The processor and peripheral functions implemented on one VLSI chip
- Embedded systems
- SOC (system on chip)

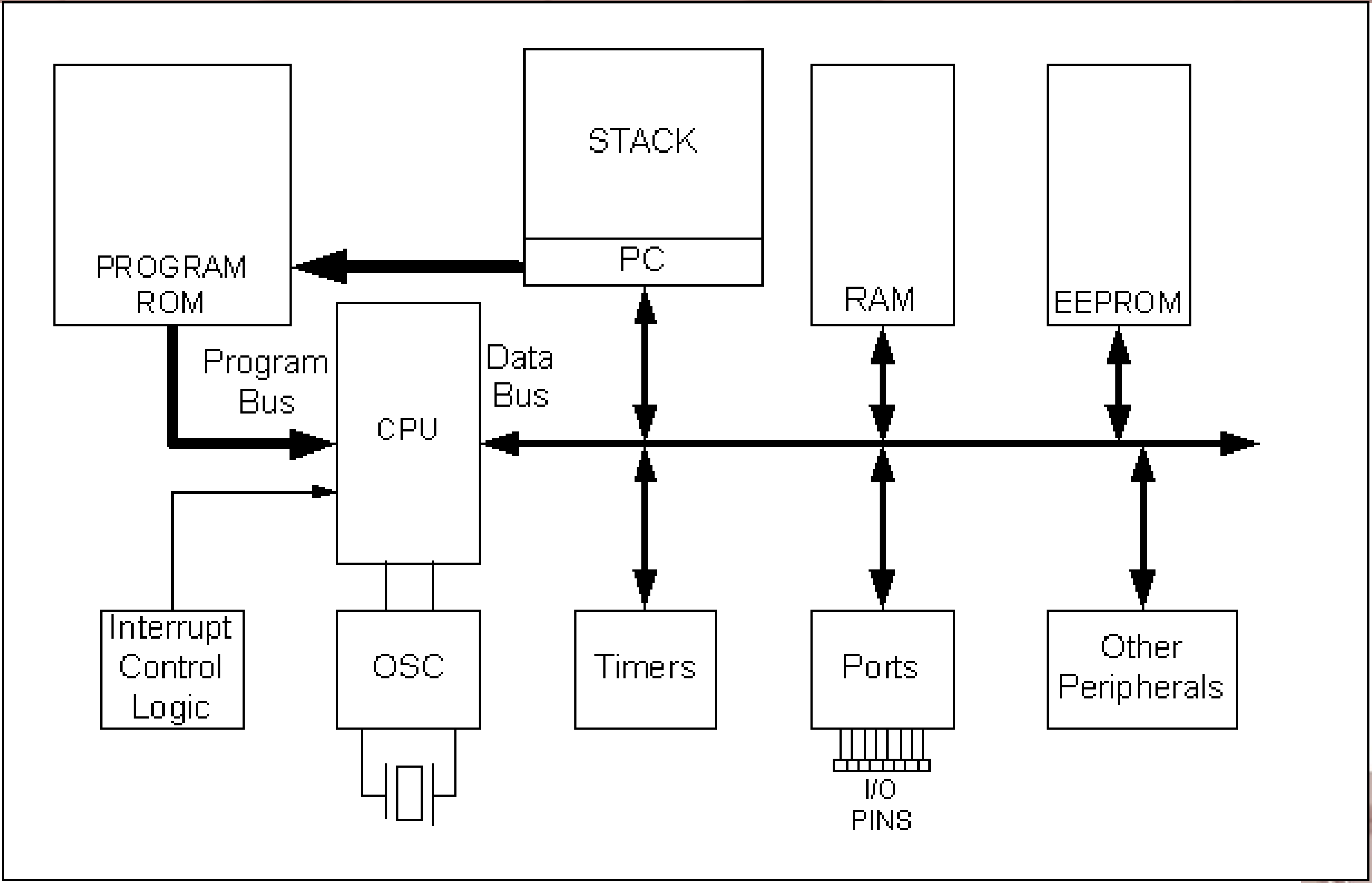
Microprocessor vs. microcontroller (MCU)



(a) General-Purpose Microprocessor System

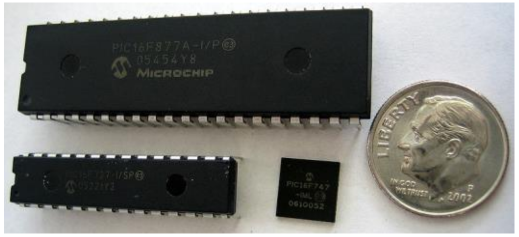
(b) Microcontroller

Simplified view of a PIC microcontroller



PIC microcontroller vs. Intel MCS-51 (8051)

- "Peripheral Interface Controller" made by Microchip Technology

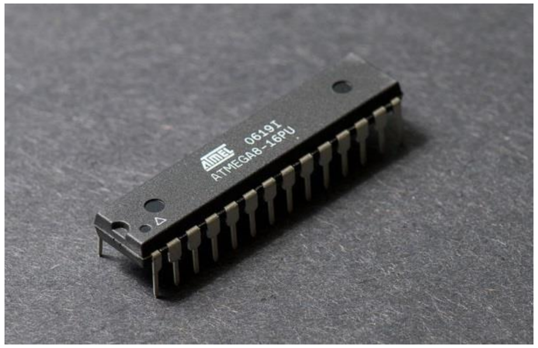


- 8-bit ALU
- Intel's original in the 1980s.
- Several companies offer MCS-51 as IP cores in FPGAs or ASICs.

68HC MCU and AVR

- 68HC microcontroller
  - 8-bit microcontroller family introduced by Motorola in 1985. Now produced by Freescale Semiconductor
  - CISC (complex instruction set computer) design

- AVR
  - 8-bit RISC MCU was sold to Atmel from Nordic VLSI



Features of PIC18 MCU

- 8-bit CPU
- Program ROM (read-only memory)
  - UV-EROM (erasable programmable ROM): need 20 minutes to erase
  - Flash (Electrical EPROM, EEPROM)
  - OTP (one-time-programmable) version: mass production
  - Masked version: burning program during IC fabrication
- Data RAM (random access memory)
  - A maximum of 4096 bytes
  - Some PIC18s use EEPROM to store critical data that are not changed not often.

Features of PIC18 MCU

- Timers
- Pulse-width modulation (PWM)
- Parallel I/O ports
- SPI, I^{2}C, controller area network (CAN) serial interface
- Universal asynchronous receiver transmitter (UART)
- 10-bit A/D converter