Outline

- The Components of a Microcomputer System
- Instruction Cycle
- Programming Languages
- How Integers Are Represented in the Computer
- Character Representation

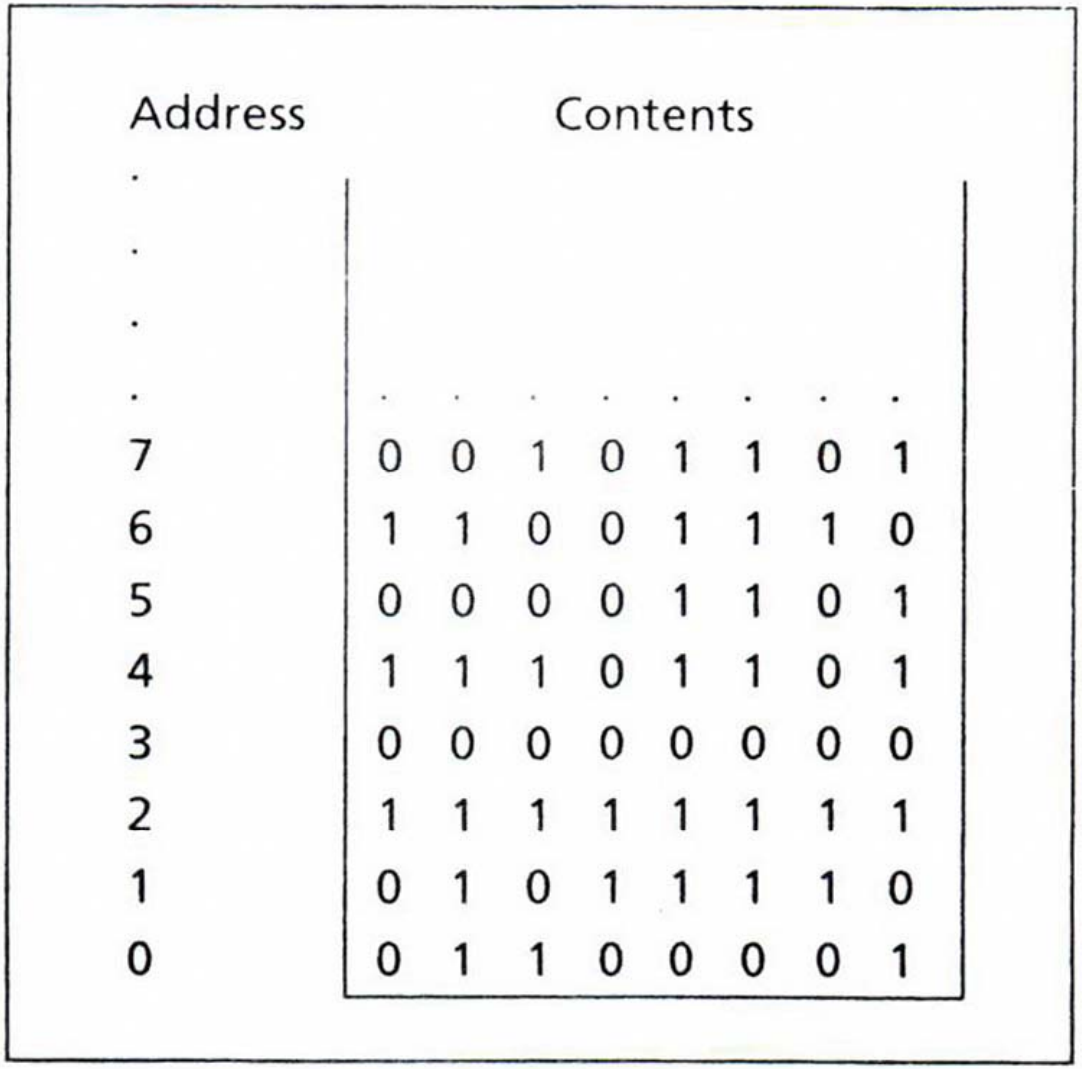
Bytes and Words

- Information processed by the computer is stored in its memory.
- The memory circuits are usually organized into groups that can store eight bits of data.
- A string of eight bits is called a **byte**.
- Each **memory byte** is identified by a number that is called **address**.
- The data stored in a memory byte are called its **contents**.

Bytes and Words

- The address of a memory byte is fixed and is different from the address of any other memory byte in the computer.
- The contents of a memory byte are not unique and are subject to change, because they denote the data currently being stored.

Memory Represented as Bytes

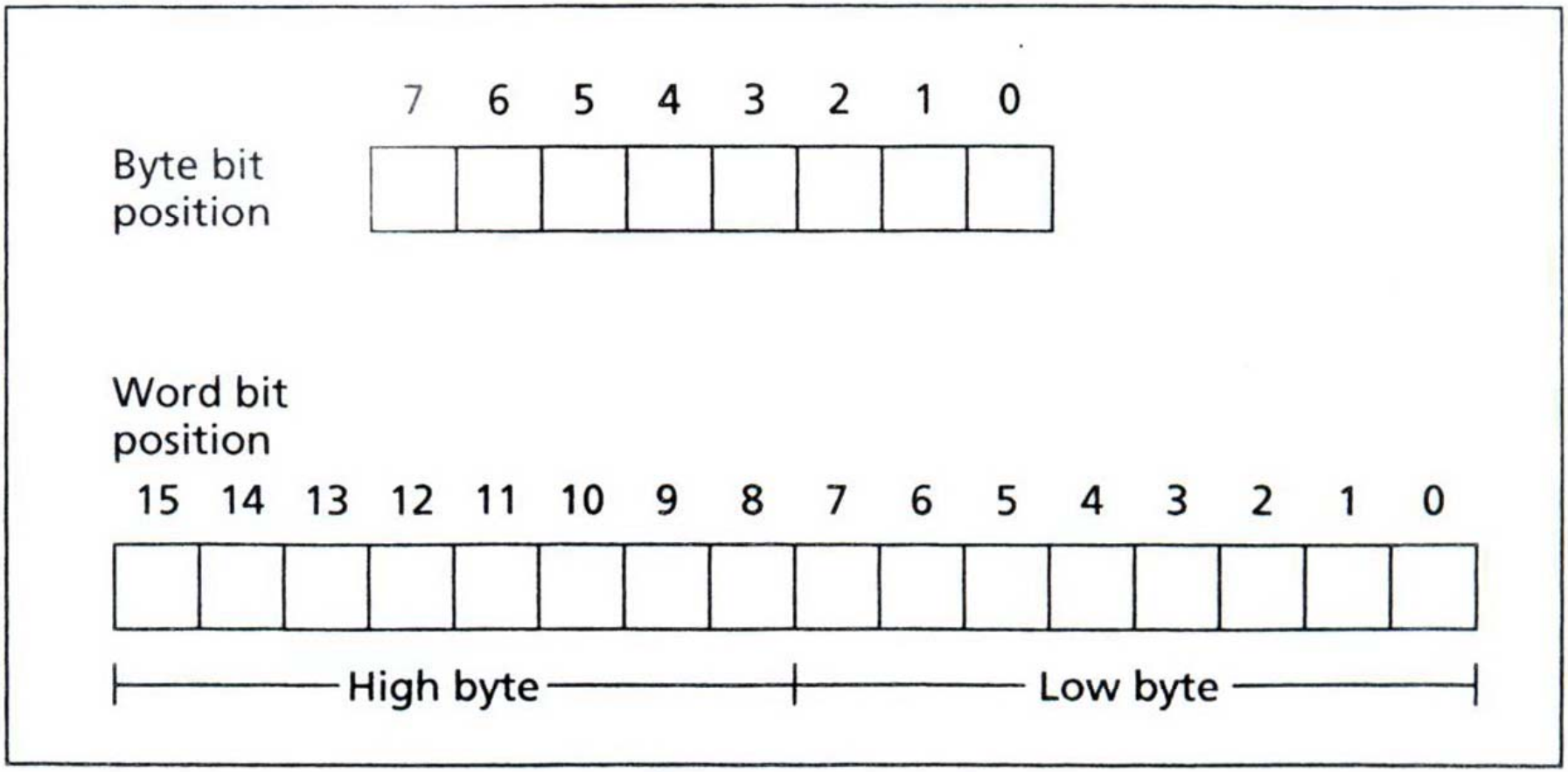


Address	Contents
.	
.	
.	
.	. . . .
7	0 0 1 0 1 1 0 1
6	1 1 0 0 1 1 1 0
5	0 0 0 0 1 1 0 1
4	1 1 1 0 1 1 0 1
3	0 0 0 0 0 0 0 0
2	1 1 1 1 1 1 1 1
1	0 1 0 1 1 1 1 0
0	0 1 1 0 0 0 0 1

Bit Position

- The positions are numbered from right to left, starting with 0.
- In a word, the bits 0 to 7 form the **low byte** and the bits 8 to 15 form the **high byte**.
- For a word stored in memory, its low byte comes from the memory byte with the lower address and its high byte is from the memory byte with the higher address.

Bit Positions in a Byte and a Word



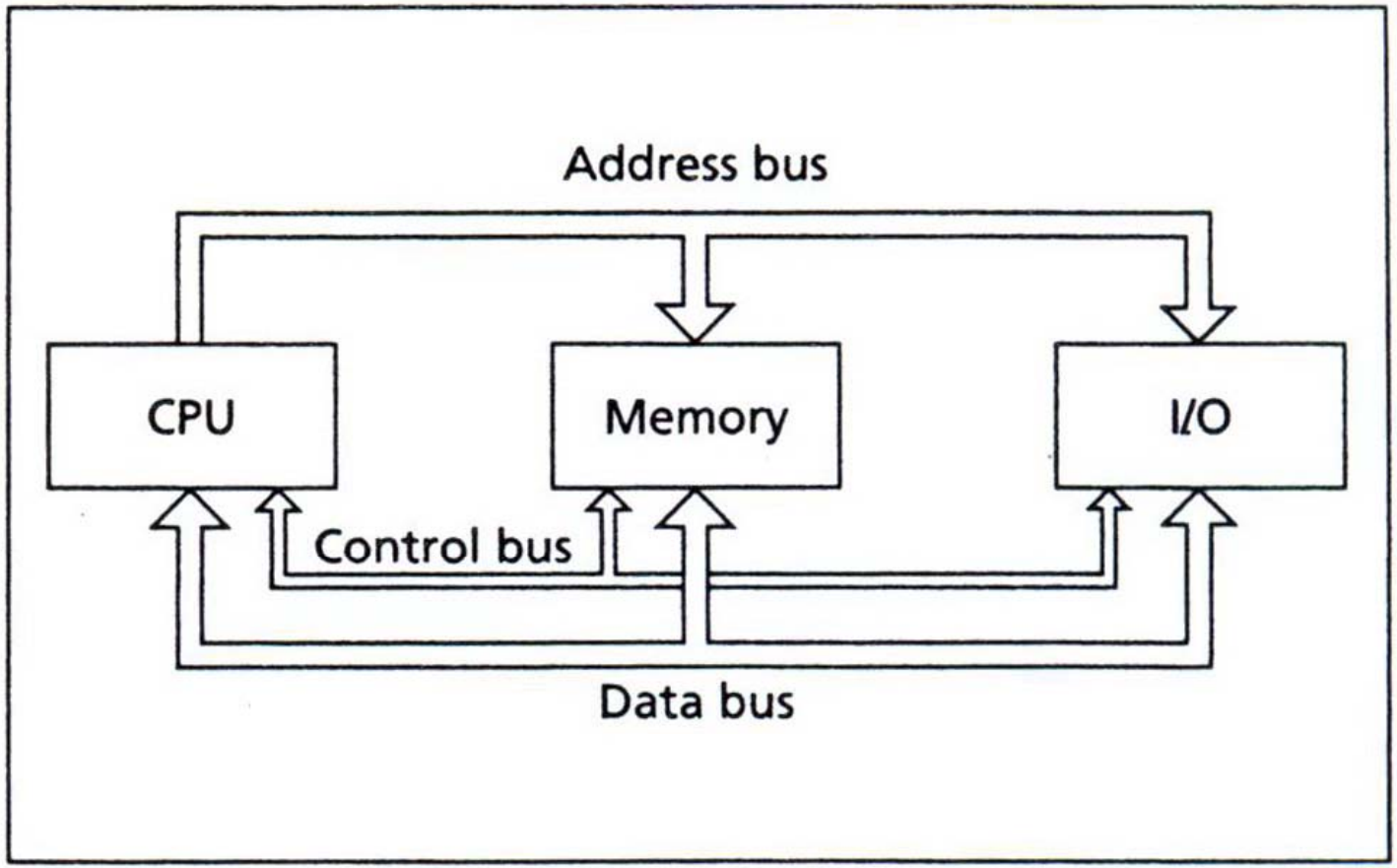
Buses

- A processor communicates with memory and I/O circuits by using signals that travel along a set of wires or connections called **buses**.
- There are three kinds of signals: address, data, and control.
- And there are three buses: **address bus**, **data bus**, and **control bus**.

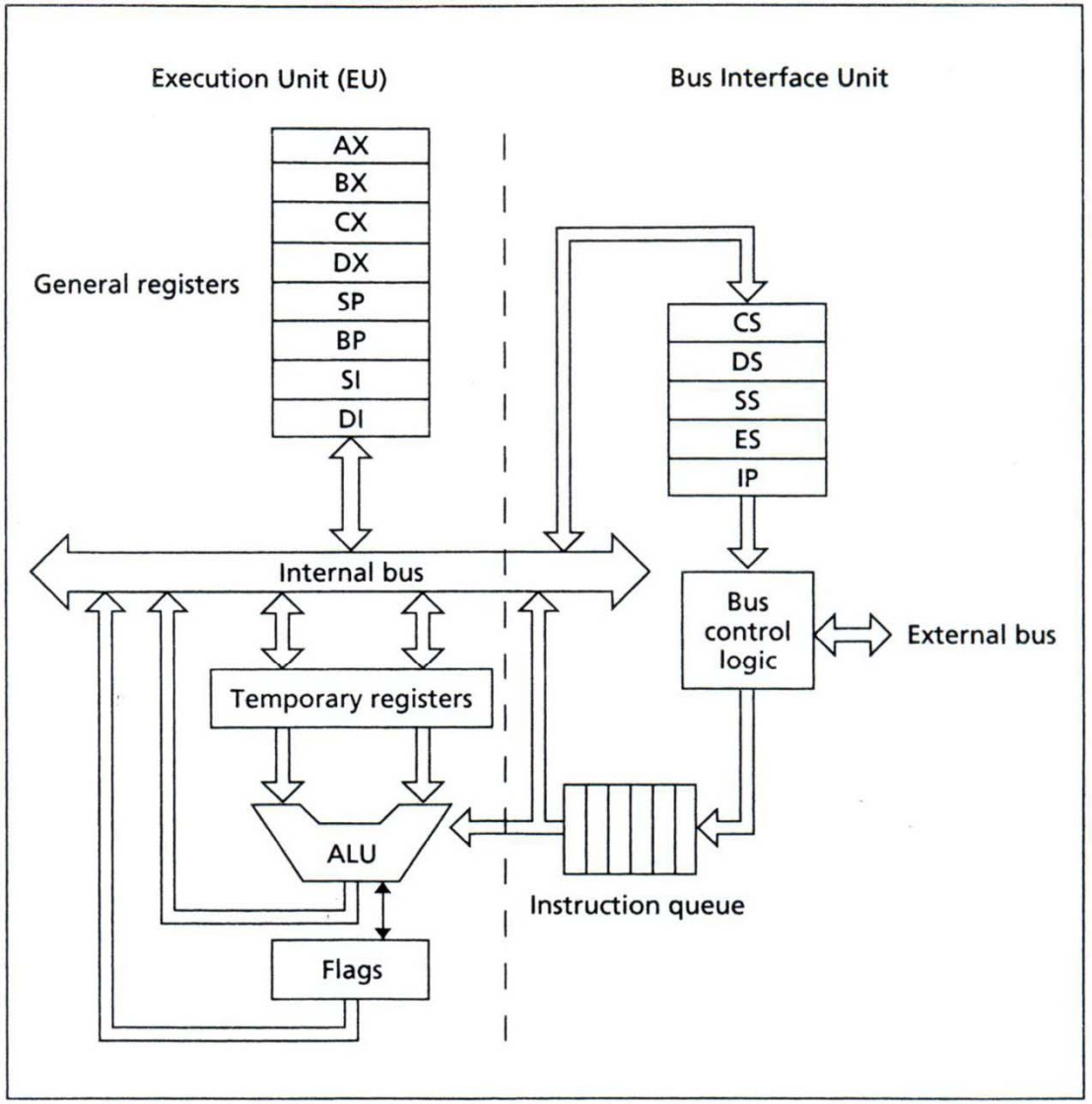
Buses

- For example, to read the contents of a memory location, the CPU places the address of the memory location on the address bus, and it receives the data, sent by the memory circuits, on the data bus.
- A control signal is required to inform the memory to perform a read operation.
- The CPU sends the control signal on the control bus.

Bus Connections of a Microcomputer



Intel 8086 Microprocessor Organization



Execution Unit (EU)

- The purpose of the EU is to execute instructions.
- The **arithmetic and logic unit (ALU)** can perform arithmetic (+, -, x, /) and logic (AND, OR, NOT) operations.
- The data for the operations are stored in circuits called **registers**.

Bus Interface Unit (BIU)

- The BIU facilitates communication between the EU and the memory or I/O circuits.
- The BIU is responsible for transmitting addresses, data, and control signals on the buses.
- The **instruction pointer (IP)** contains the address of the next instruction to be executed by the EU.

Instruction Prefetch

- While the EU is executing an instruction, the BIU fetches up to six bytes of the next instruction and places them in the instruction queue.

Machine Instruction

- The **Opcode** specifies the type of operation.
- The **Operands** are often given as memory addresses to the data to be operated on.

Fetch-Execute Cycle

Fetch

1. Fetch an instruction from memory.
2. Decode the instruction to determine the operation.
3. Fetch data from memory if necessary.

Execute

4. Perform the operation on the data.
5. Store the result in memory if needed.

Programming Languages

- The operations of the computer's hardware are controlled by its software.
- When the computer is on, it is always in the process of executing instructions.

Machine Language

- The CPU can only execute machine language instructions.
- They are bit strings.

Machine Language

Machine Instruction	**Operation**
10100001 00000000 00000000	Fetch the contents of memory word 0 and put it in register AX.
00000101 00000100 00000000	Add 4 to AX.
10100011 00000000 00000000	Store the contents of AX in memory word 0.

Assembly Language

- A more convenient language to use is **assembly language**.
- In assembly language, we use symbolic names to represent operations, registers, and memory locations.
- If location 0 is symbolized by A, the preceding program expressed in IBM PC assembly language would look like this:

Assembly Language

Assembly Instruction	**Comment**
MOV AX, A	; fetch the contents of
	; location A and put it in
	; register AX
ADD AX, 4	; add 4 to AX
MOV A, AX	; move the contents of AX
	; into location A

How Integers Are Represented in the Computer

- The **least significant bit (lsb)** is the rightmost bit.
  - In a byte or word, bit 0
- The **most significant bit (msb)** is the leftmost bit.
  - In a word, bit 15
  - In a byte, bit 7

Unsigned Integers

- An unsigned integer represents a magnitude.
- Unsigned integers are appropriate for representing quantities that can never be negative.
  - Addresses of memory locations
  - Counters
  - ASCII Code

Unsigned Integers

- The largest unsigned that can be stored in a byte is 1111 1111= FFh = 255.
- The biggest unsigned integer a 16-bit word can hold is 1111 1111 1111 1111 = FFFFh =65535.
- Note that if the least significant bit of an integer is 1, the number is odd, and it’s even if the lsb is 0.

Signed Integers

- A signed integer can be positive or negative.
- The msb is reserved for the sign:
  - 1 means negative.
  - 0 means positive.
- Negative integers are stored in the computer in a special way known as two's complement.

Two’s complement

- The one’s complement of an integer is obtained by complementing each bit; that is, replace each 0 by a 1 and each 1 by a 0.
- To get the two’s complement of an integer, just add 1 to its one’s complement.

Find the two’s complement of 5.

- 5 = 0000 000000000101
- One’s com. of 5 = 1111 1111 1111 1010
- Two’s com. of 5 = 1111 1111 1111 1011

Subtraction as Two's Complement Addition

- A subtraction can be done by bit complementation and addition.
- The circuits that add and complement bits are easy to design.

Subtraction as Two's Complement Addition

Ex.: Suppose AX contains 5ABCh and BX contains 21FCh. Find the difference of AX minus BX by using complementation and addition.

Solution: 5ABCh = 0101 1010 1011 1100

+ 1's com. of 21FCh = 1101 1110 0000 0011

+1

Difference = 1 0011 1000 1100 0000

= 38C0h

Note that 21FCh = 0010 0001 1111 1100

Subtraction as Two's Complement Addition

- A one is carried out of the most significant bit and is lost.
- The answer stored, 38C0h, is correct, as may be verified by hex subtraction.

Unsigned Decimal Interpretation

- Just do a binary-to-decimal conversion.
- It’s usually easier to convert binary to hex first, and then convert hex to decimal.

Signed Decimal Interpretation

- If the msb is 0, the number is positive, and the signed decimal is the same as the unsigned decimal.
- If the msb is 1, the number is negative, so call it –N.
- To find N, just take the two's complement and then convert to decimal as before.

Suppose AX contains FE0Ch. Give the unsigned and signed decimal interpretations.

- FE0Ch = 1111 1110 0000 1100
- For the unsigned decimal interpretations
  - FE0Ch = 65036
- For the signed decimal interpretations
  - FE0Ch = –N
  - One's com. of FE0Ch = 0000 0001 1111 0011
  - Two's com. of FE0Ch = 0000 0001 1111 0100
  - N = 01F4h = 500

ASCII Code

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange Code
- The ASCII Code system uses seven bits to code each character, so there are a total of $2^7 = 128$ ASCII codes.
- Notice that only 95 ASCII codes, from 32 to 126, are considered to be printable.

Show how the character string “RG 2z” is stored in memory, starting at address 0.

Address	**Contents**
0	0101 0010
1	0100 0111
2	0010 0000
3	0011 0010
4	0111 1010