Microprocessors (0630371)
Fall 2010/2011 – Lecture Notes # 1

The Intel Microprocessors: Architecture, Programming and Interfacing

Introduction to the Microprocessor and computer

Outline of the Lecture

- **Evolution of programming languages.**
- **Microcomputer Architecture.**
- **Instruction Execution Cycle.**

Evolution of programming languages:

- ***Machine language***- the programmer had to remember the machine codes for various operations, and had to remember the locations of the data in the main memory like: 0101 0011 0111…
- ***Assembly Language***- an instruction is an easy –to- remember form called a **mnemonic code**.

Example:

Assembly Language	*Machine Language*
`Load`	`100100`
`ADD`	`100101`
`SUB`	`100011`

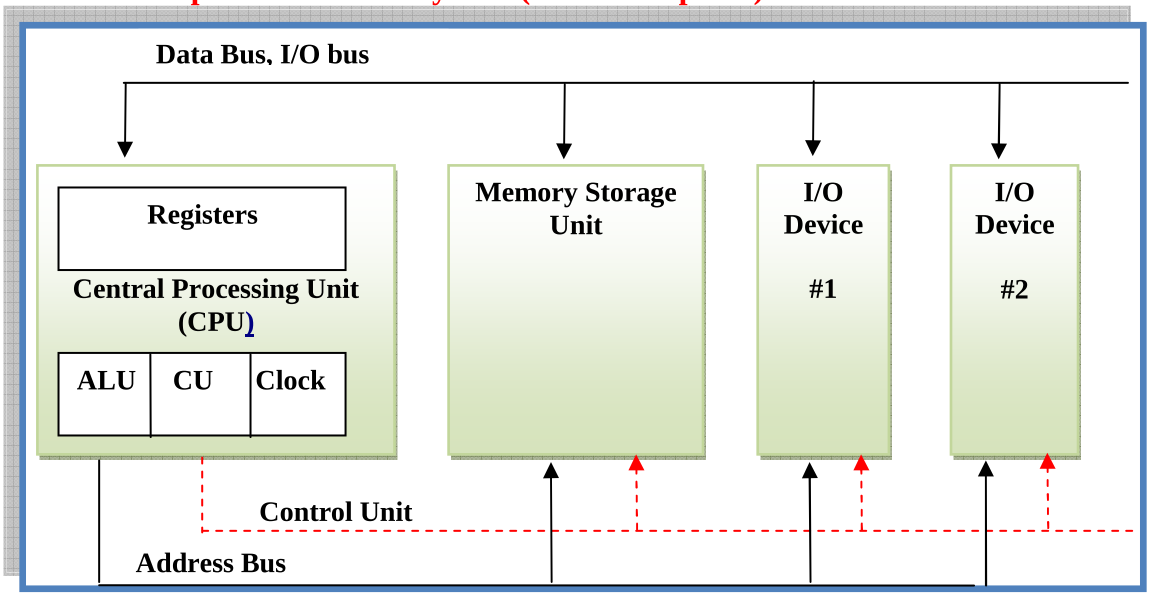
We need a program called an assembler that translates the assembly language instructions into machine language.

- ***High-level languages***

Fortran, Cobol, Pascal, C++, C# and java.

We need a **compiler** to translate instructions written in high-level languages into machine code.

Microprocessor-based system (Micro computer) Architecture

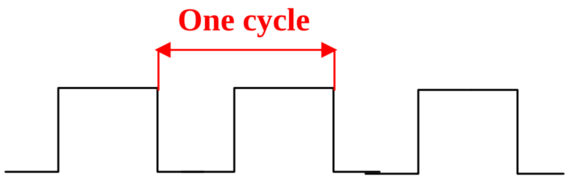


The figure shows the main components of a microprocessor-based system:

- *CPU- Central Processing Unit*, where calculations and logic operations are done.

CPU contains **registers**, a high-frequency **clock**, a control unit (**CU**) and an arithmetic logic unit (**ALU**).

  - *Clock*: synchronizes the internal operations of the CPU with other system components using clock pulsing at a constant rate (the basic unit of time for machine instructions is a machine cycle or clock cycle)



A machine instruction requires at least one clock cycle some instruction require 50 clocks.

  - *Control Unit (CU)* - generate the needed control signals to coordinate the sequencing of steps involved in executing machine instructions: (fetches data and instructions and decodes addresses for the ALU).
  - *Arithmetic logic (ALU)* - performs arithmetic operations such as addition and subtraction and logical operations such as AND, OR, and NOT.
  - *Registers*: Internal memory locations.
  - *System Bus:* The components of the computer system must communicate with other and with the outside world using a bus.

A **bus** is a bunch of wires that transfer data from one part of the computer to another.

There are three types of Bus:

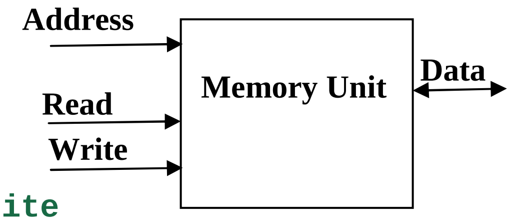
1. *Address Bus (AB):* the width of AB determines the amount of physical memory addressable by microprocessor and it holds the addresses of instructions and data when the currently executing instruction transfers data between the CPU and memory.
2. *Data Bus (DB):* The width of DB: The width of DB indicates the size of the data transferred between microprocessor and memory or I/O device.
   (Transfer instructions and data)
3. *Control Bus (CB):* The control bus uses binary signals to synchronize actions of all devices attached to the system bus.
   CB consists of control signals like ***memory read, memory write, I/O read, I/O write, interrupt acknowledge*** …

- *Memory Storage Unit:*

The memory of computer system consists of tiny electronic switches (two states: open or close). Each switch can represent a binary digit or bit. The memory unit consists of millions of such bits; bits are organized into groups as the following:

Name	Number of bytes
Bit	0 or 1 (1 Bit)
Byte	8 Bits (1 Byte)
Word	2- Bytes (16- Bits)
Double Word	4- Bytes (32- Bits)
Quad Word	8- Bytes (64- Bits)
Paragraph	16- Bytes (128- Bits)
KB.	1024- Byte

Two memory operations (Basic Operations)



Read & Write

- ***Steps in a typical read cycle:***

1. Read the address of the location.
2. Activate the **read** signal (control signal).
3. Wait for the memory to retrieve the data.
4. Read the data from the data bus.
5. Drop the memory **read** control signal to terminate the read cycle.

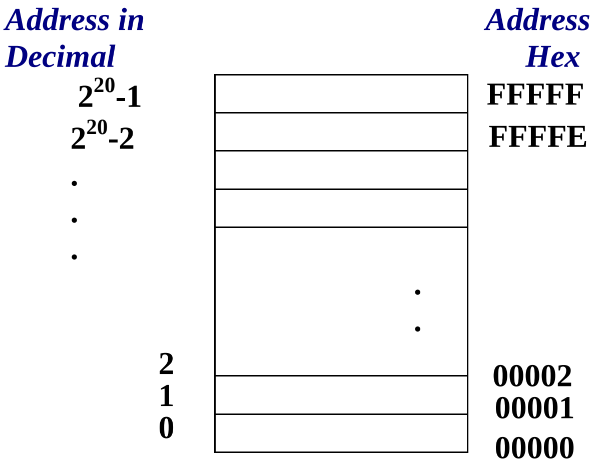
- ***Steps in a typical write cycle:***

1. Place the address of the location to be written on the address bus.
2. Place the data to be written on the data bus.
3. Activate the **write** signal.
4. Wait for the memory to store data.
5. Drop the **write** signal.

- ***Logical view of the systems (memory system)***

Memory can be viewed as consisting of an ordered sequence of bytes. Such memory is called **byte addressable memory.**

8086 microprocessor can address up to 1 MB (220 bytes) (address lines)



- ***Types of memory:***

The memory unit can be implemented using a variety of memory chips- different speeds, different technologies and different sizes.

(The two basic types are RAM or ROM)

1- ***Read Only Memories (ROM)***

Other types of ROM:

- Programmable ROM (PROM)
- Erasable PROM (EPROM)- Special equipment
- EAPROM (Electrically)

2- ***Read/write Memory***

Random access memory (RAM):

- Static RAM- used for implementing CPU registers and cache memory.
- Dynamic RAM- main memory in a computer system (Program is kept during the execution, also the data).

Instruction Execution Cycle

- The sequence of individual operations to execute a single machine instruction is called the **instruction execution cycle**.
- Executing a machine instruction requires three basic steps: **fetch**, **decode** and **execute** and two more steps are required when the instruction uses a memory operand: **fetch operand** and **store output operand**.

1. Fetch:

The control unit fetches the next instruction from the **instruction queue** and increment the **instruction pointer (IP)**

2. Decode:

The control unit decodes the instruction, to determine what the instruction will do.

3. ***Fetch operands***:

If the instruction uses an input operand located in memory, the control unit uses a read operation to retrieve the operand and copy it into internal registers.

4. ***Execute:***

The ALU executes the instruction using the named registered and sends the output to named register and/or memory.

The ALU updates status flags providing information about the processor state.

5. Store Output Operand

If the output operand is in memory, the control unit uses a write operation to store the data.

The sequence of steps can be expressed in pseudo code:

```
Loop
    Fetch next Instruction
    Advance the instruction pointer (IP)
    Decode the instruction
    If memory operand needed, read value from memory
    Execute the instruction
    If result is memory operand, write result to memory
Continue loop
```

- The processor use high speed cache memory to decrease its need to access the slower main memory and the CPU must wait one or more clock cycles until operands have been fetched from memory before the current instruction can complete its execution.

  The wasted clock cycles are called **wait states.**