UNIT 1
Computer Evolution and Performance

Computer Organization and Architecture

- **Computer Architecture** refers to those attributes of a system <u>visible to a programmer</u> or, put another way, those attributes that have a <u>direct impact on the logical execution of a program</u>.

- Eg. **Instruction set**, the **number of bits** used to represent various data types (e.g., numbers, characters), I/O mechanisms, and **techniques for addressing memory.**

- **Computer Organization** refers to the <u>operational units and their interconnections</u> that realize the architectural specifications.

- Eg. Hardware details transparent to the programmer, such as **control signals**; **Interfaces** between the computer and peripherals; and the **memory technology** used.

- Eg:

  - It is an <u>architectural design issue whether a computer will have a multiply instructions.</u>

  - It is an <u>organizational issue whether that instruction will be implemented by a special multiply unit</u> or by a mechanism that makes repeated use of the add unit of the system.

Functions and Structures

- The **hierarchical nature of complex systems** is essential to both their **design and their description**.

- The **designer need only deal with a particular level** of the system at a time.

- At **each level**, the system consists of a set of **components and their interrelationships.**

- The **behavior at each level depends only on a simplified, abstracted characterization** of the system at the next lower level.

- At each level, the designer is concerned with structure and function:

- **Structure:** The way in which the components are interrelated.

- **Function:** The operation of each individual component as part of the structure.
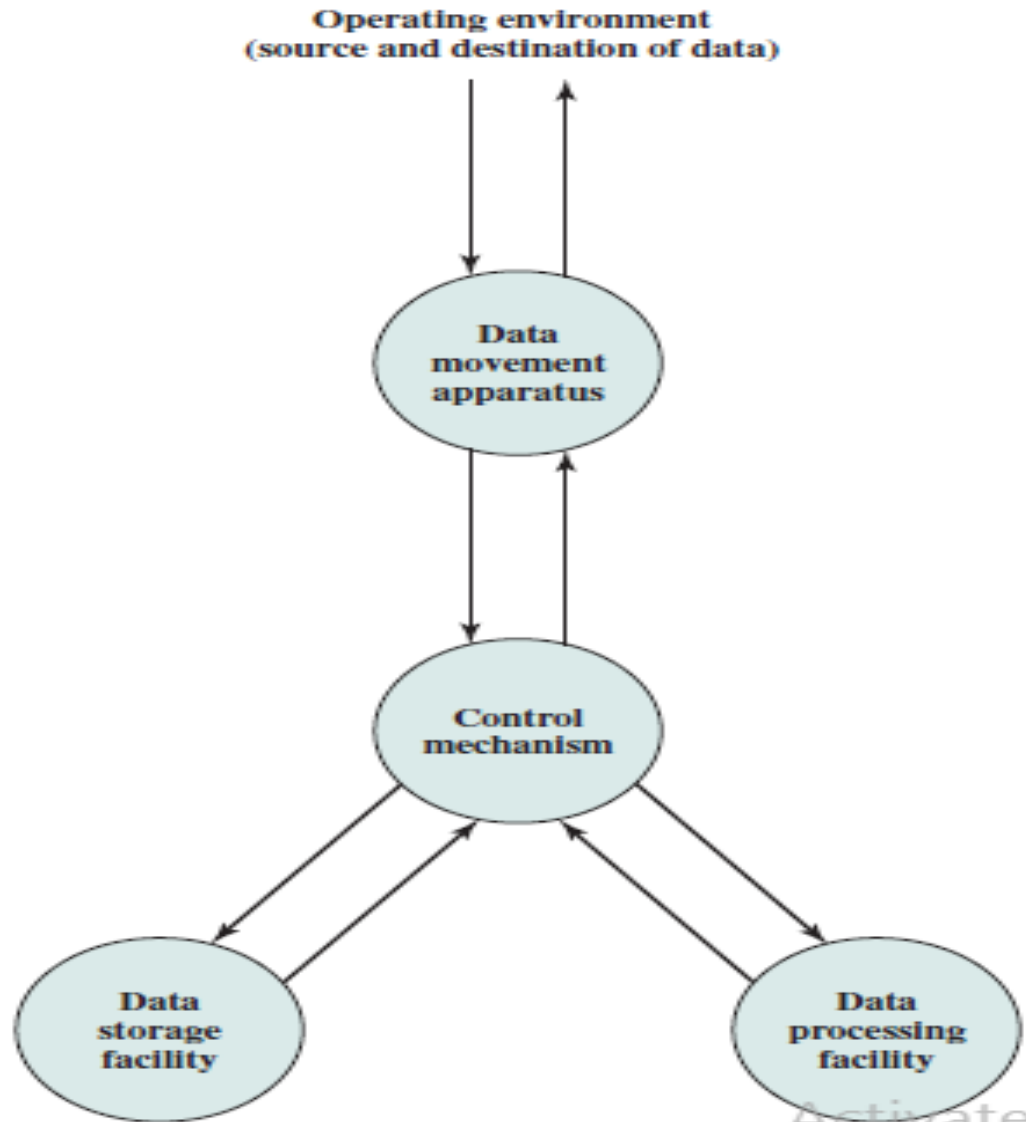
I. Structure

- There are four main structural components:

- **Central Processing Unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as processor.

- **Main Memory**: Stores data.

- **I/O Devices**: Moves data between the computer and its external environment.

- **System Interconnection**: Some mechanism that provides for communication among CPU, main memory, and I/O. A common example of system interconnection is by means of a **System Bus.**

CPU

- **Control unit:** Controls the operation of the CPU and hence the computer.

- **Arithmetic and logic unit (ALU):** Performs the computer's data processing functions.

- **Registers:** Provides storage internal to the CPU.

- **CPU interconnection**: Some mechanism that provides for communication among the control unit, ALU, and registers.

II. Functions

- The operation of each individual component as part of the structure.

- There are 4 functions
  - **Data Processing**
  - **Data Storage**
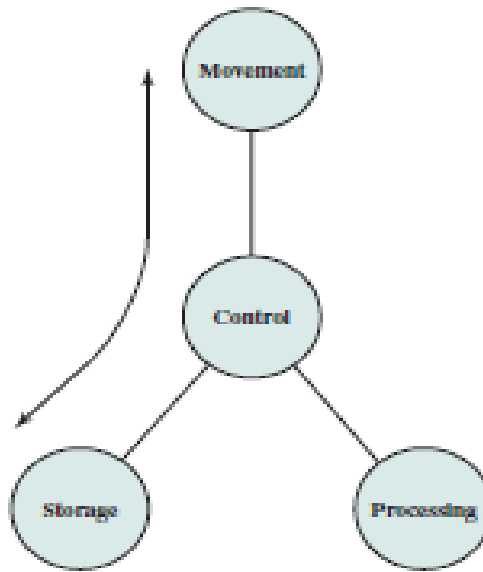 - **Data Movement**
  - **Control**

Operating environment
(source and destination of data)

Data movement apparatus

Control mechanism

Data storage facility

Data processing facility

- The computer, of course, must be able to **Process Data.** The data may take a wide variety of forms, and the range of processing requirements is broad.

- It is also essential that a computer **Store Data.** Even if the computer is processing data on the fly (i.e., data come in and get processed, and the results go out immediately), the computer must temporarily store at least those pieces of data that are being worked on at any given moment. Thus, there is at least a short-term data storage function.

- Equally important, the computer performs a long-term data storage function. Files of data are stored on the computer for subsequent retrieval and update.

- The computer must be able to **Move Data** between itself and the outside world.

- When data are received from or delivered to a device that is directly connected to the computer, the process is known as *Input– Output (I/O),* and the device is referred to as a *Peripheral.*

- *When data are moved* over longer distances, to or from a remote device, the process is known as **Data Communications.**
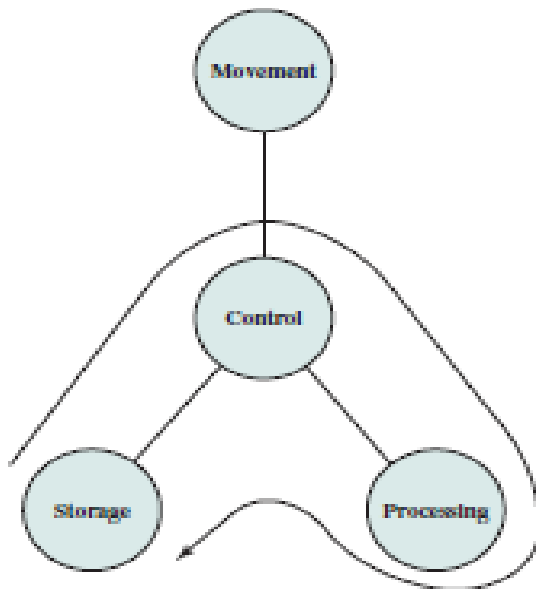
- Finally, there must be **Control** of these three functions. Ultimately, this control is exercised by the individual(s) who provides the computer with instructions.
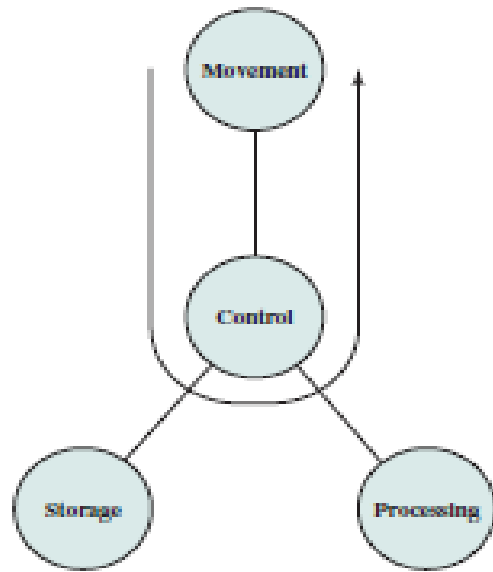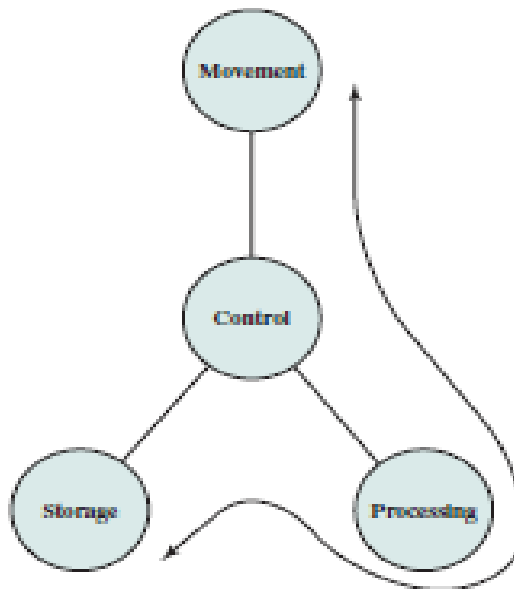
a) **Data Movement**
b) **Data Storage**
c) **Data Processing (Internal)**
d) **Data Processing (External)**

Evolution of Computers

First Generation (1940-1956) Vacuum Tubes

- The first electronic computer was designed at Iowa State between 1939-1942.

- The Atanasoff-Berry Computer <u>used the binary system(1's and 0's).</u>

- Contained **Vacuum Tubes** and <u>stored numbers or calculations by burning holes in paper.</u>

IBM Stretch - 1959

IBM Stretch - 1959

Second Generation (1956-1963) Transistors

- In 1947, the **Transistor** was invented.

- The transistor made computers smaller, less expensive and increased calculating speeds.

- Second generation computers also saw a new way data was stored.

- Punch cards were replaced with magnetic tapes.

- ALP are used for computing.

SECOND GENERATION COMPUTERS

- Transistors
- Punched cards for input
- Assembly language for computing

Third Generation (1964-1971) Integrated Circuits

- Transistors were replaced by integrated circuits(IC).

- <u>One IC could replace hundreds of transistors.</u>

- This made computers even smaller and faster.

- OS is used for UI.

- Keyboard entry is possible.

THIRD GENERATION COMPUTERS

- Integrated circuits
- Operating system as user interface to computing
- Increased speed and efficiency

Fourth Generation (1971-Present) Microprocessors

- In 1970 the Intel Corporation invented the Microprocessor: an entire CPU on one chip.

- This led to microcomputers-computers on a desk.

Fifth Generation (Present and Beyond) AI

- Having features like

  - Artificial Intelligence

  - Voice Recognition

  - Parallel Processing

  - Natural Language Processing

What is Microprocessor

- A microprocessor, sometimes called a *logic chip*, is a computer processor on a microchip.

- It is also called as **"Heart of Computer."**

- The microprocessor contains all, or most of, the **central processing unit (CPU)** functions.

- A microprocessor is designed to perform arithmetic and logic operations that make use of small number-holding areas called *registers*.

- Typical microprocessor operations include **adding, subtracting, comparing two numbers, and fetching numbers** from one area to another.

- These operations are the result of a set of instructions that are part of the microprocessor design.
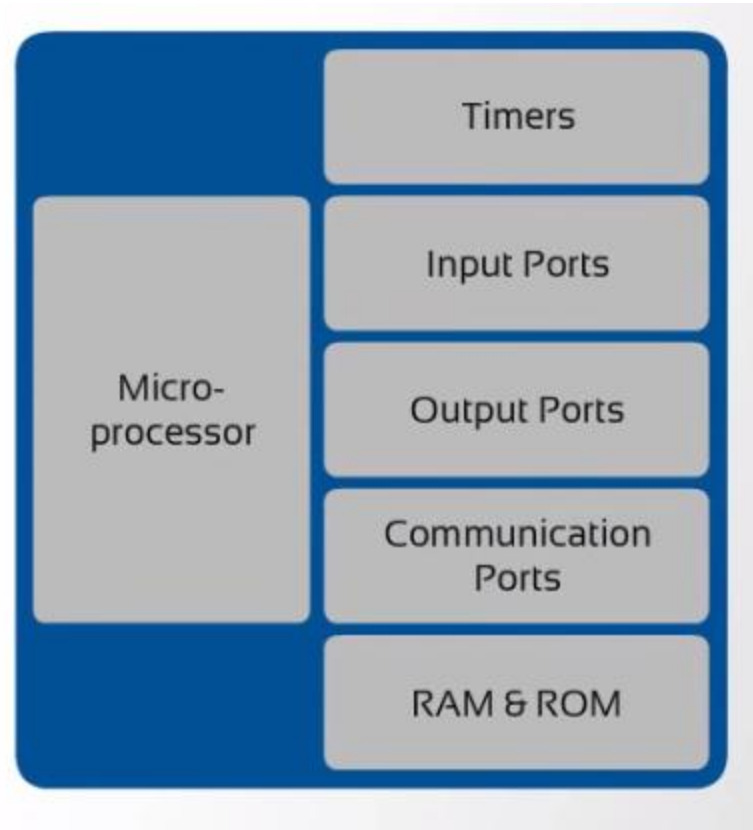
Three basic characteristics differentiate microprocessors:

- **Instruction set**: The set of instructions that the microprocessor can execute.
- **Bandwidth:** The number of bits processed in a single instruction.
- **Clock speed:** Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.
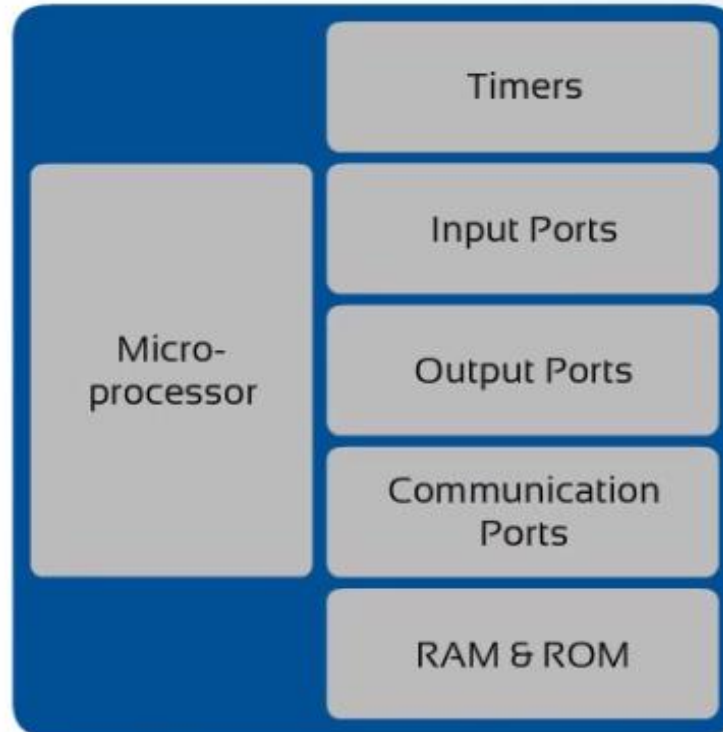
In both cases, the higher the value, the more powerful the CPU.

For example, a 32-bit microprocessor that runs at 50MHz is more powerful than a 16-bit microprocessor that runs at 25MHz.

Microprocessor Vs. Microcontroller

Microcontroller

Micro-processor	Timers
	Input Ports
	Output Ports
	Communication Ports
	RAM & ROM

Microcontroller, as an Integrated Circuit (IC), is complex than a General Purpose Processor.

Differences between a Microprocessor and a Microcontroller:

- Multipurpose
- Contains primarily the CPU
- System costs are higher
- Higher Clock speed
- Can be constantly reprogrammed as required

Microprocessor

Versus

- Specific usages
- Contains the CPU and many peripheral devices
- System costs are lower
- Cannot operate at higher Clock speed
- Requires programming only once for a particular application

Microcontroller

History of Microprocessor

MP	Introduction	Data Bus (In Bits)	Address Bus (In Bits)
4004	1971	4	8
8008	1972	8	8
8080	1974	8	16
8085	1977	8	16
8086	1978	16	20
80186	1982	16	20
80286	1983	16	24
80386	1986	32	32
80486	1989	32	32
Pentium	1993 onwards	32	
Core solo	2006	32	
Dual Core	2006	32	
Core 2 Duo	2006	32	
Core to Quad	2008	32	
I3,i5,i7	2010	64	40

4004
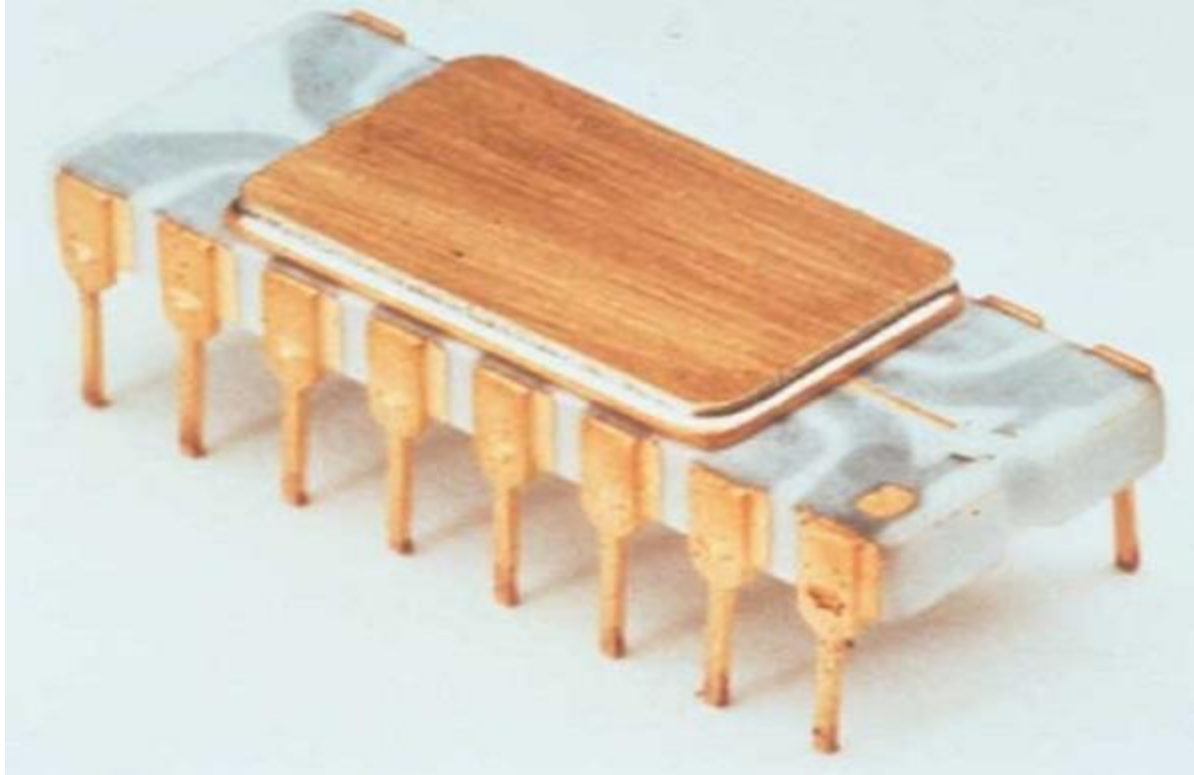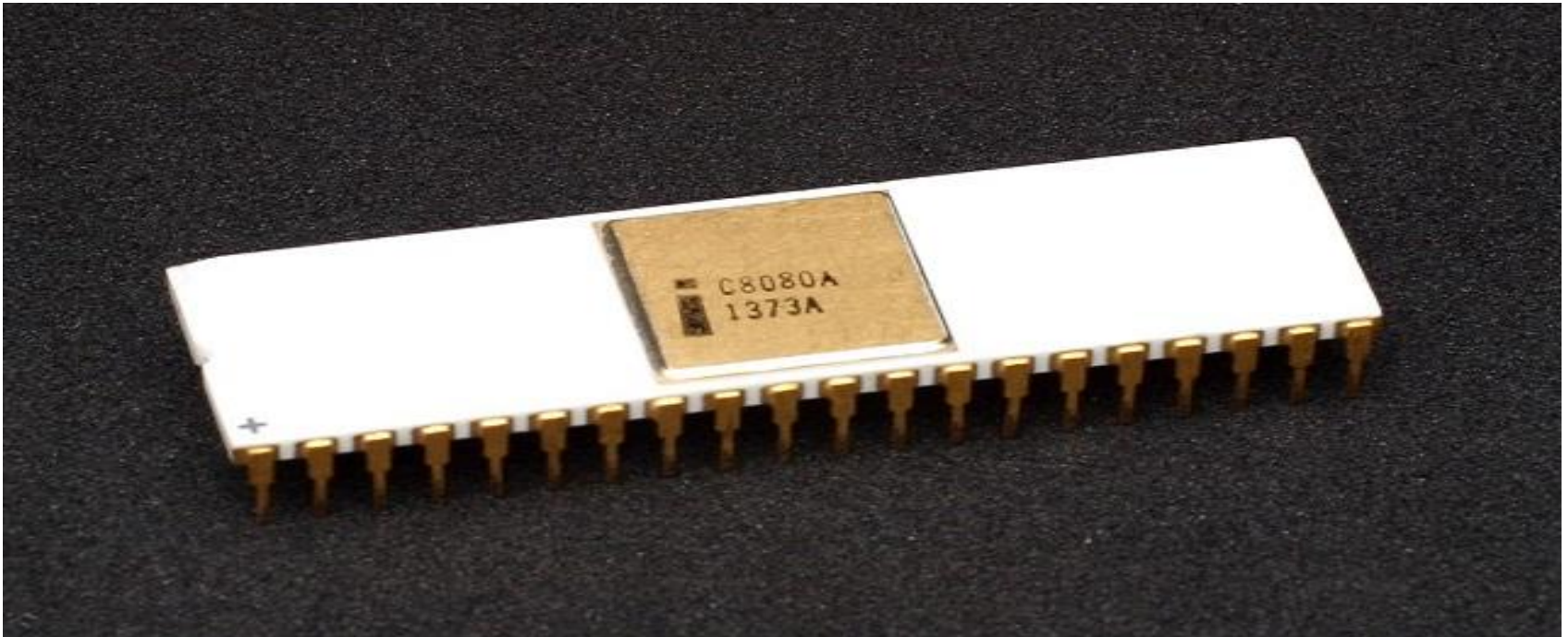

The Intel 4004 Processor was announced in **1971**. It has 2300 transistors with an initial clock speed of **108KHz**. It cost $200.

8008
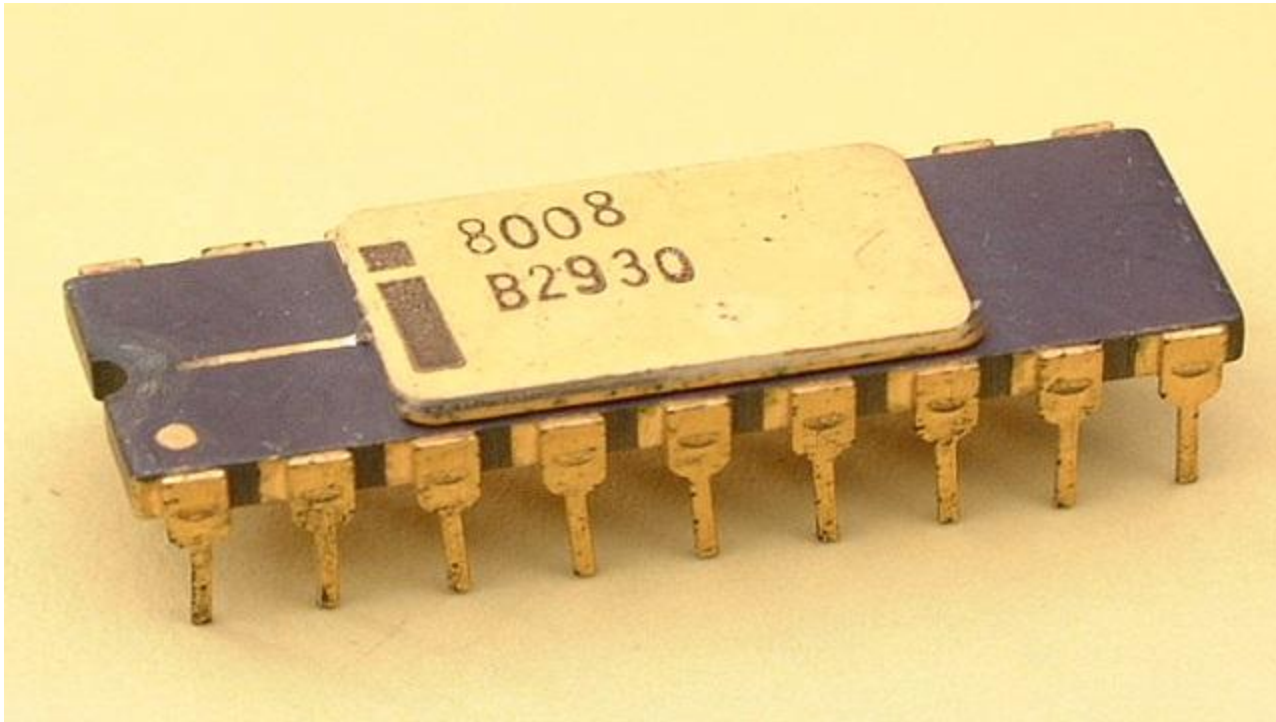
The Intel 8008 Processor was introduced in **1972**. It has 3,500 transistors with an initial clock speed of **800KHz.**

8080

The Intel 8080 Processor was introduced in **1974**. It has 4,500 transistors with an initial clock speed of **2MHz.**

8086

The Intel 8086 Processor was introduced in **1978**. It had a staggering 29,000 transistors, dwarfing the Intel 8080 processor. It has 29,000 transistors with an initial clock speed of **5MHz**.
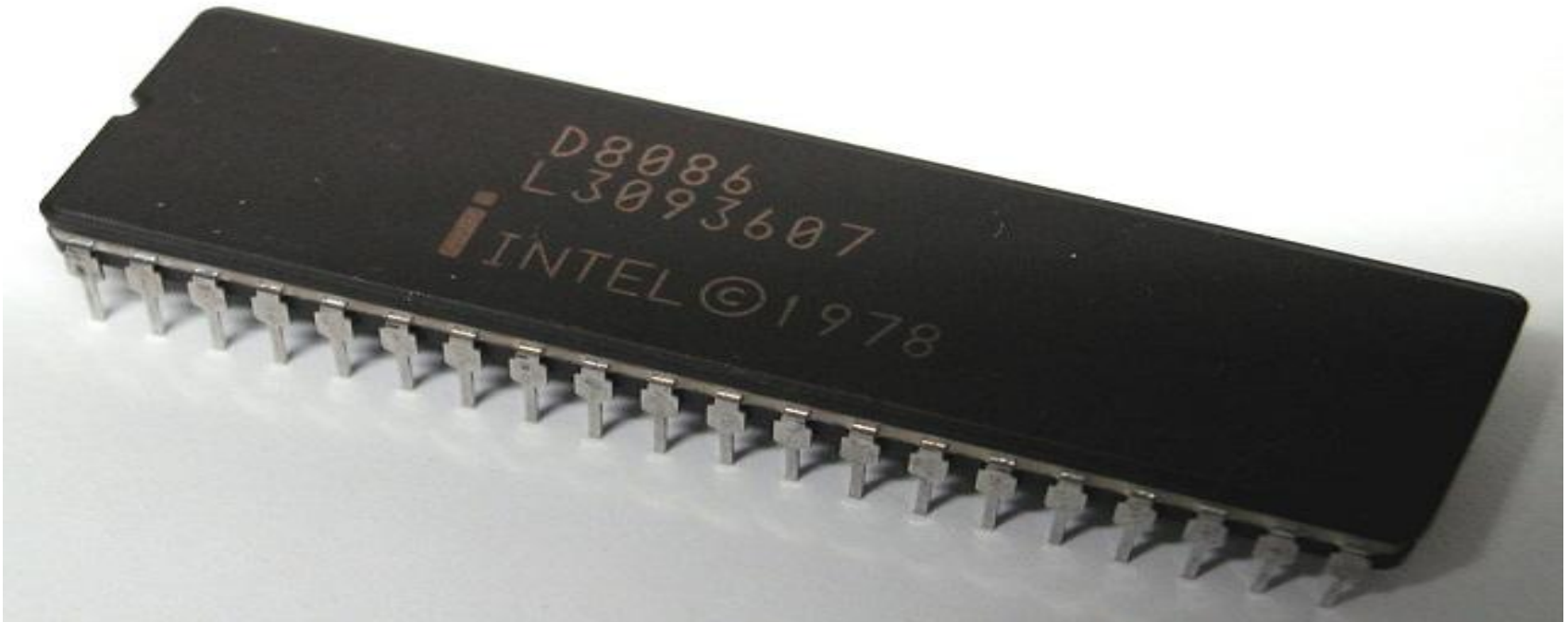
80286

The Intel 286 Processor was introduced in **1982**. It has 134,000 transistors with an initial clock speed of **6 MHz.**

80386

The Intel 386 Processor was introduced in **1985**. It has 275,000 transistors with an initial clock speed of **16 MHz.**

80486

The Intel 486 Processor was introduced in **1989**. It has 1.2 million transistors with an initial clock speed of **25MHz.**

Pentium

The Intel Pentium Processor was introduced in **1993**. It has 3.1 million transistors with an initial clock speed of **66 MHz.**

Pentium-Pro

The Intel Pentium Pro Processor was introduced in **1995**. It has 5.5 million transistors with an initial clock speed of **200 MHz.**

Pentium-II

The Intel Pentium II Processor was introduced in **1997**. It has 7.5 million transistors with an initial clock speed of **300 MHz.**

Celeron

The Intel Celeron Processor was introduced in **1998**. It has 7.5 million transistors just like the Intel Pentium II but with an initial clock speed of **266 MHz.**
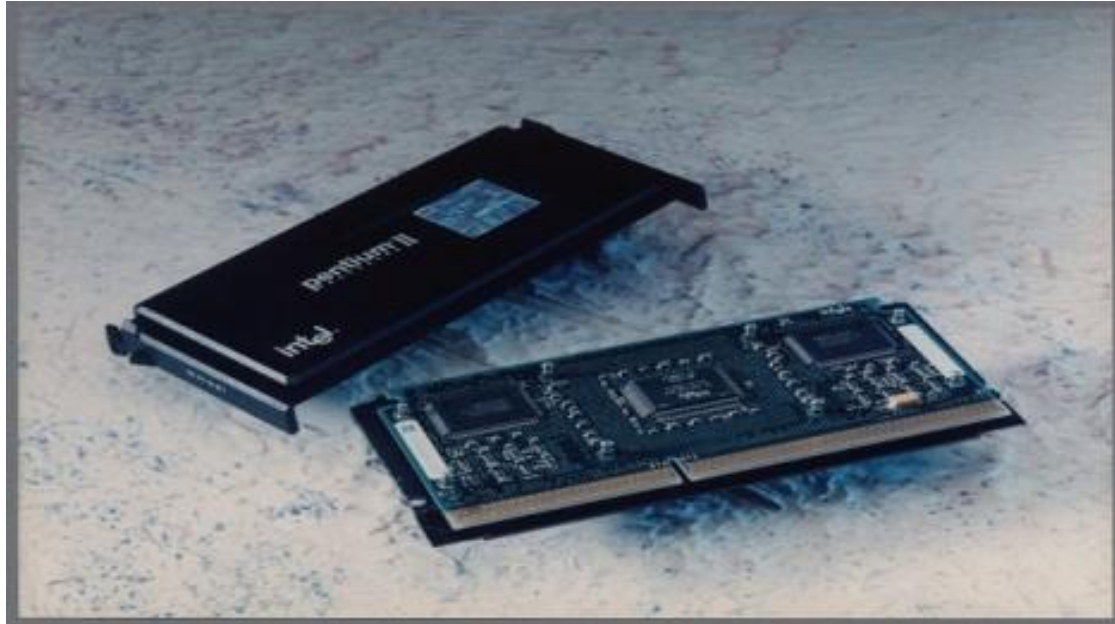
Pentium-III

The Intel Pentium III Processor was introduced in 1999. It has 9.5 million transistors with an initial clock speed of **600 MHz.**

Pentium-4

The Intel Pentium 4 Processor was introduced in **2000.** It has an astonishing 42 million transistors with an initial clock speed of **1.5 GHz.**

Xeon

The Intel Xeon 7500 Processor was introduced in **2001**. It has 42 million transistors with an initial clock speed of **1.7 GHz.**

Pentium-M

The Intel Pentium M Processor was introduced in **2003**. It has 55 million transistors with an initial clock speed of **1.7 GHz.**

Core-2

The Intel Core 2 Processor was introduced in **2006**. It has 291 million transistors with an initial clock speed of **2.66 GHz.**

Core-2-Duo

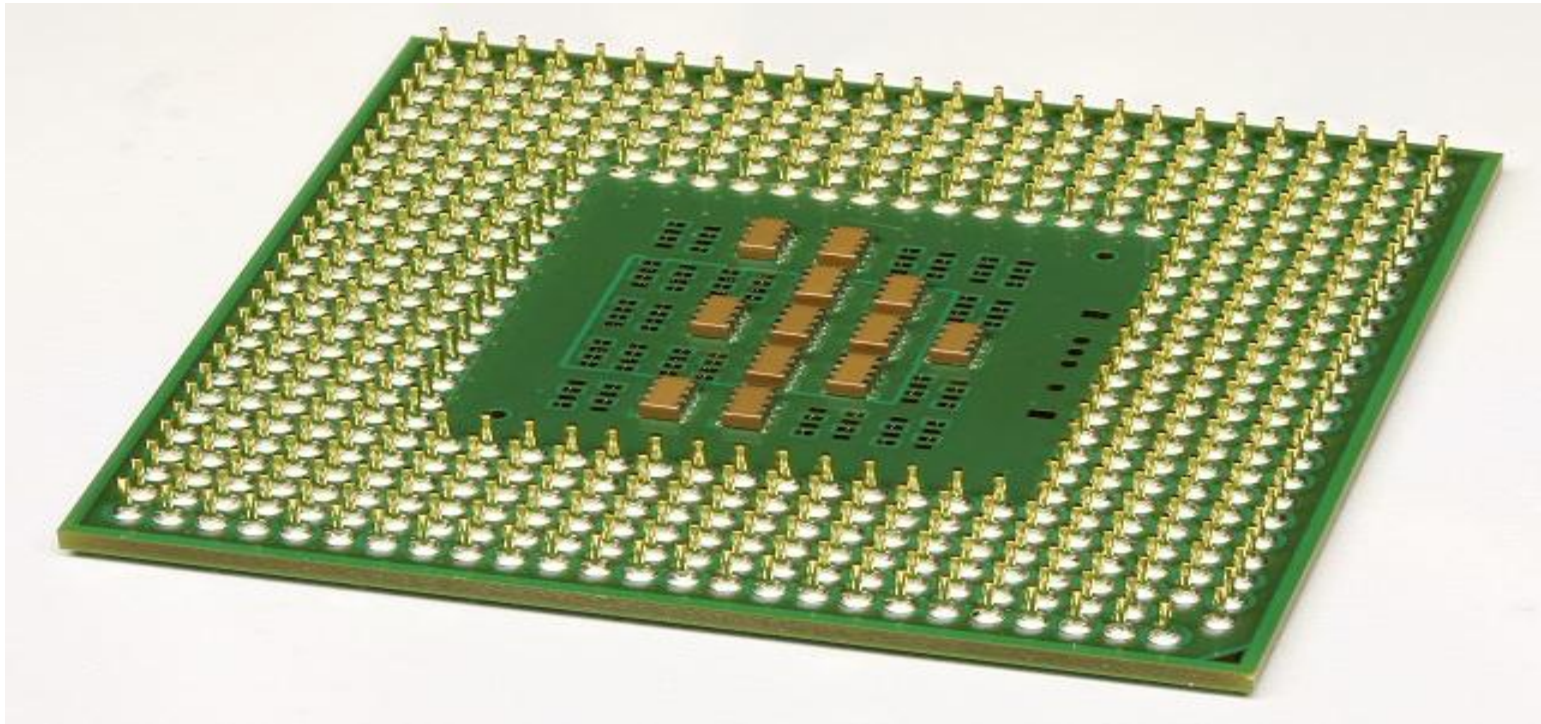The Intel Core 2 Duo Processor was introduced in **2008**. It has 410 million transistors with an initial clock speed of **2.4 GHz.**
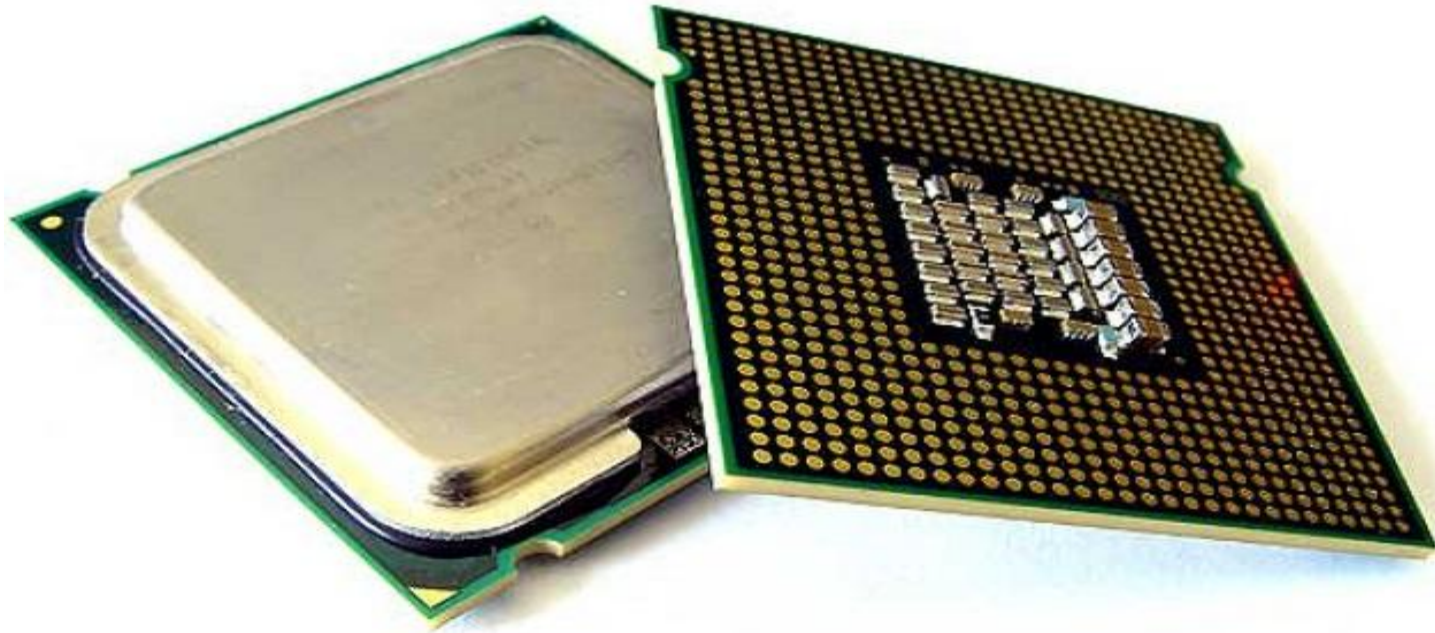
Atom

The Intel Atom Processor was introduced in **2008**. It has 41 million transistors with an initial clock speed of **1.86 GHz.**

i-Series (2ⁿᵈ Generation)

The 2nd generation Intel Core Processor was introduced in **2010**. It is a monster with 1.16 billion transistors and an initial clock speed of **2.7 GHz.**

i-Series (3rd Generation)

The 3rd generation Intel Core Processor was introduced in **2010**. It has a staggering 1.4 billion transistors and an initial clock speed of **2.9 GHz.**

i-Series (7ᵗʰ Generation)

- Introduced in **2015** and Clock speed can be upto **4.4 GHz**.

Designing for Performance

- **Things can be added to improve performance:**

  1. Microprocessor Speed
    - **Pipelining**
    - **Brach Prediction**
    - **Data Flow Analysis (Data Dependency)-** Data-flow analysis is a technique for gathering information about the possible set of values calculated at various points in a computer program
    - **Speculative execution** (see next slide)

Speculative Execution

- Using **Branch Prediction** and **Data Flow Analysis**, some processors execute instructions ahead of their actual appearance in the program execution, holding the results in temporary locations.

- This enables the processor to keep its execution engines as busy as possible by executing instructions that are likely to be needed.

2. **Performance Balance-** By increasing Hardware Requirements. (Eg. 8259 Interrupt Controller)
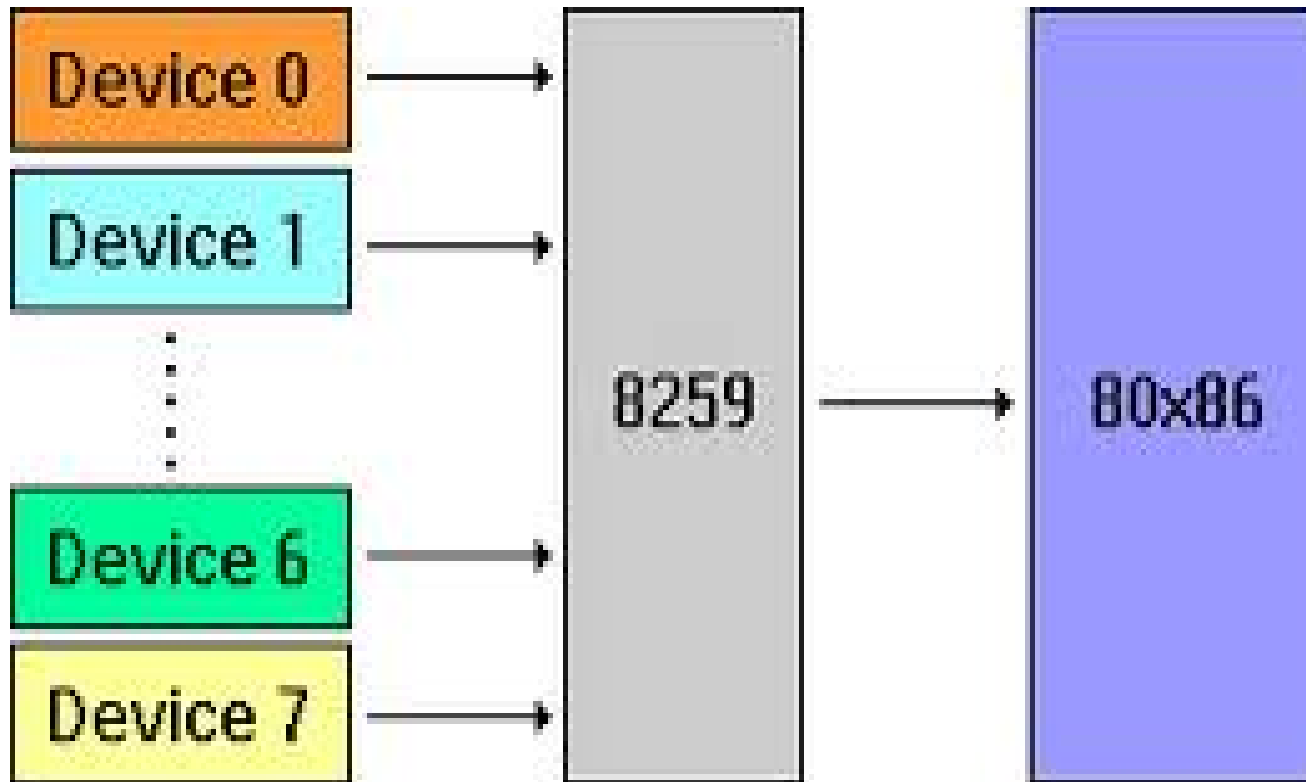
3. Improvements in Chip Organization and Architecture.

(Eg. Three level cache in i-Series)
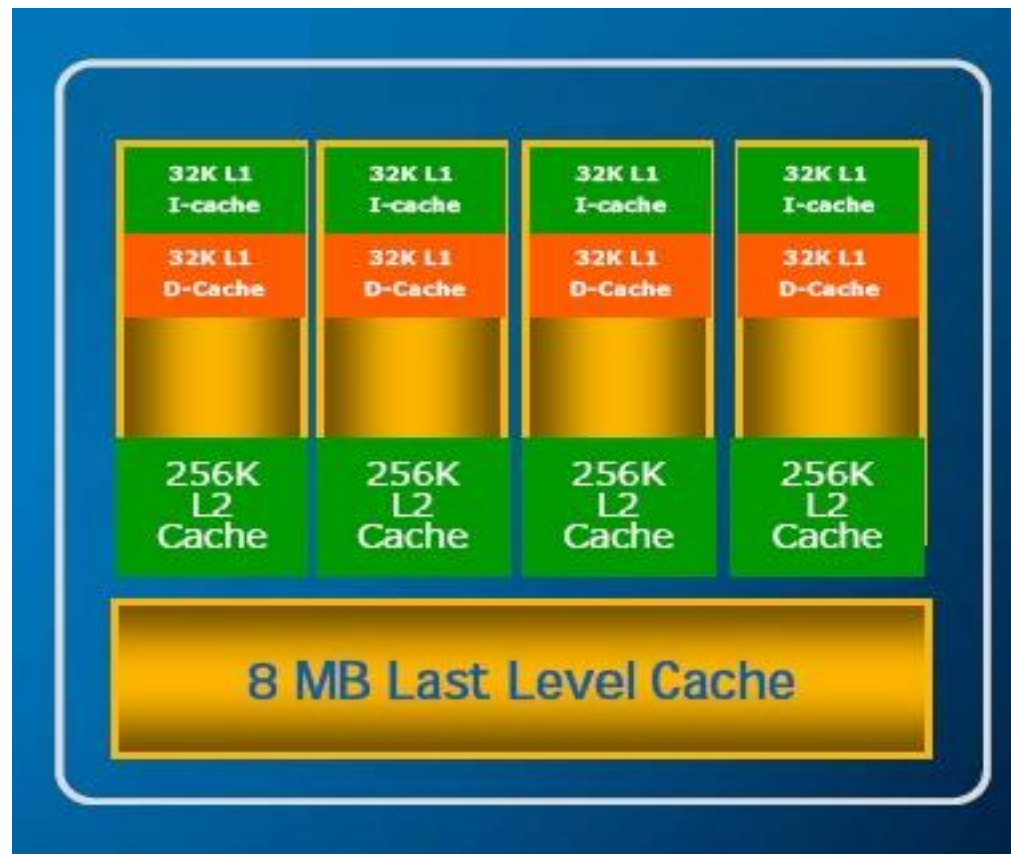
Performance Assessment

- Clock Speed and Instructions Per Second

- Benchmarks

- Amdahl's Law

- Little's Law

- **Clock Speed-**

  – Operations performed by a processor, such as **Fetching an Instruction, Decoding the Instruction, Performing an Arithmetic Operation**, and so on, are governed by a system clock.

  – Typically, **all operations begin with the pulse of the clock.** Thus, at the most fundamental level, the **speed of a processor is dictated by the pulse frequency produced by the clock, measured in cycles per second, or Hertz (Hz).**

  – The rate of pulses is known as the **Clock Rate, or Clock Speed**. One increment, or pulse, of the clock is referred to as a **Clock Cycle, or A Clock Tick**. The time between pulses is the **Cycle Time**.

- **Benchmarks-**

Lists the following as desirable characteristics of a **Benchmark Program**:

- It is written in a **High-level Language**, making it portable across different machines. (Platform Independent)

- It is representative of a particular kind of **Programming Style**, such as <u>Systems Programming, Numerical Programming, or Commercial Programming</u>.

- It can be **Measured Easily.** (Complexities)

- **Amdahl's Law:**

  – Amdahl's law was first proposed by Gene Amdahl. **Deals with the potential speedup of a program using multiple processors compared to a single processor.**

$$\text{Speedup} = \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on } N \text{ parallel processors}}$$

- **Little's Law:**
  - Total number of instructions executed by the processor in a given unit time, which will tell **Processors Throughput and Response Time.**

  - Little's law can also be applied for our web / app / db servers to relate the total no of users/requests, server's throughput & the average response time.

  - **Throughput** is number of requests processed per unit time.

  - **Response time** –It includes wait time + service time.

Computer Components: Top Level View
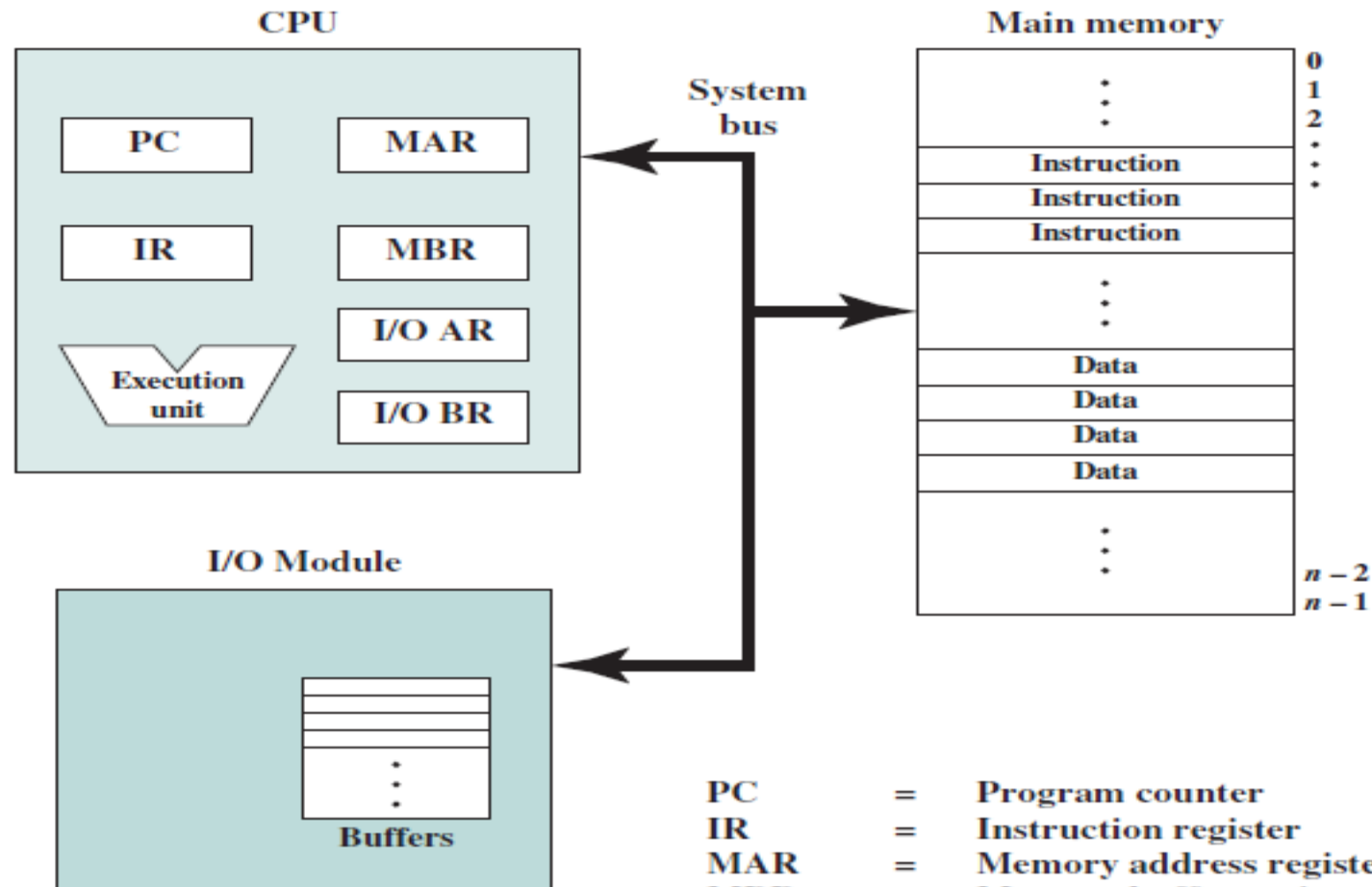
Computer Functions

- Computer functions mainly involve

  - **Instruction Fetch and Execute**

  - **Interrupts**

  - **I/O Function**

What is a Program?

- A sequence of steps (Instructions).

- For each step, an arithmetic or logical operation is done.

- For each operation, a different set of control signals is needed.

Instruction Cycle

- Two steps:
  - **Fetch**
  - **Execute**
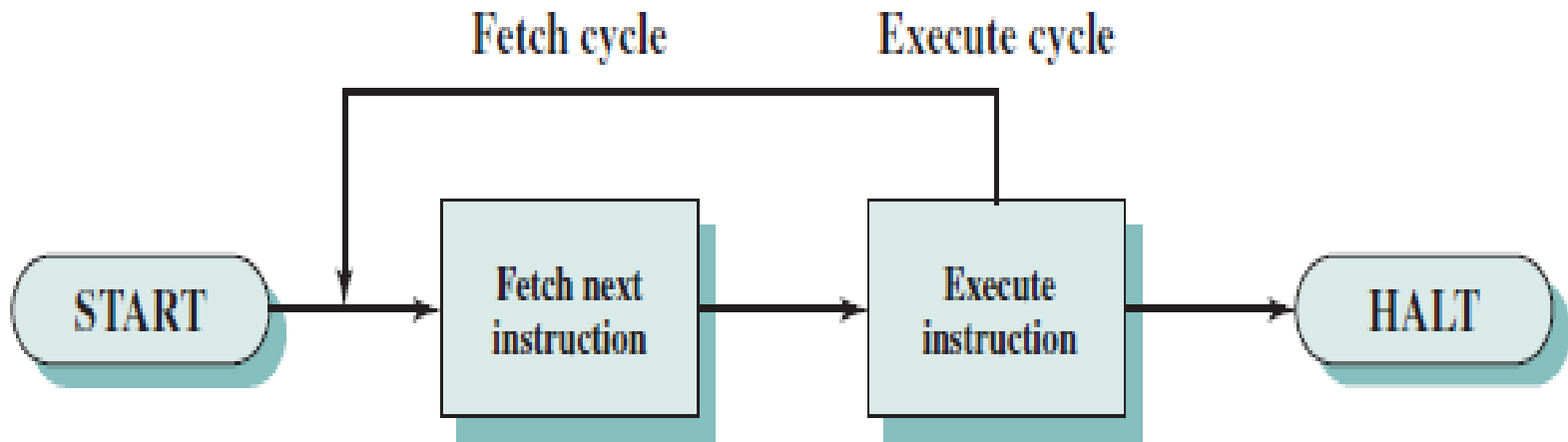
Fetch Cycle

- At the <u>beginning of each instruction cycle the processor fetches an instruction from memory.</u>

  – The **Program Counter (PC) holds the address of the instruction to be fetched next.**

  – The <u>processor increments the PC</u> after each instruction fetch so that it will fetch the next instruction in sequence.

- The fetched instruction is loaded into the **Instruction Register (IR).**

  – The processor interprets the instruction and performs the required action.

Execute Cycle

- **Processor-Memory**

  – Data transfer between CPU and main memory.

- **Processor-I/O**

  – Data transfer between CPU and I/O module.

- **Data Processing**

  – Some arithmetic or logical operation on data.

- **Control**

  – Alteration of sequence of operations.

  – e.g. Jump, Call.

- **Combination of above.**

Actions Performed by Instructions

- Data transferred from processor to memory or from memory to processor

Processor-memory

- Data transferred to or from a peripheral device by transferring between the processor and an I/O module

Processor-I/O

- An instruction may specify that the sequence of execution be altered

Control

Data processing

- The processor may perform some arithmetic or logic operation on data

- This is a general depiction of a von Neumann system:

- These computers employ a **fetch-decode-execute** cycle to run programs as follows . . .

Central Processing Unit

Program Counter

Registers

Arithmetic-Logic Unit

Control Unit

Main Memory

Input/Output System
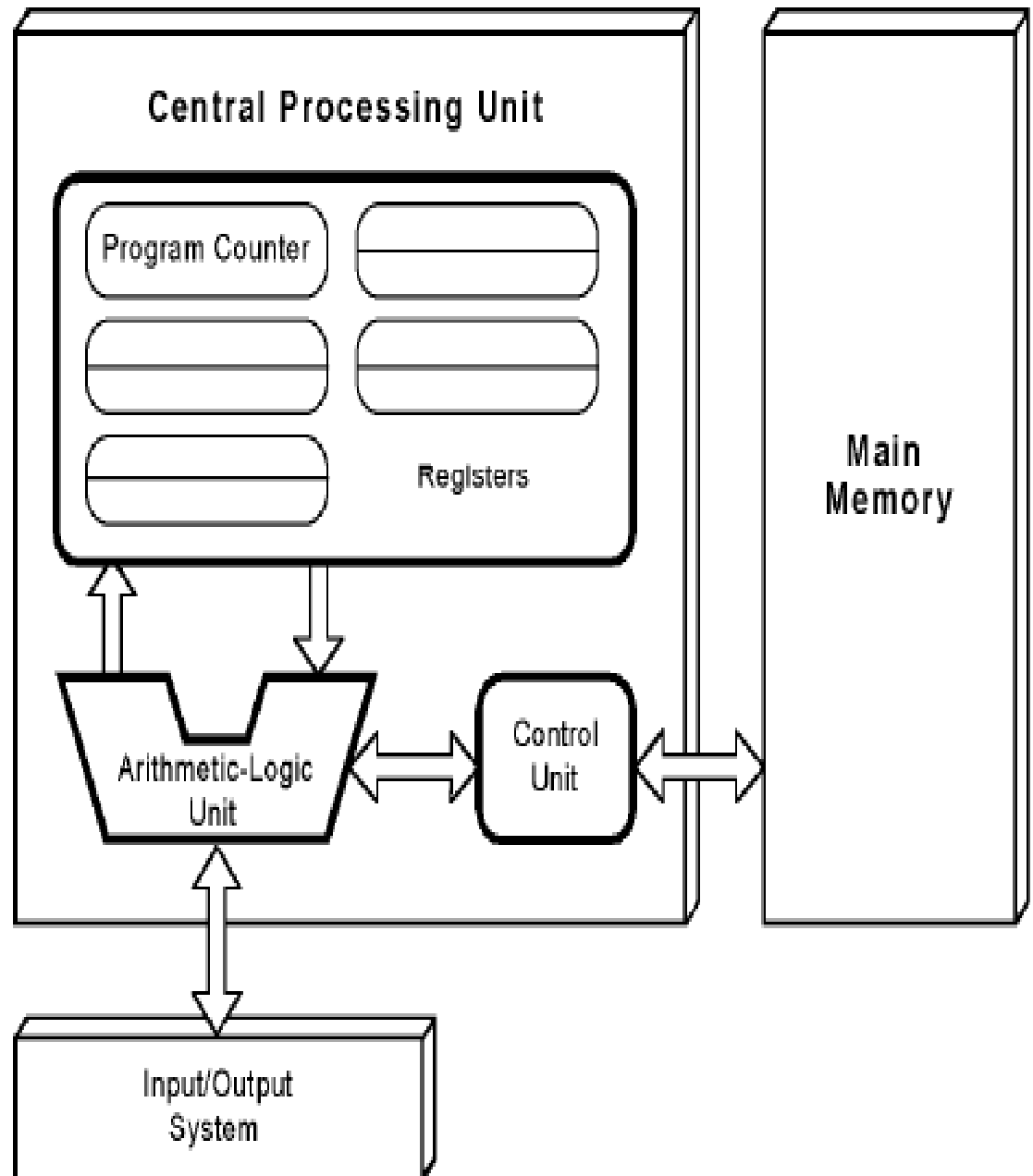
- **The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.**
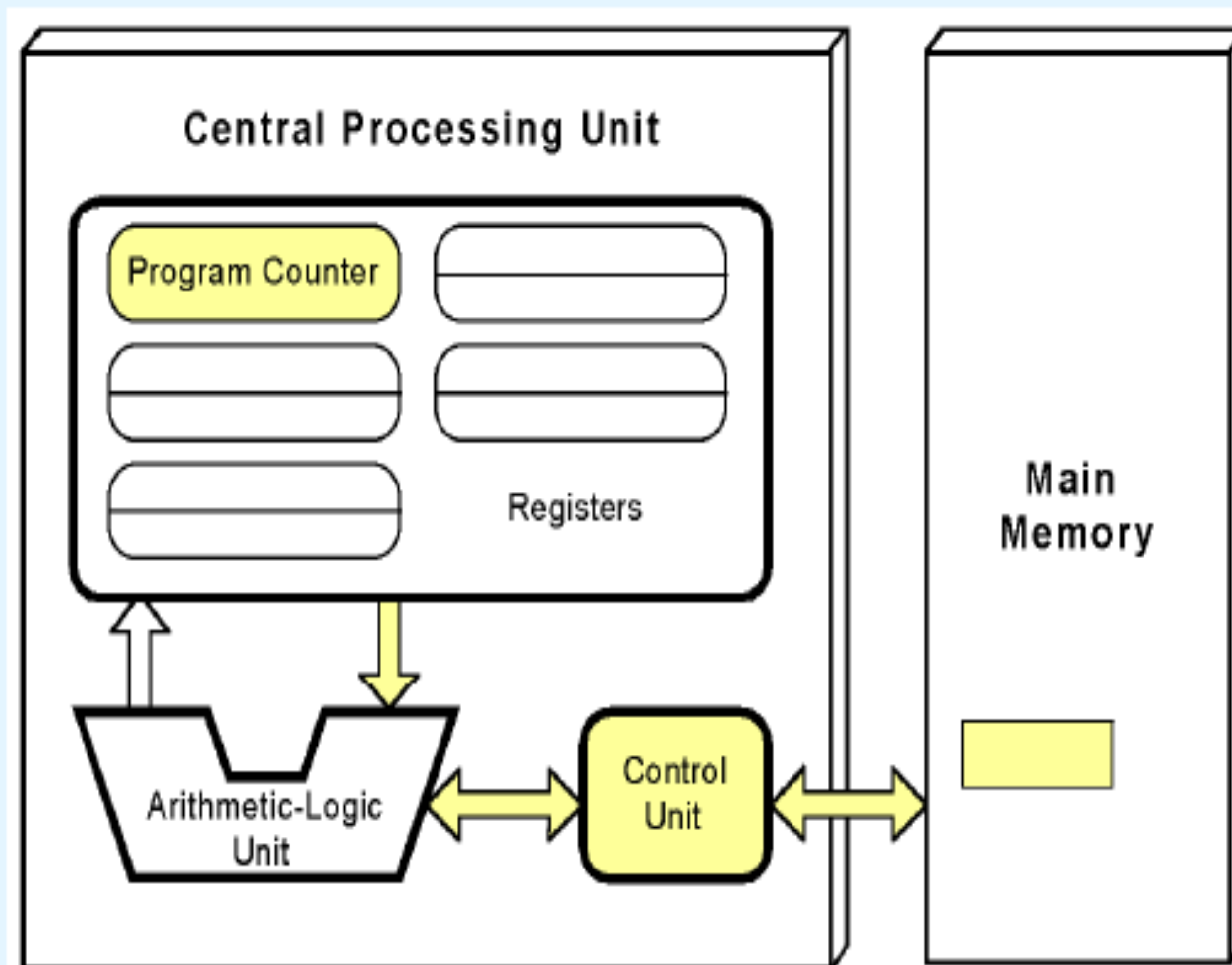
- **The instruction is decoded into a language that the ALU can understand.**
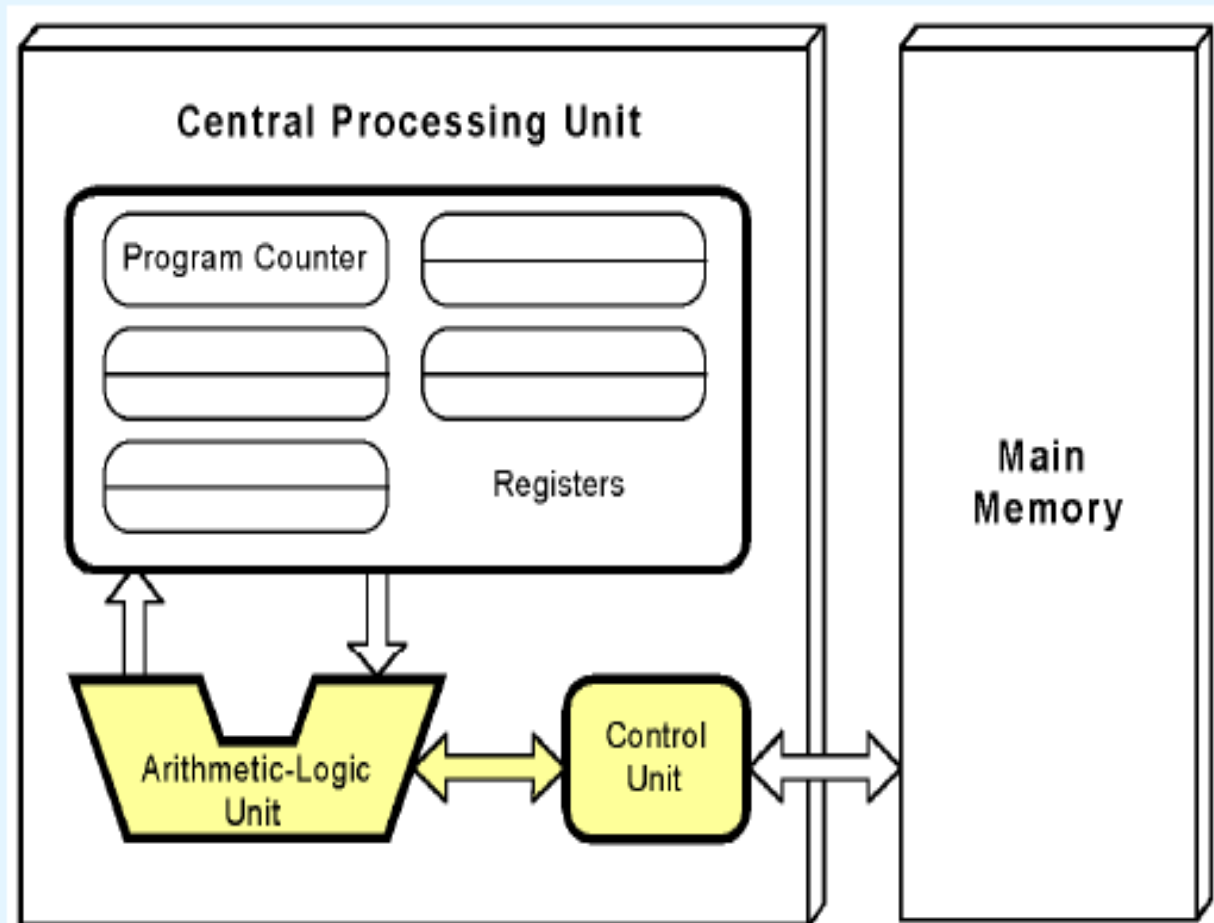
- **Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.**

Central Processing Unit

Program Counter

Registers

Arithmetic-Logic Unit

Control Unit

Main Memory
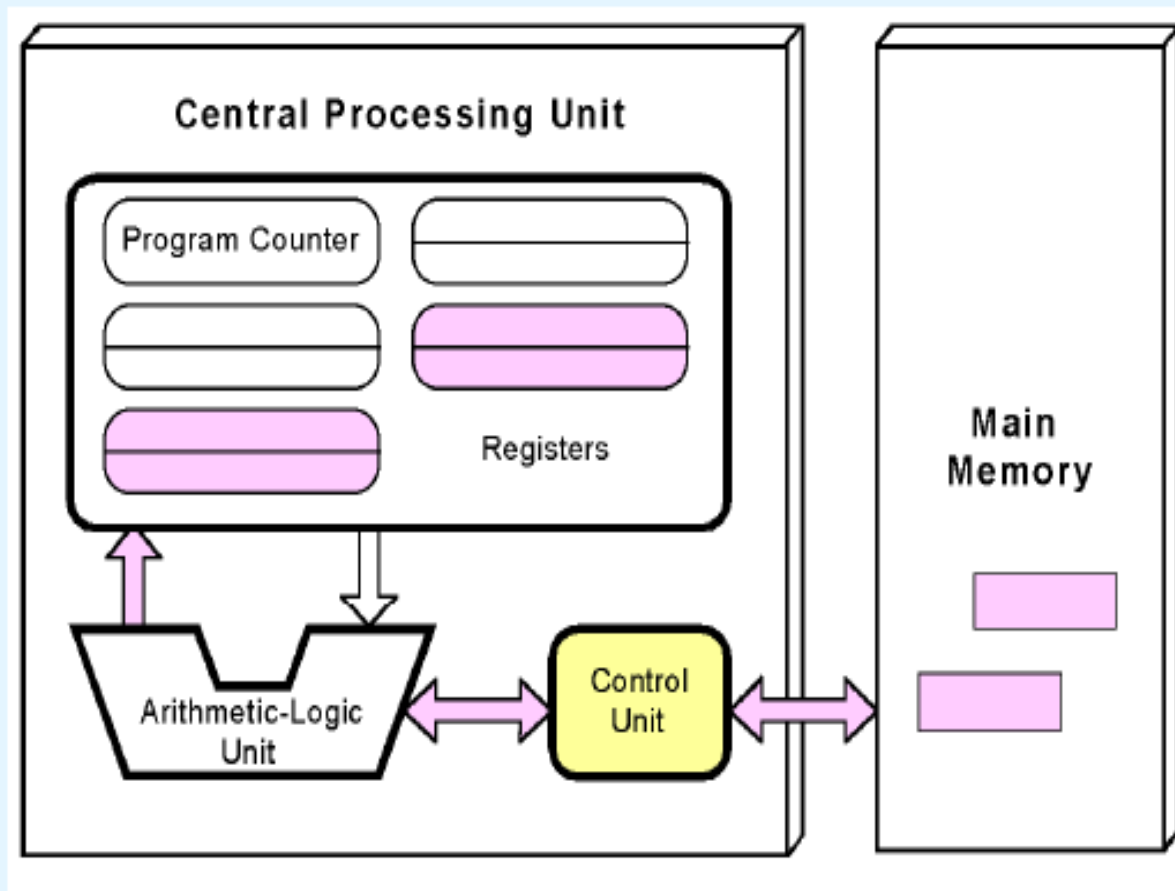
- **The ALU executes the instruction and places results in registers or memory.**

```
0                  3 4                                                    15
┌──────────────────┬─────────────────────────────────────────────────────┐
│                  │                                                     │
│     Opcode       │                    Address                          │
│                  │                                                     │
└──────────────────┴─────────────────────────────────────────────────────┘
```
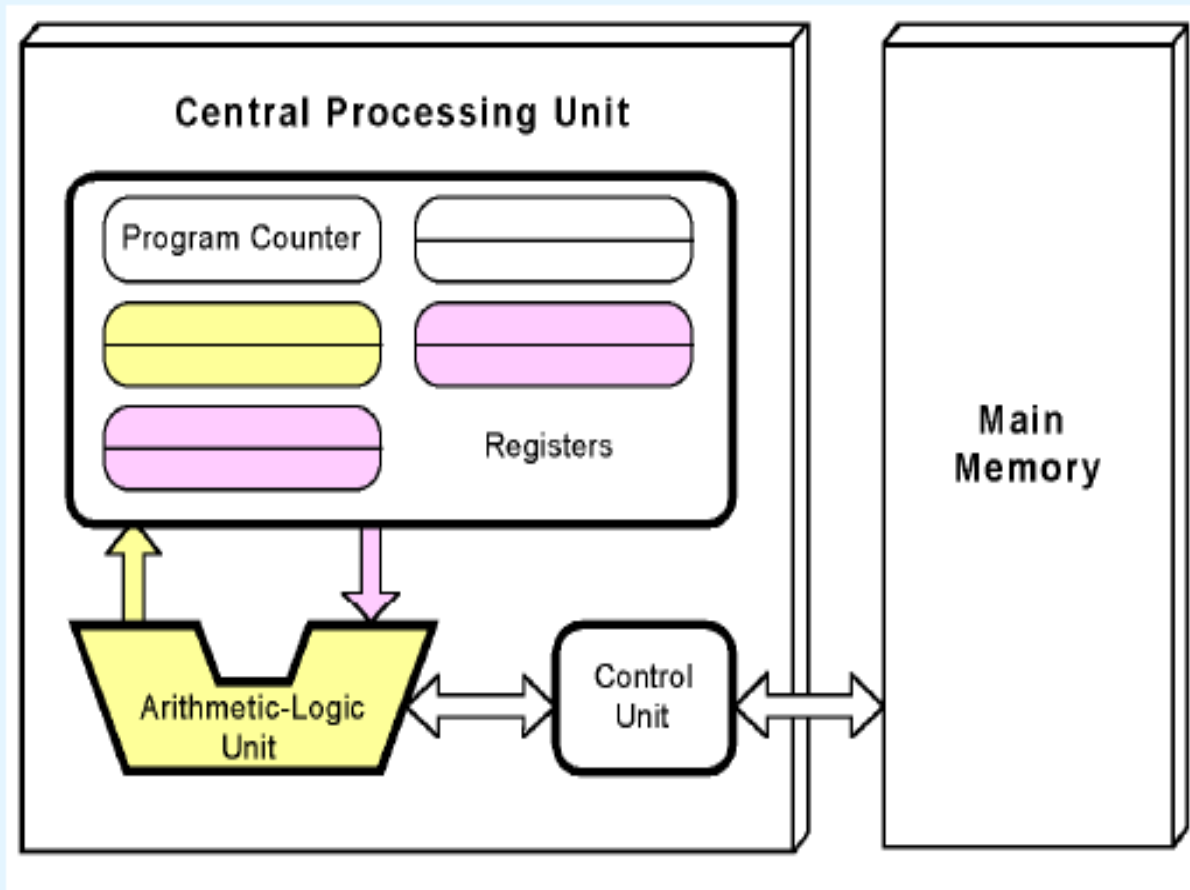
(a) Instruction format

```
0     1                                                                  15
┌─────┬──────────────────────────────────────────────────────────────────┐
│     │                                                                  │
│     │                       Magnitude                                  │
│     │                                                                  │
└─────┴──────────────────────────────────────────────────────────────────┘
```
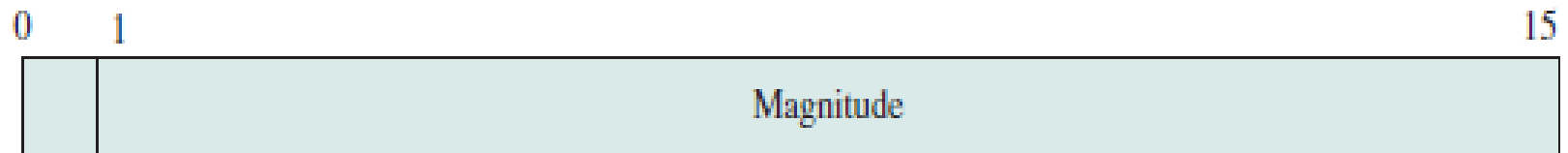
(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Example of Program Execution
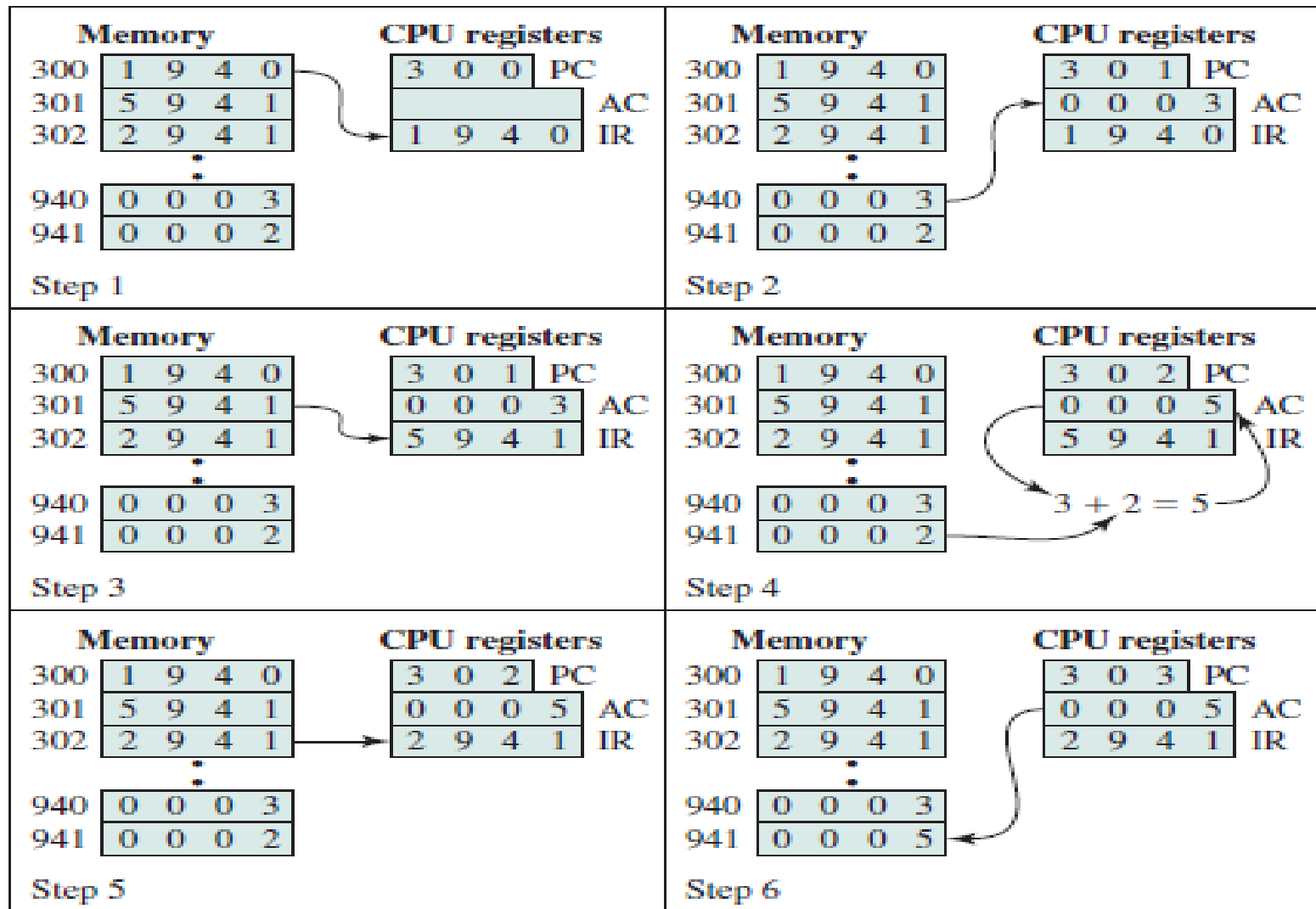
Instruction Cycle State Diagram
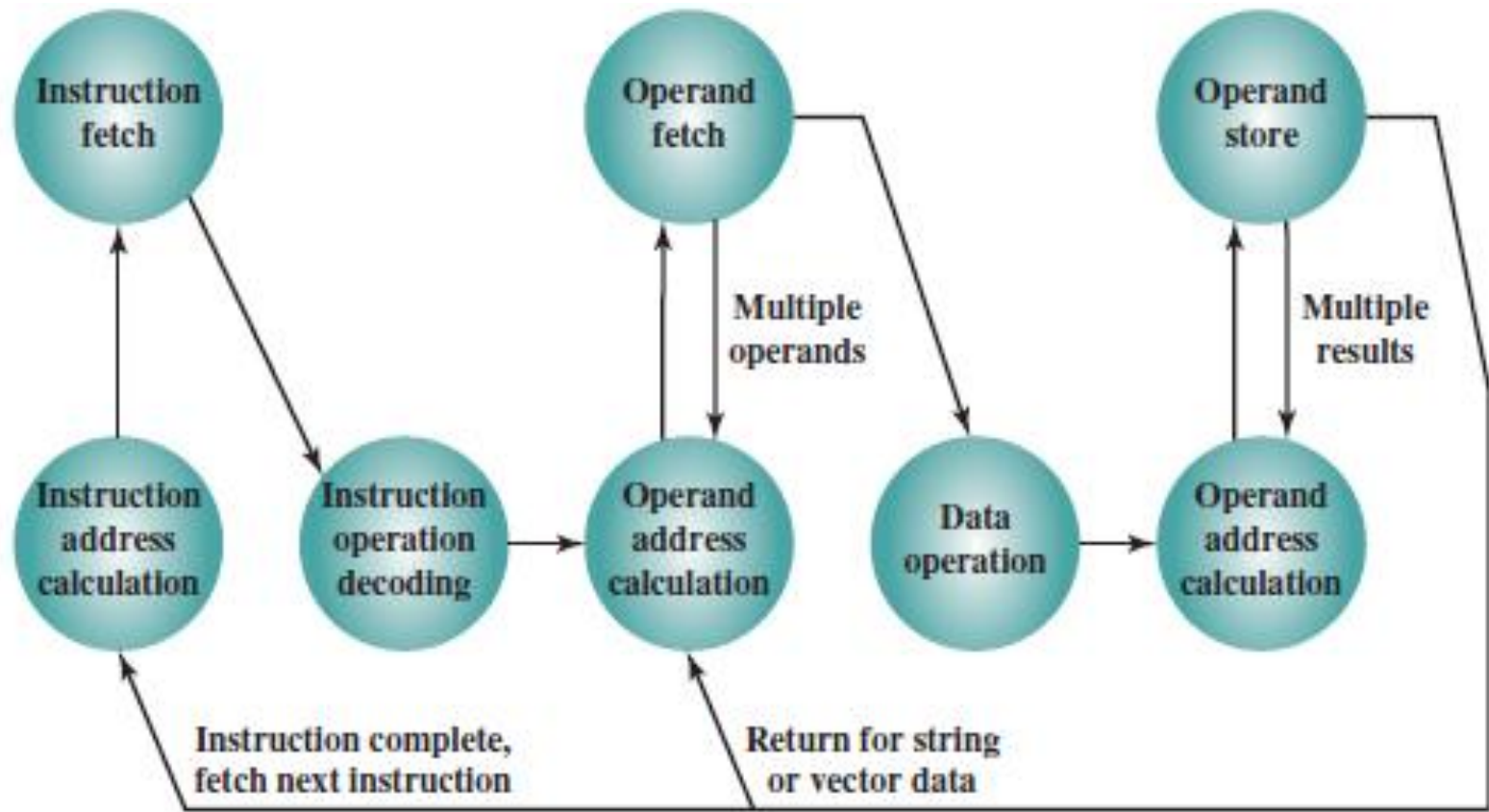
Interrupt

- In system programming, an **Interrupt** is a signal to the processor emitted by hardware or software indicating an **event that needs immediate attention.**

- Interrupt causes transfer of control to an **Interrupt service routine (ISR).** ISR is also called a **Handler**.

- When the ISR is completed, the original program resumes execution.

- Interrupts provide an efficient way to handle unanticipated events.

Interrupt Process (from three potential sources)

Hardware	Processor	Software
Interrupt Request (IRQ) sent from device to processor	Exception / Trap sent from processor to processor	Software Interrupt instruction loaded by processor

Processor halts thread execution

Processor saves thread state

Processor executes interrupt handler

Processor resumes thread execution

Interrupt Sources

- **Hardware Interrupts:** A hardware interrupt is an electronic alerting signal sent to the processor from an **external device**, either a part of the computer itself such as a Printer.

- Two types of this interrupt:
  - NMI
  - INTR: If IF flag=0.

- **Software Interrupt:** A software interrupt is caused either by an **exceptional condition in the processor itself**, or a **special instruction** in the instruction set which causes an interrupt when it is executed.

- Eg.

  DOS INT 21H, BIOS INT 10H.

  INT 00 (divide error)

  INT 01 (single step)

  INT 03 (breakpoint)

  INT 04 (signed number overflow)

- Software interrupt instructions function similarly to **subroutine calls** and are used for a variety of purposes.

- For example, computers often use software interrupt instructions to communicate with the disk controller to request data be read or written to the disk.

- **Error Conditions**: Events occurring during program execution that are exceptional enough that they cannot be handled within the program itself.

- Eg. Divide by zero error, System aborts, system faults.

- Also called as Trap or Exception.

What if two interrupts occur at same time?

When different type of interrupt (ie. software, NMI, INTR or automatic interrupts) occur at the same time, the one with the highest priority is handled first. Intel microprocessors use the following order of priority:

Interrupt Type	Priority
divide error interrupt, INT n, INT0 NMI INTR TRAP flag (single step)	highest ↕ lowest

Interrupts Vs. Procedures

Interrupts	**Procedures**
• Initiated by both *software* and *hardware.*	• Can only be initiated by *software.*
• Can handle *anticipated* and *unanticipated* internal as well as external events.	• Can handle *anticipated* events that are coded into the program.
• ISRs or interrupt handlers are memory resident.	• Typically loaded along with the program.
• Use numbers to identify an interrupt service.	• Use meaningful names to indicate their function.
• (E)FLAGS register is saved automatically.	• Do not save the (E)FLAGS register.

Classes of Interrupt

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Program Flow of Control without and with Interrupts
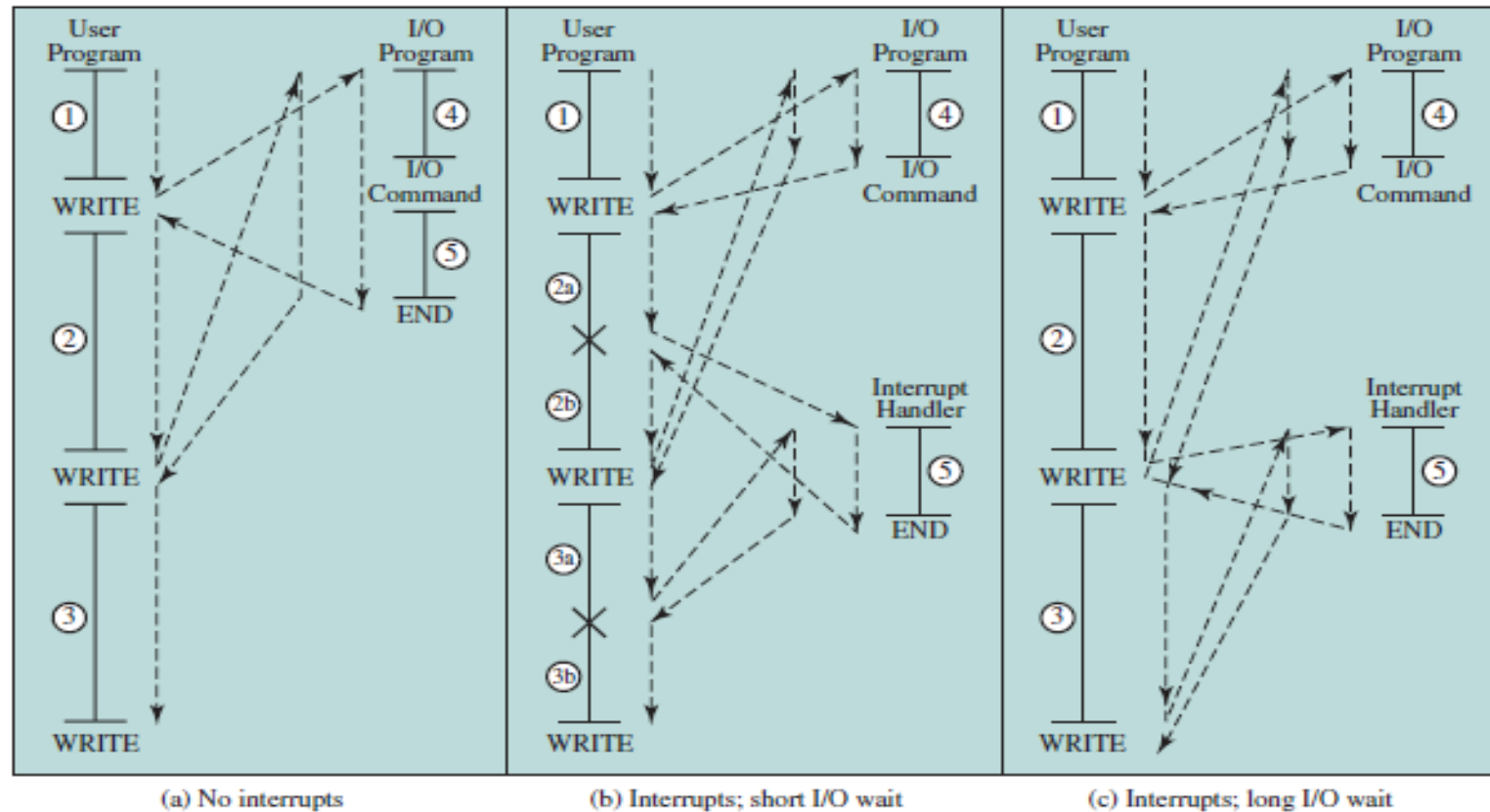
Figure 3.7 Program Flow of Control without and with Interrupts

Instruction Cycle With Interrupts
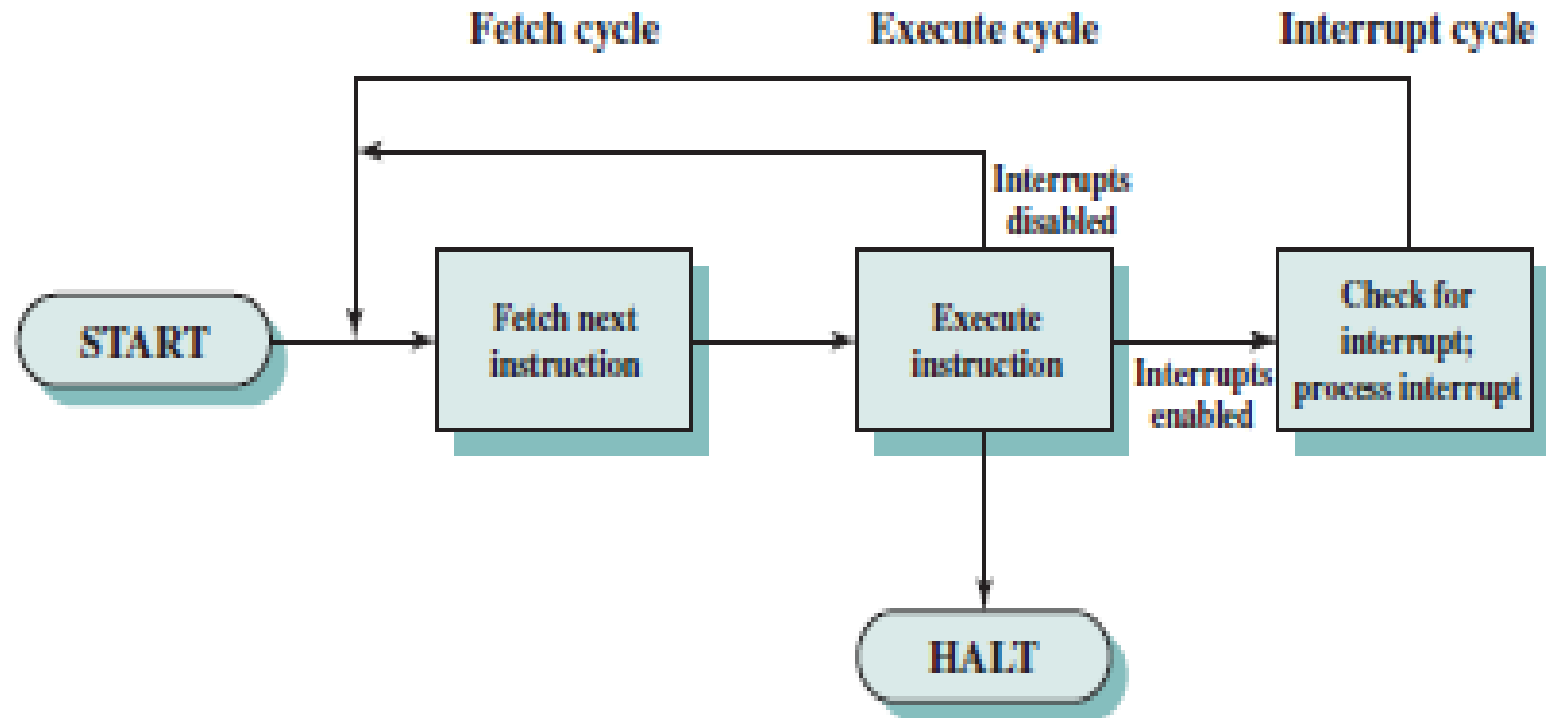
Instruction Cycle State Diagram, with Interrupts
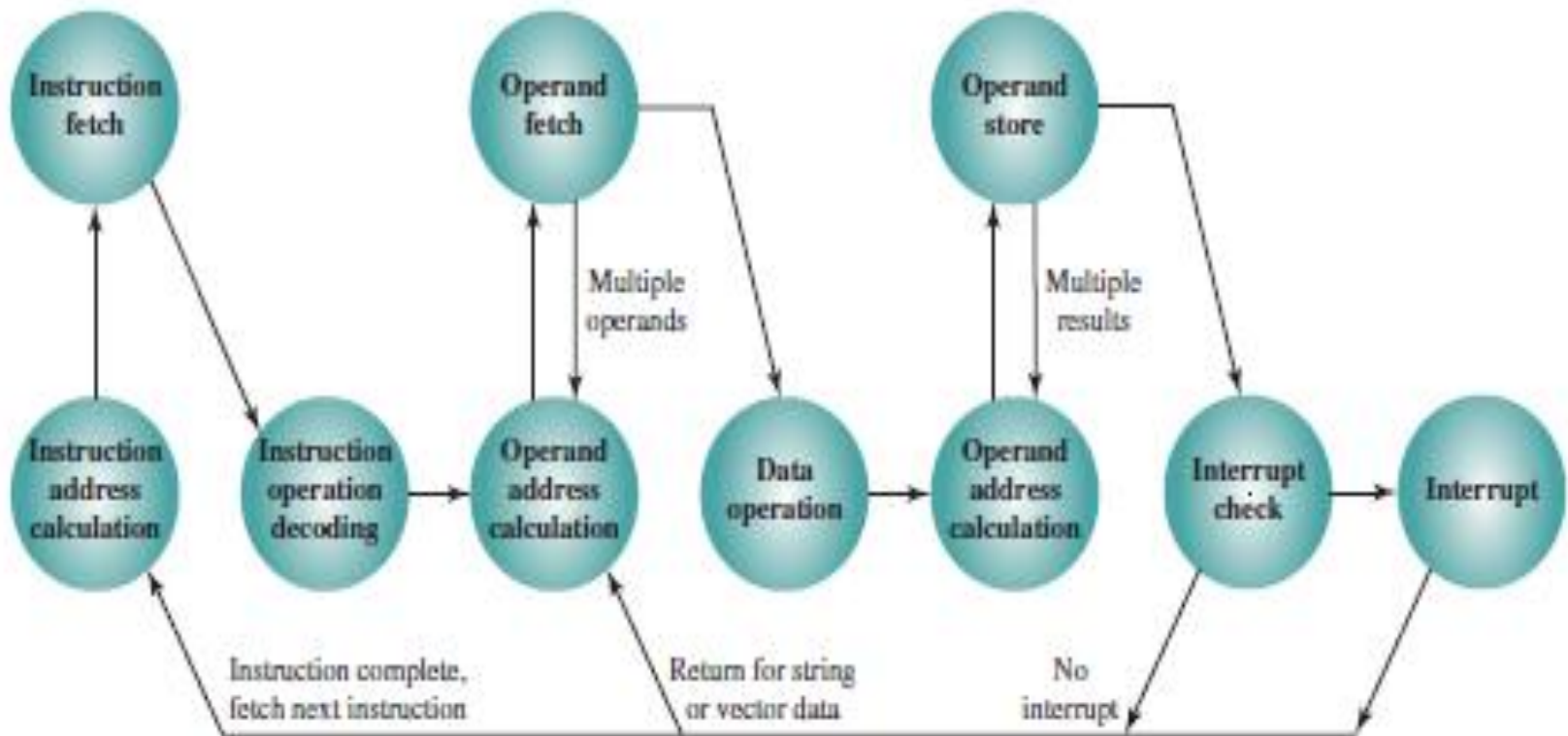
Transfer of Control with Multiple Interrupts
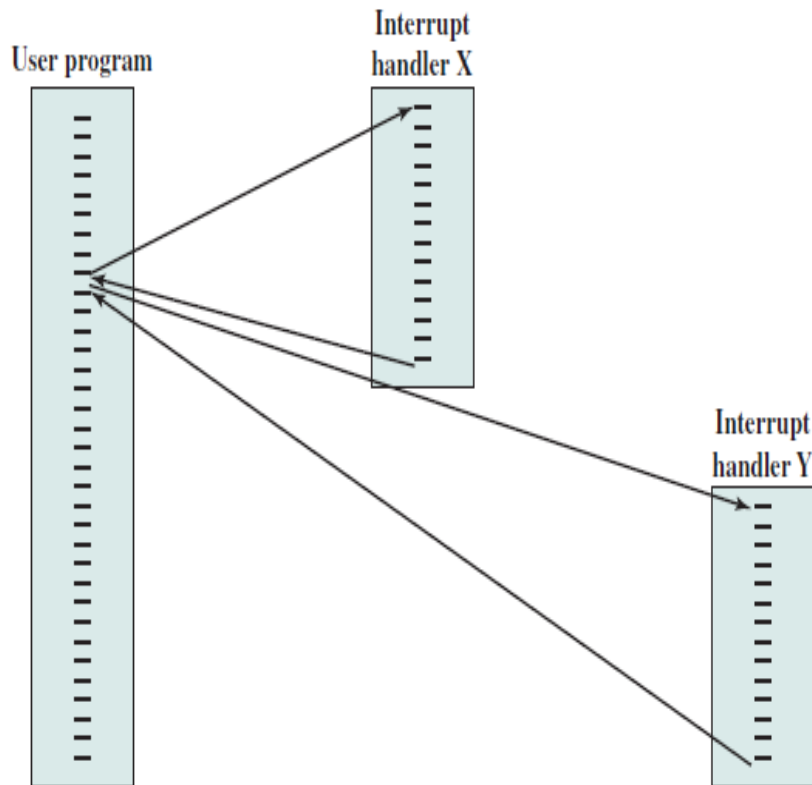
(a) Sequential interrupt processing
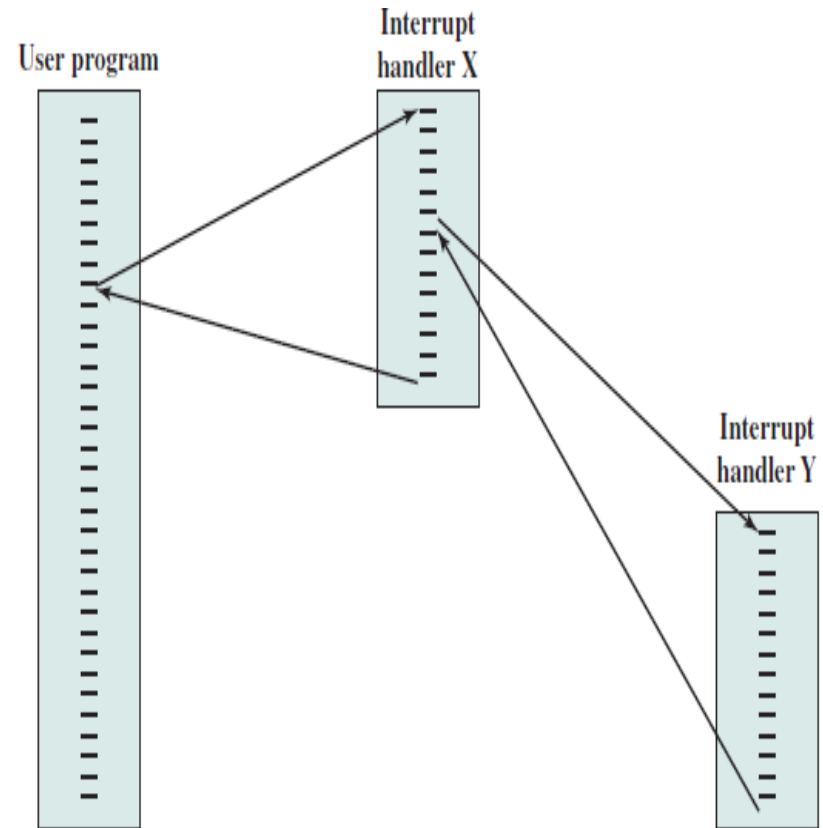
(b) Nested interrupt processing

I/O Function

- I/O module can exchange data directly with the processor.

- Processor can read data from or write data to an I/O module.

    – Processor **identifies a specific device that is controlled by a particular I/O module.**

    – **Uses I/O instructions** rather than memory referencing instructions.

- In some cases **it is desirable to allow I/O exchanges to occur directly with memory.**

  - **The processor grants to an I/O module the authority to read from or write to memory** so that the I/O memory transfer can occur without tying up the processor.

  - **The I/O module issues read or write commands** to memory relieving the processor of responsibility for the exchange.

  - This operation is known as **Direct Memory Access (DMA).**

Interconnection Structures

- **A computer consists of a set of components or modules** of three basic types (**CPU, Memory, I/O**) that communicate with each other.

- In effect, a computer is a network of basic modules. **Thus, there must be paths for connecting the modules**.

- The collection of paths connecting the various modules is called the **Interconnection Structure**.

- The design of this structure will depend on the exchanges that must be made among modules.

Review: Bus Concept
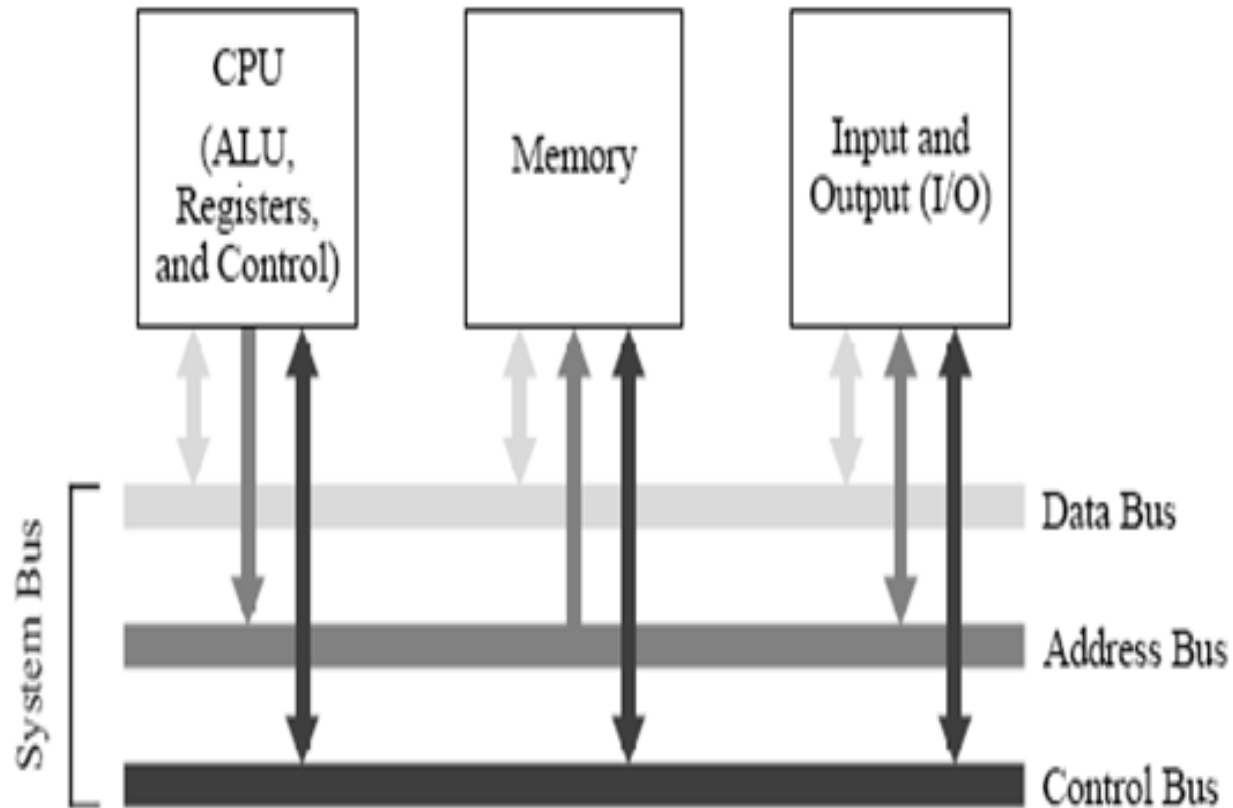
- There are normally three types of bus in any processor system:

  - **Address Bus**: this determines the location in memory that the processor will read data from or write data to.

  - **Data Bus**: this contains the contents that have been read from the memory location or are to be written into the memory location.

  - **Control Bus**: this manages the information flow between components indicating whether the operation is a read or a write and ensuring that the operation happens at the right time.

Review: CPU Building Blocks

➢ **Registers**

➢ **Control Unit (CU)**

➢ **Arithmetic Logic Unit (ALU)**
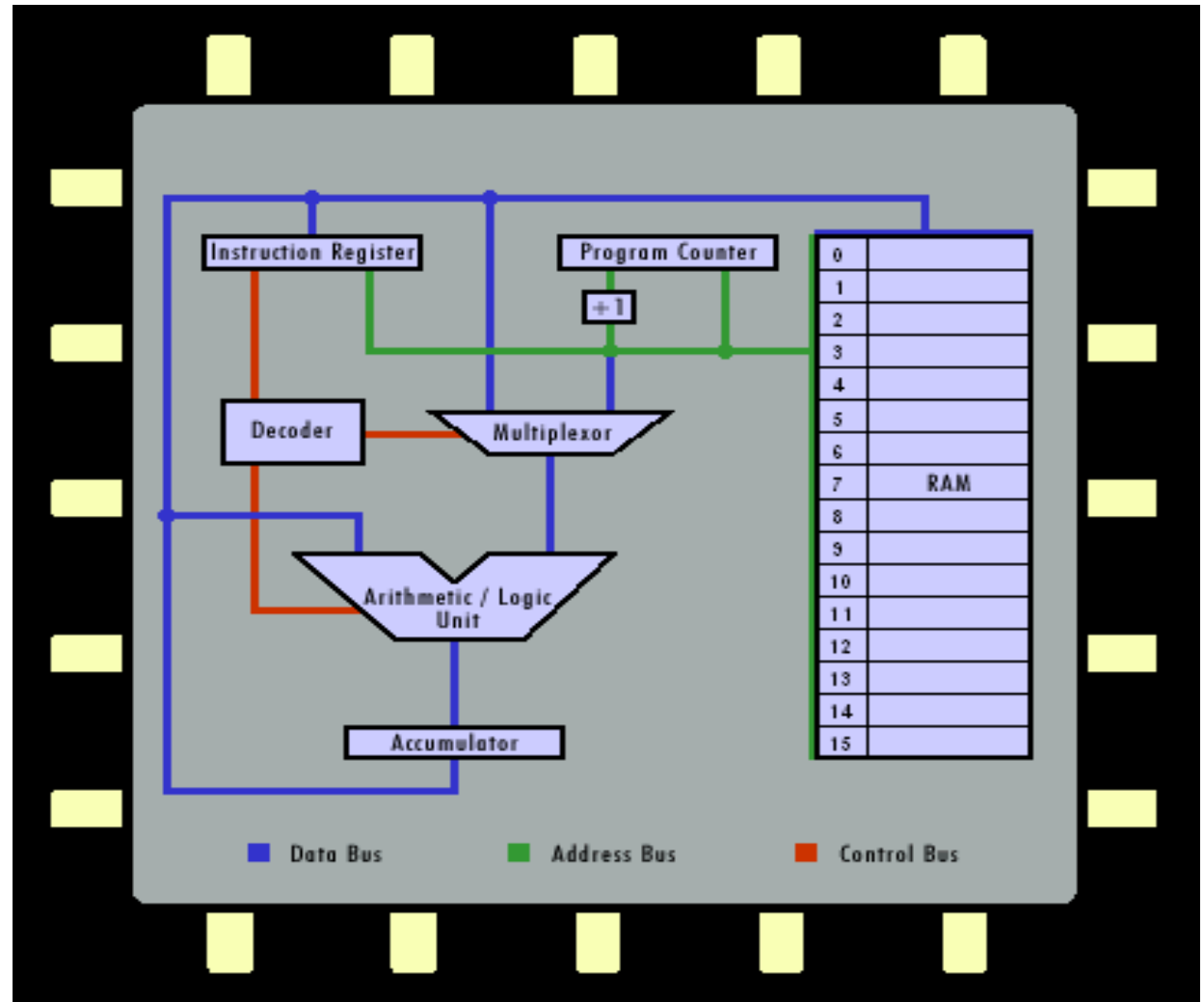
- **Memory:** Typically, a memory module will consist of *N words of equal length.* Each word is assigned a unique numerical address (0, 1, ..., *N - 1). A word of* data can be read from or written into the memory. The nature of the operation.
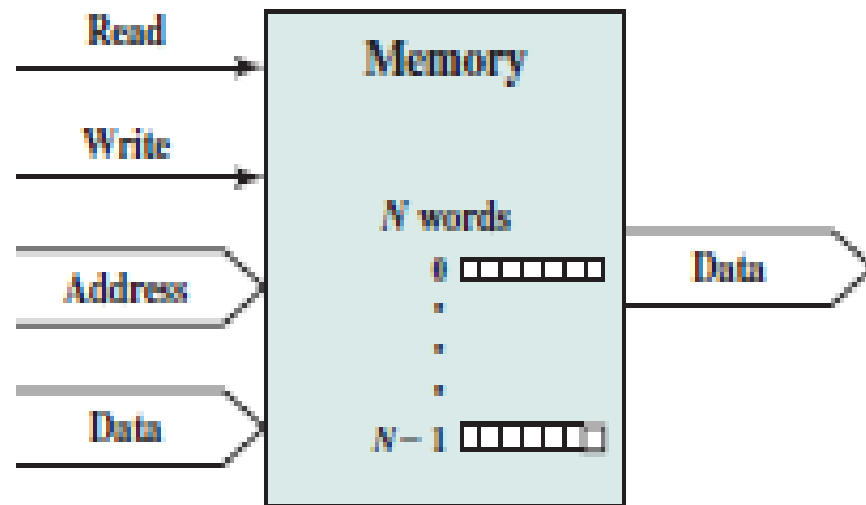
- **I/O Modules:** From an internal (to the computer system) point of view, I/O is functionally similar to memory. There are two operations, read and write. Further, an I/O module may control more than one external device. We can refer to each of the interfaces to an external device as a *port and give each* a unique address (e.g., 0, 1, ..., *M - 1). In addition, there are external data* paths for the input and output of data with an external device. Finally, an I/O module may be able to send interrupt signals to the processor.
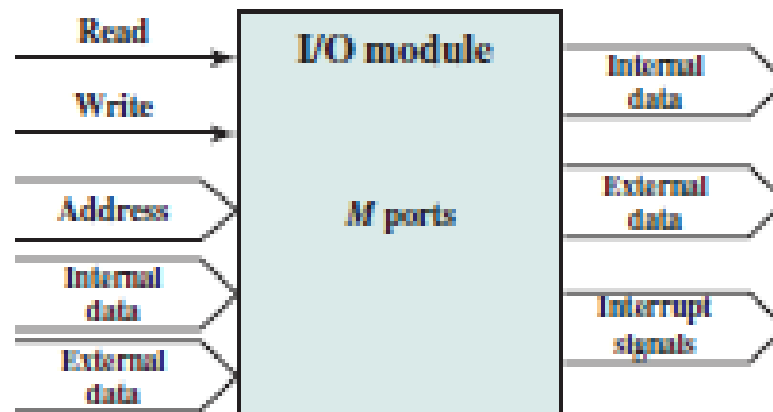
- **Processor:** The processor reads in instructions and data, writes out data after processing, and uses control signals to control the overall operation of the system. It also receives interrupt signals.
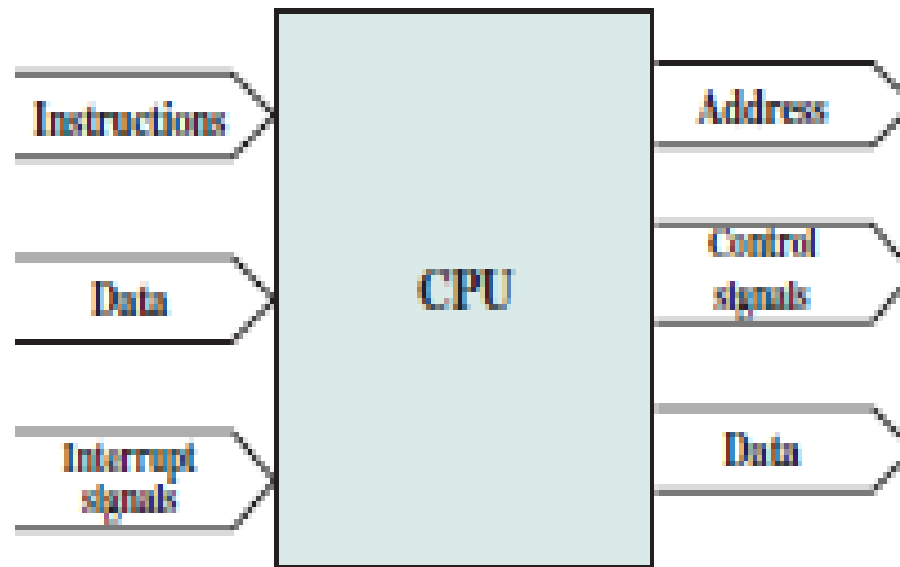
Arithmetic & Logic Unit

- Does the calculations.

- Everything else in the computer is there to service this unit.

- Handles integers.

- May handle floating point (real) numbers.

- May be separate **FPU** (math co-processor).

- May be **on chip separate FPU** (486DX +).

ALU Inputs and Outputs
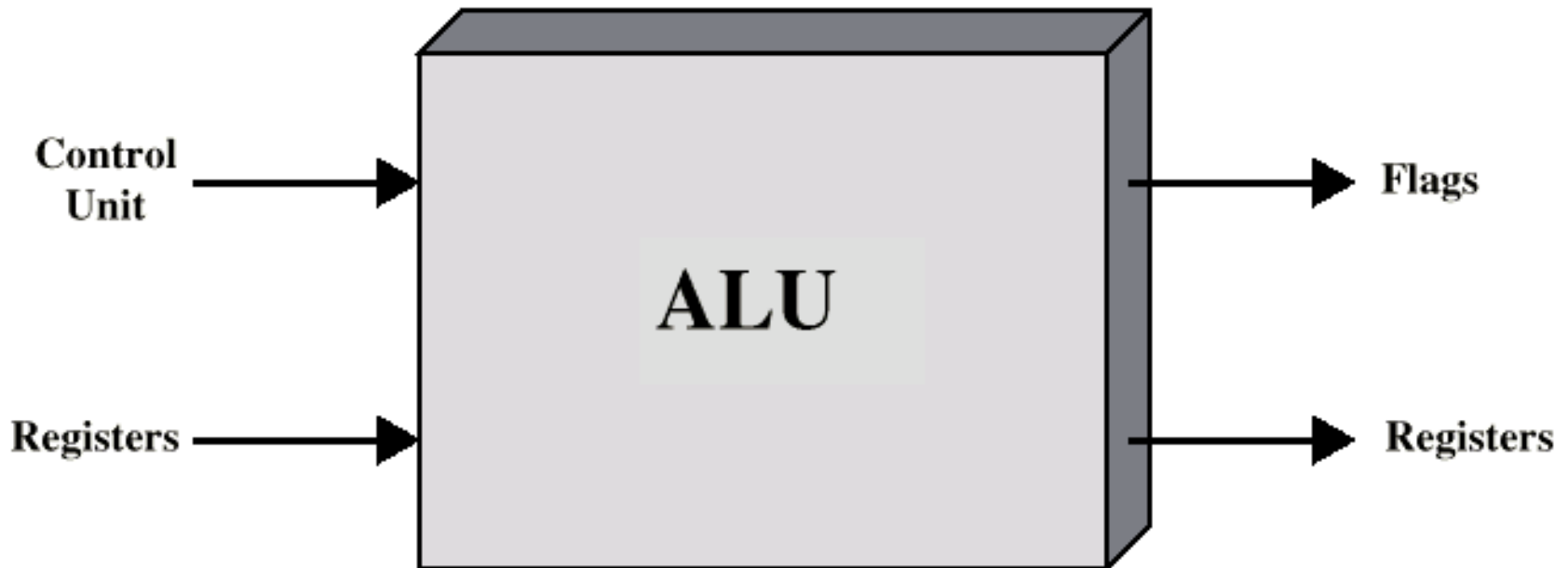
Integer Representation

- Only have 0 & 1 to represent everything.

- Positive numbers stored in binary.
  - e.g. 41=00101001

- No minus sign.

- **Two ways to represent negative numbers:**
  - **Sign Magnitude**
  - **Two's Complement**

Sign-Magnitude

- Left most bit is sign bit.
  - 0 means Positive.
  - 1 means Negative.
- +18 = 00010010
- -18 = 10010010
- Problems
  - Need to consider both sign and magnitude in arithmetic.
  - Two representations of zero (+0 and -0).

Two's Compliment

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Benefits of 2's Complement

- One representation of zero.

- Arithmetic works easily.

- Negating is fairly easy
  - 3 = 00000011
  - Boolean complement gives 11111100
  - Add 1 to LSB   11111101

Geometric Depiction of Twos Complement Integers

(a) 4-bit numbers

(b) n-bit numbers
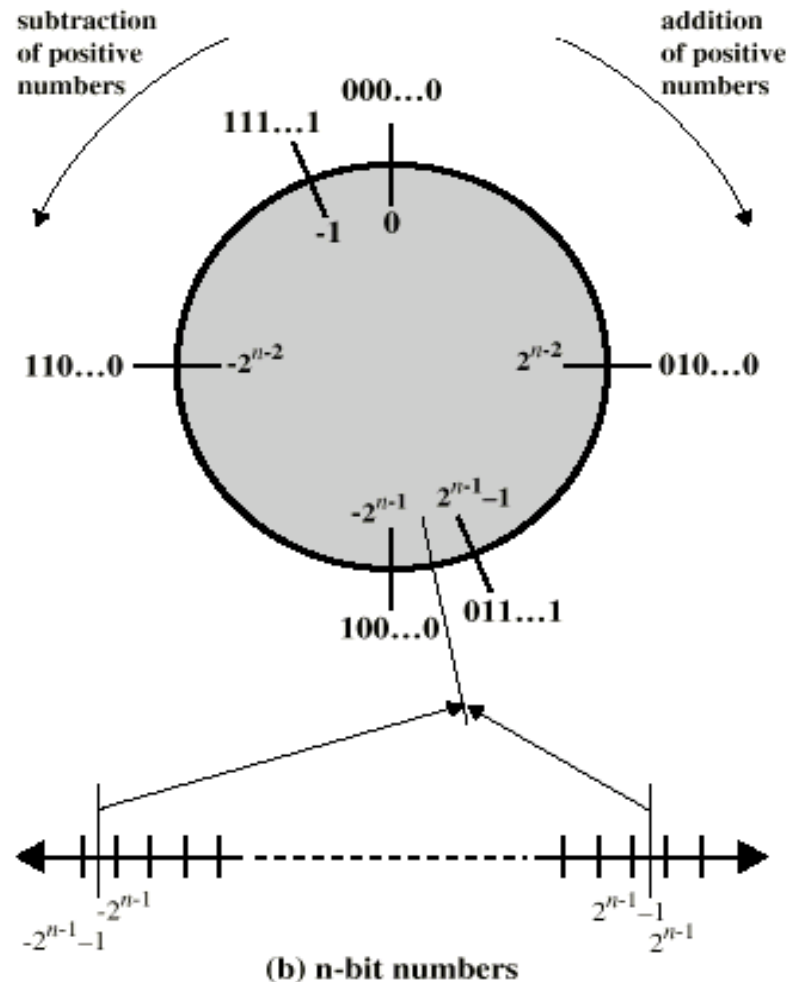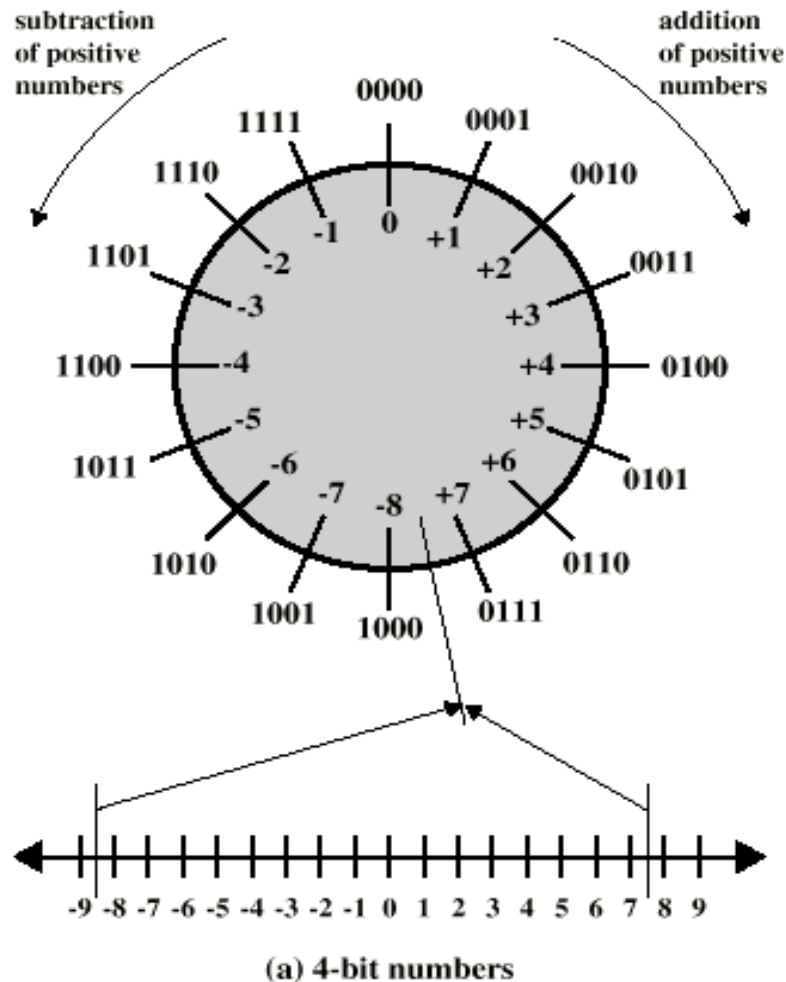
Range of Numbers

- 8 bit 2s compliment
  - **+127 = 01111111 = $2^7 - 1$**
  - **-128 = 10000000 = -2^7**

- 16 bit 2s compliment
  - **+32767 = 011111111 11111111 = $2^{15} - 1$**
  - **-32768 = 100000000 00000000 = -2^{15}**

General concept: Addition

- Decimal addition

$$
\begin{array}{r}
\textbf{(carry)} \quad 1\_ \\
19 \\
+ \ 7 \\
\hline
26
\end{array}
$$

- Binary addition

$$
\begin{array}{r}
(\text{carry}) \quad 111\_ \\
10011 \\
+ \ 111 \\
\hline
11010
\end{array}
$$

- 16+8+2 = 26

Signed and unsigned additions

- Unsigned addition in 4-bit arithmetic

```
( carry)   11_
           1011
         + 0011
           1110
```

- `11 + 3 = 14`
`(8 + 4 + 2)`

- Signed addition in 4-bit arithmetic

```
( carry)   11_
           1011
         + 0011
           1110
```

- `-8 + 6 = -2`

Subtraction

```
         0010  =   2                    0101  =   5
        +1001  = —7                    +1110  = —2
         1011  = —5                    10011  =   3

(a)  M = 2 = 0010             (b)  M = 5 = 0101
     S = 7 = 0111                  S = 2 = 0010
    —S =       1001              —S =       1110


         1011  = —5                    0101  = 5
        +1110  = —2                   +0010  = 2
         11001  = —7                   0111  = 7

(c)  M = —5 = 1011            (d)  M =   5 = 0101
     S =   2 = 0010                S = —2 = 1110
    —S =        1110             —S =        0010


         0111  = 7                     1010  = —6
        +0111  = 7                    +1100  = —4
         1110  = Overflow             10110  = Overflow

(e)  M =   7 = 0111          (f)  M = —6 = 1010
     S = —7 = 1001                S =   4 = 0100
    —S =        0111             —S =        1100
```

Hardware for Addition and Subtraction
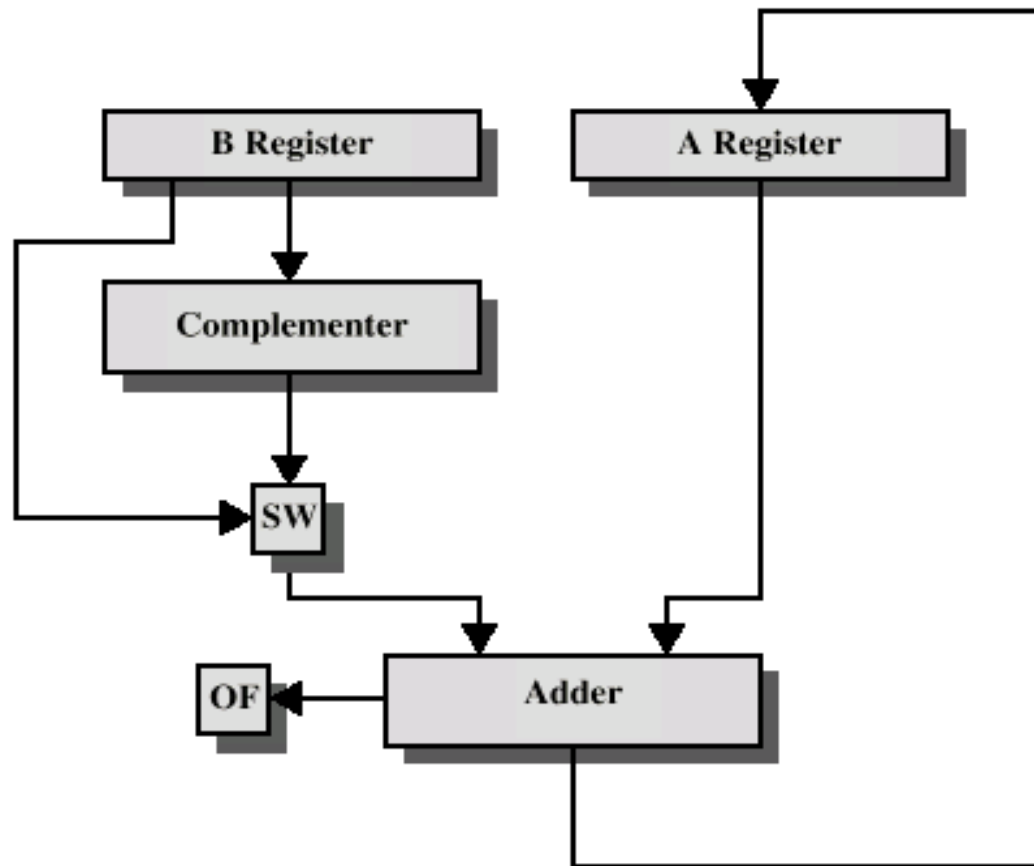
OF = overflow bit
SW = Switch (select addition or subtraction)

Multiplication

- Complex.

- Work out partial product for each digit.

- Take care with place value (column).

- Add partial products.

Multiplication Example

```
      1011        Multiplicand (11)
    ×1101         Multiplier (13)
      1011  ⎫
      0000  ⎪
      1011  ⎬     Partial products
     1011   ⎪
  10001111  ⎭     Product (143)
```

Unsigned Binary Multiplication
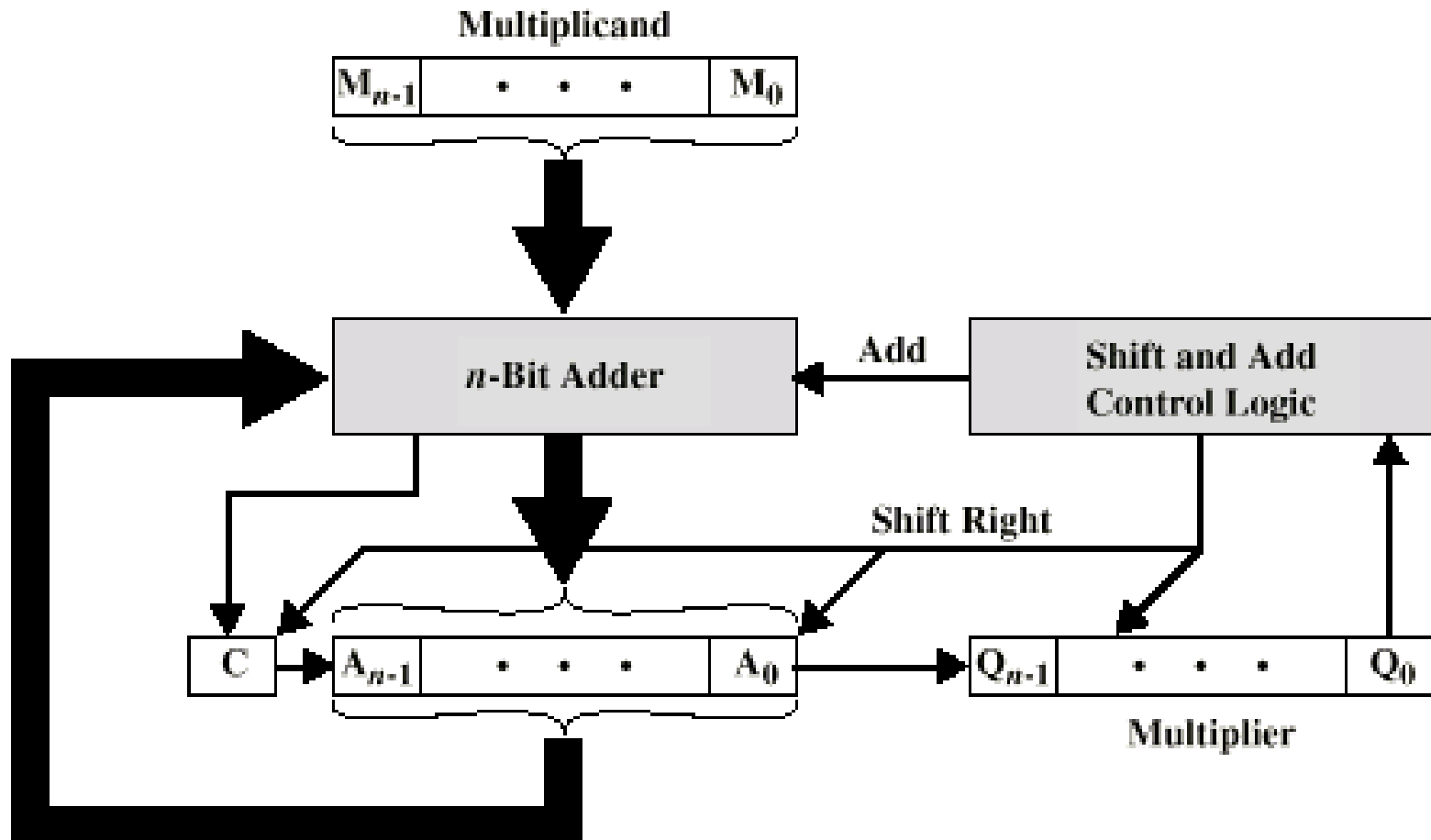
(a) Block Diagram

Flowchart for Unsigned Binary Multiplication
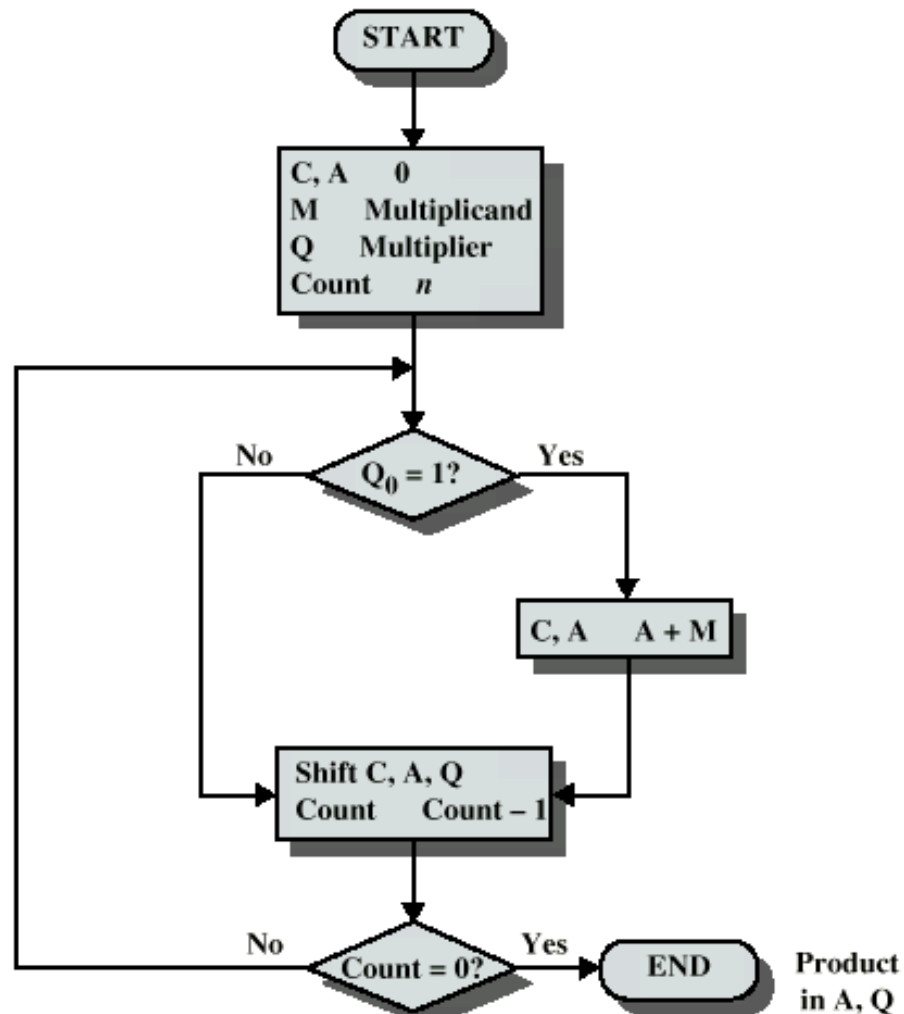
Multiplying Negative Numbers

- This does not work!

- Solution 1
  - Convert to positive if required.
  - Multiply as above.
  - If signs were different, negate answer.

- Solution 2
  - Booth's algorithm.

Booth's Algorithm
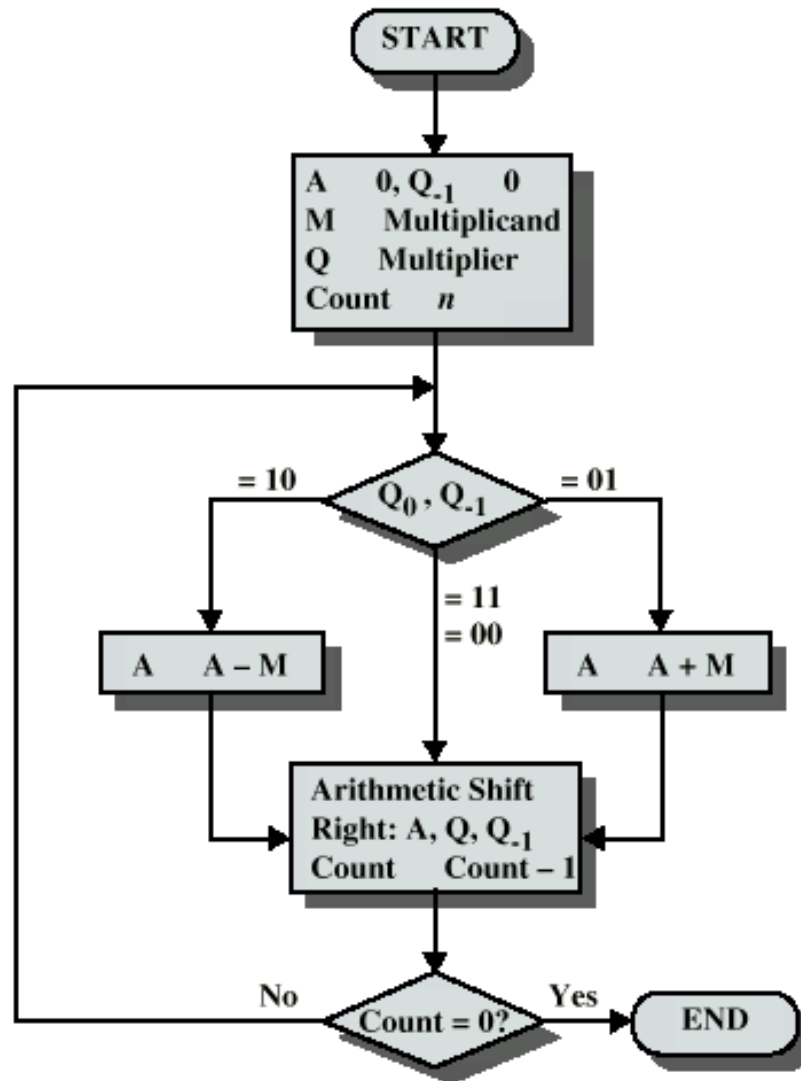
Example of Booth's Algorithm

A	Q	Q_{-1}	M	
0000	0011	0	0111	Initial Values
1001	0011	0	0111	A ← A − M } First
1100	1001	1	0111	Shift } Cycle
1110	0100	1	0111	Shift } Second Cycle
0101	0100	1	0111	A ← A + M } Third
0010	1010	0	0111	Shift } Cycle
0001	0101	0	0111	Shift } Fourth Cycle

Division

- More complex than multiplication.

- Based on long division.

Division of Unsigned Binary Integers

```
                         00001101              ←——————— Quotient

Divisor  ——→   1011  | 10010011              ←——————— Dividend
                       1011
                      ———————
                       001110
                         1011
                        ———————
                         001111
                           1011
                          ———————            ←——————— Remainder
                            100
```

Partial Remainders
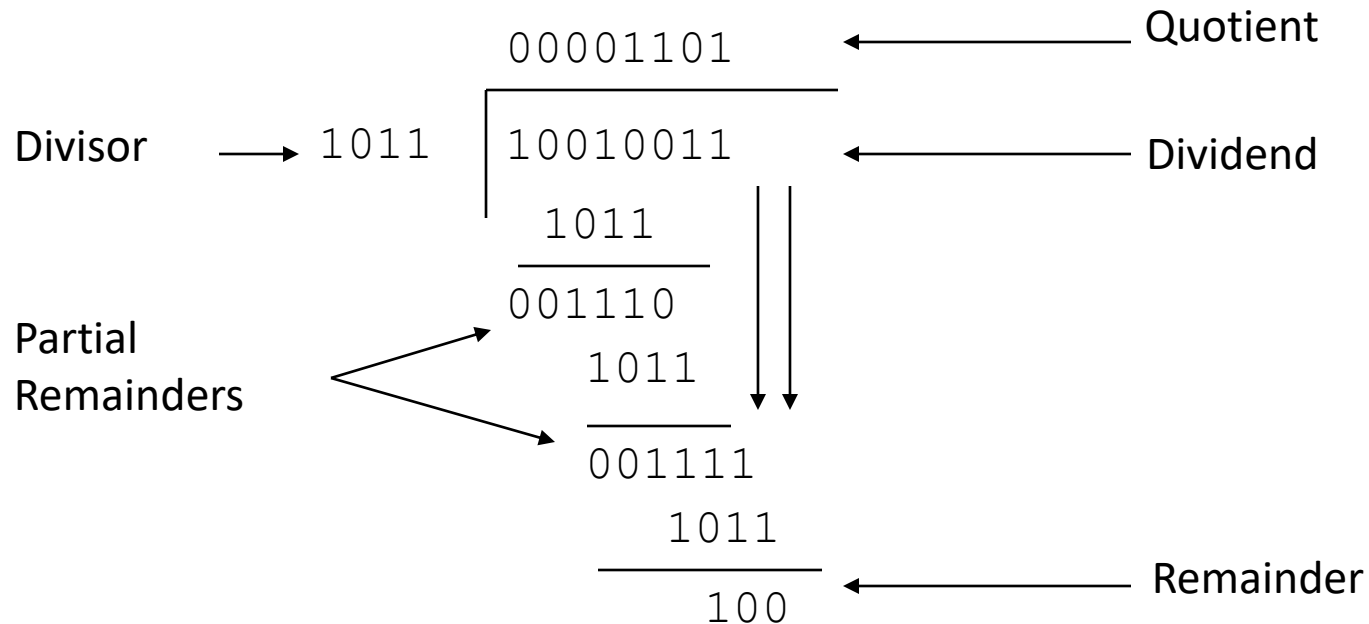
Flowchart for Unsigned Binary Division

START

$A \leftarrow 0$
$M \leftarrow$ Divisor
$Q \leftarrow$ Dividend
$Count \leftarrow n$

Shift Left
A, Q

$A \leftarrow A - M$

A < 0?

No   Yes

$Q_0 \leftarrow 1$

$Q_0 \leftarrow 0$
$A \leftarrow A + M$

$Count \leftarrow Count - 1$

Count = 0?

No   Yes

END

Quotient in Q
Remainder in A
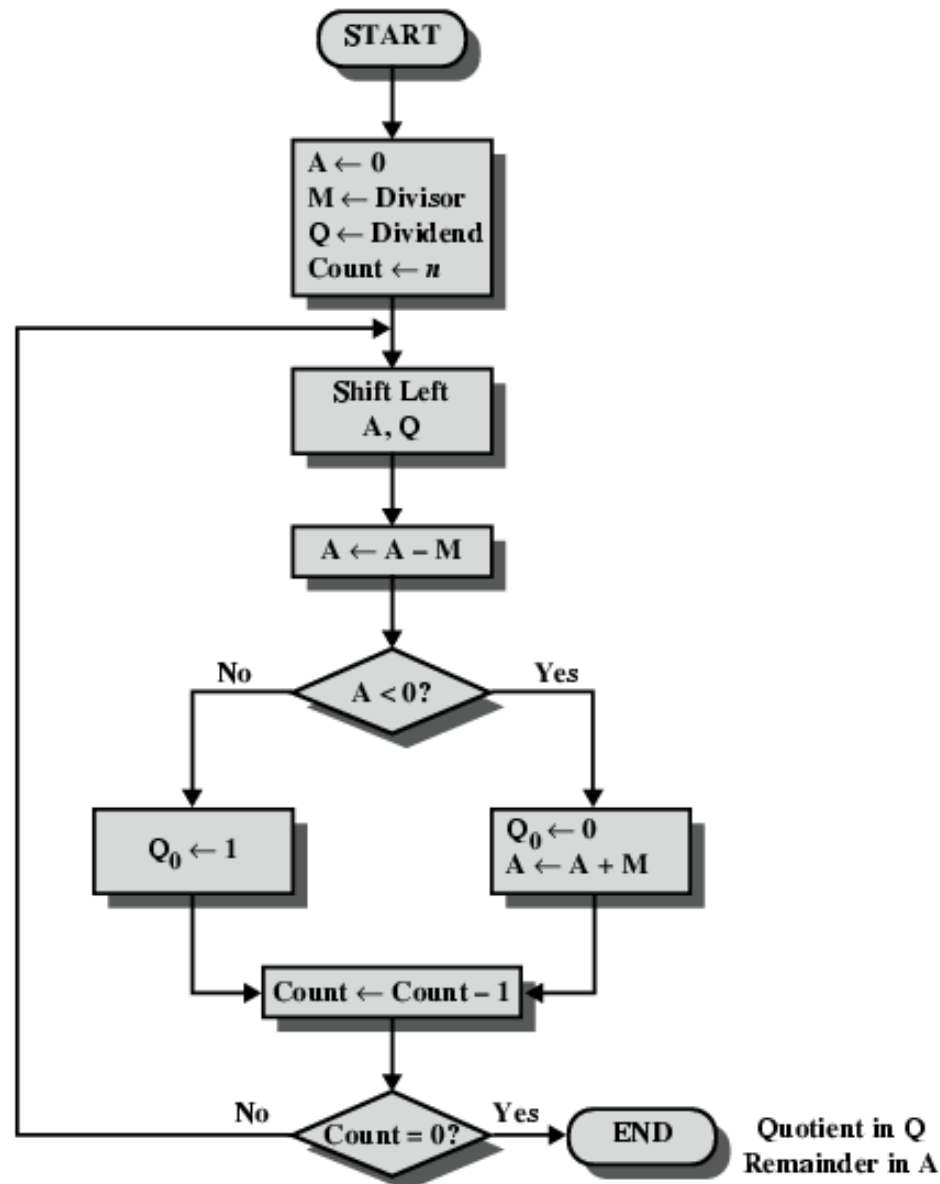
Floating Point Representations: IEEE Standards

Two Types

1. Single Precision (bias of 127)

2. Double Precision (bias of 1023)

IEEE 754 Standard

- Most common standard for representing floating point numbers
- Single precision: 32 bits, consisting of...
  - Sign bit (1 bit)
  - Exponent (8 bits)
  - Mantissa (23 bits)
- Double precision: 64 bits, consisting of…
  - Sign bit (1 bit)
  - Exponent (11 bits)
  - Mantissa (52 bits)

Single Precision & Double Precision

0	1 8	9 31
S	Biased exponent (E)	Significand (N)

Single Precision = 32 bits

0	1 11	12 63
S	Biased exponent (E)	Significand (N)
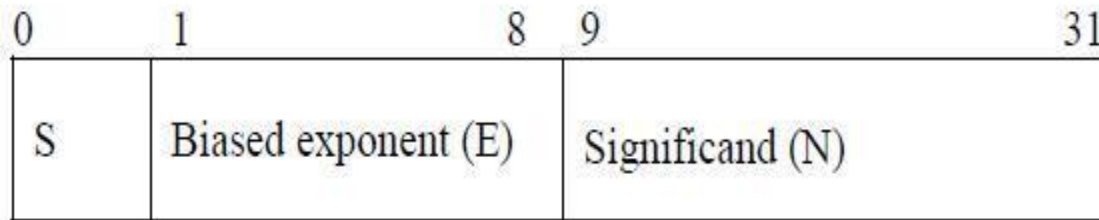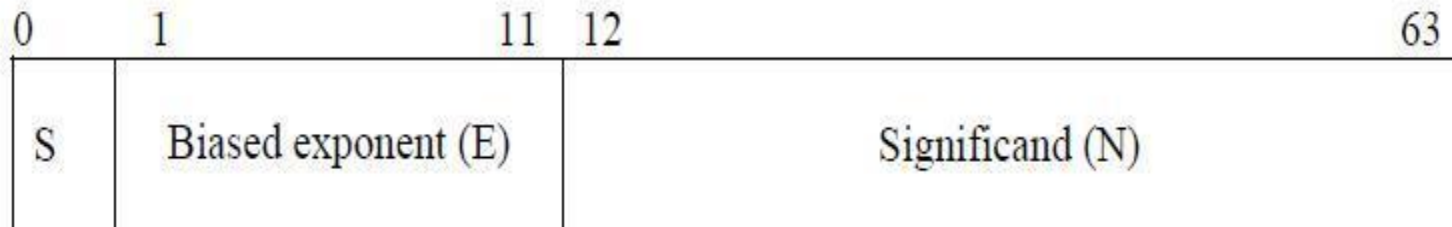
Double Precision = 64 bits

IEEE Standard 754 format

Guard Bits

- We mentioned that, prior to a floating-point operation, the exponent and significand of each operand are loaded into ALU registers.

- In the case of the significand, the length of the register is almost always greater than the length of the significand plus an implied bit.

- The register contains additional bits, called **Guard Bits**, which are **used to pad out the right end of the significand with 0s.**

Rounding

- The result of any operation on the significands is generally stored in a longer register.

- When the result is put back into the floating-point format, the extra bits must be eliminated in such a way as to produce a result that is close to the exact result. This process is called **Rounding.**

- **Round to nearest:** The result is rounded to the nearest representable number.

- **Round toward +infinity:** The result is rounded up toward plus infinity.

- **Round toward –infinity:** The result is rounded down toward negative infinity.

- **Round toward 0:** The result is rounded toward zero.

Blue Screen Error: Exception

A problem has been detected and Windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: kbdhid.sys

MANUALLY_INITIATED_CRASH

If this is the first time you've seen this stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use safe mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical Information:

*** STOP: 0x000000e2 (0x00000000, 0x00000000, 0x00000000, 0x00000000)

*** kbdhid.sys - Address 0x94efd1aa base at 0x94efb000 DateStamp 0x4a5bc705

ASCII to Hex Table

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|		
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL