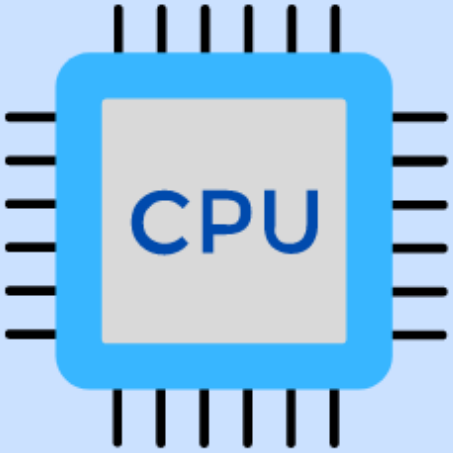
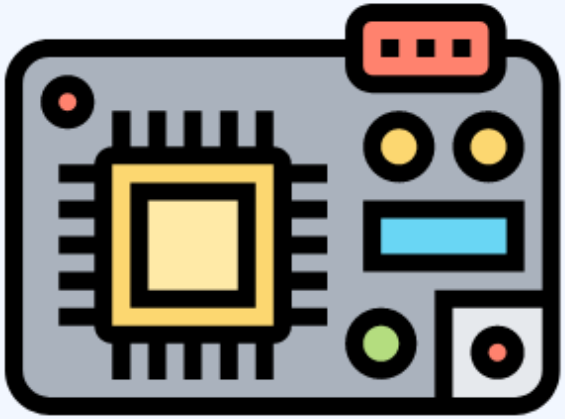
Microprocessor VS Microcontroller





Play with Circuit

Introduction to Microprocessor Technology

1

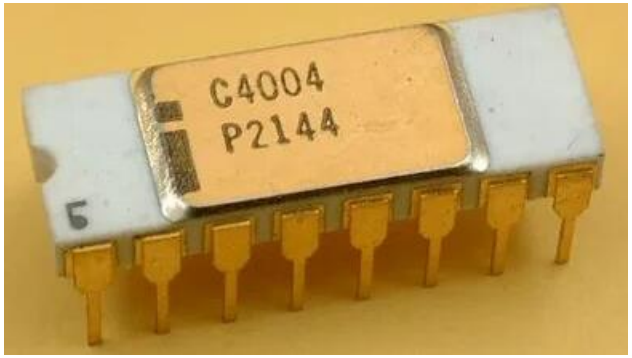
Introduction to Microprocessor Technology

- Microprocessor History and Evolution
- Microprocessor-based Systems
- Inside The Microprocessor
- Organization of a microprocessor-based System
- Memory
- Memory Map and Addresses
- Decimals, Binary, Hex and Binary Coded decimal (BCD)

Microprocessor history and evolution

Microprocessor:

- Also known as **Central Processing Unit (CPU),** is a complete computation engine that is fabricated on a single chip.
- The first microprocessor was the **Intel 4004,** introduced in 1971. The 4004 was not very powerful. All it could do was **add** and **subtract**, and it could only do that with **4 bits** at a time.
- Prior to the 4004, engineers built computers either from **collections of chips** or from **discrete components** (transistors wired one at a time). The 4004 powered one of the first portable electronic calculators.



- Intel 4004 microprocessor

Microprocessor history and evolution

Microprocessor?

- Microprocessor is a controlling unit of a **micro-computer**, fabricated on a **small chip** capable of performing ALU (Arithmetic Logical Unit) operations and communicating with the other devices connected to it.
- The word **Microprocessor** comes from the combination **of micro** and **processor.**
- **Processor:** In this context, processor means a device that performs certain operations on the numbers, specifically binary numbers, 0's and 1's.
- **Micro is a new addition:** In the late 1960's, processors were built using discrete elements but were too large and too slow. In the early 1970's the size became several thousand times smaller, and the speed became several hundred times faster. The "Micro" Processor was born.

Microprocessor history and evolution

What is a Microprocessor?

- The microprocessor is a **programmable device** that **takes in numbers in binary**, **performs on them arithmetic or logical operations** according to the **program stored in memory** and then **produces other numbers as a result.**

Programmable device:

- The microprocessor can perform different sets of operations on the data it receives depending on the sequence of **instructions** supplied in the given program.

Instructions:

Each microprocessor is designed to execute a specific group of operations. This group of operations is called an **Instruction Set**. This instruction set defines what the microprocessor can and cannot do.

Microprocessor history and evolution

Microprocessor Takes in:

The data that the microprocessor manipulates must come from somewhere. It comes from what is called **"input devices".**

- These represent devices such as a keyboard, a mouse, switches, and the like.

Microprocessor processes Numbers: The microprocessor has a very narrow view on life. It only understands binary numbers (0 and 1).

- The microprocessor recognizes and processes a group of **bits together**. This group of bits is called a **"word".**
- The number of bits in a Microprocessor's **word,** is a measure of its "abilities".

```
  10110011
+ 01001101
  --------
1 00000000
```

Microprocessor history and evolution

Bits, Bytes and Words:

- The earliest microprocessor (the Intel 8088, 8085 and Motorola's 6800) recognized **8-bit words** which equals to **1 Byte**.
- To process data larger than 8-bit **they have to break them into 8-bit pieces** and processed each group of 8-bits separately.
- Later microprocessors (8086 and 68000) were designed with **16-bit words**.
- A group of 8-bits were referred to as a **"half-word"** or **"byte".** A group of **4 bits** is called a **"nibble".** Also, 32-bits groups were given the name **"long word".**
- Today, all processors manipulate at least 32 bits at a time and there exists microprocessors that can process 64, 80, 128 bits

Microprocessor history and evolution

Arithmetic and Logic Operations:

Every microprocessor performs arithmetic operations such as **"add"** and "**subtract"** as part of its instruction set.

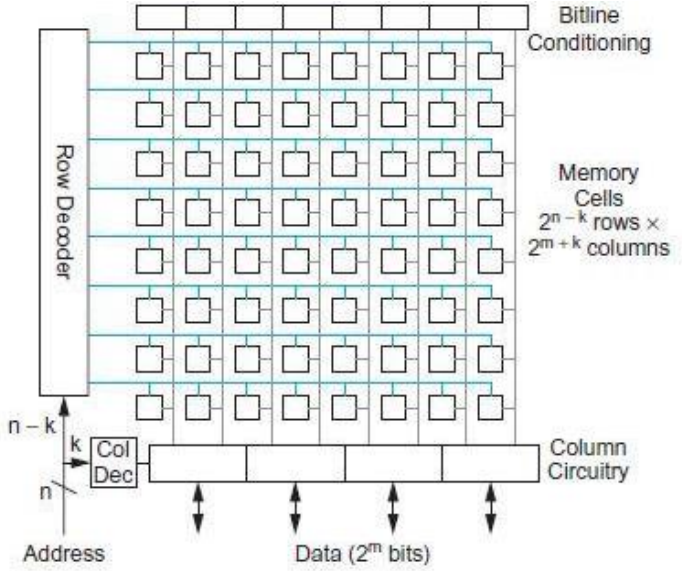
- Most microprocessors will have operations such as **multiply** and **divide**. Some of the newer ones will have complex operations such as **square root**.
- In addition, microprocessors have logic operations as well. Such as **AND**, **OR**, **XOR**, **shift left**, **shift right**, etc.
- Again, the number and types of operations define the microprocessor's instruction set and depends on the specific microprocessor.

Microprocessor history and evolution

Read/Write from/to Memory :

what is memory? Memory is the location where information is kept while not in current use.

- Memory are built using different technologies as digital storage devices.
- Also, in most kinds of memory, these storage devices are grouped into groups of 8. These **8 storage locations** can only be accessed together. So, one can only read or write in terms of bytes to and from memory.



Microprocessor history and evolution

- Memory is usually measured by the number of bytes (group of 8 bits) it can hold. It is measured in Kilobytes (KB), Megabytes (MB) Giga (GB), Tera (TB), etc.

- A Kilobyte (KB) in computer language is $\mathbf{2^{10} = 1024\ bytes}$. A Megabyte (MB) is **1024** KB ($\mathbf{2^{20}\ bytes}$), A Gigabyte is 1024 MB ($\mathbf{2^{30}\ bytes}$) and so on .

Microprocessor produces/outputs :

- For the user to see the result of the execution of the program, the results must be presented in a human readable form.
- The results must be presented on an output device. This can be the monitor, a paper from the printer, a simple LED or many other forms.

Microprocessor-based System

Microcomputer

- Microcomputer is a computer with a **microprocessor** as its CPU and other key components such as **System memory**, **I/O** etc. with system busses providing data communications amongst all the components.

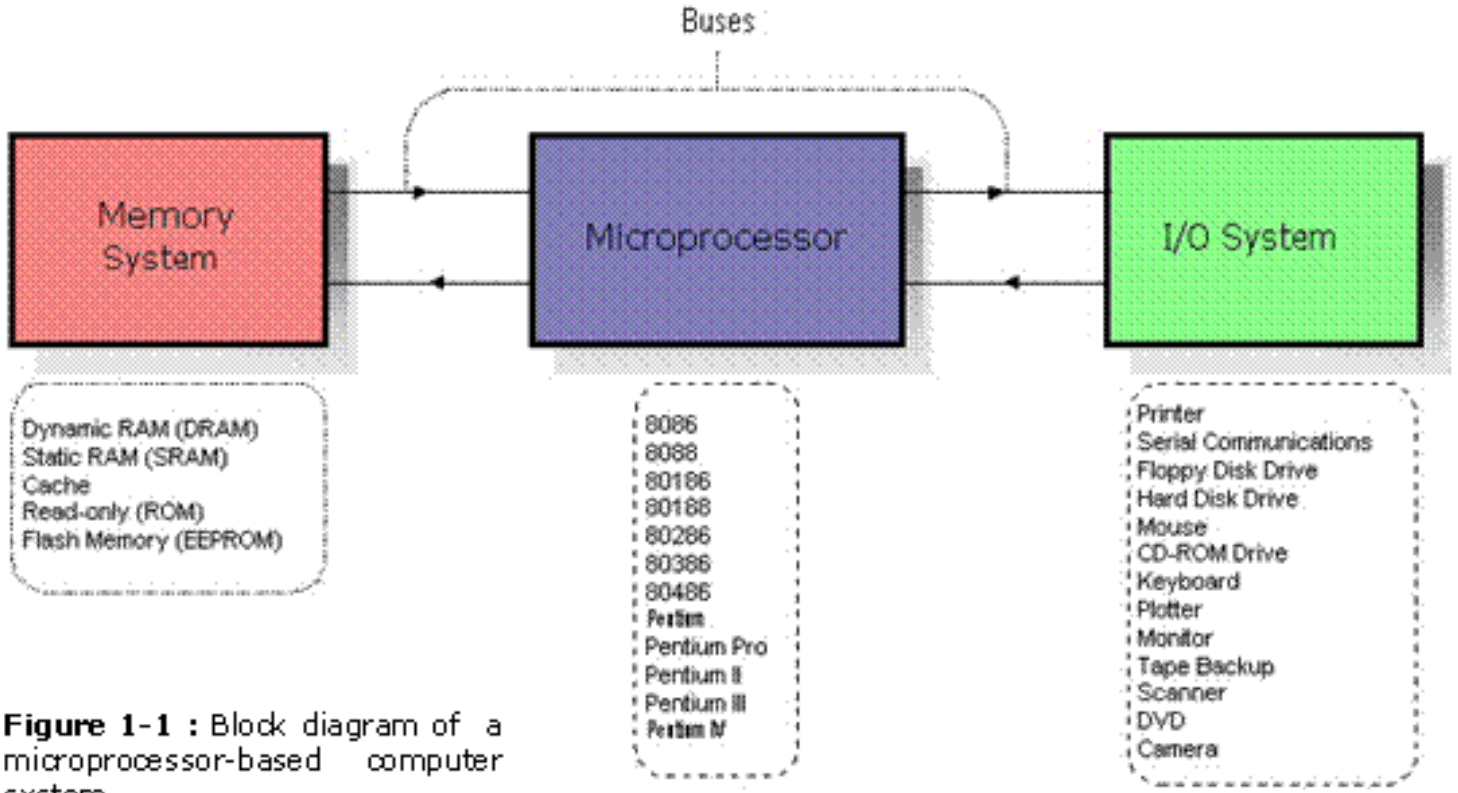


Figure 1-1 : Block diagram of a microprocessor-based computer system

Inside The Microprocessor

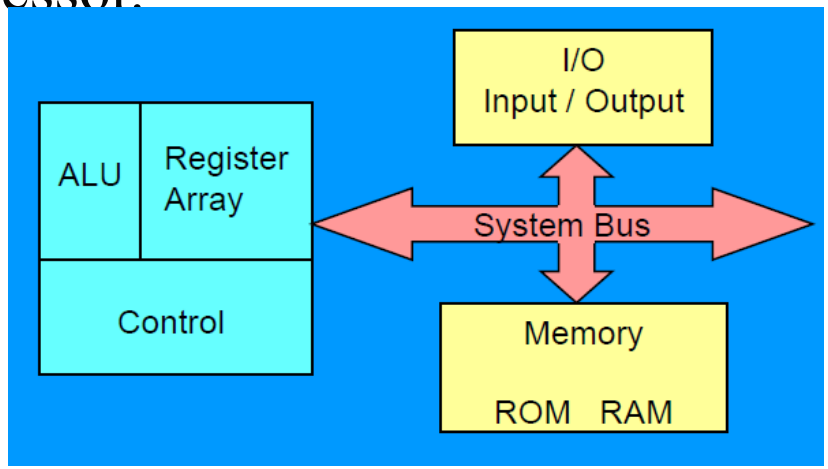
Microprocessor key components

- Microprocessor consists 3 key components of **Arithmetic and Logical Unit (ALU) , Register Array**, and a **Control Unit**.

Arithmetic and Logical Unit (ALU): Performs arithmetical and logical operations on the data received from the memory or an input device.

Register Array: consists of temporary memory locations registers identified by letters like **B, C, D, E, H, L** and **Accumulator**.

Control Unit: controls the **flow of data** and **instructions** within the computer's microprocessor.



Inside The Microprocessor

Memory

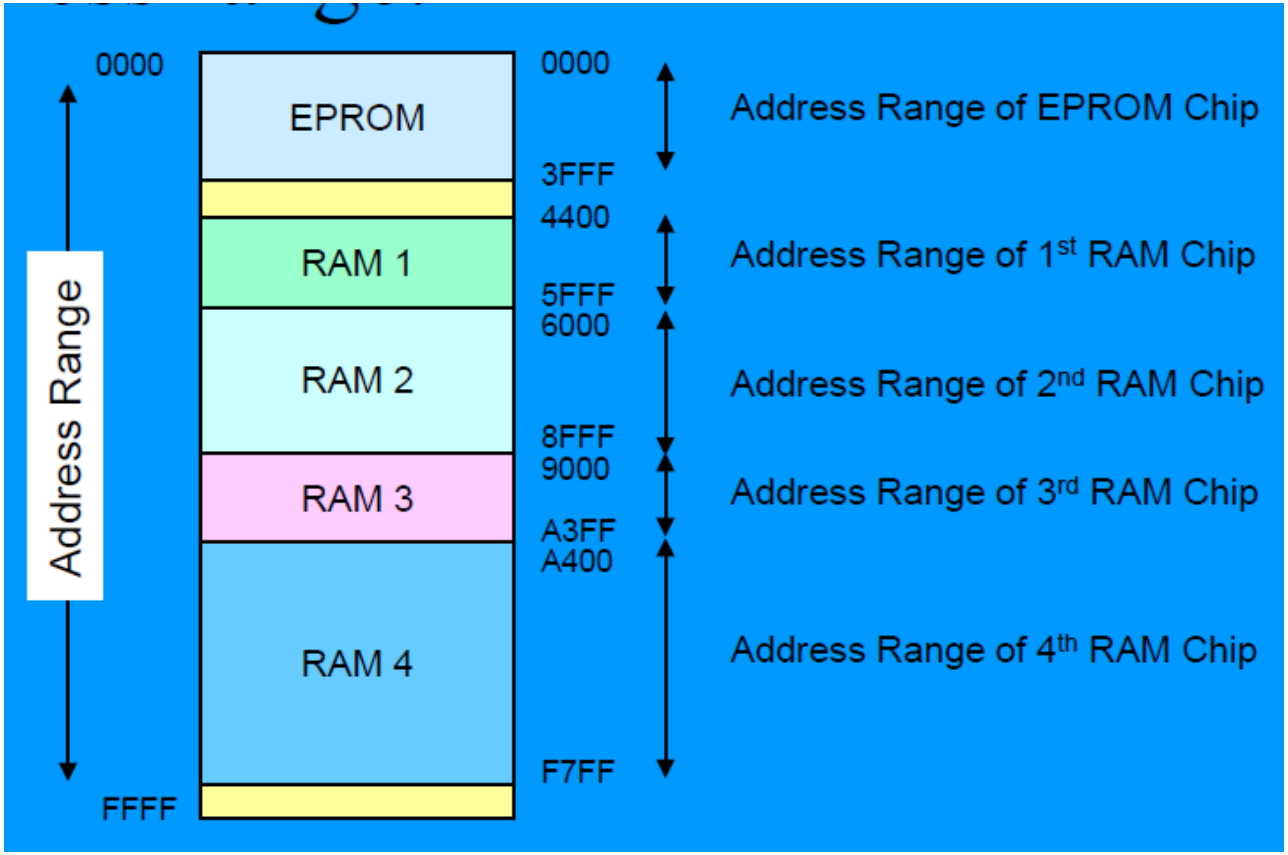
- Memory stores information such as instructions and data in binary format (0 and 1). It provides this information to the microprocessor whenever it is needed.
- Usually, there is a memory **"sub-system"** in a microprocessor-based system. This sub-system includes:

1. The **registers** inside the microprocessor
2. **Read Only Memory (ROM):** used to store information that does not change.
3. **Random Access Memory (RAM):**used to store information supplied by the user. Such as programs and data.

Inside The Microprocessor

Memory Map and Addresses

- The memory map is a picture representation of the address range and shows where the different memory chips are located within the address range.



Inside The Microprocessor

Program Execution

- To execute a program the user enters instructions in binary format into the memory.
- The microprocessor then reads these instructions and whatever data is needed from memory, executes the instructions and places the results either in memory or produces it on an output device.

The three cycle instruction execution model:

- To execute a program, the microprocessor **“reads”** each instruction from memory, **“interprets”** it, then **“executes”** it.
- In a more technical way, microprocessor **fetches** each instruction, **decodes it**, then **executes** it.
- This sequence is continued until all instructions are performed.

Inside The Microprocessor

Machine Language

- To decodes the type of operation required by microprocessor, special numbers (binary bits stream) are needed to form the part of the microprocessor's vocabulary (language base).
- Therefore, for 8-bit microprocess the maximum number word combinations or instructional opcode that can be defined is $2^8 = 256$
- Number of bits that form the "word" of a microprocessor is fixed for that particular processor.
- However, in most microprocessors, not all of these combinations are used. Certain patterns are chosen and assigned specific meanings.
- Each of these patterns forms an instruction for the microprocessor.
- The complete set of patterns makes up the **microprocessor's machine language.**

Inside The Microprocessor

The Intel 8085 Machine Language

- The 8085 (from Intel) is an 8-bit microprocessor.
- The 8085 uses a total of **246-bit** patterns to form its instruction set.
- These **246** patterns represent only **74** instructions.
- The reason for the difference is that some (actually most) instructions have multiple different formats.
- Because it is very difficult to enter the bit patterns correctly, they are usually entered in **hexadecimal** instead of **binary**.
- For example, the combination **00111100** which translates into "increment the number in the register called the **accumulator**", is usually entered as **3C**.

Inside The Microprocessor

Assembly Language

- Entering the instructions using **hexadecimal** is quite easier than entering the **binary** combinations.
- However, it still is difficult to understand what a program written in hexadecimal does.
- So, each microprocessor company defines a **symbolic code** for the instructions. These codes are called **"mnemonics".**
- The **mnemonic** for each instruction is usually a group of letters that suggest the operation performed.

Inside The Microprocessor

Assembly Language

- Using the same example from before, **00111100** translates to **3C** in hexadecimal (OPCODE)
- Its mnemonic is: "**INR A**".
- INR stands for "**increment register**" and **A** is short for accumulator.
- Another example is: **10000000**, which translates to **80** in hexadecimal.
- Its mnemonic is "ADD B".
- "Add register B to the accumulator and keep the result in the accumulator".

Inside The Microprocessor

Assembly Language

- It is important to remember that a **machine language** and its associated **assembly language** are completely machine dependent.
- In other words, they are not transferable from one microprocessor to a different one.
- For example, Motorola has an 8-bit microprocessor called the 6800.
- The 8085-machine language is very different from that of the 6800. So is the assembly language.
- A program written for the 8085 cannot be executed on the 6800 and vice versa.

Inside The Microprocessor

"Assembling" The Program

- How does assembly language get translated into machine language?
- There are two ways:
- 1st there is **"hand assembly".**
- The programmer translates each assembly language instruction into its equivalent hexadecimal code (machine language). Then the hexadecimal code is entered into memory.
- The other possibility is a program called an **"assembler",** which does the translation automatically.

Introduction to the Hexadecimal System

What is Hexadecimal?

- Is a base-16 number system, meaning it uses 16 distinct symbols: 0–9 and A–F (where A=10, B=11, ..., F=15).
- Example: $1FH$ or $1F_{16}$ represents hex number that can be converted to base 10 (decimal) as follows.

$$\mathbf{1F_{16} = 1 \times 16^1 + 15 \times 16^0 = 31_{10}}$$

Why Use Hexadecimal

- Is a compact Representation: Hex simplifies long binary strings. For example, 10101111_2 becomes AF_{16} or AFH
- Alignment with Binary: Each hex digit corresponds to 4 binary digits (nibble), making conversions straightforward.

Introduction to the Hexadecimal System

Hexadecimal conversion Table:

Hexadecimal to Decimal Table

MATH MONKS

Hexadecimal (Base 16)	Decimal (Base 10)	Binary (Base 2)
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Introduction to the Hexadecimal System

Converting Between Hexadecimal and Decimal

Hexadecimal to Decimal:

Multiply each digit by $16^{position}$ (right to left, starting at 0). Example: convert

Convert $2AF_{16}$ to decimal

$$2AF_{16} = 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 10815_{10}$$

Decimal to Hexadecimal:

- Divide the decimal number by 16 repeatedly; record remainders (convert 10–15 to A–F).
- Example: Convert 255_{10} to hex.

$$255_{10} = 255 \div 16 = 15\ (reminder\ 15) \rightarrow FF_{16}$$

Introduction to the Hexadecimal System

Hexadecimal to Binary:

- Each hex digit corresponds to **4 binary digits**. Example: Convert $3F_{16}$ to binary.

$$3_{16} = 0011_2,\ F_{16} = 1111_2 \rightarrow 0011\ 1111_2$$

Binary to Hexadecimal:

- Group binary digits into sets of 4 (starting from the right) and convert each group to hex.
- Example: Convert $11010110F_{16}$ to hex.

$$1101\ 0110_2 \rightarrow D6_{16}$$

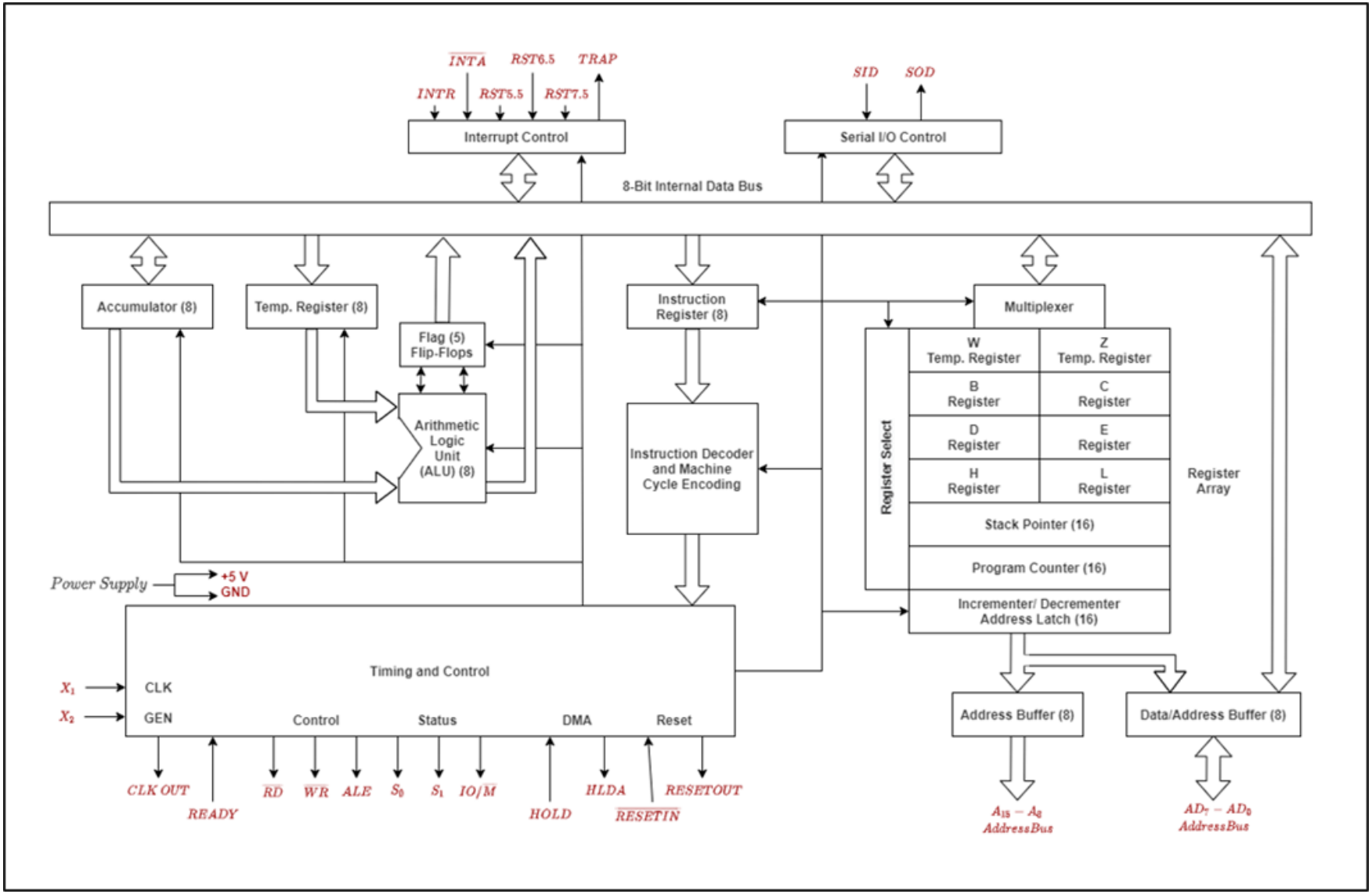
Introduction to the Hexadecimal System

Practical Uses of Hexadecimal:

- Memory addresses are often represented in hex for compactness.
- Hex is used in web design to represent RGB colors. Example: #FF5733 = Red=FF, Green=57, Blue=33.
- Hex dumps display raw binary data in a human-readable format. Example: Debugging memory or file contents.
- Hex simplifies the representation of binary-coded data in microprocessors and communication protocols.
- Compactness: Reduces long binary strings (e.g., 8 binary digits → 2 hex digits).
- Readability: Easier to interpret than binary and Fits neatly with bytes (2 hex digits = 1 byte).

Inside The Microprocessor

Intel 8085 architecture



8085 Pins Description

- Instruction with a Memory Address