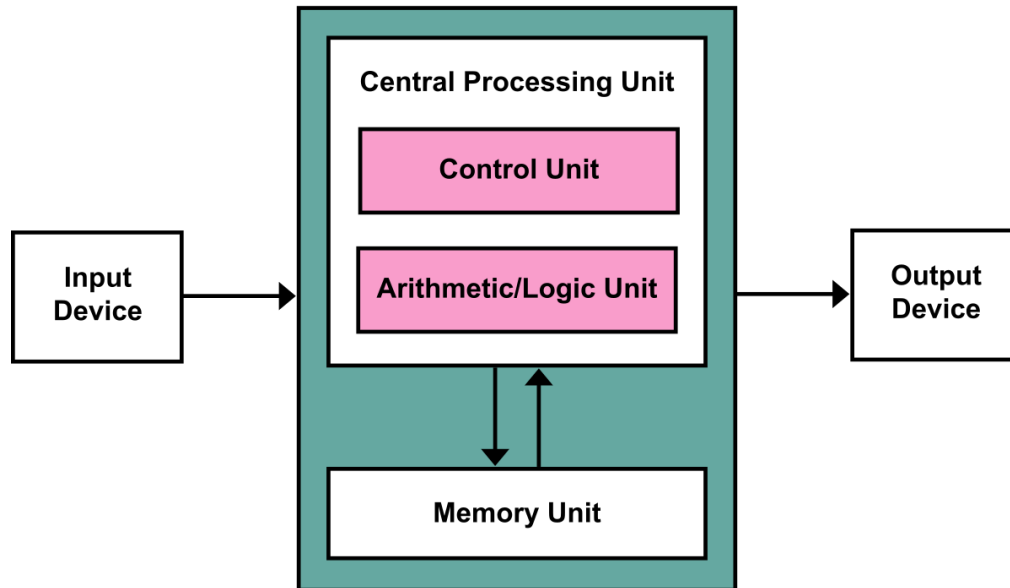


## Functional units of computer:



A computer is not a single unit but it consists of many functional units(intended to perform jobs) such as Input unit, Central Processing Unit(ALU and Control Unit), Storage (Memory) Unit and Output Unit.

1. Input Unit: Its aim is to supply data (Alphanumeric, image , audio, video, etc.) to the computer for processing. The Input devices are keyboard, mouse, scanner, mic, camera, etc

2. Central Processing Unit (CPU): It is the brain of the computer and consists of three components

Arithmetic Logic Unit(ALU): As the name implies it performs all calculations and comparison operations.

Control Unit(CU): It controls overall functions of a computer

Registers: It stores the intermediate results temporarily.

3. Storage Unit(Memory Unit): A computer has huge storage capacity. It is used to store data and instructions before starts the processing. Secondly it stores the intermediate results and thirdly it stores information(processed data), that is the final results before send to the output unit(Visual Display Unit, Printer, etc)

Two Types of storage unit

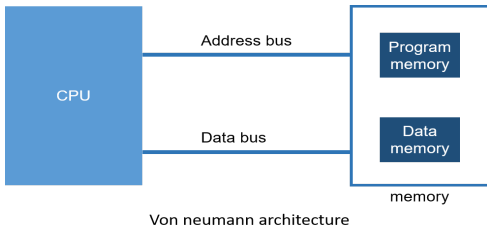
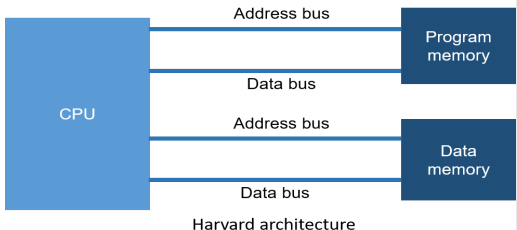
(a) Primary Storage alias Main Memory: It is further be classified into Two- Random Access Memory(RAM) and Read Only Memory(ROM). The one and only memory that the CPU can directly access is the main memory at a very high speed. It is expensive hence storage capacity is less. RAM is volatile (when the power is switched off the content will be erased) in nature but ROM is non volatile(It is permanent)

(b) Secondary Storage alias Auxiliary Memory: Because of limited storage capacity of primary memory its need arises. When a user saves a file, it will be stored in this memory hence it is permanent in nature and its capacity is huge. eg: Hard Disc Drive(HDD), Compact Disc(CD), DVD, Pen Drive, Blu Ray Disc etc.

4. Output Unit: After processing the data we will get information as result, that will be given to the end user through the output unit in a human readable form. Normally monitor and printer are used.

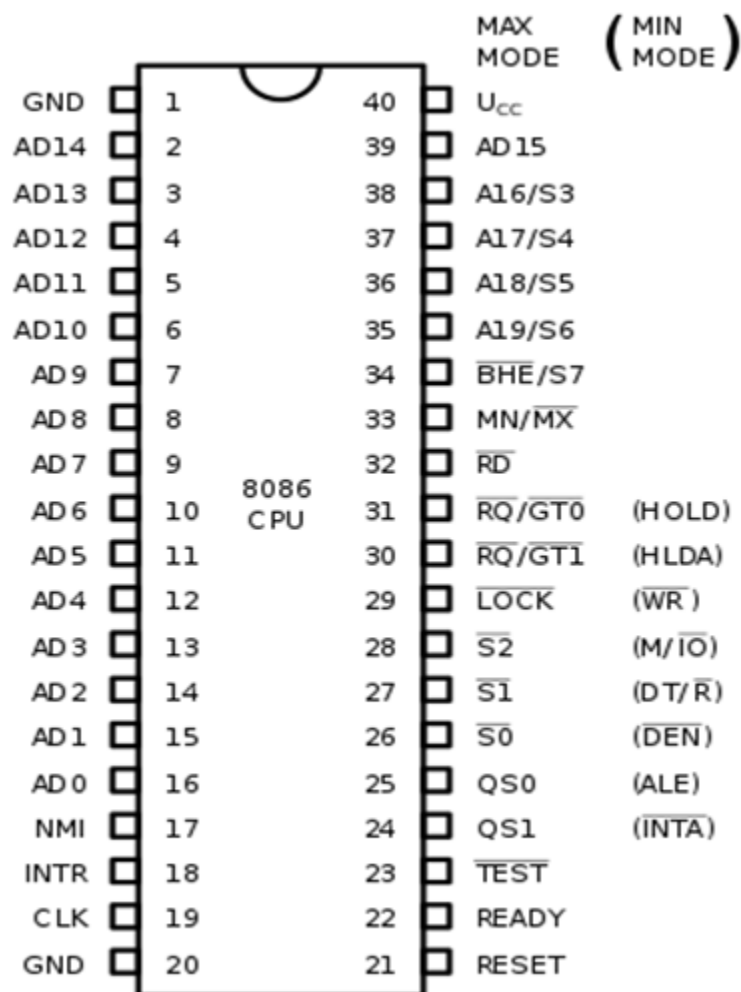
### **Difference between von neumann and harvard architecture**

Key important difference between von neumann and harvard architecture are given below

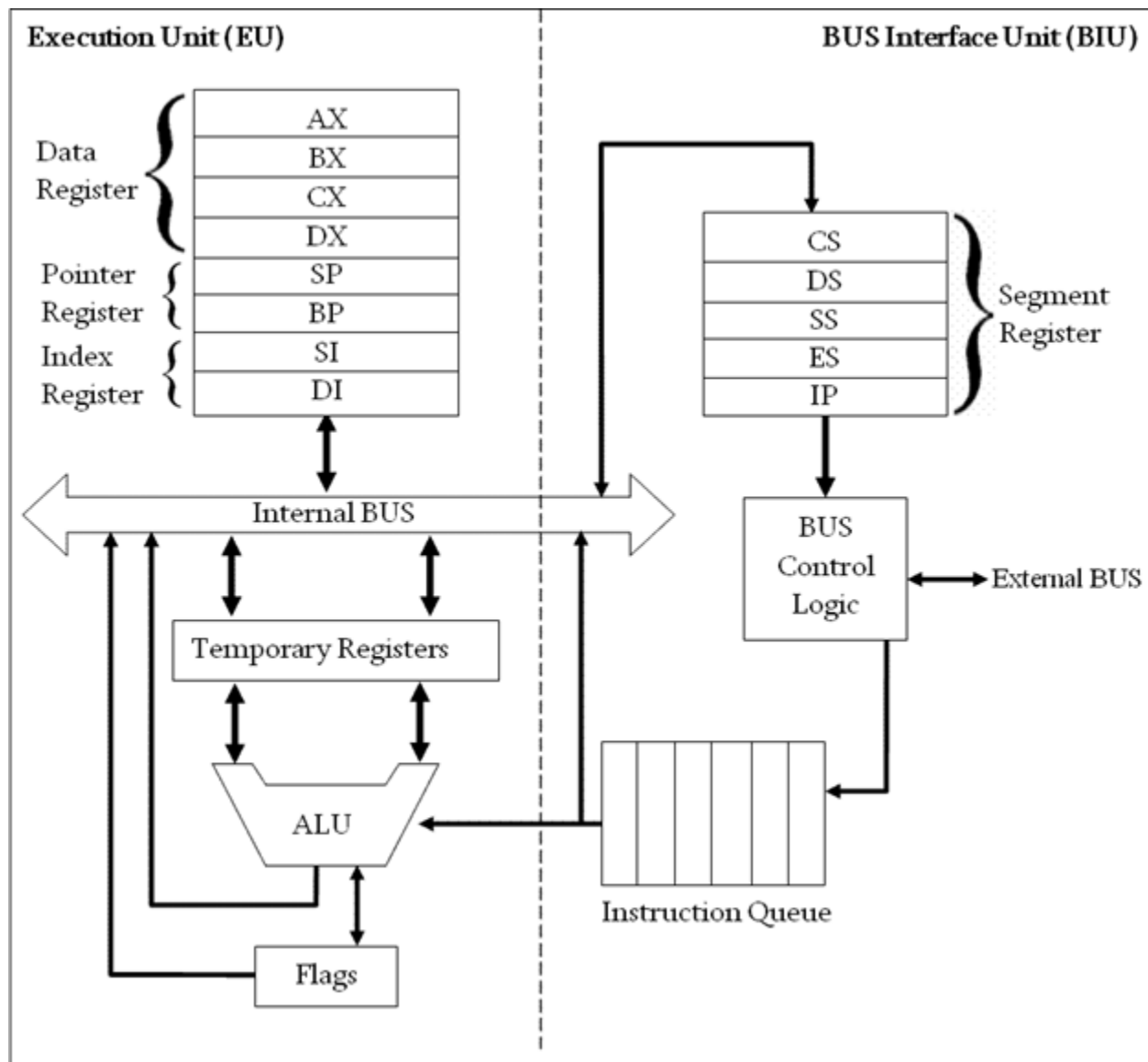
<b>Parameter</b>	<b>Von neumann architecture</b>	<b>Harvard architecture</b>
1)Defination	The architecture which uses common bus to access data memory and program memory is called as von neumann architecture	The architecture which uses separate address bus and data bus to access data memory and program memory respectively is called as harvard architecture
2)Block diagram	 <p>The diagram shows a CPU on the left connected to a single memory block on the right. The memory block contains 'Program memory' and 'Data memory'. A single 'Address bus' line connects the CPU to the memory block, and a single 'Data bus' line connects the CPU to the memory block. The entire memory block is labeled 'memory' at the bottom right.</p> <p>Von neumann architecture</p>	 <p>The diagram shows a CPU on the left connected to two separate memory blocks on the right: 'Program memory' and 'Data memory'. There are four distinct bus lines: an 'Address bus' to Program memory, a 'Data bus' to Program memory, an 'Address bus' to Data memory, and a 'Data bus' to Data memory.</p> <p>Harvard architecture</p>
3)Storage	The von neumann architecture uses single memory space for both program memory and data memory	The harvard architecture uses seperate program and data memory space
4)Access	In von neumann architecture we can not access program and data memory simultaneously	In harvard architecture we can access program and data memory simultaneously because it has separate program memory and data memory

5)Architecture	The von neumann architecture uses CISC architecture	The harvard architecture uses RISC architecture
6)Execution of instruction	Execution of instruction takes more machine cycle	Execution of instruction takes less machine cycle

### Introduction to microprocessor:



## Functional units of 8086



### What is an Assembler?

We know that assembly language is a less complex and programmer-friendly language used to program the processors. In [assembly language programming](#), the instructions are specified in the form of mnemonics rather than in the form of machine code i.e., 0 and 1. But the microprocessor or microcontrollers are specifically designed in a way that they can only understand machine language.

Thus assembler is used to convert assembly language into machine code so that it can be understood and executed by the processor. Therefore, to control the generation of

machine codes from the assembly language, assembler directives are used. However, machine codes are only generated for the program that must be provided to the processor and not for assembler directives because they do not belong to the actual program.

## **8086- Assembler directives**

### **Introduction:**

Assembler directives are the directions to the assembler which indicate how an operand or section of the program is to be processed. These are also called pseudo operations which are not executable by the microprocessor. The following section explains the basic assembler directives for 8086.

### **ASSEMBLER DIRECTIVES:**

The various directives are explained below.

1. **ASSUME** : The **ASSUME** directive is used to inform the assembler the name of the logical segment it should use for a specified segment.

Ex: **ASSUME DS: DATA** tells the assembler that for any program instruction which refers to the data segment, it should use the logical segment called **DATA**.

2. **DB** -Define byte. It is used to declare a byte variable or set aside one or more storage locations of type byte in memory.

For example, **CURRENT\_VALUE DB 36H** tells the assembler to reserve 1 byte of memory for a variable named **CURRENT\_VALUE** and to put the value 36 H in that memory location when the program is loaded into RAM .

3. **DW** -Define word. It tells the assembler to define a variable of type word or to reserve storage locations of type word in memory.

4. **DD**(define double word) :This directive is used to declare a variable of type double word or reserve memory locations which can be accessed as type double word.

5.DQ (define quadword) :This directive is used to tell the assembler to declare a variable 4 words in length or to reserve 4 words of storage in memory .

6.DT (define ten bytes):It is used to inform the assembler to define a variable which is 10 bytes in length or to reserve 10 bytes of storage in memory.

7. EQU –Equate It is used to give a name to some value or symbol. Every time the assembler finds the given name in the program, it will replace the name with the value or symbol we have equated with that name

8.ORG -Originate : The ORG statement changes the starting offset address of the data.

It allows to set the location counter to a desired value at any point in the program.For example the statement ORG 3000H tells the assembler to set the location counter to 3000H.

9 .PROC- Procedure: It is used to identify the start of a procedure. Or subroutine.

10. END- End program .This directive indicates the assembler that this is the end of the program module.The assembler ignores any statements after an END directive.

11. ENDP- End procedure: It indicates the end of the procedure (subroutine) to the assembler.

12.ENDS-End Segment: This directive is used with the name of the segment to indicate the end of that logical segment.

Ex: CODE SEGMENT : Start of logical segment containing code

CODE ENDS : End of the segment named CODE.