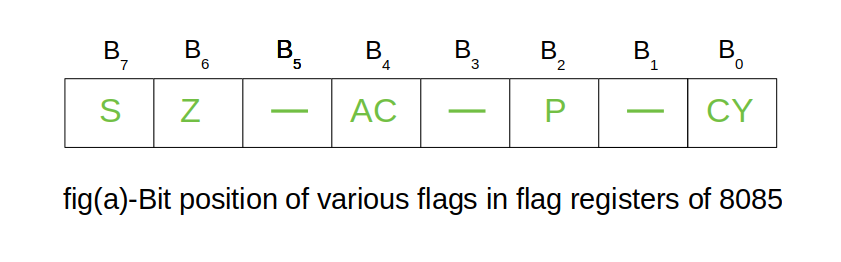
**Programming model of 8085**

**Registers in 8085:**

1. **General Purpose Registers –** The 8085 has six general-purpose registers to store 8-bit data; these are identified as- B, C, D, E, H, and L. These can be combined as register pairs – BC, DE, and HL, to perform some 16-bit operation. These registers are used to store or copy temporary data, by using instructions, during the execution of the program.
2. **Specific Purpose Registers –**

* **Accumulator:** The accumulator is an 8-bit register (can store 8-bit data) that is the part of the arithmetic and logical unit (ALU). After performing arithmetical or logical operations, the result is stored in accumulator. Accumulator is also defined as register A.
* **Flag registers:**The flag register is a special purpose register and it is completely different from other registers in microprocessor. It consists of 8 bits and only 5 of them are useful. The other three are left vacant and are used in the future Intel versions.These 5 flags are set or reset (when value of flag is 1, then it is said to be set and when value is 0, then it is said to be reset) after an operation according to data condition of the result in the accumulator and other registers. The 5 flag registers are:
  1. **Sign Flag:** It occupies the seventh bit of the flag register, which is also known as the most significant bit. It helps the programmer to know whether the number stored in the accumulator is positive or negative. If the sign flag is set, it means that number stored in the accumulator is negative, and if reset, then the number is positive.
  2. **Zero Flag:**: It occupies the sixth bit of the flag register. It is set, when the operation performed in the ALU results in zero(all 8 bits are zero), otherwise it is reset. It helps in determining if two numbers are equal or not.
  3. **Auxiliary Carry Flag:** It occupies the fourth bit of the flag register. In an arithmetic operation, when a carry flag is generated by the third bit and passed on to the fourth bit, then Auxiliary Carry flag is set. If not flag is reset. This flag is used internally for BCD(Binary-Coded decimal Number) operations. **Note –** This is the only flag register in 8085 which is not accessible by user.
  4. **Parity Flag:** It occupies the second bit of the flag register. This flag tests for number of 1’s in the accumulator. If the accumulator holds even number of 1’s, then this flag is set and it is said to even parity. On the other hand if the number of 1’s is odd, then it is reset and it is said to be odd parity.
  5. **Carry Flag:** It occupies the zeroth bit of the flag register. If the arithmetic operation results in a carry(if result is more than 8 bit), then Carry Flag is set; otherwise it is reset.

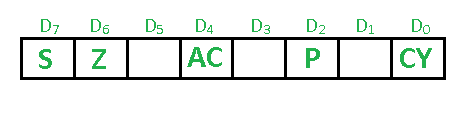
**(c) Memory Registers –** There are two 16-bit registers used to hold memory addresses. The size of these registers is 16 bits because the memory addresses are 16 bits. They are :-

* **Program Counter:** This register is used to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.
* **Stack Pointer:** It is used as a memory pointer. It points to a memory location in read/write memory, called the stack. It is always incremented/decremented by 2 during push and pop operation.

**Flag register in 8085 microprocessor**

The **Flag register** is a Special Purpose Register. Depending upon the value of the result after any arithmetic and logical operation, the flag bits become set (1) or reset (0). In 8085 microprocessor, the flag register consists of 8 bits and only 5 of them are useful. The 5 flags are:

* The **Flag register** is a Special Purpose Register. Depending upon the value of the result after any arithmetic and logical operation, the flag bits become set (1) or reset (0). In 8085 microprocessor, the flag register consists of 8 bits and only 5 of them are useful.



1. **Sign Flag (S) –** After any operation if the MSB (B(7)) of the result is 1, it indicates the number is negative and the sign flag becomes set, i.e. 1. If the MSB is 0, it indicates the number is positive and the sign flag becomes reset i.e. 0. from 00H to 7F, sign flag is 0 from 80H to FF, sign flag is 1 1- MSB is 1 (negative) 0- MSB is 0 (positive)   
   **Example:** MVI A 30 (load 30H in register A) MVI B 40 (load 40H in register B) SUB B (A = A – B) These set of instructions will set the sign flag to 1 as 30 – 40 is a negative number. MVI A 40 (load 40H in register A) MVI B 30 (load 30H in register B) SUB B (A = A – B) These set of instructions will reset the sign flag to 0 as 40 – 30 is a positive number.
2. **Zero Flag (Z) –** After any arithmetical or logical operation if the result is 0 (00)H, the zero flag becomes set i.e. 1, otherwise it becomes reset i.e. 0. 00H zero flags is 1. from 01H to FFH zero flag is 0 1- zero-result 0- non-zero result **Example:** MVI A 10 (load 10H in register A) SUB A (A = A – A) These set of instructions will set the zero flag to 1 as 10H – 10H is 00H
3. **Auxiliary Carry Flag (AC) –** This flag is used in the BCD number system(0-9). If after any arithmetic or logical operation D(3) generates any carry and passes it on to D(4) this flag becomes set i.e. 1, otherwise, it becomes reset i.e. 0. This is the only flag register that is not accessible by the programmer 1-carry out from bit 3 on addition or borrows into bit 3 on subtraction 0-otherwise   
   **Example:** MVI A 2BH (load 2BH in register A) MVI 39H (load 39H in register B) ADD B (A = A + B) These set of instructions will set the auxiliary carry flag to 1, as on adding 2B and 39, the addition of lower-order nibbles B and 9 will generate a carry.
4. **Parity Flag (P) –** If after any arithmetic or logical operation the result has even parity, an even number of 1 bit, the parity register becomes set i.e. 1, otherwise it becomes reset i.e. 0. 1-accumulator has an even number of 1 bits 0-accumulator has odd parity   
   **Example:** MVI A 05 (load 05H in register A) This instruction will set the parity flag to 1 as the BCD code of 05H is 00000101, which contains an even number of ones i.e. 2.
5. **Carry Flag (CY) –** Carry is generated when performing n bit operations and the result is more than n bits, then this flag becomes set i.e. 1, otherwise, it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset, and if (A<B) it becomes set. Carry flag is also called the borrow flag. 1-carry out from MSB bit on addition or borrow into MSB bit on subtraction 0-no carry out or borrow into MSB bit.   
   **Example:** MVI A 30 (load 30H in register A) MVI B 40 (load 40H in register B) SUB B (A = A – B) These set of instructions will set the carry flag to 1 as 30 – 40 generates a carry/borrow. MVI A 40 (load 40H in register A) MVI B 30 (load 30H in register B) SUB B (A = A – B) These set of instructions will reset the carry flag to 0 as 40 – 30 does not generate any carry/borrow.

# Addressing modes in 8085 microprocessor

The 8085 microprocessor has several **addressing modes that are used to access memory locations.**

The way of specifying data to be operated by an instruction is called addressing mode.

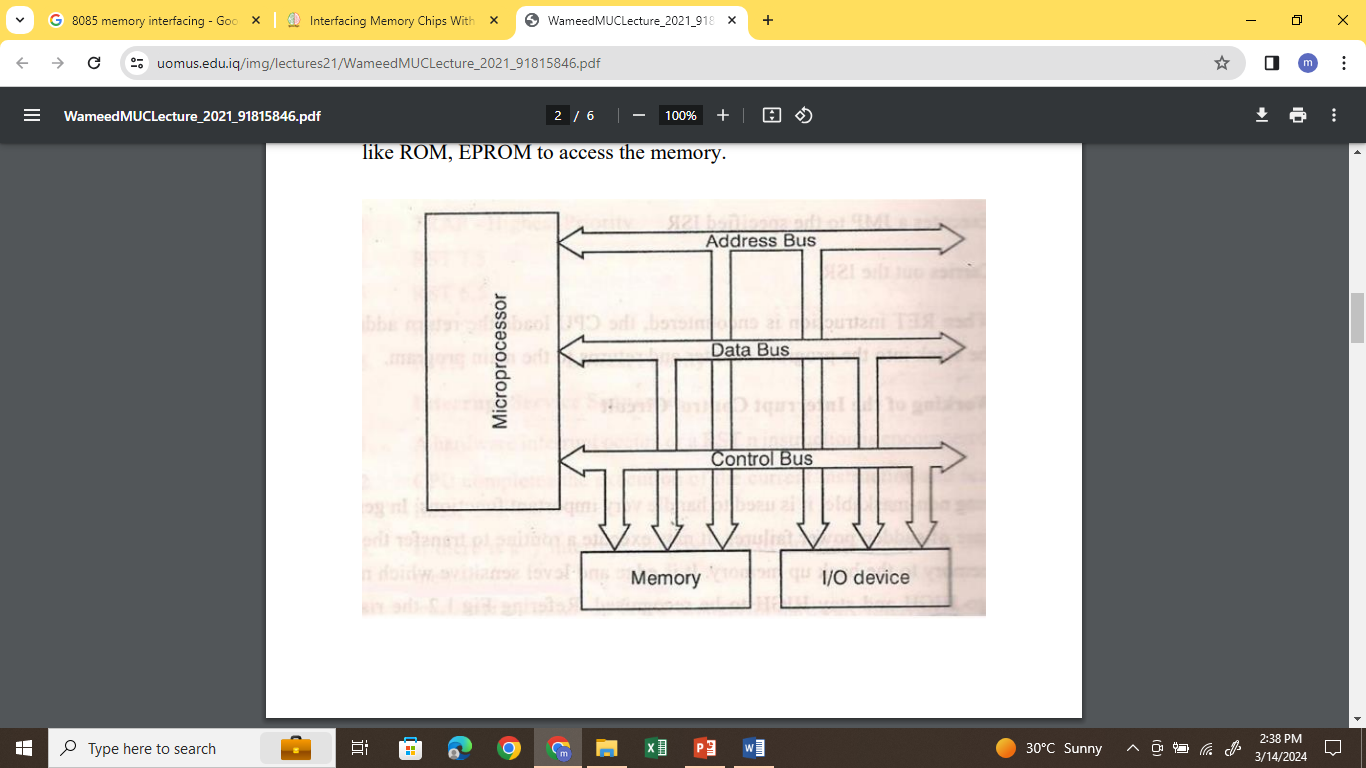
Some of the most commonly used addressing modes in the 8085 microprocessor are:

**Types of addressing modes –**   
In 8085 microprocessor there are 5 types of addressing modes: 

1. **Immediate Addressing Mode –**   
   In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.   
     
   **Examples:**   
   MVI B 45 (move the data 45H immediately to register B)   
   LXI H 3050 (load the H-L pair with the operand 3050H immediately)   
   JMP address (jump to the operand address immediately)
2. **Register Addressing Mode –**   
   In register addressing mode, the data to be operated is available inside the register(s) and register(s) is(are) operands. Therefore the operation is performed within various registers of the microprocessor.   
     
   **Examples:**   
   MOV A, B (move the contents of register B to register A)   
   ADD B (add contents of registers A and B and store the result in register A)   
   INR A (increment the contents of register A by one)
3. **Direct Addressing Mode –**   
   In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself.   
     
   **Examples:**   
   LDA 2050 (load the contents of memory location into accumulator A)   
   LHLD address (load contents of 16-bit memory location into H-L register pair)   
   IN 35 (read the data from port whose address is 35)
4. **Register Indirect Addressing Mode –**   
   In register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a register pair.   
     
   **Examples:**   
   MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)   
   LDAX B (move contents of B-C register to the accumulator)   
   STAX B (store accumulator contents in memory pointed by register pair B-C)
5. **Implied/Implicit Addressing Mode –**   
   In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.   
     
   **Examples:**   
   CMA (finds and stores the 1’s complement of the contents of accumulator A in A)   
   RRC (rotate accumulator A right by one bit)   
   RLC (rotate accumulator A left by one bit)

**Interfacing the 8085 microprocessor:**

A microprocessor is the CPU of a computer. A microprocessor can perform some operation on a data and give the output. But to perform the operation we need an input to enter the data and an output to display the results of the operation. So we are using a keyboard and monitor as Input and output along with the processor. Microprocessors engineering involves a lot of other concepts and we also interface memory elements like ROM, EPROM to access the memory.



**Interfacing Types:**

There are two types of interfacing in context of the 8085 Microprocessor.

Memory Interfacing & I/O Interfacing.

**Memory Interfacing:** Memory is an integral part of a microprocessor system, and in this section, we will discuss how to interface a memory device with the microprocessor. The Memory Interfacing in 8085 is used to access memory quite frequently to read instruction codes and data stored in memory. This read/write operations are monitored by control signals. The microprocessor activates these signals when it wants to read from and write into memory.

Basic Concepts in Memory Interfacing: For Memory Interfacing in 8085, following important points are to be kept in mind.

1. Microprocessor 8085 can access 64Kbytes memory since address bus is 16-bit. But it is not always necessary to use full 64Kbytes address space. The total memory size depends upon the application.

2. Generally EPROM (or EPROMs) is used as a program memory and RAM (or RAMs) as a data memory. When both, EPROM and RAM are used, the total address space 64Kbytes is shared by them.

3. The capacity of program memory and data memory depends on the application.

4. It is not always necessary to select 1 EPROM and 1 RAM. We can have multiple EPROMs and multiple RAMs as per the requirement of application.

5. We can place EPROM/RAM anywhere in full 64 Kbytes address space. But program memory (EPROM) should be located from address 0000H since reset address of 8085 microprocessor is 0000H.

6. It is not always necessary to locate EPROM and RAM in consecutive memory For example: If the mapping of EPROM is from 0000H to OFFFH, it is not must to locate RAM from 1000H. We can locate itanywhere between 1000H and FFFFH. Where to locate memory component totally depends on the application.

# I/O Interfacing

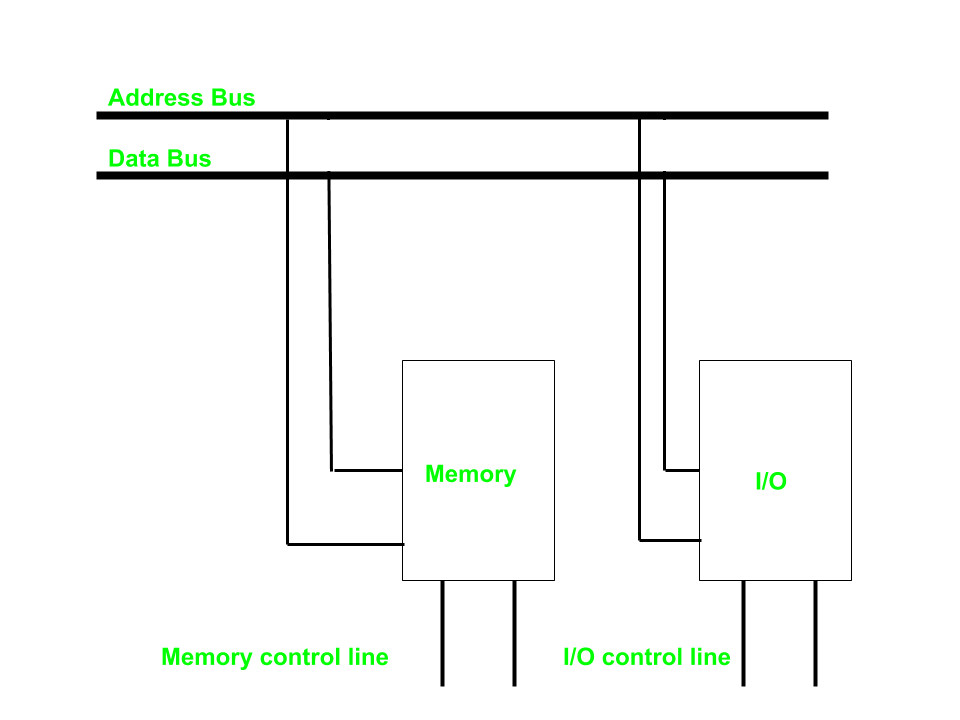
# Memory mapped I/O and Isolated I/O

As a CPU needs to communicate with the various memory and input-output devices (I/O) as we know data between the processor and these devices flow with the help of the system bus. There are three ways in which system bus can be allotted to them :

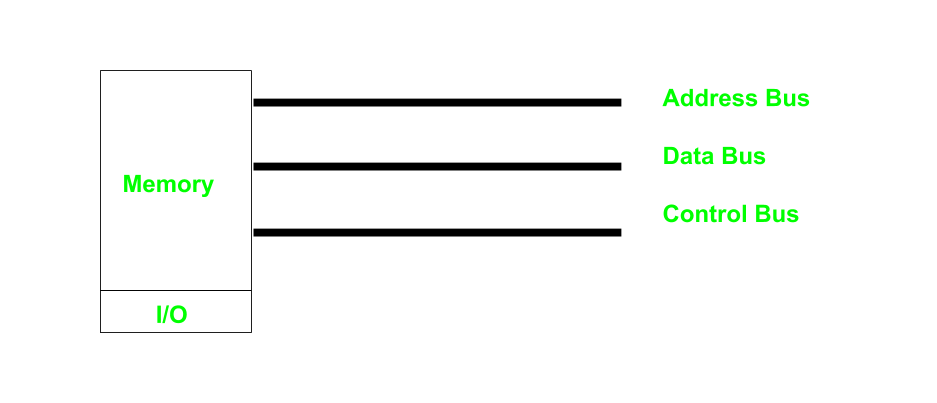
1. Separate set of address, control and data bus to I/O and memory.
2. Have common bus (data and address) for I/O and memory but separate control lines.
3. Have common bus (data, address, and control) for I/O and memory.

In first case it is simple because both have different set of address space and instruction but require more buses.

#### **Isolated I/O –**

Then we have Isolated I/O in which we Have common bus(data and address) for I/O and memory but separate read and write control lines for I/O. So when CPU decode instruction then if data is for I/O then it places the address on the address line and set I/O read or write control line on due to which data transfer occurs between CPU and I/O. As the address space of memory and I/O is isolated and the name is so. The address for I/O here is called ports. Here we have different read-write instruction for both I/O and memory.

#### **Memory Mapped I/O –**

In this case every bus in common due to which the same set of instructions work for memory and I/O. Hence we manipulate I/O same as memory and both have same address space, due to which addressing capability of memory become less because some part is occupied by the I/O.

**Differences between memory mapped I/O and isolated I/O –**

