PROGRAMMING WITH 8085
LECTURE 2

INTRODUCTION TO
8085
INSTRUCTIONS

1. Introduction

- A microprocessor executes instructions given by the user
- Instructions should be in a language known to the microprocessor
- Microprocessor understands the language of 0's and 1's only
- This language is called **Machine Language**

A Machine language program to add two numbers

00111110	;Copy value 2H in register A
00000010	
00000110	;Copy value 4H in register B
00000100	
10000000	;A = A + B

Assembly Language of 8085

- It uses English like words to convey the action/meaning called as MNEMONICS
- For e.g.
  - MOV to indicate data transfer
  - ADD to add two values
  - SUB to subtract two values

Assembly language program to add two numbers

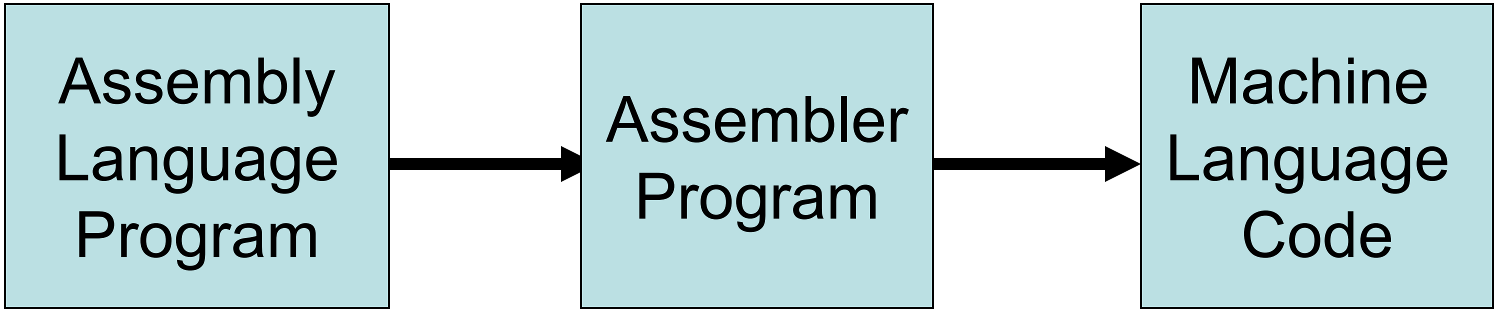
```
MVI A, 2H   ;Copy value 2H in register A
MVI B, 4H   ;Copy value 4H in register B
ADD B            ;A = A + B
```

Note:

- Assembly language is specific to a given processor
- For e.g. assembly language of 8085 is different than that of Motorola 6800 microprocessor

Microprocessor understands Machine Language only!

- Microprocessor cannot understand a program written in Assembly language
- A program known as **Assembler** is used to convert a Assembly language program to machine language

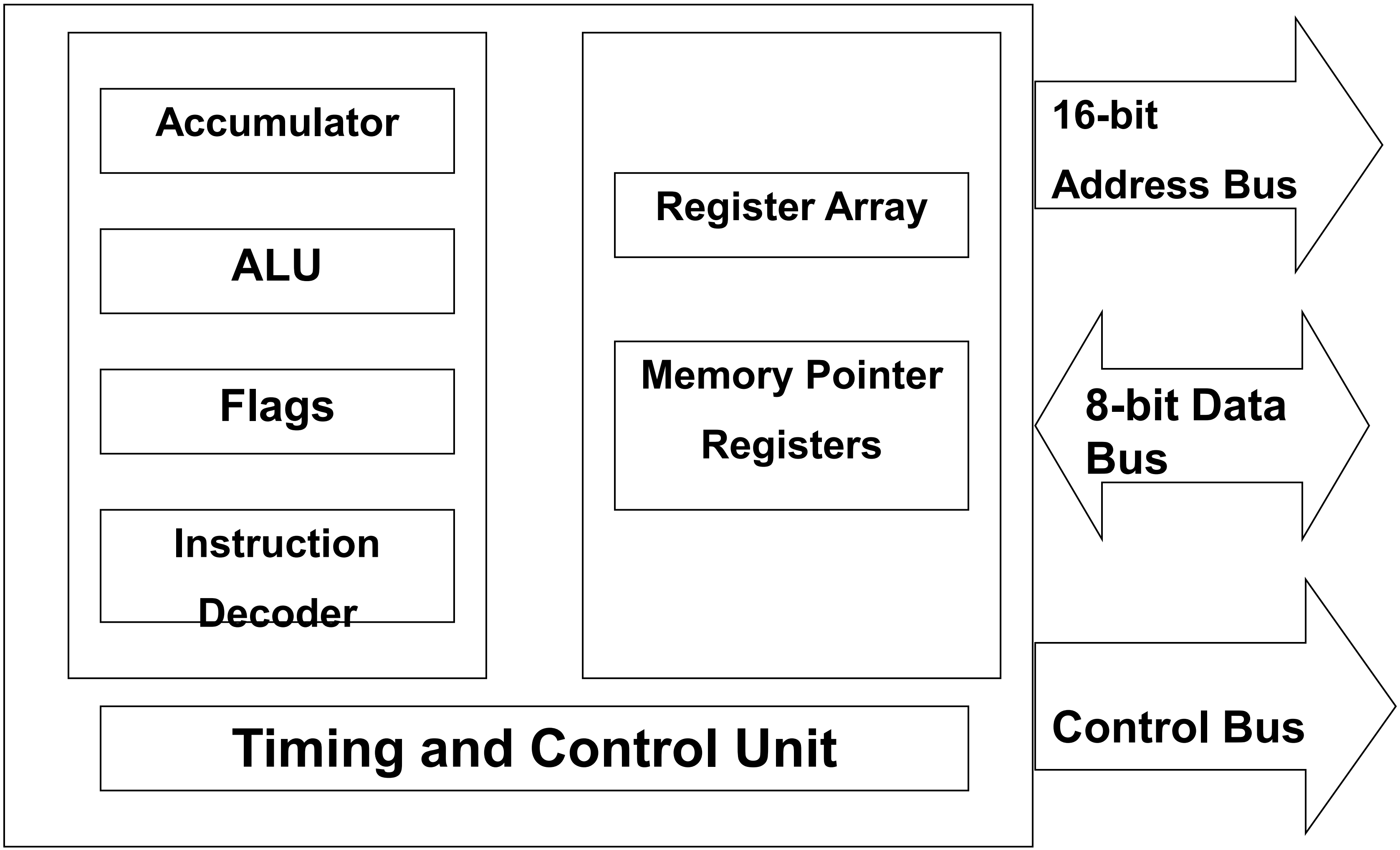


Low-level/High-level languages

- Machine language and Assembly language are both
  - Microprocessor specific (**Machine dependent**)

    so they are called
  - Low-level languages
- **Machine independent** languages are called
  - High-level languages
  - For e.g. BASIC, PASCAL,C++,C,JAVA, etc.
  - A software called **Compiler** is required to convert a high-level language program to machine code

2. Programming model of 8085



Accumulator (8-bit)	Flag Register (8-bit) S | Z | | AC | | P | | CY
B (8-bit)	C (8-bit)
D (8-bit)	E (8-bit)
H (8-bit)	L (8-bit)
Stack Pointer (SP) (16-bit)	
Program Counter (PC) (16-bit)	

8- Lines

Bidirectional

16- Lines

Unidirectional

Overview: 8085 Programming model

1. Six general-purpose Registers
2. Accumulator Register
3. Flag Register
4. Program Counter Register
5. Stack Pointer Register

1. **Six general-purpose registers**
   - B, C, D, E, H, L
   - Can be combined as register pairs to perform 16-bit operations (BC, DE, HL)
2. **Accumulator – identified by name A**
   - This register is a part of ALU
   - 8-bit data storage
   - Performs arithmetic and logical operations
   - Result of an operation is stored in accumulator

3. Flag Register

- This is also a part of ALU
- 8085 has five flags named
  - **Zero** flag (Z)
  - **Carry** flag (CY)
  - **Sign** flag (S)
  - **Parity** flag (P)
  - **Auxiliary Carry** flag (AC)

- These flags are five flip-flops in flag register
- Execution of an arithmetic/logic operation can **set** or **reset** these flags
- Condition of flags (set or reset) can be tested through software instructions
- 8085 uses these flags in decision-making process

4. Program Counter (PC)

- A 16-bit memory pointer register
- Used to sequence execution of program instructions
- Stores address of a memory location
  - where next instruction byte is to be fetched by the 8085
- when 8085 gets busy to fetch current instruction from memory
  - PC is incremented by one
  - PC is now pointing to the address of next instruction

5. Stack Pointer Register

- a 16-bit memory pointer register
- Points to a location in **Stack** memory
- Beginning of the stack is defined by loading a 16-bit address in stack pointer register

3.Instruction Set of 8085

- Consists of
  - 74 operation codes, e.g. MOV
  - 246 Instructions, e.g. MOV A,B
- 8085 instructions can be classified as

1. **Data Transfer (Copy)**
2. **Arithmetic**
3. **Logical and Bit manipulation**
4. **Branch**
5. **Machine Control**

1. Data Transfer (Copy) Operations

Copying data from a source to destination refers to data transfer function.

1. **Load** a 8-bit number in a Register
2. **Copy** from Register to Register
3. **Copy** between Register and Memory
4. **Copy** between Input/Output Port and Accumulator
5. **Load** a 16-bit number in a Register pair
6. **Copy** between Register pair and Stack memory

Example Data Transfer (Copy) Operations / Instructions

Operations	Instructions
1. **Load** a 8-bit number 4F in register B	**MVI B, 4FH**
2. **Copy** from Register B to Register A	**MOV A,B**
3. **Load** a 16-bit number 2050 in Register pair HL	**LXI H, 2050H**
4. **Copy** from Register B to Memory Address 2050	**MOV M,B**
5. **Copy** between Input/Output Port and Accumulator	**OUT 01H** **IN 07H**

Data Transfer (Copy) Operations

6. 1 byte instruction. Processor stops executing and enters wait state. — **HLT**

7. 1 byte instruction . No operation .Generally used to increase processing time or substitute in place of instruction. — **NOP**

2. Arithmetic Operations

1. **Addition** of two 8-bit numbers
2. **Subtraction** of two 8-bit numbers
3. **Increment/ Decrement** a 8-bit number

Example Arithmetic Operations / Instructions

Operation	Instruction
1. **Add** a 8-bit number 32H to Accumulator	**ADI 32H**
2. **Add** contents of Register B to Accumulator	**ADD B**
3. **Subtract** a 8-bit number 32H from Accumulator	**SUI 32H**
4. **Subtract** contents of Register C from Accumulator	**SUB C**
5. **Increment** the contents of Register D by 1	**INR D**
6. **Decrement** the contents of Register E by 1	**DCR E**

3. Logical & Bit Manipulation Operations

1. **AND** two 8-bit numbers
2. **OR** two 8-bit numbers
3. **Exclusive-OR** two 8-bit numbers
4. **Compare** two 8-bit numbers
5. **Complement**
6. **Rotate** Left/Right Accumulator bits

Example Logical & Bit Manipulation Operations / Instructions

1. Logically **AND** Register H with Accumulator	**ANA H**
2. Logically **OR** Register L with Accumulator	**ORA L**
3. Logically **XOR** Register B with Accumulator	**XRA B**
4. Compare contents of Register C with Accumulator	**CMP C**
5. Complement Accumulator	**CMA**
6. Rotate Accumulator Left	**RAL**

4. Branching Operations

These operations are used to control the flow of program execution

1.Jumps

- Conditional jumps
- Unconditional jumps

2.Call & **Return**

- Conditional Call & Return
- Unconditional Call & Return

Example Branching Operations / Instructions

1. **Jump** to a 16-bit Address 2080H if Carry flag is SET. This is conditional jump. **JNC, JZ,JNZ,JP,JM,JPE,JPO**
2. **Unconditional Jump**

   3 byte instruction. 2nd and 3rd byte specify 16 bit memory address.
3. **Call** a subroutine with its 16-bit Address
4. **Return back** from the Call
5. **Call** a subroutine with its 16-bit Address if **C**arry flag is **RESET**
6. **Return** if **Z**ero flag is **SET**

JC 2080H

JMP 2050H

CALL 3050H

RET

CNC 3050H

RZ