Introduction of Microprocessor 8085

By

Prof. K.V.Pawar

M.Sc.

Associate Professor Department of Electronics

Head Department of Electronics Science

Mahatma Fule Arts, Commerce and Sitaramji Choudhary Science Mahavidhyalaya, Warud Dist. Amravati

Mobile:9423054134 Email: kvpawar123@gmail.com

Keyboard
Mouse
Scanner
Microphone

Processor
Memory

Printer
Speaker
Monitor

Languages

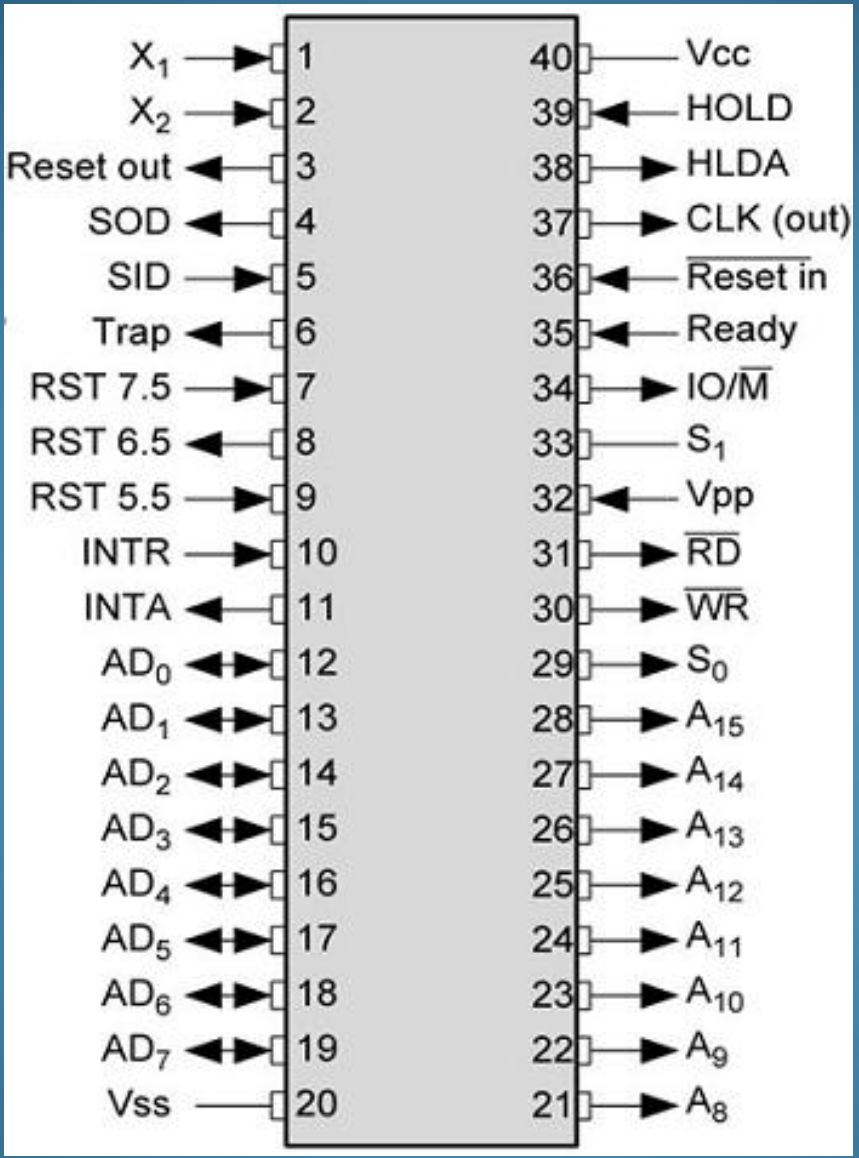
- ✓High level Language
- ✓Low level Language
- ✓Middle level Language
- ✓Machine level Language
- ✓Assembly level Language
- ✓Binary level Language
- ✓OCTAL level Language
- ✓HEXA level Language
- ✓DECIMAL level Language

- Assembler
- Interpreter
- Compiler

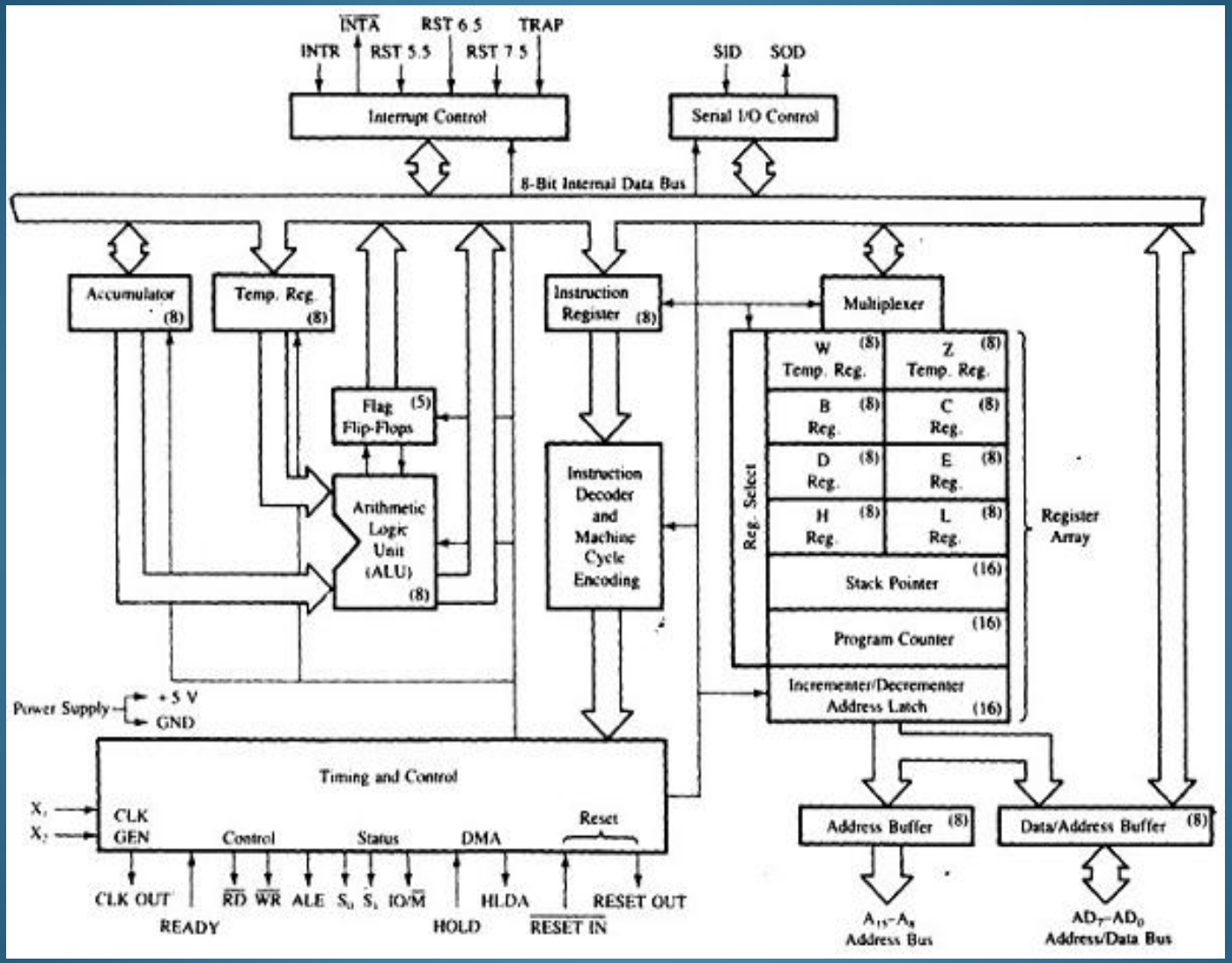
Manufacturer of Microprocessor:

INTEL, TOSHIBA, NATIONAL SEMICONDUCTOR, FAIRCHILD, ZILOG, ATMEL, CYPRUS, IBM

The 8085 Microprocessor Architecture



Pin Diagram of 8085 Microprocessor



Block Diagram of 8085 Microprocessor

Processor System Architecture

The typical processor system consists of:

- *CPU (central processing unit)*
  - *ALU (arithmetic-logic unit)*
  - *Control Logic*
  - *Registers, etc...*
- *Memory*
- *Input / Output interfaces*

Interconnections between these units:

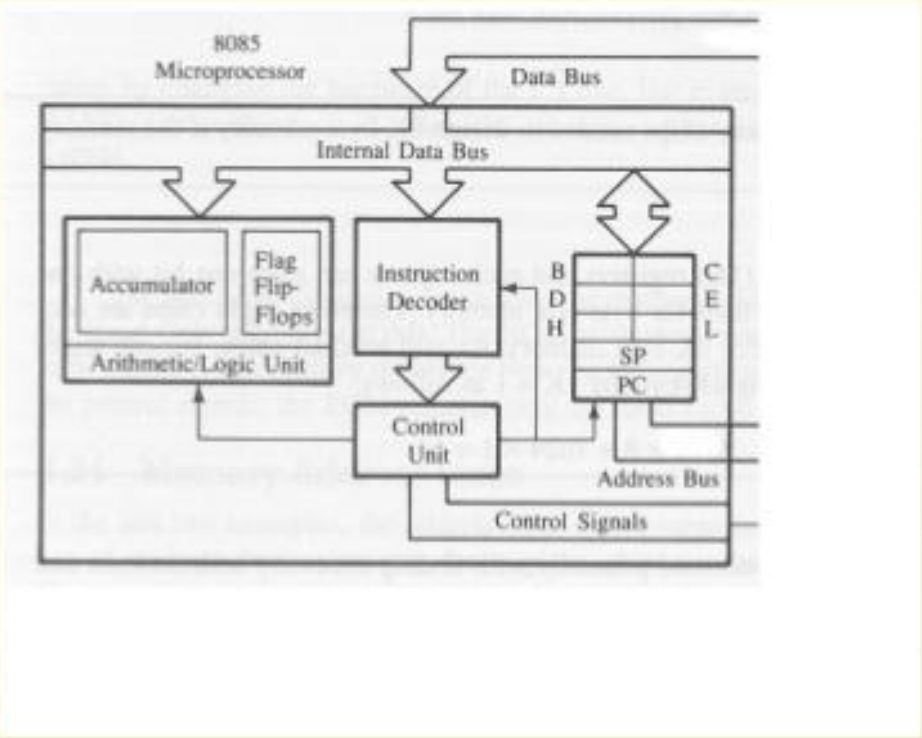
- Address Bus
- Data Bus
- Control Bus

The 8085: CPU Internal Structure

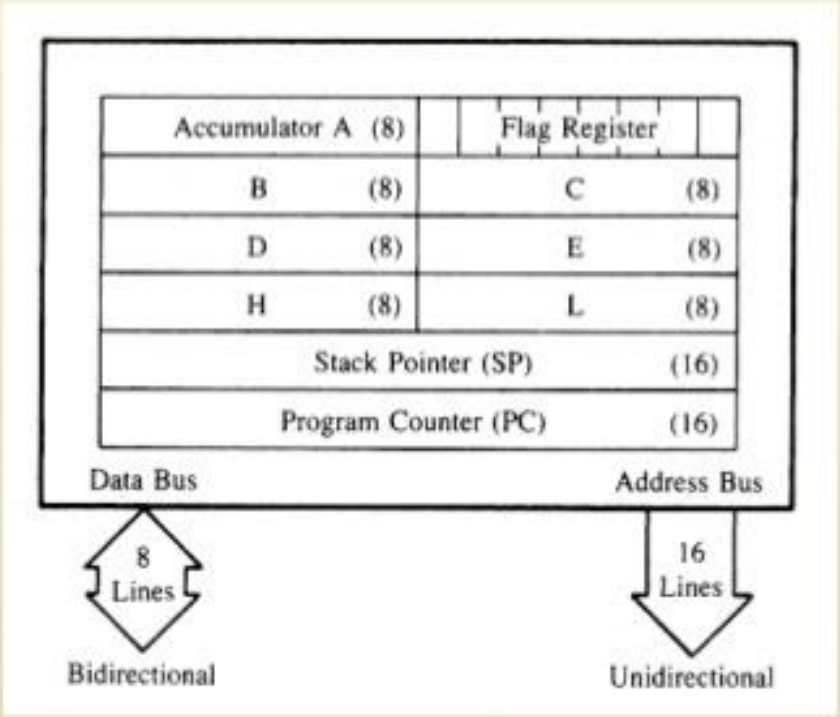
The internal architecture of the 8085 CPU is capable of performing the following operations:

- Store 8-bit data (Registers, Accumulator)
- Perform arithmetic and logic operations (ALU)
- Test for conditions (IF / THEN)
- Sequence the execution of instructions
- Store temporary data in RAM during execution

The 8085: CPU Internal Structure



The 8085: Registers



The 8085: CPU Internal Structure

Registers

- *Six general purpose 8-bit registers: B, C, D, E, H, L*
- *They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL*
- *Registers are programmable (data load, move, etc.)*

Accumulator

- *Single 8-bit register that is part of the ALU !*
- *Used for arithmetic / logic operations – the result is always stored in the accumulator.*

The 8085: CPU Internal Structure

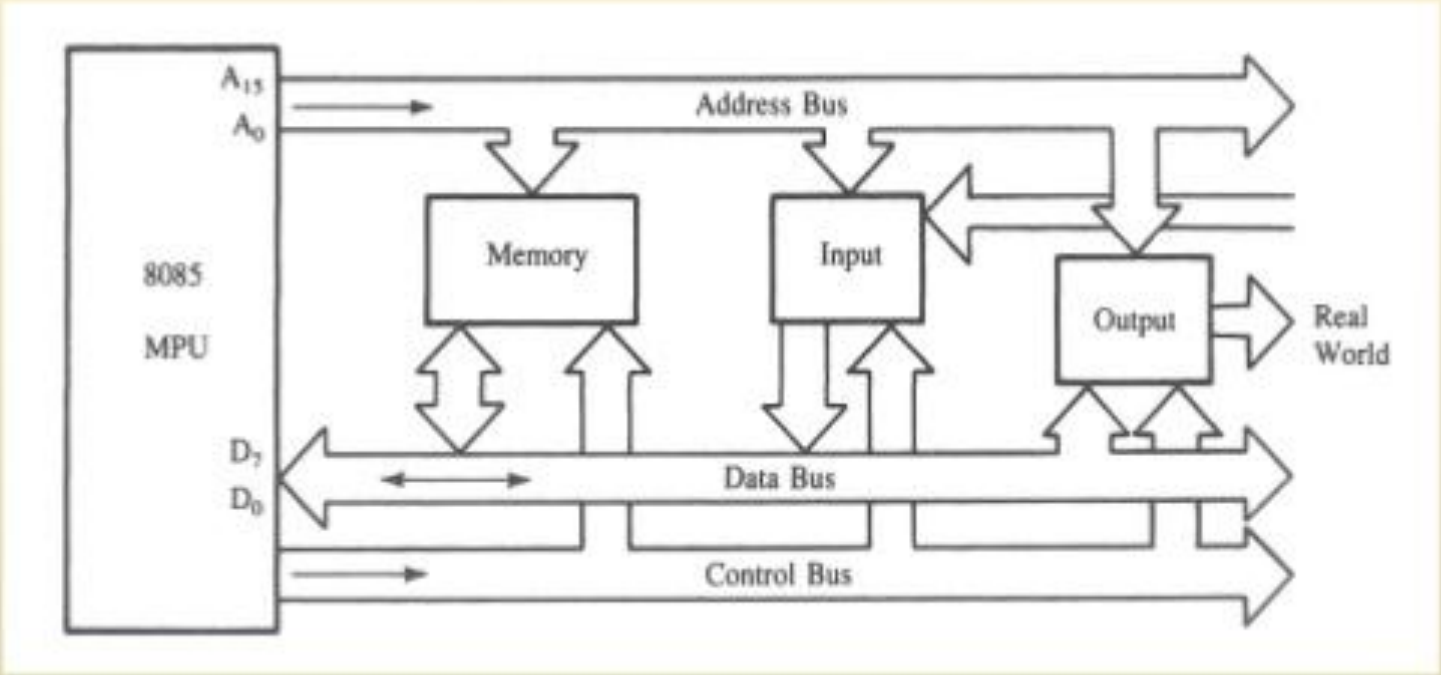
- *The Program Counter (PC)*
  - *This is a register that is used to control the sequencing of the execution of instructions.*
  - *This register always holds the address of the next instruction.*
  - *Since it holds an address, it must be 16 bits wide.*
- *The Stack pointer*
  - *The stack pointer is also a 16-bit register that is used to point into memory.*
  - *The memory this register points to is a special area called the stack.*
  - *The stack is an area of memory used to hold data that will be retreived soon.*
  - *The stack is usually accessed in a Last In First Out (LIFO) fashion.*

The 8085 and Its Busses

- The 8085 is an 8-bit general purpose microprocessor that can address 64K Byte of memory.
- It has 40 pins and uses +5V for power. It can run at a maximum frequency of 3 MHz.
  - The pins on the chip can be grouped into 6 groups:
    - Address Bus.
    - Data Bus.
    - Control and Status Signals.
    - Power supply and frequency.
    - Externally Initiated Signals.
    - Serial I/O ports.

The 8085 Bus Structure

The 8-bit 8085 CPU (or MPU – Micro Processing Unit) communicates with the other units using a 16-bit address bus, an 8-bit data bus and a control bus.



The Address and Data Busses

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.
  - So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
  - In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

The Control and Status Signals

- There are 4 main control and status signals. These are:
  - ALE: Address Latch Enable. This signal is a pulse that become 1 when the AD0 – AD7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
  - RD: Read. Active low.
  - WR: Write. Active low.
  - IO/M: This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M=1).
  - S1 and S0 : Status signals to specify the kind of operation being performed .Usually un-used in small systems.

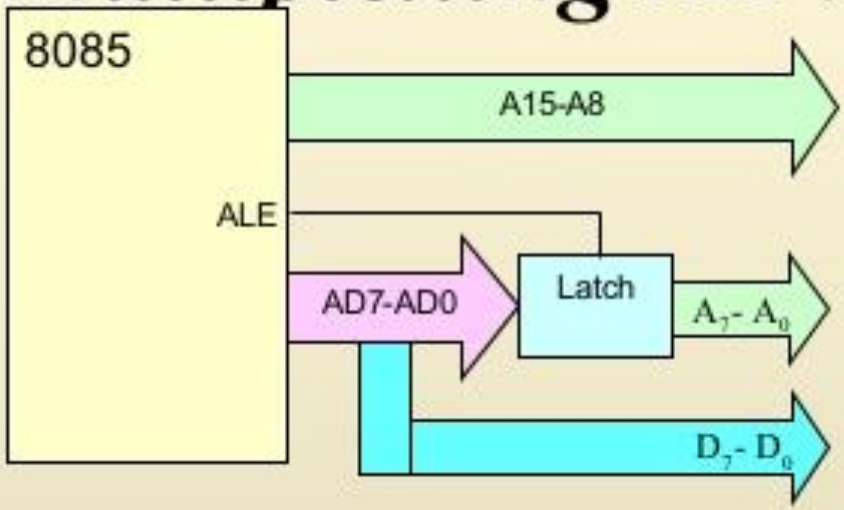
Frequency Control Signals

- There are 3 important pins in the frequency control group.
  - X0 and X1 are the inputs from the crystal or clock generating circuit.
    - The frequency is internally divided by 2.
      - So, to run the microprocessor at 3 MHz, a clock running at 6 MHz should be connected to the X0 and X1 pins.
  - CLK (OUT): An output clock pin to drive the clock of the rest of the system.
- We will discuss the rest of the control signals as we get to them.

Demultiplexing AD7-AD0

- From the above description, it becomes obvious that the AD7– AD0 lines are serving a dual purpose and that they need to be demultiplexed to get all the information.
- The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally. Also, notice that the low order bits of the address disappear when they are needed most.
- To make sure we have the entire address for the full three clock cycles, we will use an external latch to save the value of AD7– AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.

Demultiplexing AD7-AD0



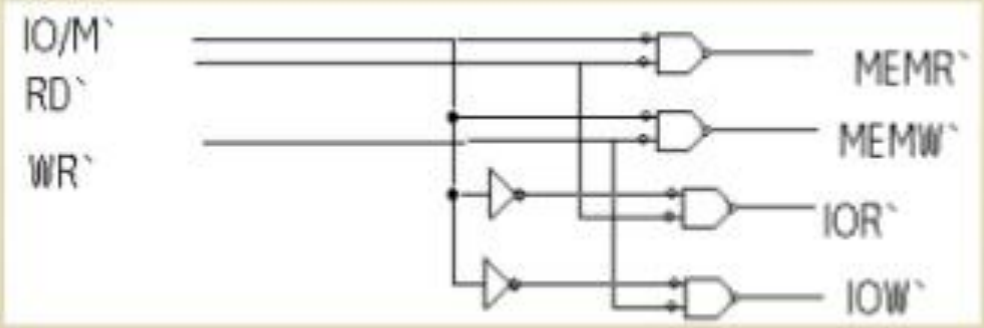
- ***Given that ALE operates as a pulse during T1, we will be able to latch the address. Then when ALE goes low, the address is saved and the AD7– AD0 lines can be used for their purpose as the bi-directional data lines.***

Cycles and States

- ***From the above discussion, we can define terms that will become handy later on:***
  - ***T- State: One subdivision of an operation. A T-state lasts for one clock period.***
    - ***An instruction's execution length is usually measured in a number of T-states. (clock cycles).***
  - ***Machine Cycle: The time required to complete one operation of accessing memory, I/O, or acknowledging an external request.***
    - ***This cycle may consist of 3 to 6 T-states.***
  - ***Instruction Cycle: The time required to complete the execution of an instruction.***
    - ***In the 8085, an instruction cycle may consist of 1 to 6 machine cycles.***

Generating Control Signals

- ***The 8085 generates a single RD signal. However, the signal needs to be used with both memory and I/O. So, it must be combined with the IO/M signal to generate different control signals for the memory and I/O.***
  - ***Keeping in mind the operation of the IO/M signal we can use the following circuitry to generate the right set of signals:***



The ALU

- ***In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.***

- ***Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer.***

Flag Register of 8085 µP

The Flags register

- There is also the flags register whose bits are affected by the arithmetic & logic operations.
  - S-sign flag
    - The sign flag is set if bit D7 of the accumulator is set after an arithmetic or logic operation.
  - Z-zero flag
    - Set if the result of the ALU operation is 0. Otherwise is reset. This flag is affected by operations on the accumulator as well as other registers. (DCR B).
  - AC-Auxiliary Carry
    - This flag is set when a carry is generated from bit D3 and passed to D4 . This flag is used only internally for BCD operations. (Section 10.5 describes BCD addition including the DAA instruction).
  - P-Parity flag
    - After an ALU operation if the result has an even # of 1's the p-flag is set. Otherwise it is cleared. So, the flag can be used to indicate even parity.
  - CY-carry flag
    - Discussed earlier

Addressing Modes in 8085

1) Immediate addressing mode

In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand.

For example: MVI B, 20H: means 20 is copied into register B.

2) Register addressing mode

In this mode, the data is copied from one register to another.

For example: MOV A, B: means data in register B is copied to register A

3) Direct addressing mode

In this mode, the data is directly copied from the given address to the register.

For example: LDA 5000H: means the data at address 5000H is copied to register A.

4**) Indirect addressing mode**

In this mode, the data is transferred from one register to another by using the address pointed by the register.

For example: MOV A, M

means data is transferred from the memory address pointed by the register whose address is stored in HL pair to the register A.

5) **Implied /Implicit addressing mode**

This mode doesn't require any operand;
the data is specified by the opcode itself.

For example: CMP.

Instruction set of 8085

1) Arithmetic instruction group
2) Logical instruction group
3) Data Transfer instruction group
4) Branch instruction group
5) Control instruction group

1) Arithmetic instruction group

ADD, SUB, ADI, SUB, ADC, SBB, DAD, INR, DCR etc.

2) Logical instruction group

ANA,ORA,CMP,ORI,XRA,XRI,RAC,RRC etc.

3) Data Transfer instruction group

MOV, MVI, LDA, LDAx, LHLD, STA, STAx, SHLD, XCHG, PUSH, POP etc.

4) Branch instruction group

JC, JNC, JP, JM, JZ, JNZ, JPE ,CC, CNC,CP, CM, RP,RM,RZ,RNZ

5) Control instruction group

HLT,DI, EI, RIM,SIM

More on the 8085 machine cycles

- *The 8085 executes several types of instructions with each requiring a different number of operations of different types. However, the operations can be grouped into a small set.*
- *The three main types are:*
  - *Memory Read and Write.*
  - *I/O Read and Write.*
  - *Request Acknowledge.*
- *These can be further divided into various operations (machine cycles).*

Opcode Fetch Machine Cycle

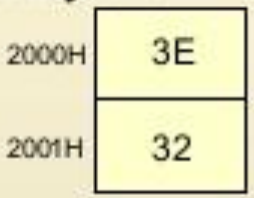
- *The first step of executing any instruction is the Opcode fetch cycle.*
  - *In this cycle, the microprocessor brings in the instruction's Opcode from memory.*
    - *To differentiate this machine cycle from the very similar "memory read" cycle, the control & status signals are set as follows:*
      - *IO/M=0, s0 and s1 are both 1.*
  - *This machine cycle has four T-states.*
    - *The 8085 uses the first 3 T-states to fetch the opcode.*
    - *T4 is used to decode and execute it.*
  - *It is also possible for an instruction to have 6 T-states in an opcode fetch machine cycle.*

Memory Read Machine Cycle

- *The memory read machine cycle is exactly the same as the opcode fetch except:*
  - *It only has 3 T-states*
  - *The s0 signal is set to 0 instead.*

The Memory Read Machine Cycle

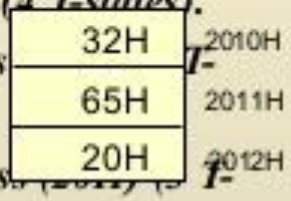
- To understand the memory read machine cycle, let's study the execution of the following instruction:
  - MVI A, 32
- In memory, this instruction looks like:



  - The first byte 3EH represents the opcode for loading a byte into the accumulator (MVI A), the second byte is the data to be loaded.
- The 8085 needs to read these two bytes from memory before it can execute the instruction. Therefore, it will need at least two machine cycles.
  - The first machine cycle is the opcode fetch discussed earlier.
  - The second machine cycle is the Memory Read Cycle.
  - Figure 3.10 page 83.

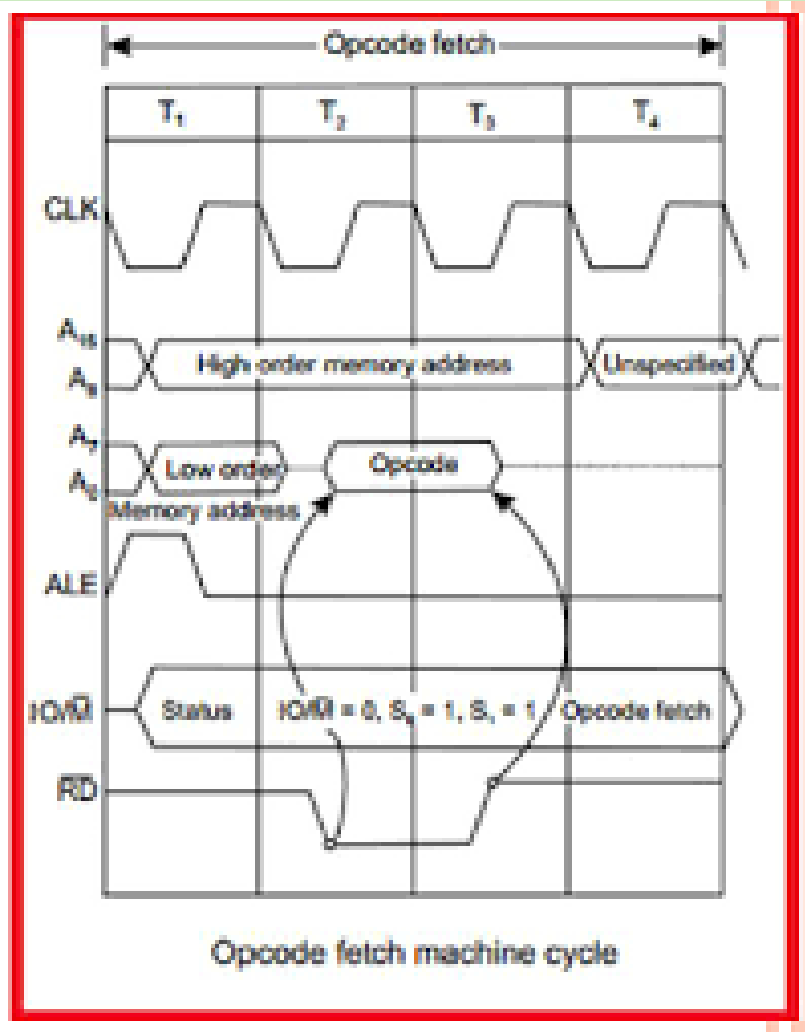
Machine Cycles vs. Number of bytes in the instruction

- *Machine cycles and instruction length, do not have a direct relationship.*
  - *To illustrate lets look at the machine cycles needed to execute the following instruction.*
    - *STA 2065H*
    - *This is a 3-byte instruction requiring 4 machine cycles and 13 T-states.*
    - *The machine code will be stored in memory as shown to the right*
    - *This instruction requires the following 4 machine cycles:*
      - *Opcode fetch to fetch the opcode (32H) from location 2010H, decode it and determine that 2 more bytes are needed (4 T-states).*
      - *Memory read to read the low order byte of the address (3 T-states).*
      - *Memory read to read the high order byte of the address (3 T-states).*
      - *A memory write to write the contents of the accumulator into the memory location.*

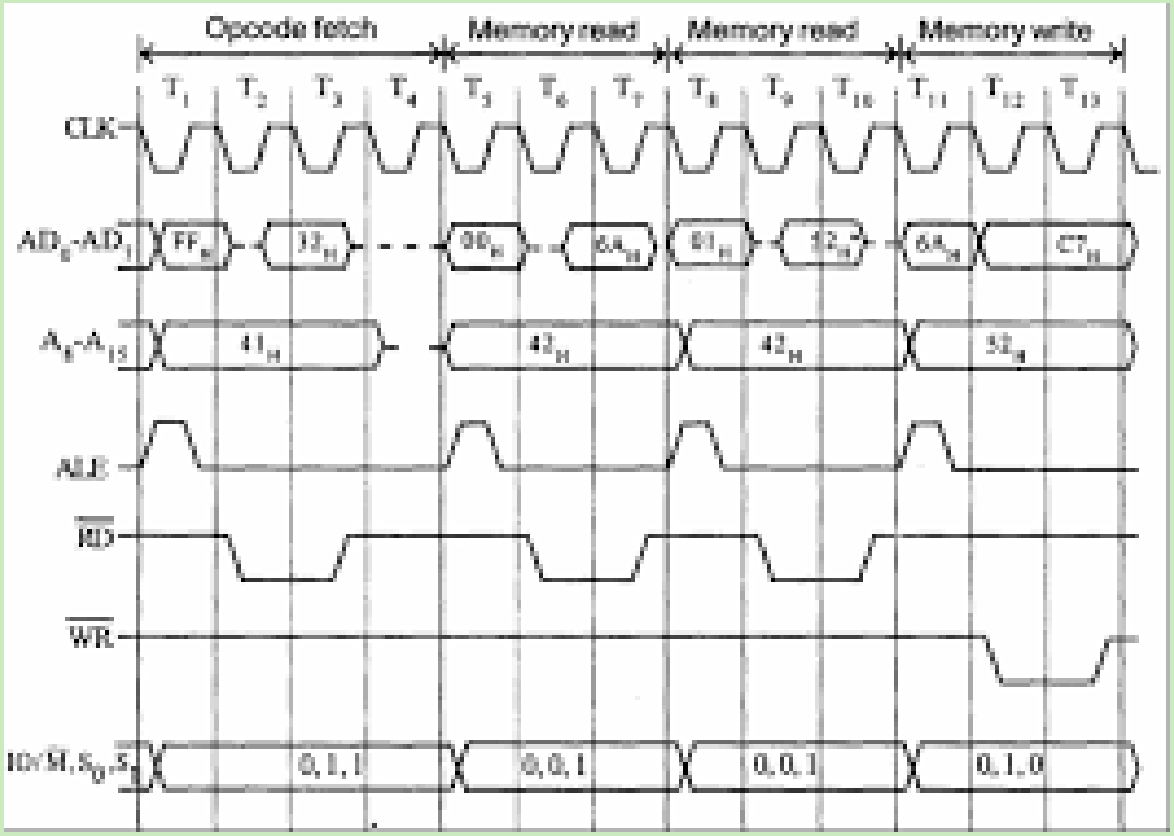


The Memory Write Operation

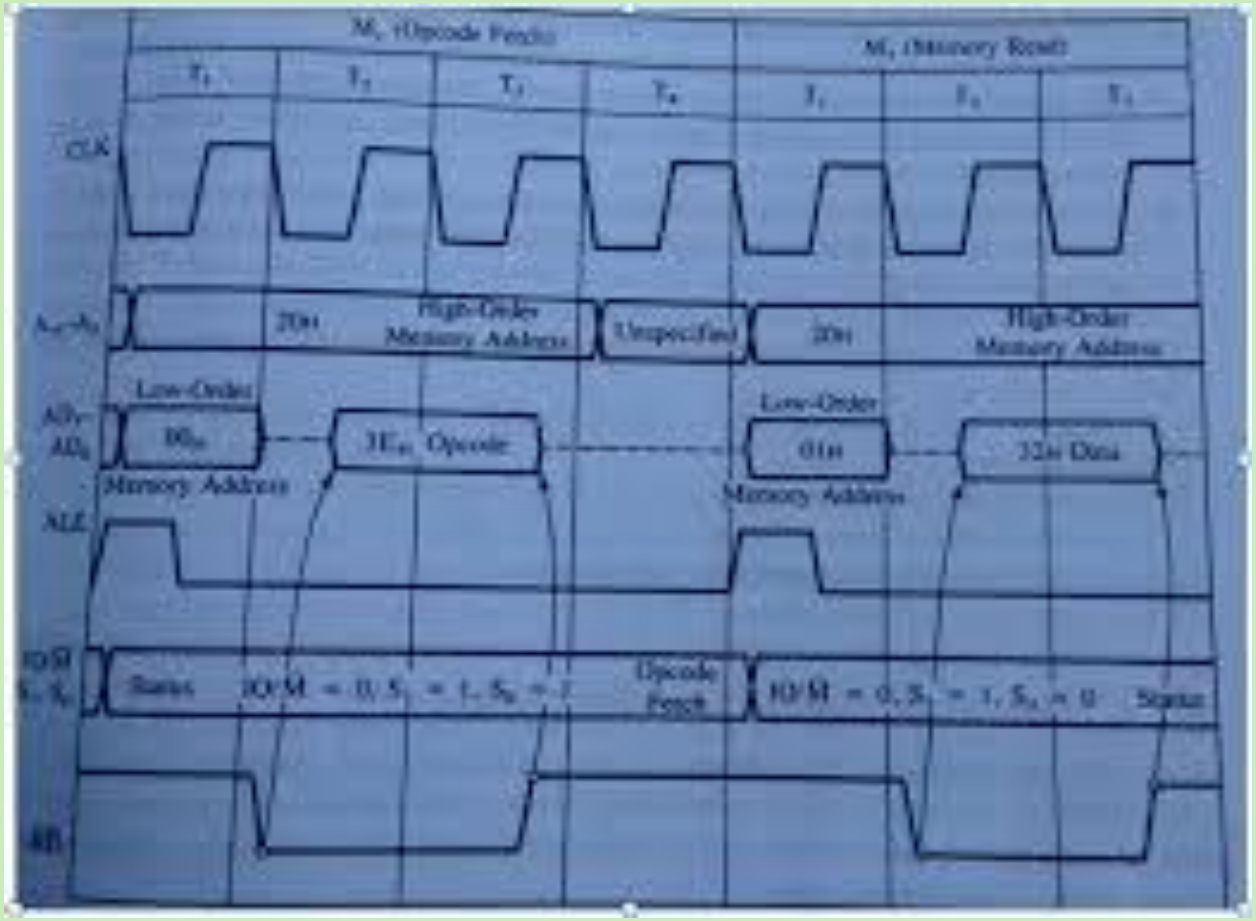
- *In a memory write operation:*
  - *The 8085 places the address (2065H) on the address bus*
  - *Identifies the operation as a memory write (IO/M=0, s1=0, s0=1).*
  - *Places the contents of the accumulator on the data bus and asserts the signal WR.*
  - *During the last T-state, the contents of the data bus are saved into the memory location.*



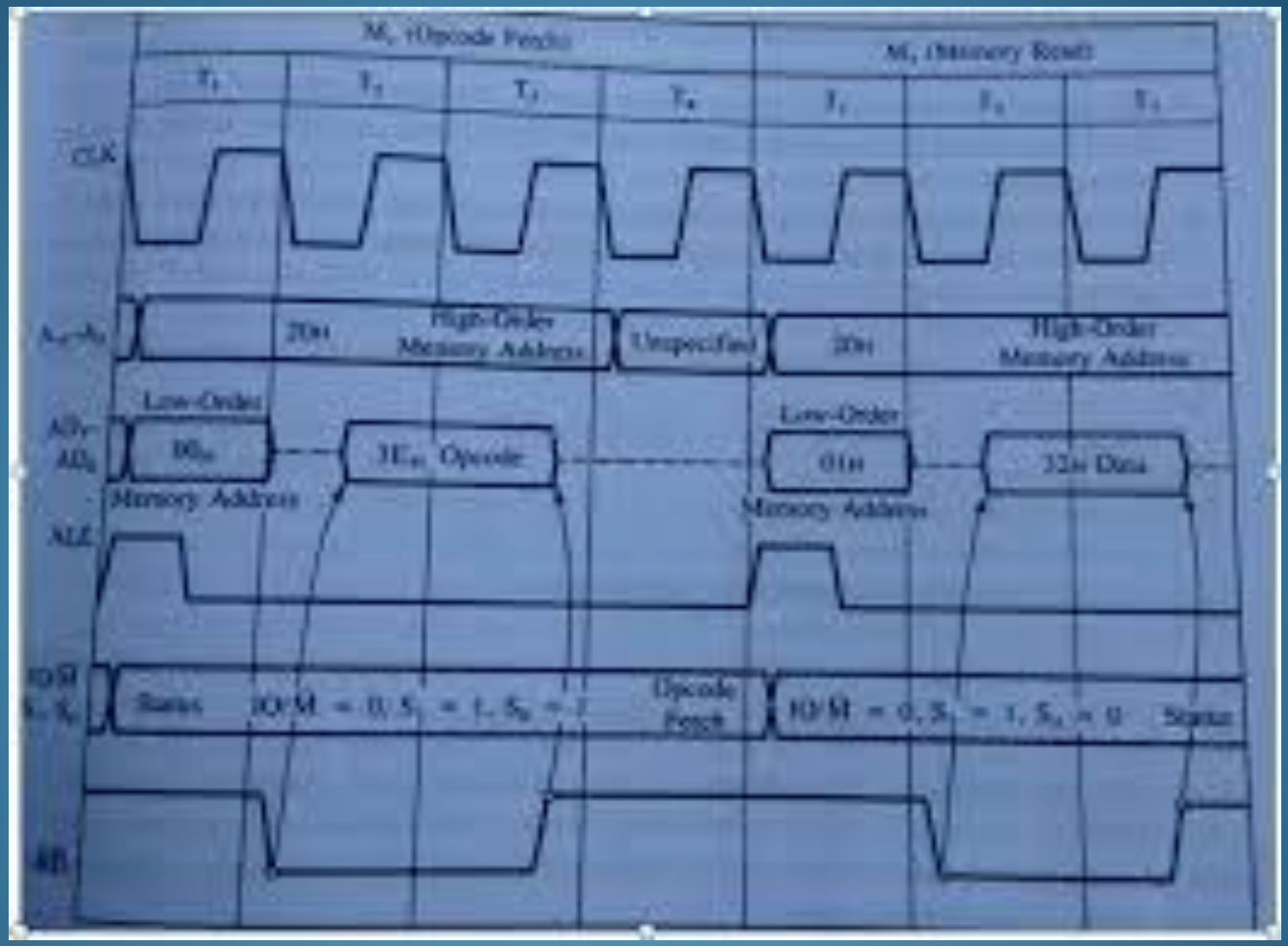
Timing Diagram for Op-code fetch



Timing Diagram for STA Instruction



Timing Diagram for Memory read Instruction



Timing Diagram for Memory write Instruction

- Algoritham
- Flowchart
- Mnemonics
- Op-Code
- Operand
- Comments

Sample Programs

Write an assembly program to add two numbers

```
MVI D, 8BH
MVI C, 6FH
MOV A, C
ADD D
OUT PORT1
HLT
```

Sample Programs

Write an assembly program to Subtract two numbers

```
MVI D, 8BH
MVI C, 6FH
MOV A, C
SUB D
OUT PORT1
HLT
```

Write an assembly program to multiply a number by 8

Program

```
 MVI A, 30H
 RRC
 RRC
 RRC
OUT PORT1
 HLT
```

Write an assembly program to find greatest between two numbers

Program

```
MVI B, 30H
MVI C, 40H
MOV A, B
CMP C
JZ EQU
 JC GRT
 OUT PORT1
 HLT

EQU:  MVI A, 01H
 OUT PORT1
 HLT
 GRT:  MOV A, C
 OUT PORT1
 HLT
```

Write a program to sort given 10 numbers from memory location 2200H in the ascending order.

```
MVI B, 09      :"Initialize counter"
START          :"LXI H, 2200H: Initialize memory
pointer"
MVI C, 09H     :"Initialize counter 2"
BACK: MOV A, M :"Get the number"
INX H          :"Increment memory pointer"
CMP M          :"Compare number with next
number"
JC SKIP        :"If less, don’t interchange"
JZ SKIP        :"If equal, don’t interchange"
MOV D, M
MOV M, A
DCX H
MOV M, D
INX H          :"Interchange two numbers"
SKIP:DCR C     :"Decrement counter 2"
JNZ BACK       :"If not zero, repeat"
DCR B          :"Decrement counter 1"
JNZ START
HLT            :"Terminate program execution"
```

Write a program to count number of l's in the contents of D register and store the count in the B register.

Sample problem
(2200H) = 04
(2201H) = 34H
(2202H) = A9H
(2203H) = 78H
(2204H) = 56H
Result = (2202H) = A9H

```
MVI B, 00H
MVI C, 08H
MOV A, D
BACK: RAR
JNC SKIP
INR B
SKIP: DCR C
JNZ BACK
HLT
```

Thanks

9423054134