Chapter-2

Assembly Language Programming (10hrs)

8085 Microprocessor Architecture:

- 8-bit general purpose microprocessor
- Capable of addressing 64K of memory (with 16 address lines).
- Has 40 pins DIP (Dual Inline Package)
- Requires +5V power supply
- Can operate with maximum 3MHz single phase clock and minimum 500 KHz clock.
- 8085 upward compatible(its instructions and features are available in 8086)
- It provides 74 instructions with 5 addressing modes

8085 Pinouts and Signals:

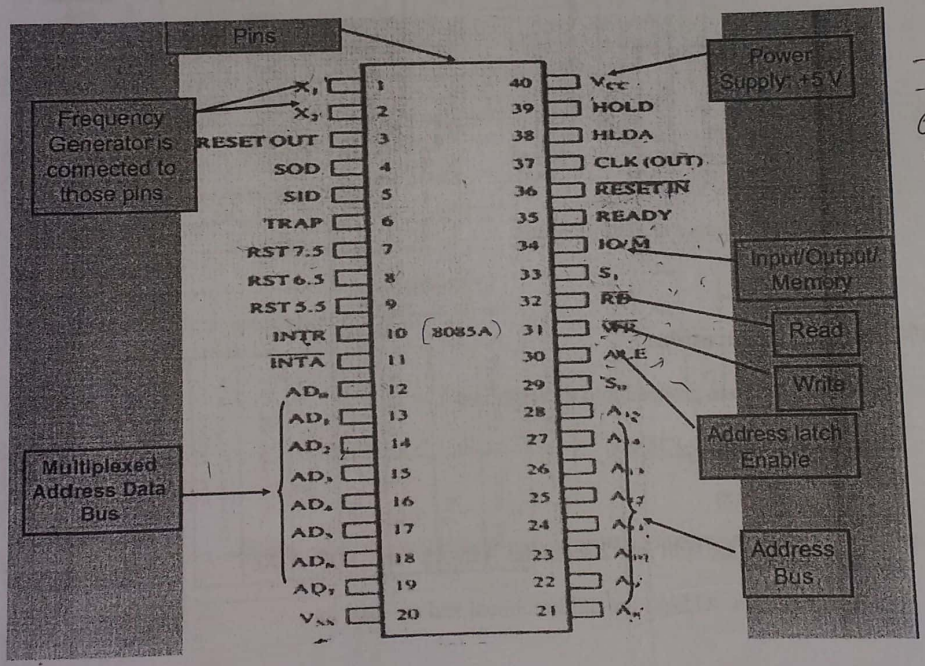


Fig. 2.1: 8085 Pinouts

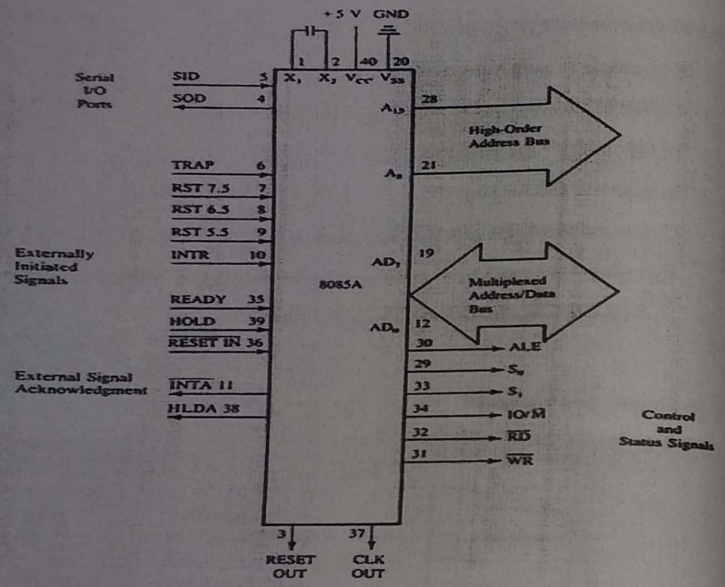


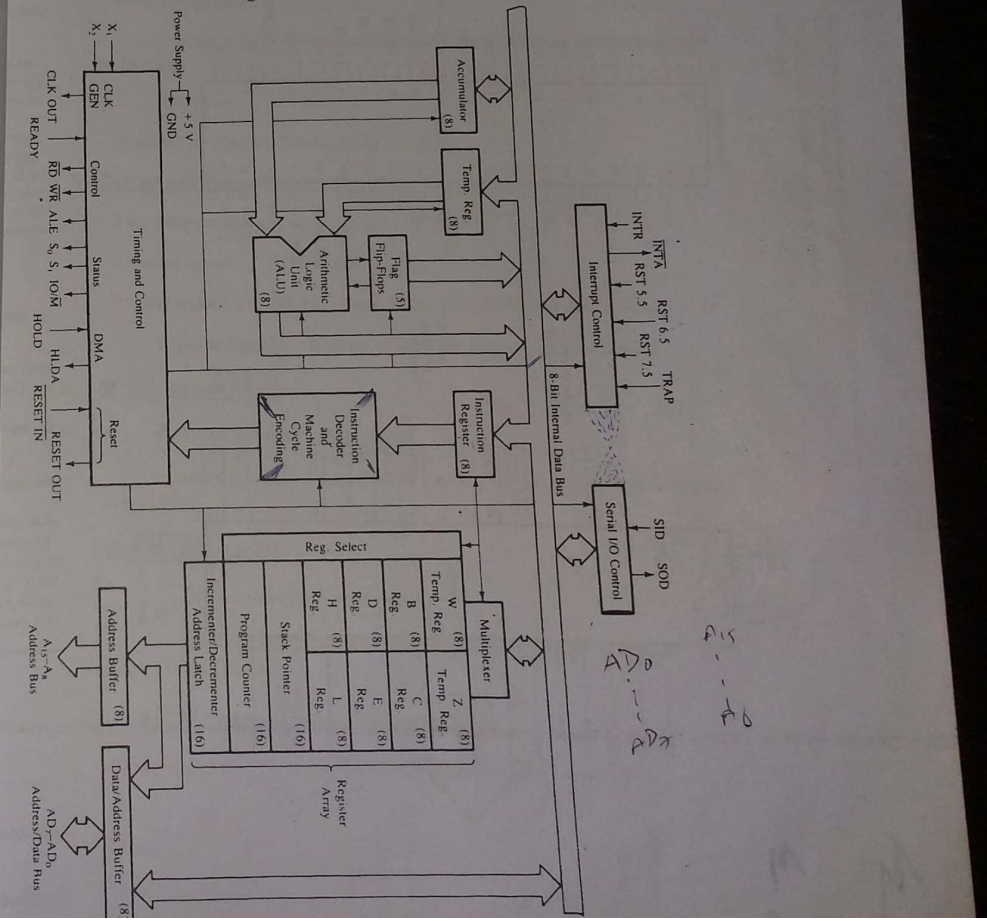
Fig. 2.2: Grouping of 8085 Pinouts

8085 Pin Configuration:

- Pins of 8085 are grouped into 6 categories:

1. **Address Bus(P21-P28):**

   - 16 signal lines
   - These lines are split into 2 segments A15-A8 and AD7- AD0.
   - 8 signal lines, A15-A8 are unidirectional and are used for



Most significant bits, called higher order address of 16-bit address.

2. **Multiplexed Address/Data Bus (P12-P19):**

- They are time multiplexed address and data bus.

- The signal lines AD7-AD0 are bidirectional.

- serve dual purpose:

. During execution of instruction cycles, these lines are used as low order address bus

. Data bus

- separated by Address Latch Enable (ALE).

3. **Control and Status Signals(P29 – P34):**

- This group includes:

2 control signals: (RD and WR.)

3 status signals : IO/ M, S1 and S0 (to identify the nature of operation) , &

1 special signal : ALE (to indicate the beginning of operation)

(i) $\overline{RD}$ **(Read) [P 32]:**

Active low read control signal

- used to read the selected I/O or memory device and make data available on data bus.

(ii) $\overline{WR}$ **(Write) [P31]:**

- Active low write control signal.

- indicated that the data on data bus are to be written on selected memory or I/O device.

(iii) **IO / $\overline{M}$ (Input – Output / Memory) [P 34]:**

- Status signal to differentiate between I/O and Memory operation.

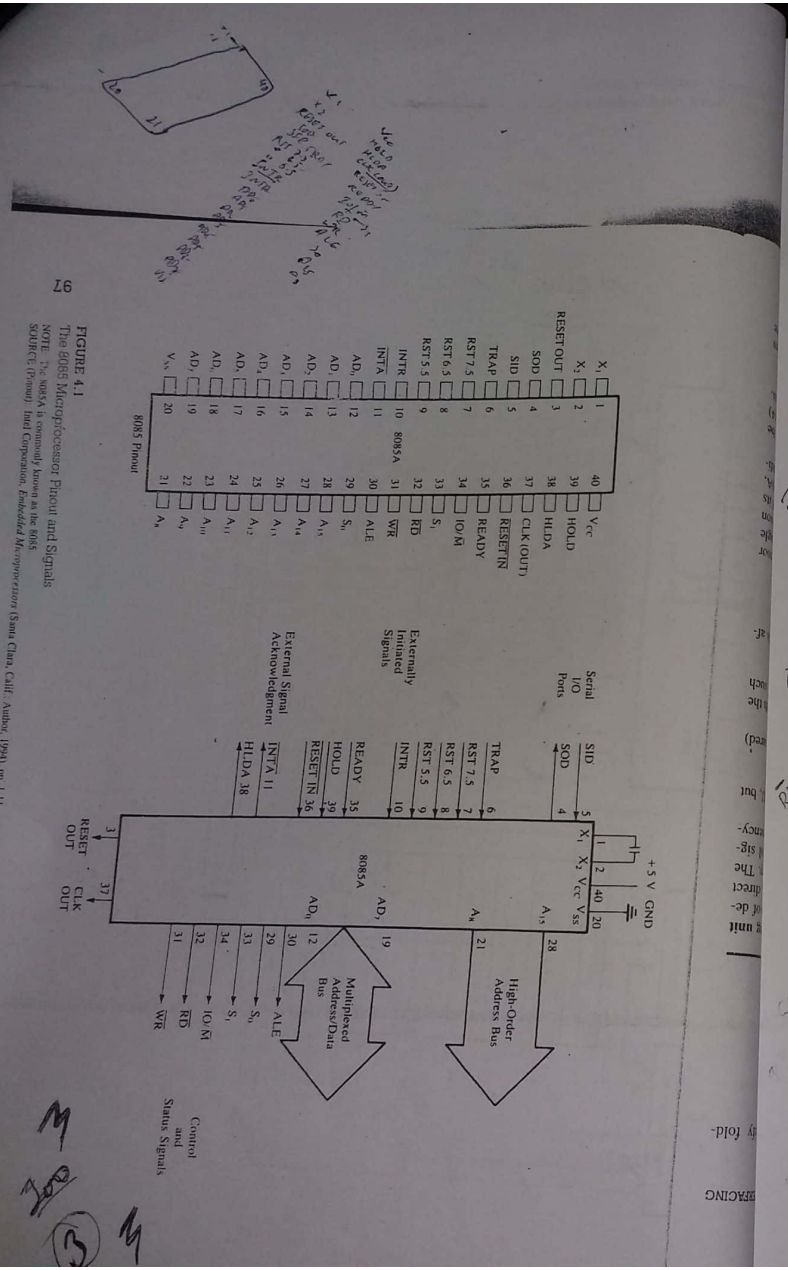


FIGURE 4.1
The 8085 Microprocessor Pinout and Signals
NOTE: The 8085A is commonly known as the 8085
SOURCE: (Pinout) Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-11.

- If IO / $\overline{M}$ = 1, input – output operation is done.

- If IO / $\overline{M}$ = 0, memory operation is done.

- These signals are combined with $\overline{RD}$ and $\overline{WR}$ control signals to generate I/O and Memory control signals ($\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$, $\overline{MEMW}$)

(iv) S1, S0 Signals)(SO= P29 , s1 = P33):

S1	S0	Operations
0	0	HLT
0	1	Write(Memory or I/O)
1	0	Read(Memory or I/O)
1	1	Opcode Fetch

Similar to IO/ M signals and identify various operations

- Rarely used in small operations

(v.) **Address Latch Enable (ALE)[P30]:**

- Active high signal.

- generated at the beginning of every machine cycle.

- Primarily used to latch the lower order address from the multiplexed bus and generate a separate set of 8 address lines A7-A0.

4. **Power Supply and Frequency:**

- *Vcc : + 5V Power Supply*

- *Vss : Ground Reference*

- ***X1, X2:***

. *A crystal RL or LC network*

. Frequency is internally divided by 2, so to operate a system at 3 MHz; the crystal should have a frequency of 6 MHz

- **CLK (OUT):**

. Used as a system clock for other devices.

5. **Externally Initiated Signals, Including Interrupts:**

- 8085 has 5 interrupt signals, 2 acknowledgement signals and 3 other signals.

- **Interrupt Signals are**:

RST 7.5, RST 6.5, RST 5.5(Inputs) :

- They are reset vectored interrupt that transfer the program control to specific location.

- cause internal restart for the system

- have higher priorities than INTR

- Among them the priority order is 7.5, 6.5 and 5.5.

TRAP (Input):

- Non-maskable interrupts and has highest priority.

- Used for critical events.

INTR (Input):

- Interrupt Request Signal

- Lowest priority signal

- used in data communication between I/O devices and microprocessor.

$\overline{INTA}$ **(Output)**

- Interrupt Acknowledgement signal.

- Microprocessor sends this signal after receiving INTR signal.

READY (Input):

- Used to delay microprocessor Read or Write Cycles until a slow responding peripheral is ready to send or receive data.

- When this signal goes low microprocessor waits and as soon as it goes high it reads the data in bus or writes in it.

HOLD (Input):

- Indicates that a peripheral devices e.g. DMA (Direct Memory Access) Controller is requesting the use of address and data buses.
  - When this signal goes high, microprocessor stops the Buses as soon as the current cycle is completed.
  - Microprocessor regains it when HOLD signal goes low.

HLDA (Output):

- This is a Hold Acknowledgement Signal and Acknowledges the HOLD request.
- Goes low when HOLD signal is removed and microprocessor takes over the Buses.

RESET – IN **[P 36]:**

- When the signal on this pin goes low, the Program Counter (PC) is set to zero.
- So, MPU is reset.

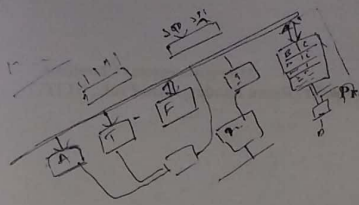
RESET OUT [P3]:

- It indicates that the microprocessor is being reset.
- This signal can be used to reset other devices,

(6.) **Serial I/O Devices [P4 and P5]:**

- SID (Serial Input Data) and SOD (Serial Output Data) used to implement serial transmission.

- In serial transmission, data are sent over a single line, one bit at a time, e.g. telephone line.



Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT

Internal Architecture of Intel 8085 Microprocessor:

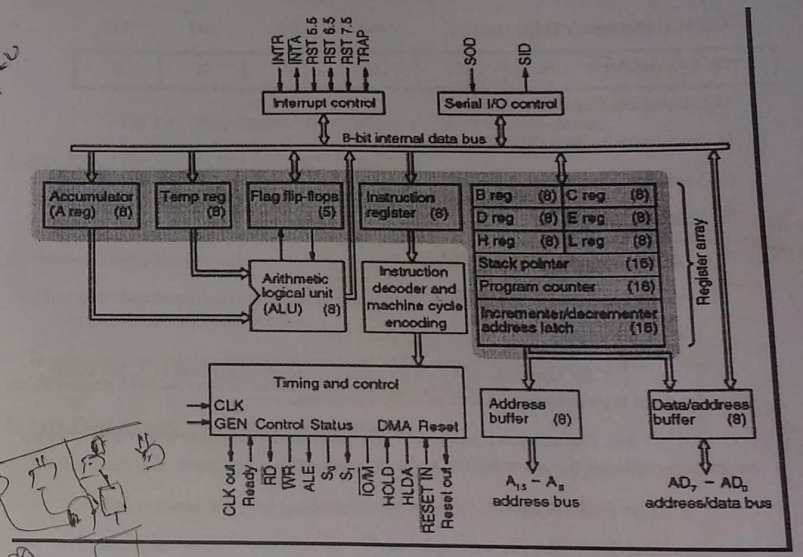
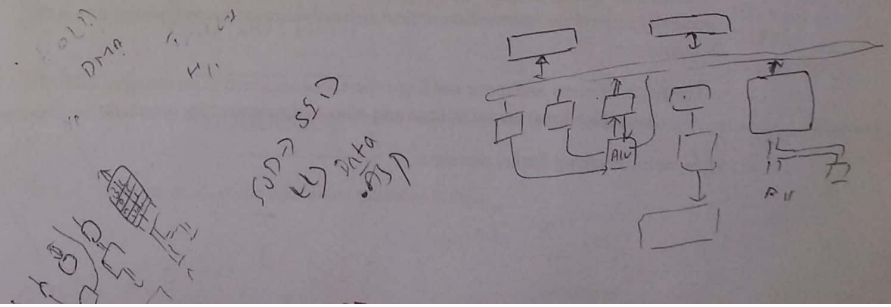


Fig 2.3: Functional Block Diagram of 8085 Microprocessor



The internal architecture of 8085 microprocessor consists of:

(1.) ALU (Arithmetic Logic Unit)

(2.) Register Array

(3.) Timing and Control Unit (CU)

(4.) Instruction Register (IR) and Decoder

(5.) Interrupt Controller

(6.) Serial I/O Control

1. **Arithmetic and Logic Unit(ALU):**

- performs the computing functions.

- It includes:

(I) the Accumulator(A) – 8 bit special purpose register used in arithmetic , logic ,load and store operations as well as in I/O operations,

(ii) The temporary registers (8 bit register used to hold data temporarily during program execution and is not accessible to the programmer.)

(iii) The arithmetic and logic circuits, and

(iv) Five flag flip-flop registers.

Flag Register:

- It is an 8 bit special purpose register (with 5 bits carrying significant information), that generally reflects the intermediate or final status of the result or conditions of data in the accumulator.

- The result of every arithmetic and logic operations are stored in accumulator, and the flags (flip-flops) are set or reset according to the result of the operation.

- Thus, helps in taking further decisions.

- The bit positions reserved for different flags in flag register array are as follows:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

Fig.2.4 . Flag Register Array (where, X = don't care conditions)

S- Sign Flag :(For –ve no. S=1 and for +ve no. S = 0)

- Used for indicating the sign of the data in the accumulator.

- For signed numbers, if D7 bit is 1, the number is viewed as negative number and if it is 0, the number is considered positive.

- Thus after the execution of arithmetic or logic operations, sign flag is set for negative value and reset for positive value.

- In arithmetic operations with signed numbers, bit D7 is reserved for indicating the sign, and remaining 7 bits are used to indicate the magnitude of a number.

Z- Zero Flag(if result = 0 , Z =1, if result ≠ 0, Z=0):

- This flag is set if the result of the ALU operations is 0 and reset if the result is non- zero.

- This flag is modified by the results in the accumulator as well as in other registers.

- For example:

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

- Here, Z =1 as the result of the operation (addition) is 00H.

The internal architecture of 8085 microprocessor consists of:

(1.) ALU (Arithmetic Logic Unit)

(2.) Register Array

(3.) Timing and Control Unit (CU)

(4.) Instruction Register (IR) and Decoder

(5.) Interrupt Controller

(6.) Serial I/O Control

1. **Arithmetic and Logic Unit(ALU):**

- performs the computing functions.

- It includes:

(I) the Accumulator(A) – 8 bit special purpose register used in arithmetic , logic ,load and store operations as well as in I/O operations,

(ii) The temporary registers (8 bit register used to hold data temporarily during program execution and is not accessible to the programmer.)

(iii) The arithmetic and logic circuits, and

(iv) Five flag flip-flop registers.

Flag Register:

- It is an 8 bit special purpose register (with 5 bits carrying significant information), that generally reflects the intermediate or final status of the result or conditions of data in the accumulator.

- The result of every arithmetic and logic operations are stored in accumulator, and the flags (flip-flops) are set or reset according to the result of the operation.

- Thus, helps in taking further decisions.

Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT

- The bit positions reserved for different flags in flag register array are as follows:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

Fig.2.4 . Flag Register Array (where, X = don't care conditions)

S- Sign Flag :(For –ve no. S=1 and for +ve no. S = 0)

- Used for indicating the sign of the data in the accumulator.

- For signed numbers, if D7 bit is 1, the number is viewed as negative number and if it is 0, the number is considered positive.

- Thus after the execution of arithmetic or logic operations, sign flag is set for negative value and reset for positive value.

- In arithmetic operations with signed numbers, bit D7 is reserved for indicating the sign, and remaining 7 bits are used to indicate the magnitude of a number.

Z- Zero Flag(if result = 0 , Z =1, if result ≠ 0, Z=0):

- This flag is set if the result of the ALU operations is 0 and reset if the result is non- zero.

- This flag is modified by the results in the accumulator as well as in other registers.

- For example:

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

Here, Z =1 as the result of the operation (addition) is 00H.

- Zero flag is also set if data in register becomes 00H after increment or decrement operation.

AC- Auxiliary Carry Flag (AC = 1 if carry occurs from bit D3 – D4, during addition):

- This flag is set whenever a carry occurs from lower nibble to upper nibble, i.e. when a carry is generated from by digit D3 and passed onto digit D4.
- Used for BCD operations.

P- Parity Flag (P= 1 for even parity & P = 0 for odd parity):

- Parity is defined as the number of 1s in the data in accumulator.
- After arithmetic and logical operations, if the result has even number of 1s , this flag is set and reset if it has odd number of 1s.
- So, parity flag is set for even parity and reset for odd parity.
- E. g. For data byte, 0000 0011, P=1, for even no. of 1s although magnitude of no. is odd.

C- Carry Flag (C= 1, for overflow during addition & borrow during subtraction):

- In an arithmetic operations such as addition and subtraction involving two 8 bit numbers, there is a situation when an add result is overflow from highest order bit or a borrow may be required in subtraction.
- In both cases, the carry flag is set, otherwise it is reset.

```
        1011 0101                1011 0101
      + 0110 1100              - 1100 1100
      ---------------          ---------------
Carry 1 0010 0001        Borrow 1 1110 1001
```

2. **Register Array:**

- 8085 has 8 and 16 bit registers.
- 8 bit registers are B, C, D, E, H, L, Accumulator, and Flag.
- There are two 16 bit registers:
  - Program Counter (PC),

  - Stack Pointer (SP), and
- 8 bit general purpose registers can be used as 8-bit or 16- bit register Pair.
- Register pair HL may hold data Pointer and it is used in register indirect addressing mode

  E.g. MOV M, A

(Copies content of A to the memory location contained by HL pair.)

- General purpose registers (GPRs) are Accessible to the programmer.

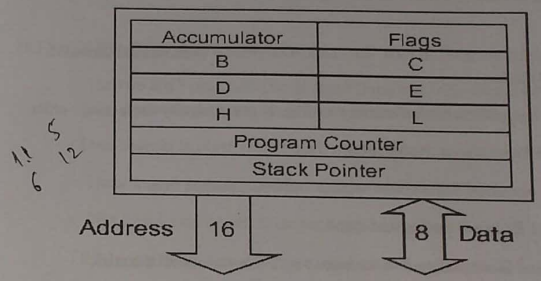


Fig 2.4: Register Array in 8085 microprocessor

Temporary Register W and Z:

- They hold 8-bit data during the execution of some instruction.
- They are used internally and are not available to the programmer.

Temporary Register (near ALU):

- It is an 8 bit register and is used to hold data for brief time period, during and arithmetic/logic operations.
- Not available to programmer.

16-bit registers:

(i) **Program Counter(PC):**

- Deals with sequencing and execution of instruction.

- contains 16 bit address of the next instruction to be executed.

- Acts as a pointer to the next instruction to be executed.

- It is operated by the processor and points the instruction after the processor has fetched the previous one.

(ii) Stack Pointer (SP):

- Stack is the area of Read/Write (RAM) memory used to hold data that will be retrieved soon.

- The beginning of stack is defined by loading a 16-bit address in the Stack Pointer (SP).

-The stack is usually accessed in First –In – Last –Out (FILO) basis or Last- In-First-Out (LIFO) basis.

- Stack writing instructions fill the memory position in progressively decreasing order.

- Instructions for Stack access: PUSH, POP

- During PUSH operation, higher order register content is pushed first.

E.g PUSH B, then B pushed and then C.

- Decrementing and Incrementing is always done by 2 bytes, since all the stack operations involve register pair.

3.) Timing and Control Unit:

- This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals.

- E.g. . $\overline{RD}$ And $\overline{WR}$ are control signals indicating the availability of data on the data bus.

(4.) Instruction Register and Decoder:

- When instruction is fetched from memory, it is loaded in the Instruction Register (IR).

- Then the decoder decodes the instruction and establishes the sequence of events to follow.

Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT

- The instruction register is not programmable and cannot be accessed through any instruction.

(5.) Interrupt Control:

- 8085 has 5 interrupt signals: TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR

- $\overline{INTA}$ is Interrupt Acknowledgement Signal.

- Valid interrupt may occur for at least 160 ns.

(6.) Serial I/O Control:

- The two 8085 pins: SID (Serial Input Data) and SOD (Serial Output Data) are specially designed for software controlled serial I/O.

- Data transfer is controlled through two instructions: SIM and RIM.

- These instructions are used for interrupt and serial I/O.

- SIM instruction is used to output data serially from the SOD line.

- RIM instruction is used to input serial data through the SID line.

- Bit D7 of RIM and SIM is separated for SID and SOD lines respectively.

8085 Bus Configuration:

- The 8085 microprocessor performs its functions, using three sets of communication lines, called buses:

(i) The address bus,

(ii) The data bus, and

(iii) The control bus.

(i) Address Bus:

- It is a group of 16 lines generally identified as A0 to A15.

- The address bus is unidirectional, i.e. bits flow in one direction from microprocessor to peripheral devices.

- Thus, MPU uses address bus to identify a peripheral or memory location.
- In a computer system, each peripheral or memory location is identified by a binary number, called an address, and the address bus is used to carry a 16-bit address.
- 16 address lines can address up to 2^{16} = 65,536 memory locations, or its memory addressing capability is 2^{16} = 64K .

(ii) Data Bus:

- It is a group of 8 lines used for data flow.
- These lines are bidirectional i.e. data flow in both directions between the MPU and memory and peripheral devices.
- MPU uses data bus to transfer information.
- The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF. (2^8 = 256) .
- The largest number that can appear on data bus is $(1111\ 1111)_2 = (FF)_H = (255)_{10}$.

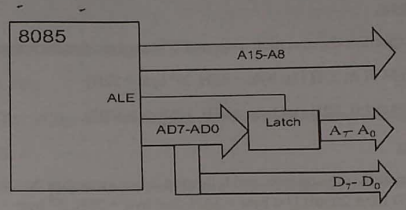
(iii) Control Bus:

- It is comprised of various signal lines that carry synchronization signals.
- MPU uses such lines to provide timing signals.

MPU generates specific control signals for every operation (such as Memory or I/O Read or Write) it performs

Demultiplexing AD7-AD0 in 8085 microprocessor:

- AD7-AD0 lines in 8085 microprocessor serve a dual purpose i.e A7-A0 as address bits and D7- D0 as data bits.
- So, they need to be demultiplexed to get all the information.
- The higher order bits of the address remain on the bus for 3 clock periods, but lower order bits remain only for 1 clock period and they would be lost if they are not saved externally.
- So to make them available for entire 3 clock period, the external latch is used and ALE signal is used to enable this latch.



- When ALE =1, the address can be latched, and when ALE = 0, the address is saved.

8085 INSTRUCTIONS

- An instruction is a command to the microprocessor to perform a given task, on specified data.
- Set of instructions is known as program.
- Each instruction has 2 parts: (i) opcode, (ii) operand
- Opcode (operation code) represents the task to be performed and Operand is the data to be operated on.

Instruction Word Size:

- 8085 instruction set is classified into the following 3 groups according to word size or byte size:

(1.) 1-byte instructions,

(2.) 2- byte instructions, and

(3.) 3- byte instructions.

(1.) 1-byte instructions:

- It includes the opcode and operand in the same byte.

e.gs. (i) MOV C, A [byte = 4FH]

(ii) ADD B [byte = 80H]

(iii) CMA [byte = 2FH]

(2.) 2-byte instruction:

- Here 1st byte specifies the operation code, and 2nd byte specifies the operand.

E.gs. (i) MVI A, 32H [1st byte = 3EH, 2nd byte = 32H]

(ii) MVI B, F2H [1st byte = 06H, 2nd byte = F2H]

(3.) 3-byte instruction:

- Here, 1st byte specifies the opcode, and following 2-bytes specifies the 16-bit address. E.gs. (i) LDA 2050H [1st byte = 3AH, 2nd byte = 50H, 3rd byte = 20H]

(ii) JMP 2085H [1st byte = C3H, 2nd byte = 85H, 3rd byte = 20H]

8085 Instruction Sets

Since the 8085 is an 8-bit device it can have up (256) instructions.

• However, the 8085 only uses 246 combinations that represent a total of 74 instructions.

– Most of the instructions have more than one format.

8085 instructions can be grouped into five different groups:

- Data Transfer (Copy) Instructions
- Arithmetic Instructions
- Logic and Bit Manipulation Instructions
- Branch Instructions
- Machine Control Instructions

1. **Data Transfer (Copy) Instructions:**

- These instructions perform the following six operations:

(i) Load an 8-bit number in a register.

(ii) Load 16-bit number in a register pair.

(iii) Copy from register to register.

(iv) Copy between register and memory.

(v) Copy between I/O and accumulator.

(vi) Copy between registers and stack memory.

Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT

They are:

(i) MVI Rd, data [Loads 8-bit data in a register.]

E.g. MVI B, 4FH

(ii) MOV Rd, Rs [Copy data from source register Rs into destination register Rd]

E.g. MOV B, A

(iii) LXI Rp, 16-bit [Loads 16-bit number in a defined registers pair.]

E.g. LXI B, 2050H

(iv) OUT 8-bit (port address) [Send (write) data byte from the accumulator to an output port /device.]

E.g. OUT 01H

(v) IN 8-bit (port address) [Accepts (reads) data byte from an input port/device and place it in the accumulator]

e.g. . IN 07H

(vi) LDA 16-bit memory address [Copy the data byte into A from the memory specified by 16-bit memory address]

e,g LDA 2050H

(vii) STA 16-bit memory address [Copy the data bye into memory specified by 16-bit address from A]

e.g. STA 2070H

(vii) LDAX Rp [Copy the data byte into A from the memory specified by the address in the register pair]

E.g. LDAX B

(viii) STAX Rp [Copy the data byte from A into the memory specified by the address in the register pair]

E.g. STAX D

(ix) MOV M, R [Copies the content of specified register to the memory location specified by HL pair]

e.g MOV M , B

(x) MOV R, M [Copies the content of the memory location specified by HL pair to the register specified.]

e.g. MOV R, D

(xi) SHLD address [Store H and L Register Direct]

e.g. SHLD 2050H (here, assume HL registers contain 01H and FFH respectively.

then after executing above instruction, 2050H = FFH and 2051H = 01H respectively.)

(xii) LHLD address [Load H and L Register Direct]

e.g. LHLD 2050H (here, assume 2050H = FFH and 2051H = 01H respectively then after executing above instruction, [H]= 01H and [L] = FFH respectively.)

(xiii) XCHG [Exchange contents of H and L with D and E respectively]

e.g. XCHG

2. Arithmetic Instructions:

The main arithmetic operations performed by arithmetic instructions are :

(i) Addition (ii) Subtraction, and (iii) Increment / Decrement [by 1]

- Addition & Subtraction are performed on any 8-bit number, or contents of register, or contents of memory location
- Added/subtracted to /from contents of accumulator and the result is stored in accumulator (A).
- Increment / Decrement can be performed on any 8 or 16 bit register content.

The different arithmetic instructions of 8085 microprocessor are:

(I) Addition:

(i) ADD R [Add the contents of specified register to the contents of accumulator (A) and store the final result in A.]

e.gs. ADD B [let A= 93H and B= B7H, now result after ADD B , in A = 14AH so Flags set, S=0, Z=0 and CY=1]

(ii) ADI 8-bit data [Add 8-bit data to the contents of A]

e.g. ADI 40H

(iii) ADD M [Add contents of memory to A, the address of memory is in HL register]

e.g. ADD M

(iv) ADC R [Adds the contents of specified register to A including carry]

e.g ADC B

(v) ADC M [Adds with carry the content of memory address stored in HL pair to the content of A]

e.g. ADC M

(vi) ACI 8-bit data [Adds with carry the 8-bit data in the instruction to A]

e.g. ACI 33H

(vii) DAD Rp [Adds contents of specified register pair to the content of HL pair i.e. used for 16 bit addition]

e,g, DAD B (suppose initially [HL] = 0242H and [BC] = 0123H)

after this instruction result in [HL] = 0242H + 0123H = 0365H

(ii) Subtraction:

(i) SUB R [Subtracts the content of specified register from the content of Accumulator (A) and store the result in A]

e.g. SUB B

(ii) SUB M

(iii) SUI 8-bit data

(iv) SBB R

(v) SBI 8-bit data

ADC R

(iii) Increment/Decrement:

(i) INR R ; Increment content of R by 1

(ii) INR M ; Increment content of HL pair by 1

(iii) INX Rp [e.g . INX B; increments the content of BC by 1]

(iv) DCR R ; Decrement content of R by 1

(v) DCR M ; Decrement content of HL pair by 1

(vi) DCX Rp [e.g. DCX B; decrements the content of BC by 1]

(iii) Logical and Bit Manipulation Instructions:

These instructions perform :

(i) AND, (ii) OR, (iii) X-OR (exclusive OR) (iv) Compare (v) Rotate Bits

- These instructions implicitly assume that the accumulator is one of the operand and place the final result in A.

- The various instructions available under this category are:

(i) ANA R ; Logically ANDs the content of specified register to the content of A and store the final result in A

(ii) ANI 8-bit data

(iii) ANA M

(iv) ORA R ; Logically ORs the content of specified register to A

(v) XRA R ; Exclusive ORs the content of R and A

(vi) XRI 8-bit data

(vii) XRA M

* Logical instructions are used for **masking** (hide certain bit) operation

E.gs. ANI 80H (Masks all other bits except MSB)

ANI 01H (Masks all other bits except LSB)

Compare Instructions:

(i) CMP R (ii) CPI 8-bit data (iii) CMP M

For these instructions:

If A>R , Cy and Z flags are reset.

If A<R , Cy is set and Z is reset.

If A= R , Z is set and Cy is reset.

(So, here we compare the data by checking the carry flags after executing these instructions.)

Complement(Logical NOT or does 1s complement):

(I) CMC (Complements Cy Flag)

(ii) CMA (Complements contents of Register A)

Other bit-manipulation instructions:

(i) RAL ; Rotate Accumulator Left through Carry- bit D7 is placed in the carry flag and carry flag is placed in LSB (9-bit rotation)

(ii) RAR ; Rotate Accumulator Right through Carry- bit D0 is placed in the carry flag and carry flag is placed in MSB (9-bit rotation)

(iii) RLC ; Rotate Accumulator Left – D7 bit is placed in D0 as well as in carry flag (8-bit rotation)

(iv) RRC ; Rotate Accumulator Right – D0 bit is placed in D7 as well as in carry flag (8-bit rotation)

Note: RLC can be used to multiply a number by 2 and RRC can be used to divide a number by 2.

(iv) **Branch Instructions:**

- These are used to change the program sequence.

Unconditional:

(i) JMP 16-bit address ;Unconditional jump to specified address

Conditional:

(i) JZ 16-bit address ; Jump to specified address if Zero Flag is set (if Z=1) or result is zero

(ii) JNZ 16-bit address ; Jump to specified address if Not Zero (if Z=0)

(iii) JC 16-bit address ; Jump to specified address if Carry (i.e.if carry flag , CY=1)

(iv) JNC 16-bit address ; Jump to specified address if Not Carry(i.e. if carry flag, CY =0)

(v) JP 16- bit address ; Jump to specified address on Positive (i.e. if sign flag, S=0)

(vi) JM 16-bit address ; Jump to specified address on Negative (i.e. if sign flag, S=1)

(vii) JPE 16-bit address ; Jump on even parity (i.e. if parity flag, P=1)

(viii) JPO 16-bit address; Jump on odd parity (i.e. if parity flag, P=0)

(ix) CALL 16-bit address; Subroutine call

(x) RET ; Return to main program from the subroutine.

(v) **Machine Control Instructions:**

(i) HLT ; Stop and wait for the next program

(ii) NOP ; No operation

Addressing Modes in 8085 Microprocessor:

- The various ways in which a processor can access data, during program execution are referred to as its addressing modes.
- In other word, the way in which the operand is specified is called its addressing modes.
- Best selection of addressing mode helps in reducing the computational complexity in any program.
- In 8085 microprocessor, different 5 types of addressing modes are available. They are:

(i.) **Immediate Addressing Mode:**

- The operand is the data present in the instruction i.e. the data is present in the instruction itself.

- Operands can be of 8 or 16-bits in size.

e.gs. MVI A, 33H ; Loads the 8-bit data 33H in Accumulator register.

LXI B, 2050H ; Loads the 16-bit data 2050H in BC register pair.

(ii.) **Direct Addressing Mode:**

- In this addressing mode, the effective address of memory or I/O port/device is defined in the instruction.

e.gs. LDA 2050H ; Loads the data stored in memory location 2050H to Accumulator

STA 2070H ; Stores the data in Accumulator to the memory location 2070H.

IN 20H ; Reads (Loads) the content of output port 20H to Accumulator.

OUT 30H ; Sends (Stores) the content of Accumulator to the output port 30H

(iii.) **Register Addressing Mode:**

- In this addressing mode, registers are used to define the data , i.e. both source and destination.

e.gs. MOV A, B ; Copies the content of Register B to Accumulator.

ADD D ; Adds the content of Accumulator and D register and store the result in Accumulator.

(iv.) **Register Indirect Addressing Mode:**

- In this addressing mode, the register pair are used to define the memory address of data.

e.gs. LDAX B ; Copy the data byte into A from the memory specified by the address in the register pair BC

STAX D; Copy the data byte from A to the memory specified by the address in the register pair DE

(v.) **Implied Addressing Mode:**

- In this addressing mode, the operand for the instruction is hidden and it is normally the Accumulator (A).

e.gs. CMA ; Complements (1s complement) the value of Accumulator

RRC ; Rotate Accumulator Right – D0 bit is placed in D7 as well as in carry flag

RLC ; Rotate Accumulator Left – D7 bit is placed in D0 as well as in carry flag

Register Transfer Language (RTL)

- Every digital modules in digital system are best defined by the registers they contain and the operations that are performed on the data stored in them.
- The operations executed on data stored in registers are called microoperations. The result of the operation may replace the previous binary information of a register or may be transferred to another register. Examples of microoperations are shift, count, clear, and load.
- It is possible to specify the sequence of microoperations in a computer by explaining every operation in words, but this procedure usually involves a lengthy descriptive explanation. So, we use symbolic notation to describe these microoperations.
- The symbolic notation used to describe the microperation transfers among registers is called a Register Transfer Language (RTL).
- The term 'Register Transfer' implies the availability of hardware logic circuits that can perform a stated microoperation and transfer the result of operation to the same or another register. And the 'Language' refers to programming language that denotes the procedure for writing symbols to specify a given computational process. Thus, a register transfer language is a system for expressing in symbolic form the microoperation sequences among the registers of a digital module.
- In RTL, the registers are generally denoted by capital letters and the information transfer from one register to another register is designated in symbolic form by means of a replacement operator. The general example of register transfer language can be as:

R1 ⟵ R2

This statement denotes a transfer (replacement) of content of register R2 into register R1 but the content of register R2 does not change after the transfer.

- Moreover, in RTL we can also show the predetermined control condition. For example:

P: R1 ⟵ R2

Here, P is a control signal generated in the control section. The above example specifies that the content of register R2 is only transferred to R1, if and only if P=1.

- We can also define the different instructions used in 8085 and 8086 microprocessors using RTL as follows:

MOV A, B ; A⟵ B
ADD B ; A⟵ [A] + [B]

Stack and Subroutine:

- The stack is a group of memory locations in a R/W memory that is used for temporary storage of binary information during execution of a program.
- The starting memory location of stack is defined in the main program, and space is reserved.
- The method of information storage resembles a stack of books.
- Stack works on the basis of rule Last-in- First- Out (LIFO) or First-in-Last-Out.
- The beginning of stack is defined in the program by using the instruction:

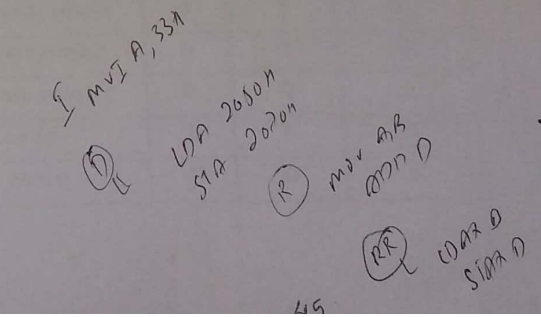
LXI SP, 16-bit address

which loads a 16-bit memory address in the stack pointer register of the microprocessor.

E.g. LXI SP, 2099H

Here, Stack Pointer(SP) register is loaded with the memory address 2099H and storing of data bytes begins from 2098H and continues in the decreasing order of memory address.

- PUSH and POP instructions are used to insert into and retrieve data from the stack
- Because 2 data bytes are being stored at a time, the 16-bit memory address in stack is decremented by 2 while storing the data byte and address is incremented by 2 while retrieving the data byte.



TIMING DIAGRAM 8085

- Timing Diagram is a graphical representation of instruction execution in steps with respect to the clock.
- The execution time is represented in T-states.
- The following control signals, status signals and buses must be shown in the Timing Diagram:

  - Higher Order Address Bus,

  -Lower Order Address Bus

  - ALE - $\overline{WR}$, - $\overline{RD}$, - IO/ $\overline{M}$, S1 and S0

T-state:

- Defined as one subdivision of operation performed in 1 clock period.
- 1 T-state is precisely equal to 1 clock period.

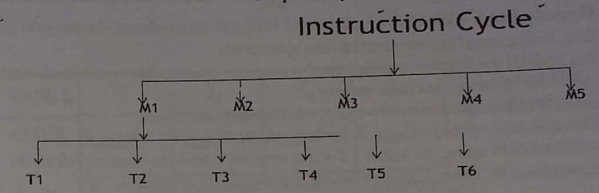
Machine Cycle:

- Time required to perform 1 operation may be opcode fetch, memory read, memory write or I/O read or I/O write is called a machine cycle.
- Consists of 4 to 6 T-states.

Instruction Cycle:

- Time required to execute an instruction is called an instruction cycle .
- In 8085 microprocessor, the instruction cycle consists of 1 to 5 machine cycles.

Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT



M_n –Machine Cycle

T_n –T State

Machine Cycles of 8085

- The 8085 Microprocessor has 5 basic machine cycle:

1. Opcode Fetch Cycle (4T- 6T states)
2. Memory read Cycle (3T state)
3. Memory Write Cycle (3T state)
4. I/O Read Cycle (3T state)
5. I/O Write Cycle (3T state)

Combination of Status Signals

Machine Cycle	IO/ $\overline{M}$	S_1	S_o
Opcode fetch	0	1	1
Memory read	0	1	0
Memory write	0	0	1
I/O read	1	1	0
I/O write	1	0	1

1. Opcode Fetch Machine Cycle

- Each instruction of the processor has one byte opcode.
- The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory.
- Hence, every instruction starts with opcode fetch machine cycle.
- The time taken by the processor to perform opcode fetch is 4-6 T states.
- When it has 4T states , the initial 3 T-states are used to fetch the instruction from the memory and 4th T-state is used to decode and execute the instruction.

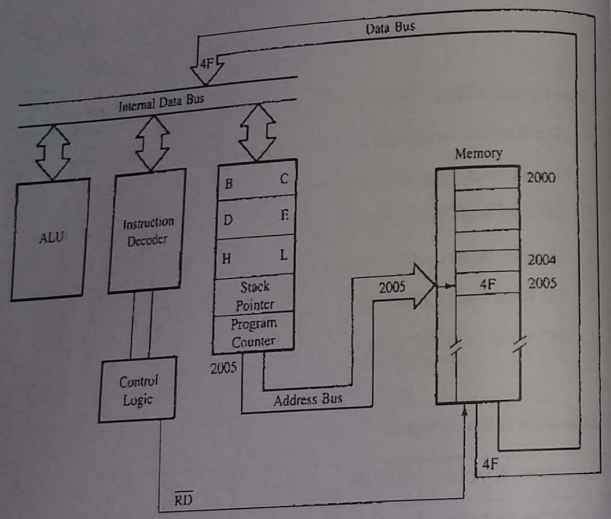
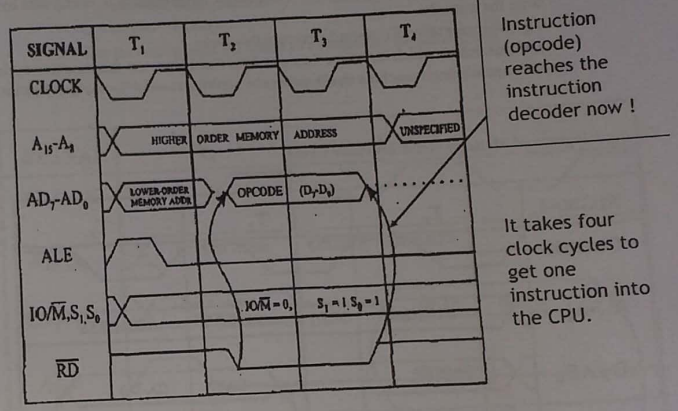


Fig. 2.5: Data Flow Diagram from Memory to MPU

Timing Diagram of Opcode Fetch Machine Cycle:



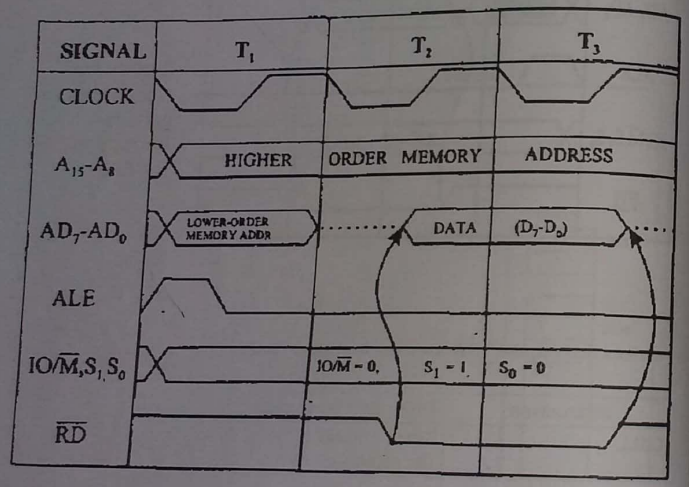
Instructions that require T5 & T6 states for Opcode Fetch:

Instruction	Operation performed
CALL	Stack pointer is decremented by 1
CALL conditional PUSH R_p	Stack pointer is decremented by 1
DCX R_p	Regester pair decremented by 1
INX R_p	Regester pair incremented by 1
PCHL	HL pair trancfered to PC
SPHL	HL pair trancfered to SP
RET conditional	The condition of flags checked

2. Memory Read Machine Cycle of 8085:

- The memory read machine cycle is executed by the processor to read a data byte from the memory.
- The processor takes 3T states to execute this cycle.
- The instruction which has more than one byte word size, use this machine cycle after opcode fetch machine cycle.

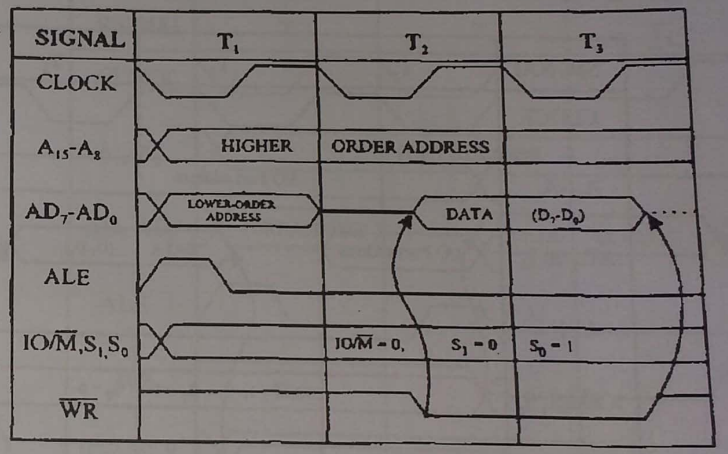
Timing Diagram of Memory Read Machine Cycle:



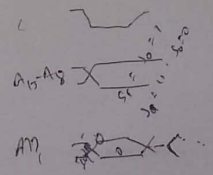
3. Memory Write Machine Cycle of 8085:

- The memory write machine cycle is executed by the processor to write a data byte in a memory location.
- The processor takes 3T states to execute this machine cycle.

Timing Diagram of Memory Write Machine Cycle



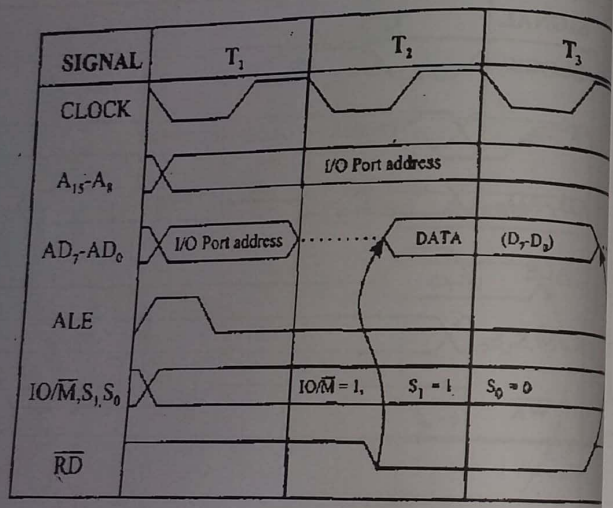
Note: In memory write machine cycle , there is no tri-state in T2 state.



4. I/O Read Machine Cycle of 8085:

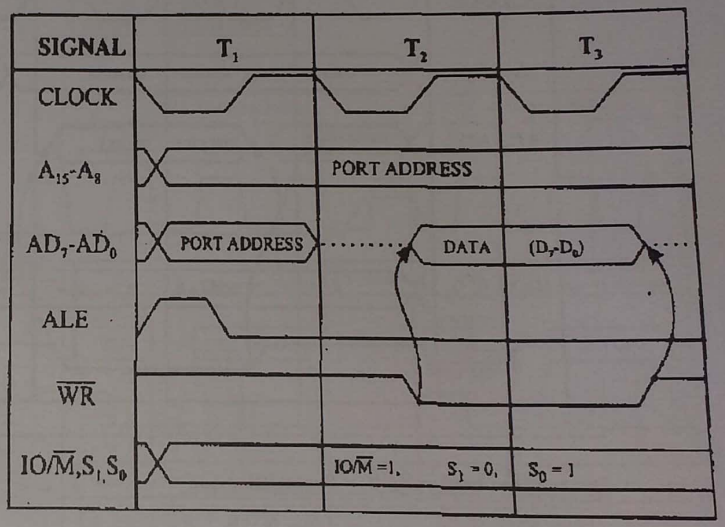
- The I/O Read Machine Cycle is executed by the processor to read a data byte from the I/O Port or from the peripheral, which is I/O mapped in the system.
- The processor takes 3T states to execute this task.
- The IN instruction uses this machine cycle during the execution.

Timing Diagram of I/O Read Machine Cycle:

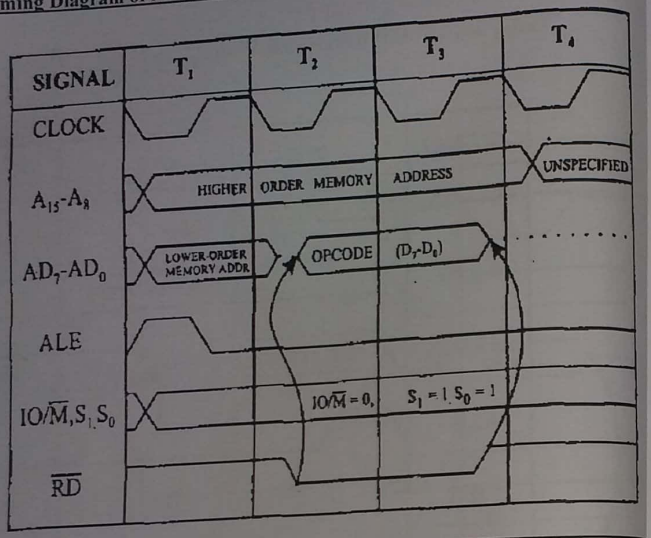


5. I/O Write Machine Cycle of 8085 (Used by OUT instruction)

Timing Diagram of I/O Write Machine Cycle:



Timing Diagram of MOV R1,R2



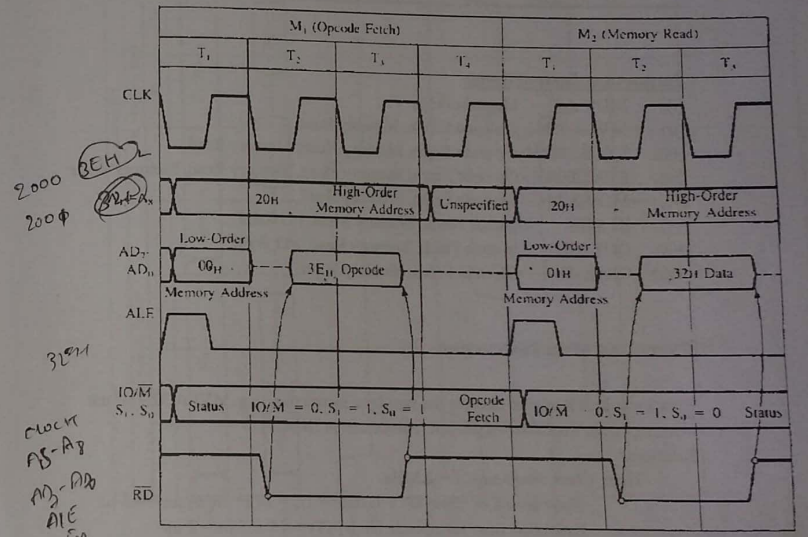
Instructions having similar Timing Diagram:

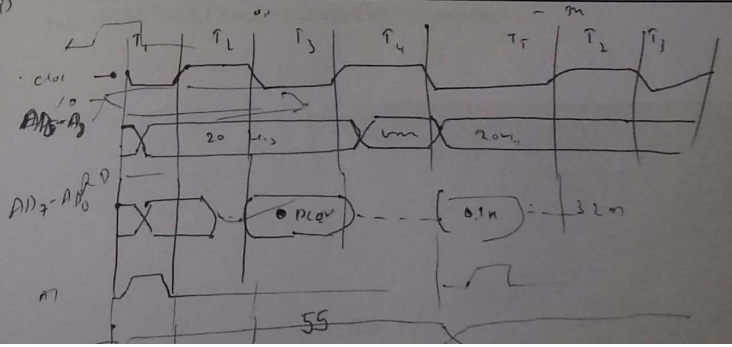
ALL MOV R1,R2	ALL ADD R
ALL ADC R	CMA,CMC,STC
ALL CMP R	ALL INR R
ALL DCR R	ALL ORA R,ANA R,XRA R
ALL ROTATE INSTRUCTIONS	ALL SUB R
ALL SBB R	RIM,SIM
DAA	EI,DI
NOP	XCHG

Timing Diagram of MVI A, 32H:

Assume Memory Address as :

2000H	3EH (Opcode of MVI A)
2001H	32H (data)





Instructions having Similar Timing Diagram like that of MVI A,32H:

ADI DATA	ANI DATA
ACI DATA	ORI DATA
SUI DATA	XRI DATA
SBI DATA	MVI R,DATA

Important Instructions:

(i). MOV A,C - Op code Fetch
(ii). MVI A,33H - Op code Fetch , Memory Read
(iii). LXI B,2065H- Op code Fetch, Memory Read, Memory Read
(iv). STA C00AH - Op code Fetch, Memory Read, Memory Read, Memory Write
(v). MOV A,M- Op code Fetch, Memory Read
(vi). IN 84H- Op code Fetch, Memory Read, I/O Read
(vii). OUT 80H- Op code Fetch, Memory Read, I/O Write
(viii). STAX B - Op code Fetch, Memory Write

Execution of an Instruction:

How long does it take to execute the two-byte instruction (e.g. MVI A,32H) if the working clock frequency of the microprocessor is 2MHz?

Solution:

Here, Clock frequency, f = 2 MHz

Time period or Time for 1 T-state = 1/f = 1/(2* 10^6) μs = 0.5 μs

Execution time for opcode fetch (4T) = 4 * 0.5 μs = 2 μs

Execution time for memory read (3T) = 3* 0.5 μs = 1.5 μs

Execution time for instruction = 2 μs + 1.5 μs = 3.5 μs

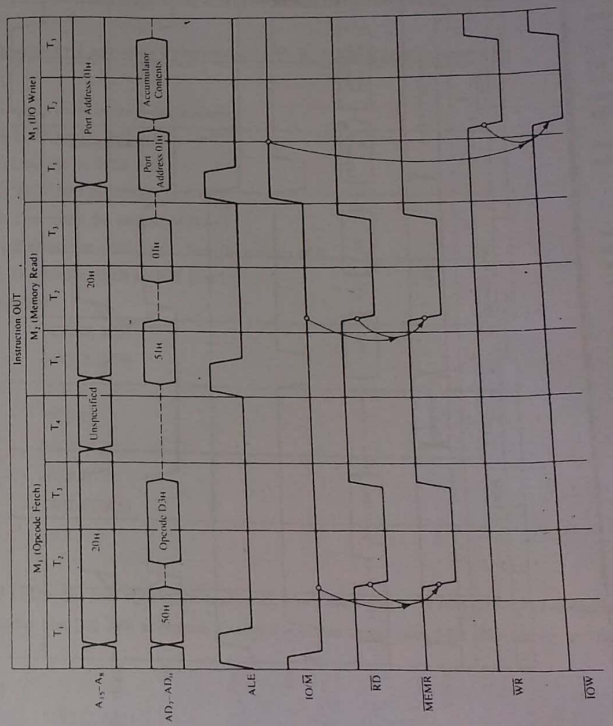


FIGURE 5.1
8085 Timing for Execution of OUT Instruction

data byte indicat-
ulator. Figure 5.2
) that of the OUT
of the input port
order address bus
ad ($\overline{IOR}$) signal.
aced on the data

)UT instruction;
) data flow from
output port.

esult out of the
ose; during the
cumulator con-
h that informa-
should we en-
s of that latch?
ne latch should
ss of an output
task is to gen-
e the presence
ing IO/$\overline{M}$ and
bus, and use
narized as fol-
lemultiplexed

evice address

rate a device
rted.

I/O device.
l generate a
is combined
able an out-

ith address
ons as a de-
e inverters.
erwise, the

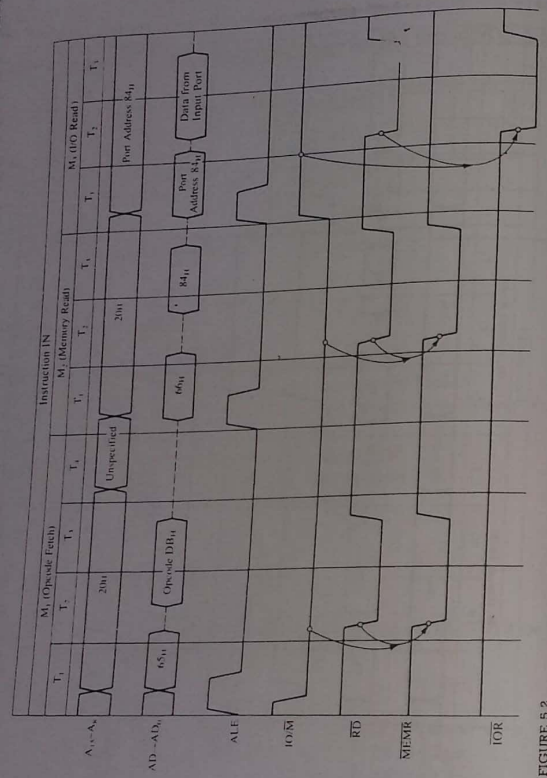


FIGURE 5.2
8085 Timing for Execution of IN Instruction

Assembly Language Programs(ALPs in 8085 Microprocessor):

1. Write a program to perform the following:
   a). Load the no: 1BH in D
   b) Load the no. B5H in B
   C). Increment the content of B by 1.
   d). Decrement the content of D by 1.
   e). Subtract the content of D from the content of B.
   f). Display the result at OUT port 1.

   Solution:

```
MVI D, 1BH
MVI B, B5H
INR B
DCR D
MOV A, B
SUB D
OUT PORT1
HLT
```

2. WAP to load the data byte in the register C. Mask the high- order bits (D7-D4) and display the low order bits (D3-D0) at output. Exclusive-OR the result with 57H and display at OUT PORT2.

Solution:

```
MVI C, A8H
MOV A, C
ANI OFH
OUT PORT1
XRI 57H
OUT PORT 2
HLT
```

3. WAP to load the byte 8EH in register D and F7H in register E. Mask the higher order bits (D7 – D4) from both the data bytes , EX-OR the low order bit (D3-D0) and display the answer.

Solution:

```
MVI D, 8EH
MOI A, D
ANI
MVI D, 8EH
MVI E, F7H
MOV A, D
ANI OFH
MOV D, A
MOV A, E
ANI OFH
XRA D
OUT PORT I
HLT
```

4. Write a program to load two unsigned numbers in register B and C respectively. Subtract C from B. If the result in 2's complement convert the result in absolute magnitude and display it port 1. Otherwise display the result.

Solution:

```
MVI B, byte 1
MVI C, byte 2
MOV A, B
SUB C
JNC label 1
CMA
ADI 01H
label1: OUT PORT1
```

Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT

```
HLT
```

5. Write an ALP to do the following:

a) Load A with byte 1.

b) Load B with byte 2.

c) Compare the equality of the contents of A and B

d) If two nos. are equal, display 01 otherwise display 00H at port 1.

Solution:

```
MVI A, byte 1
MVI B, byte 2
SUB B
JNZ loop
MVI A, 01H
OUT PORT 1
HLT
loop: MVI A, 00H
OUT PORT
HLT
```

6. The following block of data is stored in memory location from CO55 to C05A H. Transfer the entire block of data to the locations C080 to C085 H in reverse order.

Data: 22, A5, B2, 99, 7F, 37

Solution:

```
LXI H, C055 H
LXI D, C085 H
MOV B, 06H
next: MVI A, M
STAX D
INX H
DCX D
DCR B
```

```
JNZ next
HLT
```

7. Write a program to find larger of two nos. 1st no in C001 and 2nd no in C002 and result in C003 H.

Solution:

```
LXI H, C001 H
MOV A, M
INX H
CMP M
JNC loop
MOV A, M
loop: STA C003 H
HLT
```

8. Write an ALP to find the smallest no in a data array. Data from location C000H to C005 H.

Solution:

```
LXI H, C000H
MVI C, 06H
MOV A, M
DCR C
loop: INX H
CMP M
JC loop1
MOV A, M
loop1: DCR C
JNZ loop
STA C0C0 H
HLT
```


9. Write an ALP to multiply two nos: eg 05 H × 08 H

Solution:

```
MVI A, 00H
MVI B, 08H
MVI C, 05H
loop: ADD B
DCR C
JNZ loop
STA C000H
HLT
```

10. Write an ALP for the following addition.

$1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 + 9^2$

Solution:

```
MVI A, 00H
MVI B, 09H
loop1: MOV C, B
loop2: ADD B
DCR C
JNZ loop2
DCR B
JNZ loop 1
OUT PORT 1
HLT
```

11. Write an ALP to count the no. of 1 in the given string '10100110' and display the result at COCOH.

Solution:

```
MVI A, A6 H
MVI B, 00H
MVI C, 08H
loop1: RAL
JNC loop2
INR B
loop2: DCR C
JNZ loop 1
MOV A, B
STA COCOH
HLT
```

12. The following data are stored in memory location starting from C0B0 to C0B9 H . Take a test no. 48. Find out how many times the no 48 is repeated. Display the result at C0C0H .

DADA: 12, 23, 34, 45, 48, 56, 48, 67, 48, 89.

Solution:

```
LXI H, C0B0 H
MVI B, 00H
MVI C, 0A H
loop1: MOV A, M
CPI 48 H
JNZ loop2
INR B
loop2: INX H
DCR C
JNZ loop1
MOV A, B
```



```
STA COCO H
HLT
```

13. 8 bit multiplication, product is 16 bit. The multiplicand is loaded in the two consecutive memory locations 2501H and 2502H. The multiplier is stored in 2053 H. Store the product in 2504 and 2505 H.

Solution:

```
LHLD 2501 H
XCHG
LDA 2503 H
LXI H, 0000
MVI C, 08
loop1: DAD H
RAL
JNC loop2
DAD D
loop2: DCR C
JNZ loop1
SHLD 2504 H
HLT
```

14. Write an ALP to divide two nos. The dividend is in C001 and divisor is in C002. Store the quotient in C0C0 H and remainder in C0C1.

Solution:

```
LXI H, C001 H
MOV A, M
INX H
MOV B, M
MVI C, 00H
loop1: CMP B
JC loop2
INR C
SUB B
JNZ loop1
loop2: STA COC1 H
```

```
MOV A, C
STA COCO H
HLT
```

15. To arrange 54, EB, 85, A8 & 99 in descending order. These numbers are stored in the memory location 2501 to 2505 H. The count = 05 is restored in 2500 H. Results are to be stored in 2601 to 2605 H.

Solution:

```
LXI D, 2601 H
LXI H, 2500 H
MOV B, M
Start CALL Subroutine 1
STAX D
CALL Subroutine2
INX D
DCR B
JNZ start
HLT
Subroutine 1:
LXI H, 2500H
SUB A
Loop1: INX H
CMP M
JNC loop 2
MOV A, M
Loop2: DCR C
JNZ Loop 1
RET
Subroutin2:
LXI H, 2500H
MOV C, M
Loop1
```



```
INX H
CMP M
JZ Loop2
DCR C
JNZ Loop1
Loop2: MVI A, 00H
MOV M, A
RET
```

Note: Subroutine1 gives the largest number of array. Subroutine2 finds the largest number and replaces it by 00.

16. Write an ALP in 8085 to find whether the given number is palindrome or not.

Solution:

```
lxi h,1221h
mvi c,04h
mov a,l
again: rrc
       dcr c
       jnz again
       sub h
       jnz again1
       lxi h,ffffh
       hlt
again1: lxi h,0000h
        hlt
```

17. Write an ALP in 8085 to find the number of 1's in given 10 bytes.

Solution:

```
lxi h,2000h
lxi d,3000h
again1: mvi c,08h
        mvi b,00h
        mov a,m
rotate: ral
        jnc again
        inr b
again: dcr c
        jnz rotate
        mov a,b
        stax d
        inx h
        inx d
        mov a,l
        cpi 0ah
        jnz again1
        hlt
```

18. Write an ALP to divide two numbers: dividend is stored in memory location 2000H and divisor in memory location 2001H.

Solution:

```
;to divide the two number in hex
mvi e,00h
lda 2000h
mov b,a
lda 2001h
mov d,a
mov a,b
again1:sub d
       jc loop
       inr e
       jmp again1
loop:  adc d
       sta 3000h
       mov a,e
       sta 3001h
```



```
hlt
```

19. Write an ALP to find the successive Fibonacci series (up to 10^{th} term) and store the result starting from memory location 3000H.

Solution:

```
mvi a,0
mvi b,1
lxi h,3000
mvi c,0ah

loop:  mov m,a
       inx h
       add b
       mov d,a
       mov a,b
       mov b,d
       dcr c
       jnz loop
       hlt
```

20. Write an ALP to subtract two 16-bit numbers stored at memory locations starting from 2000H and 2003H and store the result at memory locations starting from 200AH.

Solution:

```
LHLD 2000H
XCHG
LHLD 2003H
MOV A,L
SUB E
MOV L,A
MOV A,H
SBB D
MOV H,A
SHLD 200AH
HLT
```

21. Write an ALP in 8085 to add two 32-bit numbers

Note: 4 memory locations are required to store each byte of 32-bit data and we assume two starting memory locations as 2000H and 3000H for adding them

LXI H, 2000H; starting address of first 32-bit data

LXI B, 3000H; starting address of second 32-bit data

LXI D, 4000H; starting address to store added result

MOV C, 04H; loop counter

addnextbyte :LDAX B; load data byte from memory pointed by BC pair to

; accumulator

MOV B, M; load the data byte stored in memory location pointed by HL pair ;to B

ADC B; Adds the content of A and B i.e. first byte is added

STA X D; store the result in memory locations pointer by DE pair

INX H

INX B

INX D

DCR C

JNZ addnextbyte

HLT

22. Write and ALP in 8085 to find the square root of a number.

Hint:

Initializa two registers one with initial value 00H (say, C) and other with 01H (say, E), load the number from memory location to find square root to A, start the loop to subtract E from A and increment its value by 2, and increment C, compare E with A , repeat the loop till E equals A. When E equals A, C contains the desired square root of a number.

Solution:

```
LXI H, 2000H
MOV A, M    ; Load the number 2000H to Accumulator to find its square root
MVI C. 00H  ; Clear C to store square root
MVI E, 01H  ; Load 01H in E to compare
Loop: SUB E ; Subtract E from A to check whether A equals E
      INR E
      INR E
      INR C
      CPI 00H; Check whether after subtraction result is 0,i.e either E equals A
      JNZ Loop; if E is not equal to A, jump to Loop
      MOV A, C
      STA 2001H ; Store result or square root of a number stored in 2000H in 2001H
      HLT
```

Code Conversion Programs:

BCD TO BINARY Conversion:

The conversion of a BCD number into its binary equivalent employs the principle of *positional weighting* in the given number:

For example: $72_{10} = 7 \times 10 + 2$

The digit 7 represents 70,based on its second position from the right. Therefore, converting 72_{BCD} into its binary equivalent requires multiplying the second digit by 10 and adding the first digit.

Algorithm:

1) Separate an 8-bit packed BCD number into two 4-bit unpacked BCD digits: BCD1 and BCD2
2) Convert each digit into its binary value according to its position.
3) Add both binary numbers to obtain the binary equivalent of the BCD number.

Program:

```
LDA 2000H; Load BCD number from memory location 2000H
MOV B, A; Store in B for future use
ANI OFH; Mask most significant four bits
MOV C, A;  Save unpacked BCD1 in register C
MOV A, B ; Get BCD again
ANI FOH ;  Mask least significant four bits
RRC ; Rotate right and convert most significant four bits to least significant four bits as BCD2
RRC
RRC
RRC
MOV B, A; Save unpacked BCD2 in B register
XRA A ; Clear accumulator to store sum
MVI D, OAH ; Set D as a multiplier of 10
```



```
SUM: ADD D; Add 10 until B=0
     DCR B
     JNZ SUM
     ADD C; Add BCD1
     STA 2001H ; Store converted binary equivalent or result in memory location 2001H
     HLT       ; Terminates the program
```

BINARY TO BCD Conversion:

The conversion of binary to BCD is performed by dividing the number by the powers of ten; the division is performed by subtraction method.

For example, assume the binary number:

$1111\ 1111_2 = FFH = 255_{10}$

To represent this number in BCD requires twelve bits or three BCD digits, labeled here as BCD_3 (MSB), BCD_2, and BCD_1 (LSB)

= 0010 0101 0101 (equivalent of 2, 5 and 5)

BCD_3 BCD_2 BCD_1

The conversion can be performed as follows:

1. If the number is less than 100, goto step 2, otherwise divide by 100 or subtract 100 repeatedly until the remainder is less than 100. The quotient is the most significant BCD digit, BCD3.
2. If the number is less than 10, goto step 3, otherwise divide by 10 repeatedly until the remainder is less than 10. The quotient is BCD2.
3. The remainder from step2 is BCD1.

Program:

```
LXI SP, 27FH          : Initialize stack pointer
LDA 6000H             : Get the binary number in accumulator
CALL SUBROUTINE       : Call subroutine
HLT                   : Terminate program execution
```

;Subroutine to convert binary number into its equivalent BCD number:

```
SUBROUTINE:
    PUSH B          : Save BC register pair contents
    PUSH D          : Save DE register pair contents
    MVI B, 64H        : Load divisor decimal 100 in B register
    MVI C, 0AH        : Load divisor decimal 10 in C register
    MVI D, 00H      : Initialize Digit 1
    MVI E, 00H      : Initialize Digit 2
STEP1: CMP B        : Check if number < Decimal 100
    JC STEP 2       : if yes go to step 2
    SUB B           : Subtract decimal 100
    INR E           : update quotient
    JMP STEP 1        : go to step 1
STEP2: CMP C        : Check if number < Decimal 10
    JC STEP 3       : if yes go to step 3
    SUB C           : Subtract decimal 10
    INR D           : Update quotient
    JMP STEP 2        : Continue division by 10
STEP3: STA 6100H        : Store Digit 0
    MOV A, D        : Get Digit 1
    STA 6101H       : Store Digit 1
    MOV A, E        : Get Digit 2
    STA 6102H       : Store Digit 2
    POP D       : Restore DE register pair
    POP B           : Restore BC register pair
    RET         : Return to main program
```


BINARY TO ASCII Conversion:

Program:

```
LXI H, C000H ; Point the address to which binary byte is stored
LXI D, C00AH ; Point the address to which converted ASCII value need to be stored
MOV A, M ; transfer byte
MOV B, A ; Save Byte
RRC    ; Shifts high-order nibble to the position of low-order nibble
RRC
RRC
RRC
CALL ASCII ; Call conversion routine
STAX D ; Store first ASCII Hex in COOAH
INX D  ; Point to next memory location, get ready to store next byte
MOV A, B ; Get number again for second digit
CALL ASCII
STAX D
HLT
ASCII:  This subroutine converts a binary digit between 0 to F to ASCII Hex code
ANI OFH ; Mask high-order nibble
CPI OAH ; Is digit less than 10?
JC CODE ; If digit is less than 10 goto CODE to add 30H
ADI 07H ; Add 7H to obtain code for digits from A to F.
CODE: ADI 30H ; Add base number 30H
RET    ; Return to main program
```

ASCII TO HEX Conversion:

Algorithm

1) Load the ASCII number in to the accumulator.
2) Subtract 30_H from the number.
3) Compare the number with 0A.
4) If number < 0A then store result and exit else go to step 5.
5) Subtract 07_H from the number and store result.
6) Terminate the program.

Program:

```
LDA 2000H
SBI 30H
CPI OAH
JC AHEAD
SBI 07H
AHEAD: STA 2001H
HLT
```

Microprocessors- Chapter-2
Prepared By: Er. Ramu Pandey, Asst. Prof. , NCIT

Counter and delay:

Timing delay using one register:

MVI C, FFH ---------- 7 T state

Loop DCR C ---------- 4 T state

JNZ Loop -------- 10 or 7 T state

Consider a micro computer with 2 MHZ frequency

Clock period, $T = 1/f = ½ = 0.5\ \mu$ sec

Delay for inst. Outside the loop, T_0 = No of T state × T

$= 7 \times 0.5$

$= 3.5\ \mu$ sec

Delay for inst inside the loop, T_L = No of T state × T * (N_{10})

$= (14\times 0.5 \times 10\text{-}6\times 255)$

$= 1785\ \mu$sec

Now, $T_{LA} = T_L\text{-}3\times 0.5$

$= 1785 - 1.5$

$= 1.7835$ ms

$T_D = T_0 + T_L$

$= 3.5 + 1785$

$= 1788.5$

$= 1.7885$ ms

Time delay for register pair:

LXI B, 2384 H ..... 10 T state

Loop DCX B ..... 6 T state

MOV A, C ..... 4 T state

ORA B ..... 4 T state

JNZ loop 10/7

Delay calculation:

T_0 = T state * T
= 10 × 0.5 × 10-3
= 109 ms
T_L= No of T state * T * (N_{10})
= 24 * 0.5 * 90.92
Total Delay = T0+ TL

Time delay using a loop within a loop:

MVI B, 38 H …….. 7 T
Loop2 MVI C, FFH ……7 T
Loop 1 DCR C……..4 T
JNZ loop1 ------10/7 T
DCR B ------ 4 T
JNZ loop2 ------ 10/7 T

Delay calculation:

To= 7 * 0.5 = 3.5 μ sec
T_{L1}= 14 * 0.5 * 255 – 3*0.5
T_{L2}= (T_{L1}+ 21*0.5) N_{10}– 3
Total delay = To+ T_{L2}

Q. Write a program to count continuously in hexadecimal from FFH to 00H in a system with a 0.5 μ s clock period. Use registers C to set up a 1 ms delay between each count and display the number at one of the o/p ports.

Solution:

MVI B, 00H
Next: DCR B
MVI C, count
Delay: DCR C
JNZ delay

MOV A, B
OUT port 1
JMP Next

Delay calculation:

T_L= T state * T* (T_{10})
= 14 × 0.5 × count
= 7 × count μs
To= 35 × T
= 35 × 0.5
= 17.5 μ s
T_D= T_L + To
1 ms = 7 * count * 10-6+ 17.5 * 10-6
Count = 140.35
= 8C H

Q. Write a program to generate a continuous square wave with the period 500 μ sec. Assume the system clock period is 325 μ sec and use bit Do to output the square wave.

Solution:

MVI D, AA
ROTATE: MOV A, D
RLC
MOV D, A
ANI 01 H
OUT PORT 1
MVI B, count
delay: DCR B
JNZ delay
JMP ROTATE

Delay calculation:

T_L= 14 * 325 * 10-9* count or 14 * 325 * (count - 1) + 11 T-state * 325

T0= 46 * 325 * 10-9

T_D= T_L+ T_O

250 = (52.4)10= 34 H