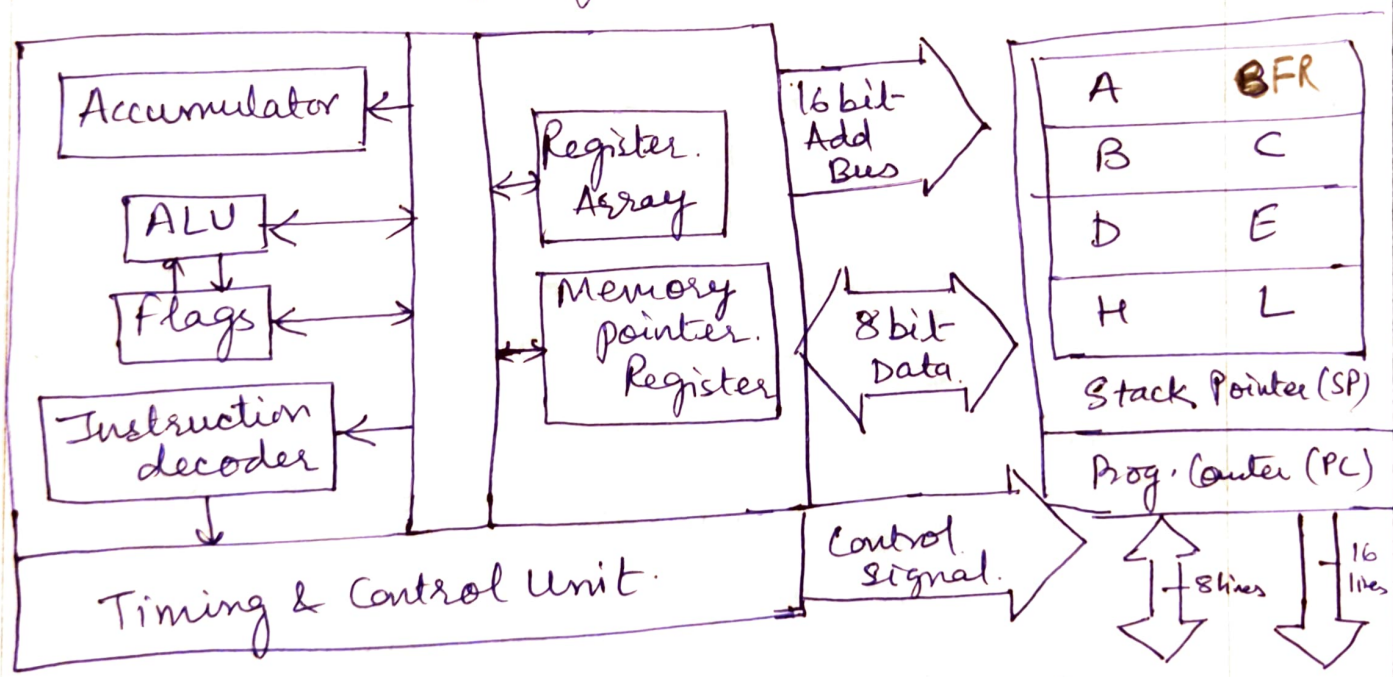
UNIT #02

PAGE NO. .....

## Software Architecture, Register and Signals:

### 8085 Programming Model →

A model is the conceptual representation of the real object. It can take many form. The following figure shows the programming model and hardware specified model -




Programming model of 8085

## Instruction set classification of 8085 -

### 1) DATA TRANSFER INSTRUCTION :-

a) MOV : Rd , Rs
M , Rs
Rs , M
} copy from the source to the destination.

b) MVI : Rd , data
M , data
} move immediate 8-bit data.

c) LDA : 16-bit Address - The content of memory location specified by the 16-bit Address.

d) LXI $R_p$ : 16 bit data - load 16 bit data in $R_p$.

e) LHLD : 16 bit Address - copies the content of ~~load 16 bit data in~~ memory location pointed out by the address into Reg. L and copy the content of next memory location into H.

f) STA 16 bit Address - The content of the accumulator are copied into the memory location.

g) SHLD 16 bit address - The content of L are stored in the memory location specified by 16 bit address and content of H. one placed in next memory location.

h) STAX 16 bit address – The content of A one copied into the memory location specified by the content of the operand.

i) XCHG – (Exchange H and L with DE), H one exchanged with the register D, content of Register L exchanged with Register E.

j) SPHL – Copy H & L to the Stack pointer Register. The content of H & L loaded into the SP register.

k) XTHL – Exchange H & L with top of Stack. The content of L Register exchanged with the stack location pointed by content of SP Register.

l) PUSH Rp (Push the Rp into the Stack) – The SP register is decremented and the content of high order register (B, D, H, A) are copied into that location.

The Stack pointer (SP) Register is decremented again the content of low order Register one Copied in that location.

m) POP Rp – The content of memory pointer pointed out by the SP Register are copied to the low order register of the operand. The Stack Pointer is incremented by 1 and the content of the memory location are copied to the high order Register.

n) OUT 8 bit port – The content of accumulator one copied into the I/O port specified by the operand.

OUT 01H.

o) IN 8 bit Port – The content of the accumulator are copied into the I/O port specified by the operand.

IN 04 H.

PAGE NO. .....

## 2) ARITHMETIC INSTRUCTION :-

a) ADD R / M ] The content of Register or memory are added to the content of the A, and result stored in A.

b) ADC R / M ] The content of R and M & the carry flag are added & result is stored in the A.

c) ADI 8 bit data — The 8 bit data is added to the content of the A, and result stored in the accumulator.

d) ACI 8 bit data — Add the immediate value with accumulator and carry.

e) LXI Rp, 16 bit data — It store 16 bit data into the Register pair designed as per the Operand.

f) DAD Rp — Add the Rp to H and L Register.

g) SUB R / M ] - Subtract the register or the memory from the A.

h) SBB R / M ] - Subtract the source and borrow from the A

i) SUI 8 bit data - Subtract the immediate value from the A

j) SBI 8 bit data - Subtract the immediate from the A with borrow.

K) INR R / M ] - Increment the Register or memory by one.

l) INX R - Increment Register pair by 1

m) DCR R / M ] - Decrement the Register or memory location by 1.

n) DCX R - Decrement Register pair by 1

o) DAA - (Decimal adjust accumulator) -

* The content of A are changed from a binary value to two 4-bit BCD digit. If the value of low order 4 bit in the A is greater than 9 or if the AC flag is set. The ins add 6 to the low order and high order four bits.

PAGE NO. .....

## 3) LOGICAL INSTRUCTION :-

1) CMP R / M ] - compare the Register or memory with the A

2. CPI 8 bit data — Compare immediate data with the A.

3. ANA R / M ] - Logical AND Register or memory with the A

4. ANI 8 bit data - Logical AND immediate with the A.

5. XRA R / M ] - EX-OR Register or memory with the A.

6. XRI, 8 bit data - XOR immediate with the accumulator

7. ORA R / M ] - Logical OR ~~or~~ register and memory with the A

8. ORI 8 bit data — Logical OR immediate with the accumulator.

9. CMA – (Complement the accumulator), no flags are affected.

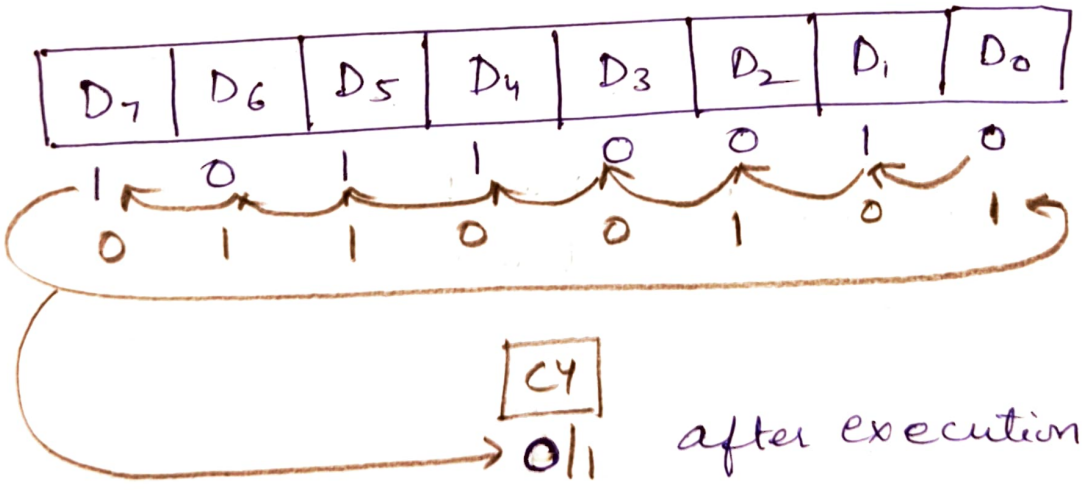
10. CMC (Complement Carry) – the carry flag is complemented, no other flags are affected.

11. STC (Set Carry) – Set the carry flag.

12. Rotate Instruction –

a) RLC (Rotate accumulator Left)

Each binary bit of the accumulator is rotate left by one position Bit $D_7$ is placed in the position of $D_0$ as well as the carry flag. Carry flag is modified according to the bit $D_7$.

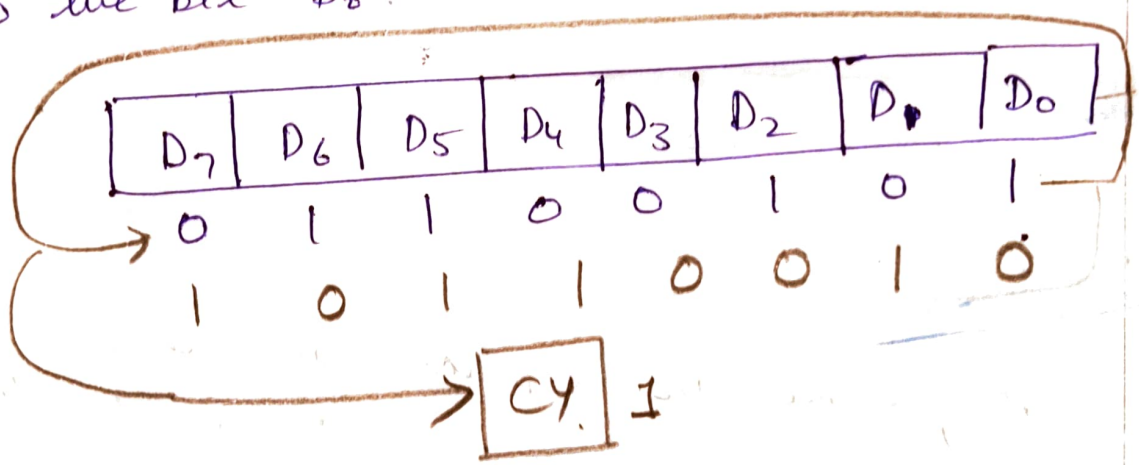



After RLC

PAGE NO. .....

## b) RRC (Rotate accumulator Right)

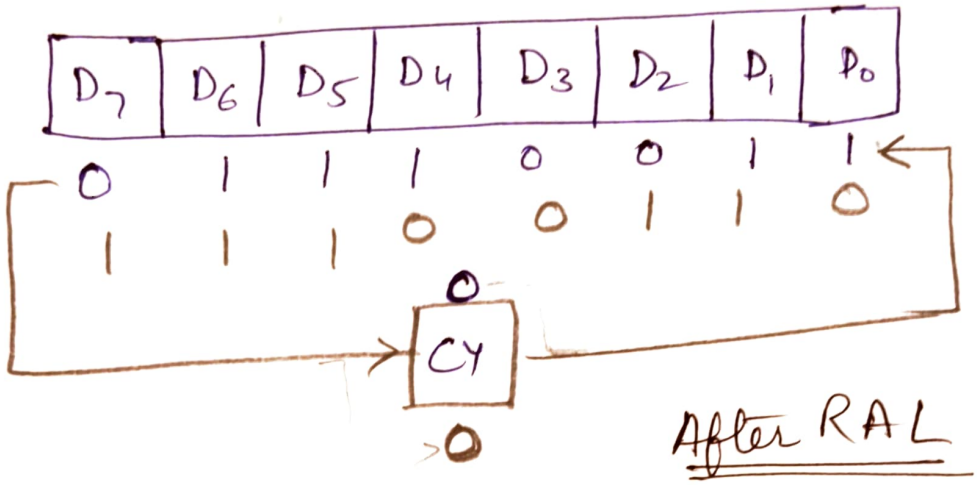
Each binary bit of the accumulator is rotated right by one position. Bit $D_0$ is placed in the position of $D_7$ as well as in the carry flag. $C_y$ is modified according to the bit $D_0$.




After RRC

## c) RAL (Rotate accumulator left through carry)

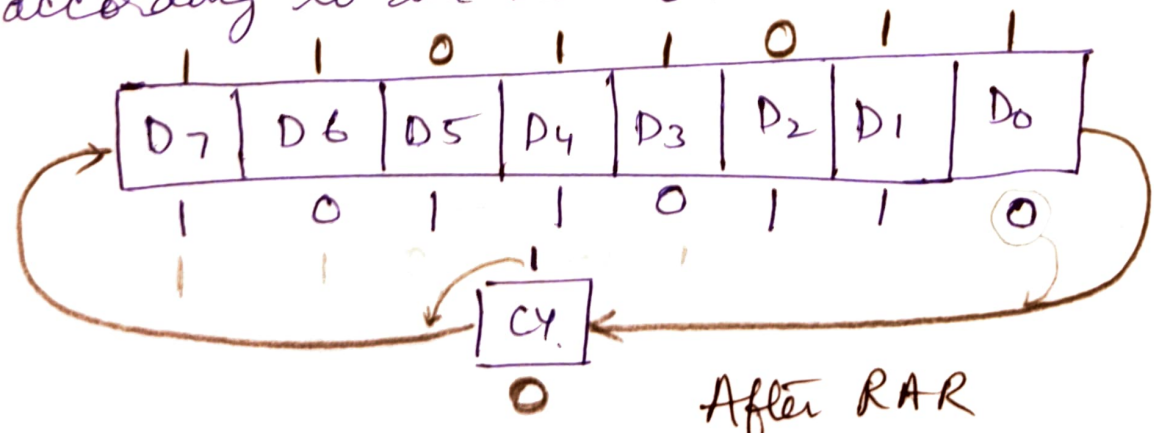
Accumulator is rotated left by one position through the carry flag. Bit $D_7$ is placed in the carry flag & the carry flag is placed in the $D_0$. CY flag is modified according to $D_7$ bit.




After RAL

## d) RAR (Rotate accumulator right through carry)

Each binary bit of the accumulator is rotated right by one position through the carry flag. Bit $D_0$ is placed in the carry flag & the carry flag is placed in the most significant position $D_7$. CY is modified according to the bit $D_0$.




After RAR

## (4) CONTROL INSTRUCTIONS :-

1) NOP (No operation) - No operation is performed.
2) HLT (Halt & enter wait state) - The CPU finish the execution of the current instruction & stops further execution An interrupt or Reset is necessary for the exit from the halt state (unaffected)
3) DI (Disable Interrupt) - The interrupt-enable flip flop is Reset and all the interrupts are disabled.
4) EI (Enable Interrupt) - The interrupt-enable flip flop is set and all the interrupts are enabled.
5) RIM (Read Interrupt Mask) - It is used to read the status of interrupt-7.5, 6.5 and 5.5 & read serial data input bit
6) SIM (Set Interrupt Mask) - The instruction used to implement the interrupt 7.5, 6.5, 5.5 and Serial data output.

## 5) BRANCH INSTRUCTIONS :-

### (A) Jump

a) JMP, 16 bit address — Jump to the memory location unconditionally.

b) Conditional Jump, 16 bit address –

| | | |
|---|---|---|
| JC | Jump on Carry | Cy = 1 |
| JNC | Jump on not carry | Cy = 0 |
| JP | Jump on Positive | S = 0 |
| JM | Jump on minus | S = 1 |
| JZ | Jump on zero | Z = 1 |
| JNZ | Jump on no zero | Z = 0 |
| JPE | Jump on parity even | P = 1 |
| JPO | Jump on parity odd | P = 0 |

### B) Call Instruction –

a) Call, 16 bit address (unconditional call)

b) Conditional Call, 16 bit address

| | | |
|---|---|---|
| CC | Call on Carry | Cy = 1 |
| CNC | Call on no carry | Cy = 0 |
| CP | Call on Plus | S = 0 |
| CM | Call on minus | S = 1 |
| CZ | Call on zero | Z = 1 |
| CNZ | Call on non zero | Z = 0 |
| CPE | Call on even Parity | P = 1 |
| CPO | Call on odd Parity | P = 0 |

## (c) RET. (Return).

a) RET, 16 bit address — Return from the subroutine

b) Conconditional RET instructions.

| | | |
|---|---|---|
| RC | Return on carry | CY=1 |
| RNC | Return on no carry | CY=0 |
| RP | Return on Plus | S=0 |
| RM | Return on minus | S=1 |
| RZ | Return on zero | Z=1 |
| RNZ | Return on no zero | Z=0 |
| RPE | Return on Parity even | P=1 |
| RPO | Return on odd Parity | P=0. |

c) PCHL: Load Program counter with the HL content.

d) RST (0-7): Restart from the location specified by the memory location.

NOP - usually used for delay or to replace instructions during debugging

PUSH Rp → memory location pointed by SP

SP

d) PCHL – Load the program counter with the HL content.

e) RST (0-7) – Restart from the location specified by the memory location.

## Types of Instruction (As per the byte):

1. **1-byte Instructions**: Those instructions required only 1 memory address for execution. for example –

```
ADD B
MOV A, C
INR R
```

2. **2-byte Instructions**: Those instruction required 2 memory address for execution.

For example –

```
MVI C, 00H
IN 18H
```

3. **3-byte Instructions**: Those instructions which required 3 memory address for execution counted as 3 byte instructions.

For example –

```
LHLD, 2050H.
STA, 3052H.
```

# Addressing Modes of 8085

The way of specifying data to be operand an instruction is called addressing modes.

## Types of Addressing Mode -

There are 5 type of Addressing mode.

### 1) Immediate Addressing mode:

In immediate addressing mode the source operand is always data. If the data is 8-bit, than the instruction will be of 2 bytes, if the data is of 16 bit then the instruction will be of 3 byte.

Example - MVI B, 45H.

LXH H, 3050H.

## 2) Register Addressing Mode :

In Register Addressing mode, the data to operated is available inside the Register and register is operand.

The operation is performed within various register of the microprocessor.

Example - MOV A, B
ADD B
INR A

## 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.

Example - LDA 2050
LHLD 3000

## 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 the Register Pair.

Example- MOV A, M

(Move the content of memory location pointed by the H, L pair to the accumulator)

So, LDAX B

(Move content of B-C Register to the accumulator)

## 5) Implied / Implicit Addressing Mode:

In this instruction the operand is hidden and the data to be operated is available in the instruction itself.

Example — CMA (Complement content of the accumulator)

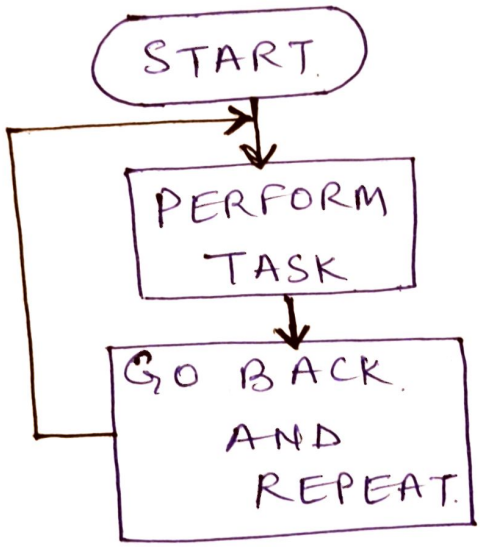
RRC (Rotate accumulator right by one bit)

## Programming Technique :

The programming technique used to instruct the microprocesor to repeat task is called **Looping**.

Loops described in the 2 sections -

1) **Continuous Loop** - it is set by unconditional Jump instruction as per shown in the flow chart.



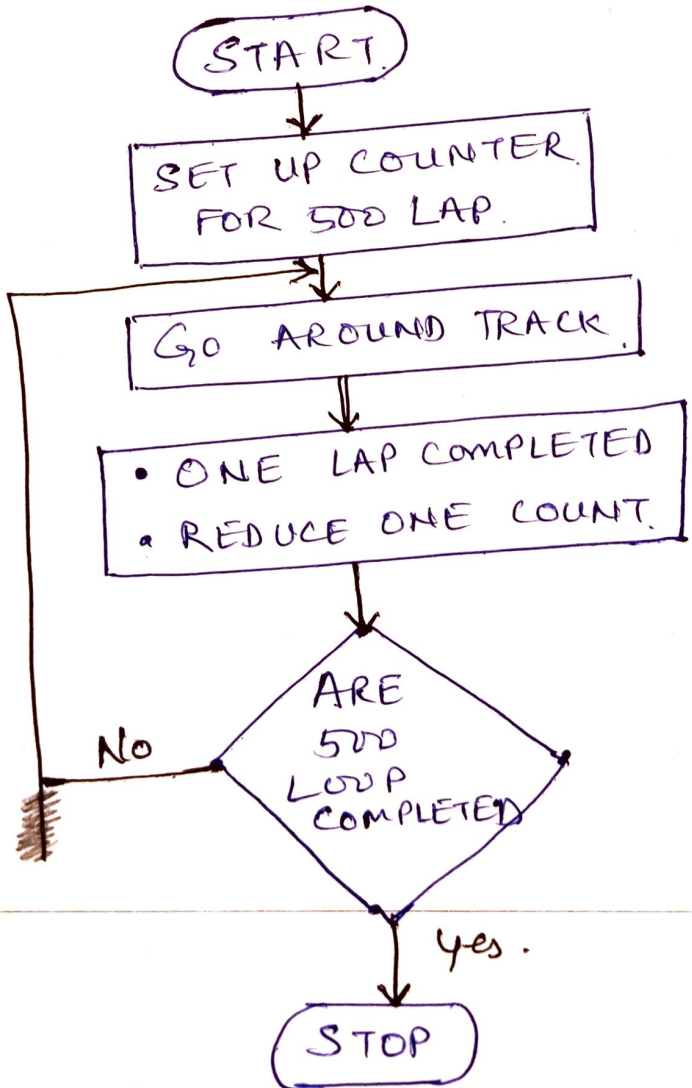

Continuous Loop

A program with continuous loop does not stop repeating the task until the system is Reset.

PAGE NO. .....

## 2) Conditional loop -

It is set by the conditional Jump instruction. These instruction check carry flag (zero/any) and Repeat the specified task if the condition are satisfied.




Laps in the loop Race

- Counter is set up by loading an appropriate count in a register.
- Counting is performed by either decrementing or incrementing the counter.
- Loop is set up by the conditional JUMP instruction.
- End of count is indicated by the flag.

## Conditional loop, Counter and Indexing:

Another type of loop includes indexing along with a counter. Data types are stored in memory locations and those data bytes referred to their memory locations.

PAGE NO. .....

## Program Addition with Carry :-

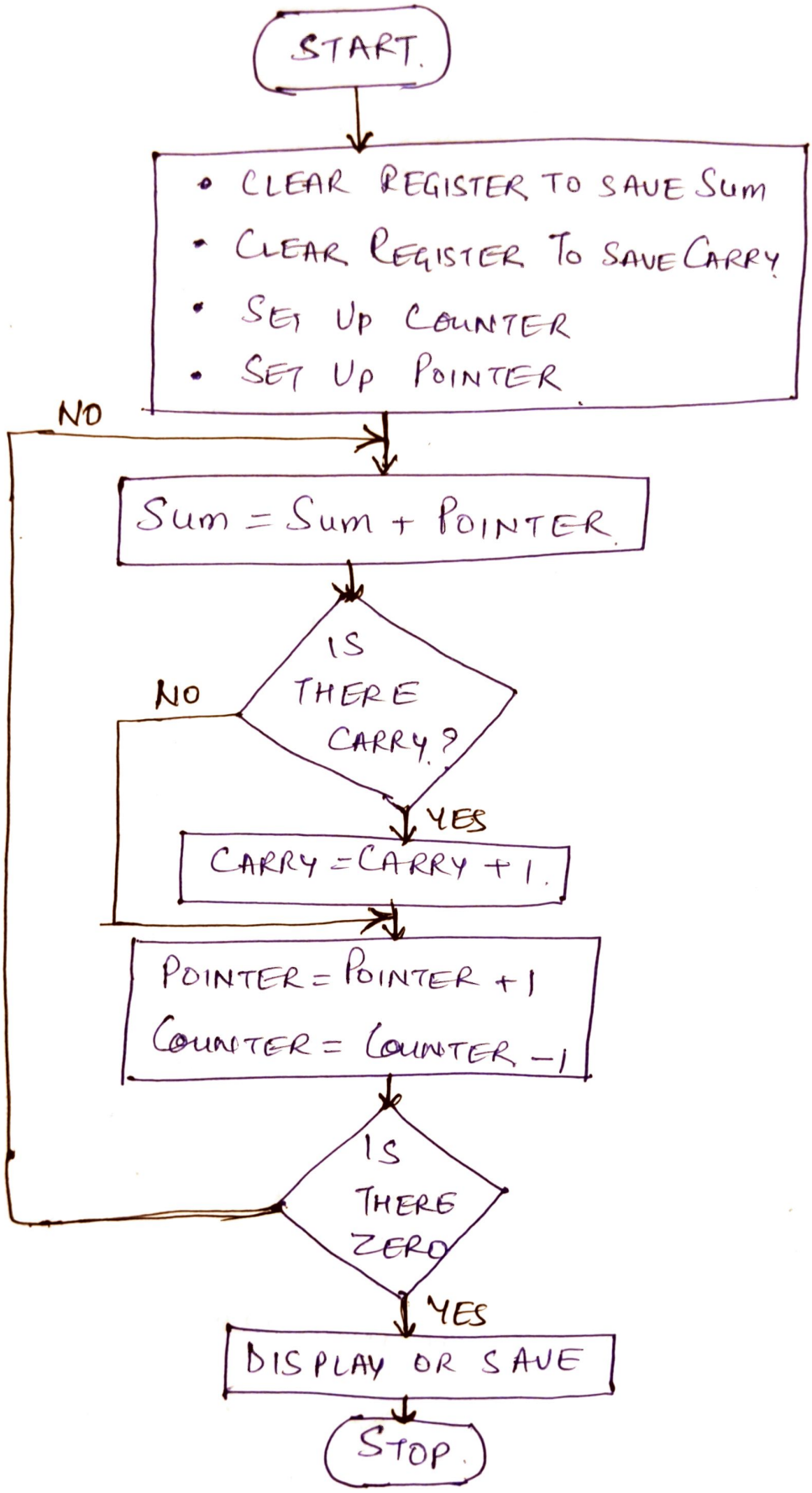
Six byte of data are stored to memory location. Add all the data byte. Use B register to save any carry generated while adding the data byte. Display the entire sum at two O/P port, or store the sum at two consecutive memory location XX70 and XX71 H.

## Problem Analysis :-

→ Memory related instruction so two blocks are used in the general flow chart, data acquistion and data processing can be combined in one instruction.

→ The 4th block - temporary storage of partial. Result - is unneccessary because the sum can be stored in the accumulator.

→ The data processing blocks needs to be expanded to account for carry.




Flow chart for addition with carry.

PAGE NO. .....

| | Opcode | Operand | Comment |
|---|---|---|---|
| | XRA | A | Clear to save sum. |
| | MOV | B, A | Clear to save carry |
| | MVI | C, 06H | Set up register C, as the Counter. |
| | LXI H | , XX05H | Set up HL as memory pointer. |
| LOOP: | ADD | M | Add byte to memory |
| | JNC | NEXT | If no carry do not increment the carry flag |
| | INR | B | If carry save carry bit |
| NEXT: | INX | H | Point to next memory location. |
| | DCR | C | One addition is couple decrement counter. |
| | JNZ | LOOP | If all bytes are not added go back to next byte. |
| | OUT | PORT.1 | Display low order byte to sum at Port 1. |
| | MOV | A, B | Transfer carry to Accumulator |
| | OUT | PORT 2 | Display carry digit |
| | HLT. | | End of the Program. |

## Debugging of the Program -

If the program is not executed as per the asked value, we can attempt to debug the program by observing the execution of instruction. It is called dynamic debugging.

## Tools for dynamic Debugging -

- Single Step
- Register Examine
- Break point.

1. **SINGLE STEP:** It allows you to execute one instruction at a time and to observe the result of each instruction. Single step facility built with a hard wired logic circuit.

The following errors can easily identified by -

- Incorrect Address
- Incorrect jump location for loop.
- Incorrect data or missing code.

2. **REGISTER EXAMINE:** The keys allows to examine the content of the microprocessor Register. When appropriate key is pressed the monitor program can display the content of the Register.

PAGE NO. .....

3. **BREAK POINT** — It is the software routine that allows you to execute the program as per the applied section. The breakpoint can be set by RST instruction. When push key has been pressed the program execution will reach at the terminating point.

# TIMING PROCEDURE OF MICROPROCESSOR:

1) **INSTRUCTION CYCLE**: It is time required to complete the execution of an instructions. The 8085 instruction cycle consist of one to six machine cycle. or one to six operations.

2) **MACHINE CYCLE**: It is defined as the time required to complete one operation for accessing memory, I/o or acknowledging an external required. This cycle may consist of 3 to 6 state.

3) **T-STATE**: It is defined as one subdivision of the operation performed in one clock period. These subdivision are internal state synchronized with the system clock and each T-state is equal to one clock period.

PAGE NO. .....

The microprocessor external communication function can be divided into 3 categories-

1. Memory Read and Write
2. I/O Read & Write
3. Request Acknowledge.

These instructions can be divided into further operation. These are opcode fetch, memory Read and memory write. It examine the signal on various bus. Opcode fetch, memory Read, memory write examine the signal on various bus in relation to the system clock.

## Opcode Fetch Machine Cycle:-

The instruction needs to get this machine code from the memory register, where it is stored before the microprocessor can begin to execute the instruction.

## Memory Read / Write Machine Cycle:

It executes the memory Read & write cycles. It is used for the 2 byte and 3 byte Instructions.

There is no direct relationship between the number of byte an instruction and no. of machine cycle required to execute that instruction.

Opcode Fetch = 1 machine cycle (4 T-state)

Memory Read = 1 machine cycle (3 T-state)

Memory write = 1 machine cycle (3 T-state)

I/o Read = 1 machine cycle (3 T-state)

I/o Write = 1 machine cycle (3 T-state)

Stack Related Instruction = 1 machine cycle (6 T-state)

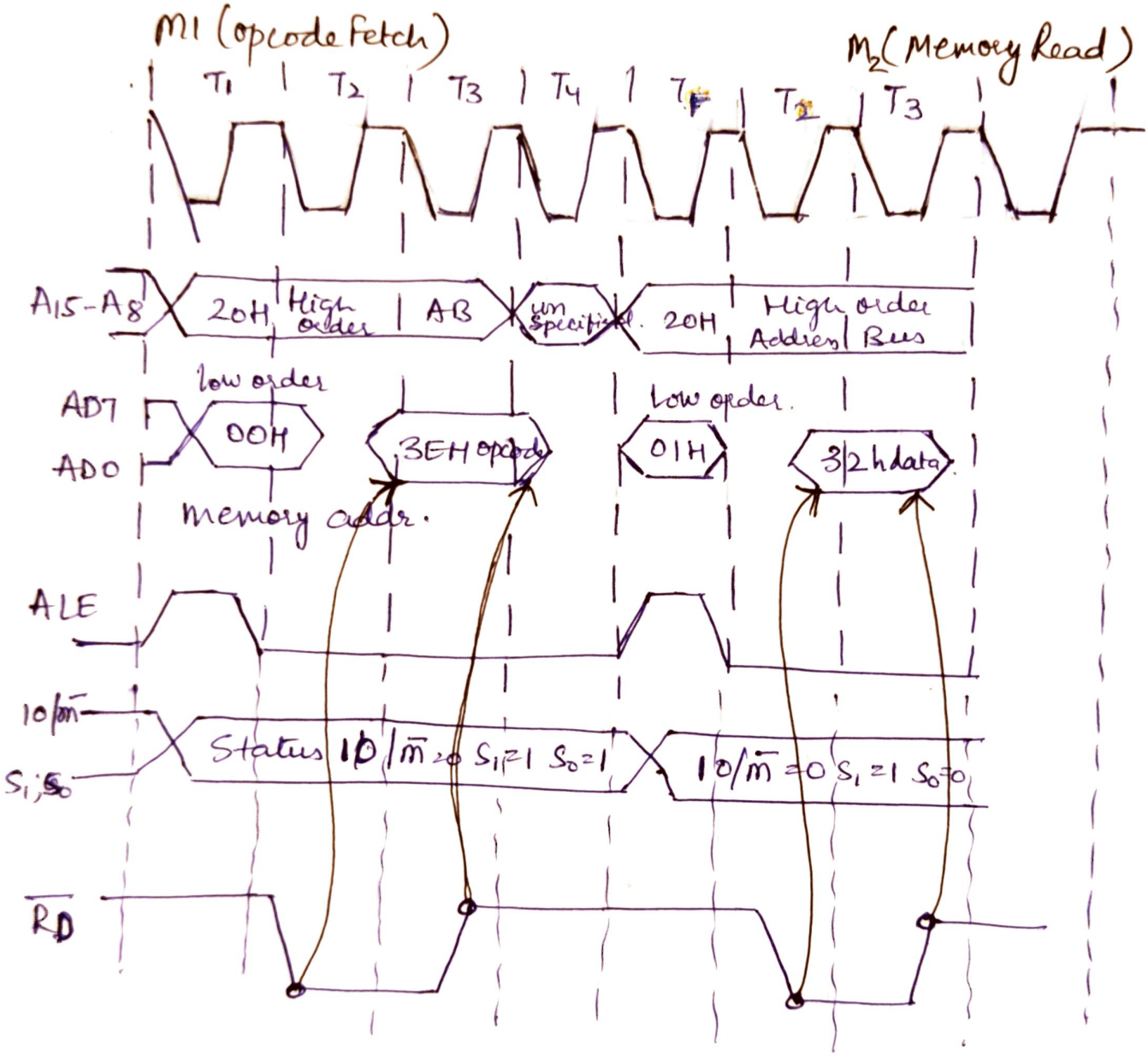
PAGE NO. .....

## Examples of Machine Cycle :-

| | Instruction | Machine Cycle | T-state | Cycles |
|---|---|---|---|---|
| 1. | ADC Reg. | 1 | 4 | F |
| 2. | CMP M | 2 | 7 | FR |
| 3. | INR M | 3 | 10 | FRW |
| 4. | LDA Add | 4 | 13 | FRRR |
| 5. | JMP Add | 3 | 10 | FRR |
| 6. | JNC Add | 2 (If condition not true) | 7 | FR |
| | | 3 (If condition True) | 10 | FRR |
| 7. | MOV M, R | 2 | 7 | FW |
| 8. | MVI R, data | 2 | 7 | FR |
| 9. | STA Addr. | 4 | 13 | FRRW |
| 10 | LHLD addr. | 5 | 16 | FRRRR |

# Timing Diagram of MVI A, 32H

| Memory Location | Machine code | Instruction |
|---|---|---|
| 2000H. | 3EH. | MVI A, 32H. |
| 2001H. | 32H. | load byte, 32H in the accumulator. |

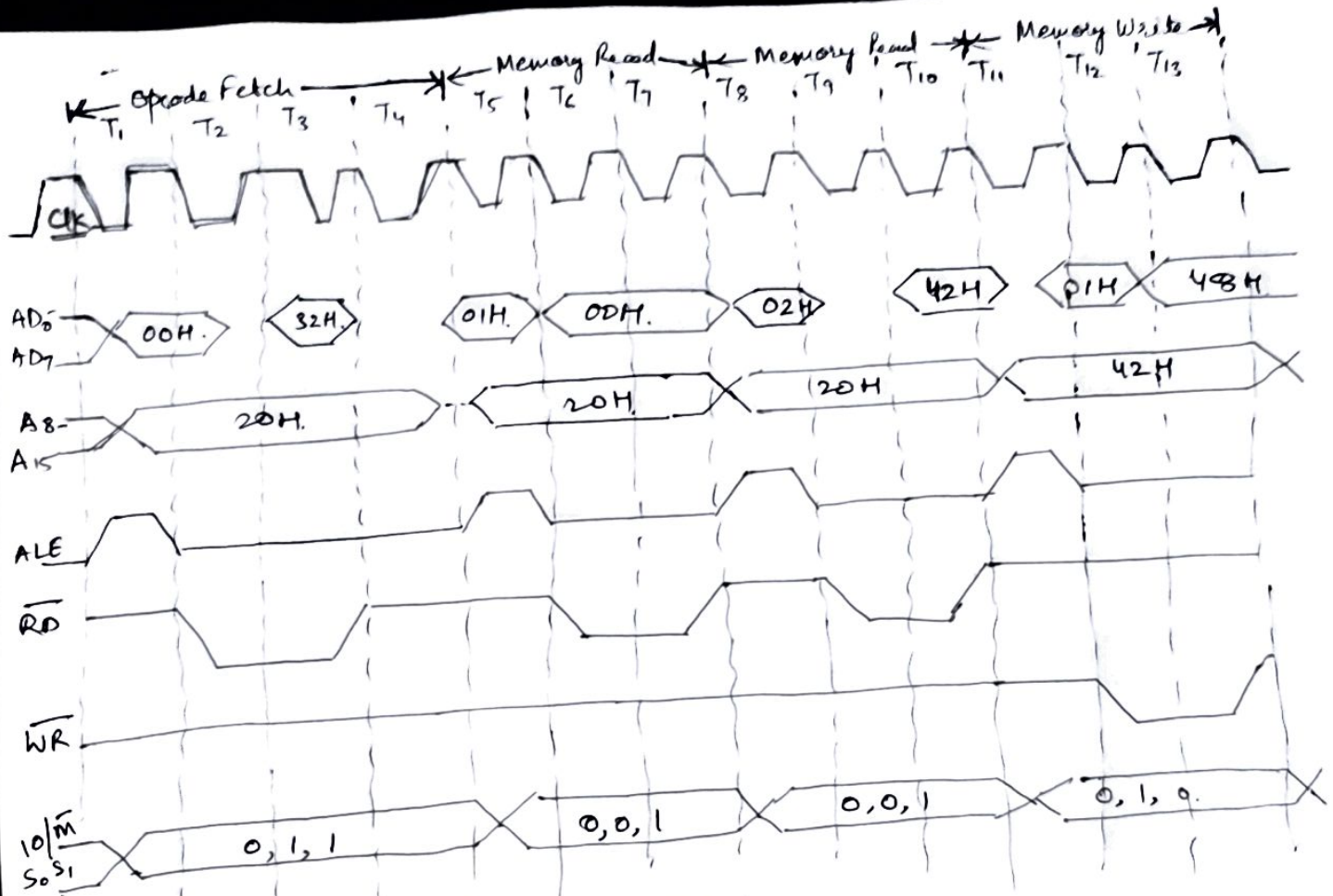



Timing Diagram of MVI A, 32H.

PAGE NO. .....

## Timing Diagram of STA Instruction:-

| Address | $A_H = 42H.$ | |
|---|---|---|
| 2000 | STA 4200 | 32 |
| 2001 | | 00 |
| 2002 | | 42 |




STA TIMING DIAGRAM.