MICROPROCESSOR AND C PROGRAMMING

(16SMBEPH2)

(Brief notes for reference)

By

Dr. V. S. Kumar

Assistant Professor

Department of Physics

Swami Dayananda College of Arts and Science

Manjakkudi

Note: Students are instructed to refer book for study and reference respectively for further elaborate points as prescribed by the University.

UNIT: 1

BASICS OF DIGITAL COMPUTER.

1) COMPONENTS OF DIGITAL COMPUTER :-

HISTORY :-

In 1623, the mechanical machines are used for some arithmetic operation.

Blaise pascal 1642 used a mechanical adding machine.

In 1671 Gottfried Leibniz made an automatic multiplication and division machine.

In 1823 charle's Babbage made a multi-step machine.

In 1930 1st digital computer was introduced.

Evolution of digital computer :-

In 1946 ENIAC the digital computer was build.

In 1946 (vonneumann).

GENERATION OF COMPUTERS :-

1st generation ⇒ 1946-1954 ⇒ Electric valves

↳⇒ IBM 704, 709, UNIVAC 1 ⇒ Assembly language

Eg :- ~~Lang~~.

2nd generation ⇒ 1955-1964 ⇒ Transistor

↳⇒ IBM 620, IBM 7090 ⇒ High level language

Eg :

3rd generation ⇒ After 1965 ⇒ ICS ⇒ IBM 370,

↳⇒ CDC 7600 ⇒ µp programming.

4th generation ⇒ µp.

Now-a days for the operation of the system silicon technology can be replaced by gallium arsenide.

Due to the speed of electron in secondary one.

It is five times faster.

DIGITAL COMPUTER :-

A digital computer is a programmable machine specially designed for computing purpose.

A micro computer is a small digital computer.

A hardware include electronic, magnetic, mechanical devices.

COMPONENTS:

- CPU
- Memory
- I/O devices.

2m PROGRAMME:

A set of sequence of instruction to perform a particular task.

2m SOFTWARE:

Software is a set of programme.

2m PERIPHERALS:

Parts of input output and memory units.

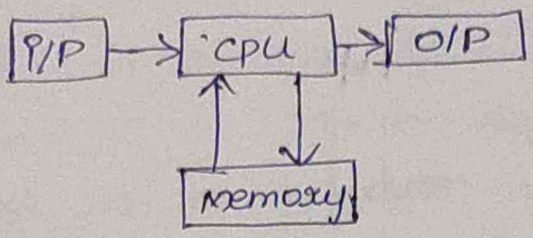
2m FIRMWARE:

A program or set of programmes stored in read only memory (ROM) and other devices.

Eg: monitor, Editor, Interpreter.

(change mach to ASS).

2nd BLOCK DIAGRAM :-

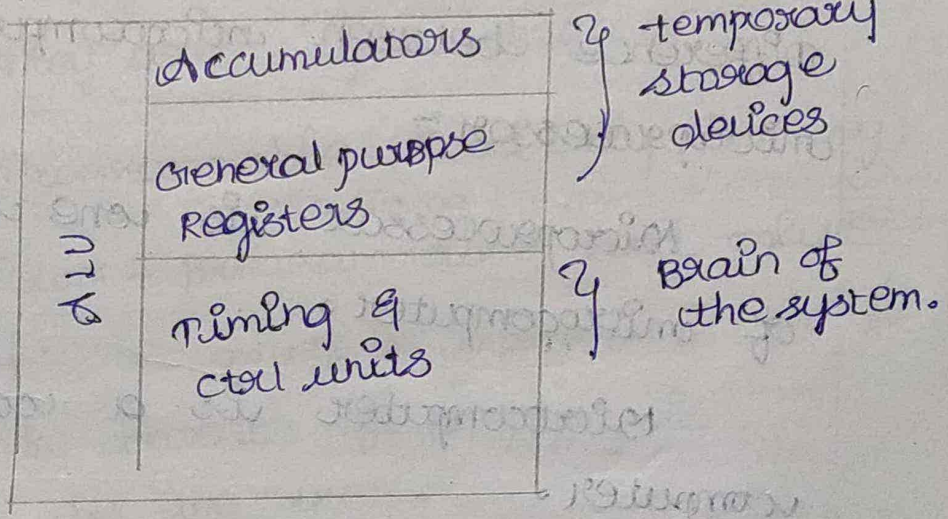


Note :-

CPU of a microcomputer is a microprocessor.

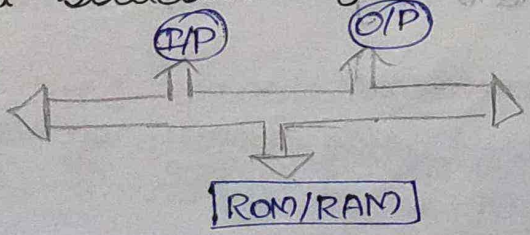
A microcomputer is a small digital computer.

CPU : CENTRAL PROCESSING UNIT :-

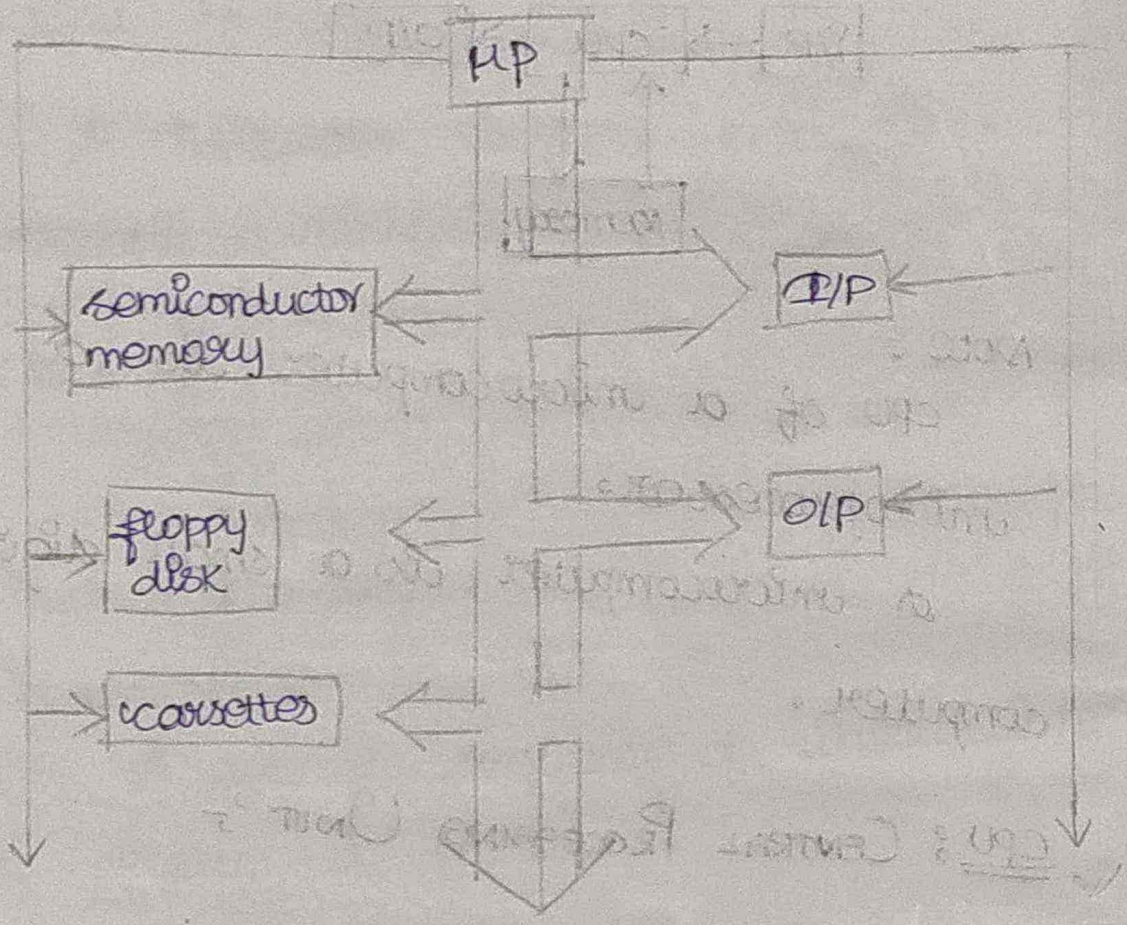


ALU ⇒ Arithmetic Logic Unit.

Formal structure of microcomputer :-



TYPICAL ORGANISATION :-



Difference between microcomputer and microprocessor :-

Microprocessor is one compound of microcomputer.

Microcomputer is a complete computer.

In microcomputer cpu functions are performed by microprocessor.

semiconductor memory :-

It is the main memory element of a microcomputer-based system & used to

ROM - Read Only Memory.

It is a non-volatile permanent storage.

It has a random access property

It contain assembler, compiler, monitor, debugging package, other permanent programmes.

It is used for sine, cosine, square root, log etc.

users can not write anything only they can read.

they are simple cheap dense.

PROM - Programmable Read Only Memory:

Prom - programmable Read only Memory.

The contents of a PROM is decided by the user.

The user can write permanent programmes in a PROM.

The programmer uses the PROM content is refered to as PROM programmer.

EPROM - Erasable programmable Read only Memory.

The contents are erased by exposing E·PROM to high intensity, short wave UV light for 10 to 20

An UV source with wavelength of 2537 Å unit is used.

Process of charging the content is non-convenient. The unit has to be removed from the board for exposer do UV source.

User cannot erase the content of a single memory because entire memory will be erased.

They are cheap reliable and widely used.

E^2PROM :-

Electronically Erasable programmable read only memory ⇒ E^2PROM.

they need not be erase from a microcomputer board for erasing.

The change in the content is made in milli second.

About 10 millisecond is required to write each byte.

A single byte of data or entire data can be erased in 10 millisecond.

RAM – Random Access Memory:

RAM – Random Access memory

this memory is volatile.

when the paver is turned off the content will be destroyed.

Two types are available in this memory.

Static Ram:- (memory)

static Ram, is made up of flipflops.

Dynamic Ram:

Dynamic memory is made up of MOS transistor – gates. It store's like a charge.

the large number of transister gate can be placed on a memory chip it has high density.

Dynamic memory is faster than static memory.

External Organisation :-

2nd

As we know, the data is stored in memory cells called bits.

Many chips as a total number of N bits each having unique address.

(Each bit will have different data.)

The address of bit ranges from 0 to (N-1) bits.

For reading a bit the address of bit is refered to the memory.

The memory does the work by reading the content of this address and give it on output pin.

NOTE :-

If content is to be written at a particular address. The chip is supplied with 1) 1 bit to be written

2) Address at which it is to be written.

3) A signal indicating that a write operation on the system to

be performed.

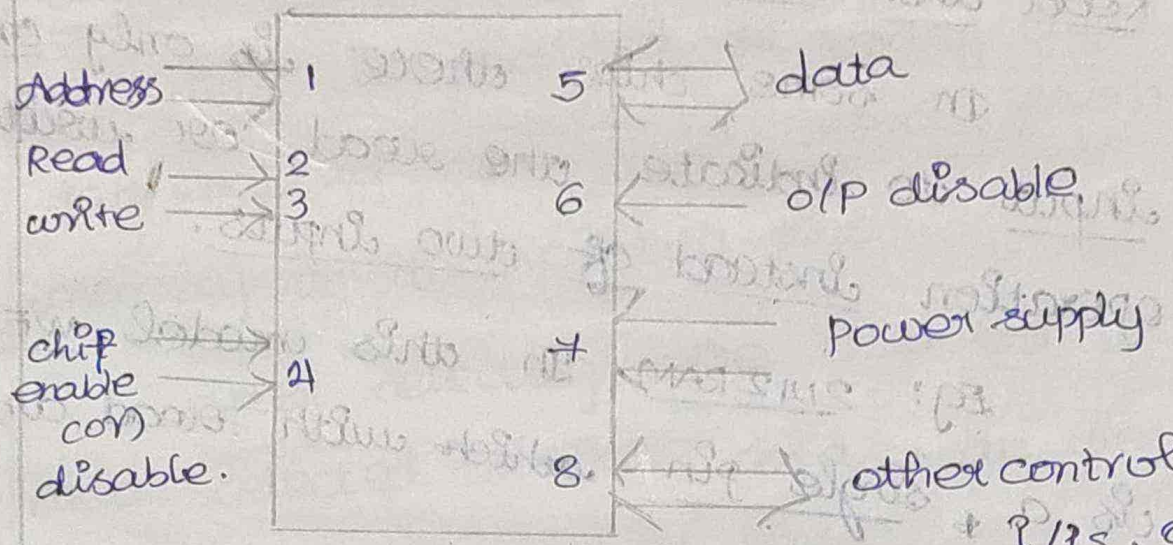
organisation has ⇒ bit, byte, nibble (four)

organisation.

8 bit = 1 byte.

N × 4 bit ⇒ Nibble organisation.

(5M) RAM – TYPICAL CONFIGURATION :-



Address Input :-

* All memory chips will have pins on which the address of the bit to be read or write is to be supplied.

* According to memory size, the number of pins are available.

Eg :- In a 2K byte memory it has 11 address input (i/ps) - (A_0 to A_{10})

Here, A_0 is LSB
A_{10} is MSB

* For a larger chip (16K), the address are split into some divisions.

The address input pins is under address multiplexing.

Read and write inputs:

In some chips there is only one input to indicate the read or write operation instead of two inputs.

Eg: 2142 RAM. In this model WE is a single pin with read and write operation.

For read operation (1) the input is high. For write operation (0) the input is low

Chip enable or disable?

Before any operation the chip must be in an active position and hence, it is initially enabled by activating the input.

Data Input & output

However, for an EPROM or PROM data is input at the pins. As these are as the output which are provided for data input to the chip. In write operation and data output from the chip in read operation.

output disable input :-

This is provided on many chips for enabling the data read from the memory on to the data pins. After selecting the chip (CS) it also selected before data can be read from the output pins.

Power supply input :-

All chips required input of power supply as denoted by Vcc. Mostly it is 5volt (Vcc).

other control Input / output signal :-

i) Address latching

In some chips input is provided at the falling (or) raising edge.

of which the chip latches the address available on its address pin.

* After the address has been latched inside the chip. The external address on address pins is no more used.

(ii) Ready Output :-

* After the chip selection this goes low and ALE input is active.

* If it goes high at raising edge of the next cycle. This output can be used by some microprocessors to extend the micro cycle.

(iii) Address Strobe :-

In chips requiring address multiplexing strobe inputs are provided.

using these inputs the chip is informed if LSB or MSB of the address have been placed in address pin.

(X) Microprocessor Architecture:- UNIT-2

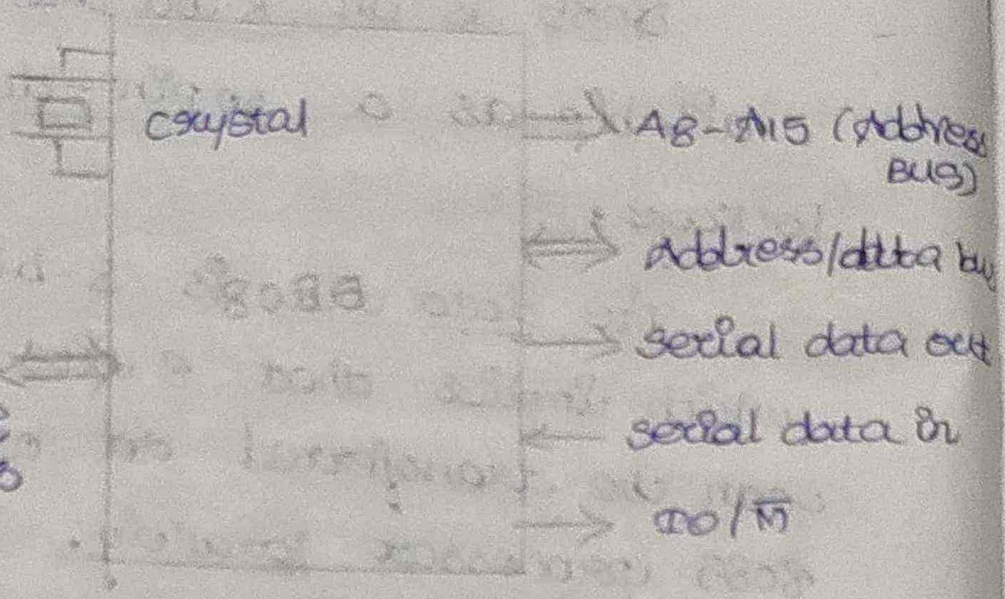
8085 is an 8 bit microprocessor. available as a 40 pin - dual inline package.

The data bus is 8-bit wide which implies that 8 bits of data can be transferred to or from 8085 processor parallely.

There are 8 pins which are dedicated to transmit 8 MSB bits of memory address.

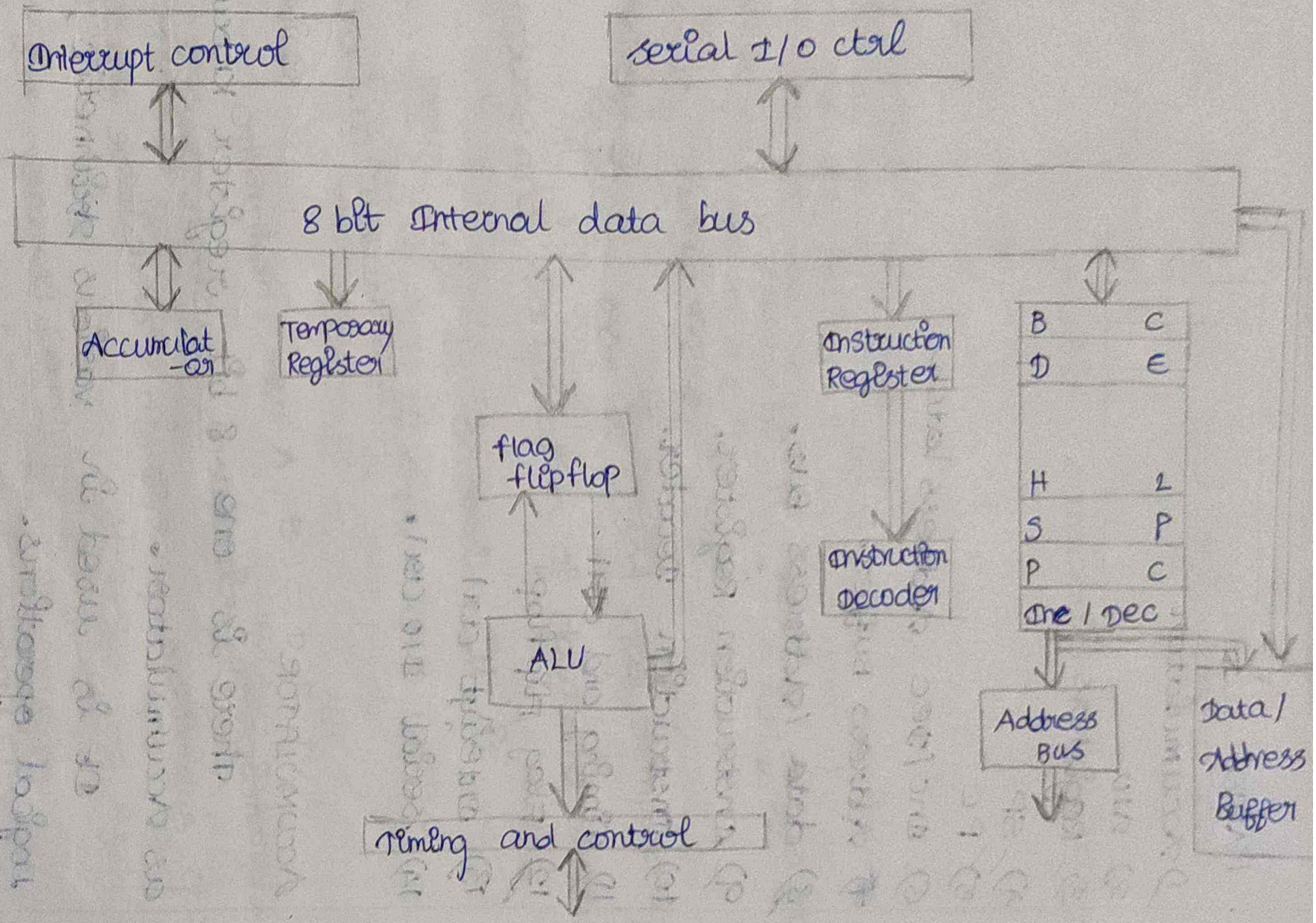
The LSB of 8 bit of address are retracted on the 8 lines on which data is transmitted. Thus, the data and apart of the address are transmitted over a set of shared lines. This is known as multiplexing.

Hence, 8085 has 16 bit address transmission capability.



2^{16} Memory locations can be addressed by 8085.

Each location is a 8-bit of data is transferred in parallel between the 8085 and the memory. So, 8085 can address upto 64 KB of memory.


Interrupt control
serial I/O ctrl
8 bit Internal data bus
Accumulat -or
Temposary Register
flag flipflop
ALU
Instruction Register
Instruction Decoder
B C
D E
H L
S P
P C
Inc / Dec
Address Bus
Data / Address Buffer
Timing and control

PARTS:-

1) Accumulator
2) ALU
3) Registers
4) SP
5) PC
6) Inc/Dec Address latch
7) Address Buffer. → The content stored in SP and PC is located in Add. Buf and Add. data buffer to commu. with CPU - The memory & I/O chips connected to these buses. the CPU can exchange desired data with mem & I/O chips.
8) data / Address Bus.
9) Instruction Register. → 8-bit reg. when an instru is fetched from memory then it stored in instruc. Regis. Instru. decoder decods the info present in Instr. Reg.
10) Instruction Decoder.
11) Timing and ctrl
12) Flag Flip flop
13) Interrupt ctrl
14) Serial I/O ctrl. → It ctrls serial data commu by using these 2 instru SID and SOD.

1) ACCUMULATOR

There is one 8 bit register known as Accumulator. It is connected to internal data bus and ALU

It is used in various Arithmetic, Logical operations. I/O & Load/Store operations.

Eg:- Addition of two 8-bit Integers one of the operands must be in A and other must be in memory or another registers.

ALU:

All Arithmetic and logic operations are performed.

Eg: Addition, subtraction, Logical AND, Logical OR etc..

REGISTERS:

As we know, the various registers (A, B, C, D, E, H, L) SP, PC having different memory locations. Each hold 8-bit data

Register can be used for temporary storage and manipulation of data and instruction. Data will be stored till it is send to the memory or input/output devices.

other than, the A, B, C, D, E, H, L Registers we have instruction registers, status registers, temporary registers etc.

PC ⇒ Programme Counter."

For the sequential execution of instructions the 16 bit register can be used. It is also known as memory pointer.

The function of programme counter is to point the memory address from which the next byte is to be fetched.

When 8-bit is being fetched. The programme counter is incremented by one point to the next location.

S.P :- Stack pointer

It is used by the programmer to maintain a stack LIFO in the memory. It holds the address of stack top.

It is used to save the content of a register during execution.

Flag Register

i) carry (CY)(S)
ii) zero (Z)
iii) sign (S)
iv) parity (P)
v) auxiliary carry (AC)

It is an 8-bit register having five bit flip flops which holds either 0 or 1 depending upon the result stored in accumulator.

i) Carry - This status flag holds carry out of the MSB resulting from the execution of an arithmetic operations.

If there is a carry from addition or subtraction (borrow) or comparision the CS flag is set to one otherwise zero.

ii) A zero status flag is set to one. If the result of arithmetic or logical operation is zero. Eg: 11 + 11 = 10 (carry 1)

For non-zero result it is zero.

iii) Sign (S)

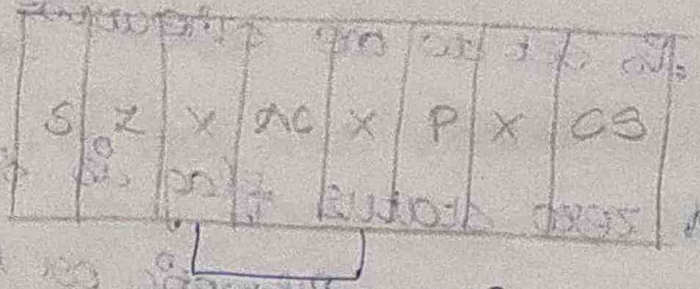
The sign status flag is set to one. If the MSB of the result of an arithmetic or logical operation is one otherwise zero.

iv) Parity (P)

This is set to one when the result of the operation contains even number of ones and it is zero when it is odd number of ones.

Auxillary carry (AC) :

This holds carry out of bit 3-4 resulting from the execution of an arithmetic operation.



undefined bits

Data bus - Address bus

Intel 8085 is a 8-bit microprocessor.

It's data bus is 8-bit wide. So, 8 bit of data can be transmitted in parallel from (or) to the microprocessor.

Intel 8085 requires a 16-bit wide address bus as the memory addresses are of 16-bit

the 8 MSB of the addresses are transmitted by the address bus (A8-A15).

the 8 LSB of addresses are transmitted by the address or data bus.

(AD0 - AD7)

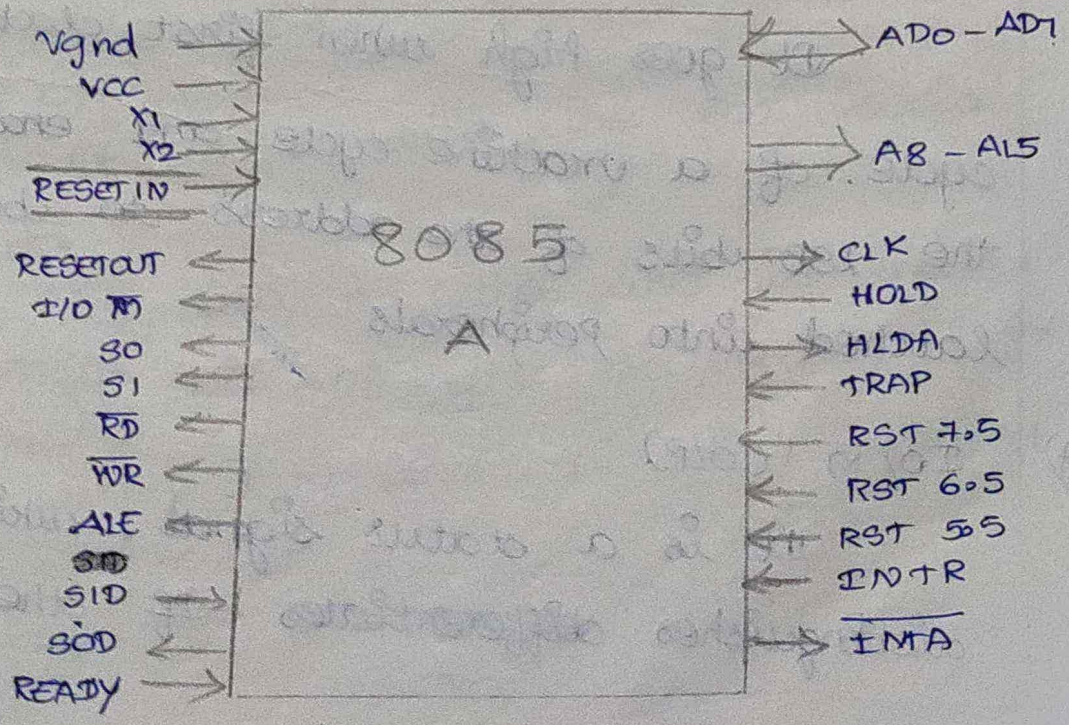
The data and address are transmitted at different moment by AD bus. At a particular time it transmits data or address (time shared mode). This is called multiplexing.

16-bit is initially transmitted by microprocessor

Then, 8 LSB of address is latched. So that, complete 16-bit addresses remains available for further operation.

Now, 8-bit AD bus becomes free and it is available for data transmission.

Pin Configuration of 8085



1) $A_8 - A_{15}$ (o/p) pin

These are the address bus and used for the MSB of the memory address (or) 8-bits of input/output address.

2) $AD_0 - AD_7$ (I/O)

Multiplexed address, Address, databus used for LSB of 8-bits of memory address (or) [I/O] address during first clock cycle of memory cycle. They are used for data during second and third cycle.

3) ALE (output)

It goes high when first clock cycle of a machine cycle and enables the LSB bits of the address to be latched into peripherals

4) $IO/\overline{M}$ (o/p)

It is a status signal which distinguishes ~~differentiates~~. If the address is for memory (or) I/O.

When it is 1 the address on the address bus is for I/O devices.

When it is (zero) the address on the address bus is for memory.

5) S_0, S_1 (o/p)

S_1	S_0	operation
0	0	Halt.
0	1	write
1	0	read
1	1	Fetch.

6) $\overline{RD}$ (o/p)

Signal to control read operation

7) $\overline{WR}$ (o/p)

Signal to control write operation.

8) SID (i/p)

It is a data line for serial input.

The data on the line is loaded into 7th-bit of the accumulator. (temporary storage) when RIM instruction is executed.

9) SOD (O/P)

It is a data line for serial output.

The 7th bit of accumulator is outputted when SIM instruction is executed.

10) READY (I/P)

It is used by microprocessor to sense if the peripherals is ready, to transfer data (or) not.

If READY is 1 peripheral is ready. If READY is 0 then the microprocessor has to wait till READY becomes 1.

11) CLK (O/P)

CLK output for user, this can be used for other digital IC's. It's frequency is same at which the processor operates.

12) HOLD (I/P)

This implies that another Master is requesting the use of AD BUS having received a HOLD request the microprocessor relinquishes the use of the buses, as soon as the current machine cycle is completed.

The processor regains the bus after HOLD removal.

HLDA (O/P)

Signal for HOLD acknowledgement.

After HOLD is removed HLDA is zero.

RST 5.5, RST 6.5, RST 7.5

There are Interrupts when an Interrupt is recognized the next Inspection is executed from a fixed location in the memory.

RST 7.5, 6.5 and 5.5 are the restart Interrupts.

They cause an Interrupt restart to the automatic one.

Each of them has a programmable mask.

TRAP (I/P)

This has the highest priority among Interrupts. It is a non-maskable Interrupt.

It is unaffected by any mask.

(5m) INTR (I/P)

It is an Interrupt signal. So, the above priority shows the lowest grade for this pin.

When it goes high the PC (Programmable counter), does not increment its content.

The microprocessor suspends, it normal sequence of Instructions. After completing the instruction at hand it goes to CALL (CALL) instruction.

The INTR line is sampled in the last state of the last machine cycle of an instruction.

The microprocessor acknowledges the Interrupt signal and issues an INTA signal. The INTR is enabled or disabled by softwares. An Interrupt issue by I/O devices to transfer data to the microprocessor without the wastage of time.

INTA (O/P)

Interrupt acknowledgement sent by the microprocessor after the INTR is received.

(10m) TIMING AND CONTROL UNIT :

While power is on the Input voltage stablizes. The microprocessor enters a loop for beginning the fetch operation the next.

~~the next~~ Instruction and execution is the ~~further~~ step.

The two tasks of fetching and executing Instructions are repeatitively executed by the microprocessor until power is off.

These two tasks are sequential in nature. (only one Instruction is dealt with during a certain time slot) Each of these fetch and execute are performed in time slot of fixed (or) varying length.

The FC operation (Fetch cycle) is performed in a time slot of fixed width whereas, easy EC-operation execute cycle.

the EC operation require variable width time slot.
the length being dependent on the instruction to be executed, therefore, Instruction cycle is equal to fetch cycle + execute cycle.

(or)

$$IC = FC + EC$$

FETCH CYCLE :-

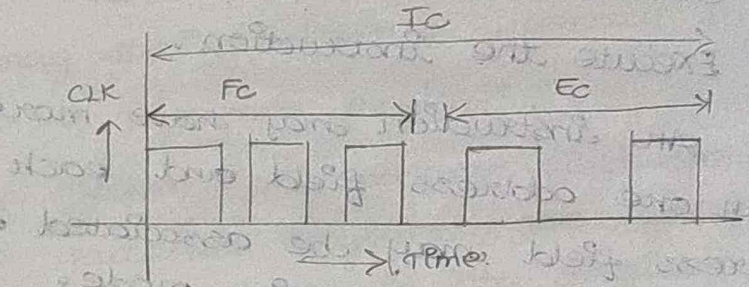
From the memory the instruction byte containing the operation code can be brought to microprocessor.

Note that the instruction itself may be more than 1 byte long containing data or operand address.

Programme counter register within the microprocessor keeps track of the address of the next instruction to be executed.

EXECUTIVE CYCLE :-

The time required for an execute cycle depends on the instruction type. If the instruction is only a byte long, the operands are in the general purpose registers. Then, only the decoding and execute operation are performed

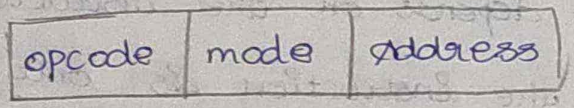


ADDRESSING MODE :-

The addressing mode specifies a rule for interpreting (or) modifying the address field of the instruction. Because, it is to be done before the operand is actually referred.

Systems used addressing mode technique for the purpose of accomodating one or both of the provisions

FORMAT :



BASIC OPERATIONS OF THE CYCLE :-

① Fetch the instruction from the memory.

② Decode the instruction.

③ Execute the instruction.

④ An instruction may have more than one address field and each address field may be associated with its particular addressing mode.

There are two modes,

1) Implied mode.
2) Immediate mode.

IMPLIED MODE :-

In this implied mode, the operands are specified implicitly in the definition of the instruction.

Eg:- CMA ⇒ COMPLEMENT ACCUMULATOR.

All register reference instruction that use an accumulator are implied mode instructions.

IMMEDIATE MODE :-

An immediate mode instruction has an operand field rather than an address field.

The address field of an instruction may specify either a memory word or a processor register.

When the address field specifies a processor register. The instruction is said to be in the register mode.

① REGISTER MODE :-

In this register mode, the operands are in register that reside within the CPU.

The particular register is selected from a register field in the instruction.

Eg:- CMPR ⇒ Compare Register with accumulator.

MOV
ADD

REGISTER INDIRECT MODE :-

The selected register contains the address of the operand rather than the operand itself. Before using a register indirect mode instruction the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.

Eg:- LDA 2020 [Code is alloted to the accumulator in the first instruction address is specified for an operand indirectly] after the address the operand can be identified.] [The data can be stored in a particular register which has a particular address considering a table of data in memory.]

To distinguish the various addressing modes it is essential to differentiate the address part of the instruction and the effective address used by the control when executing the instruction.

EFFECTIVE ADDRESS :-

Effective address is defined as the memory address obtained from the computation dictated by the given addressing mode.

DIRECT ADDRESSING MODE :-

In direct address mode, the effective address is equal to the address part of the instruction. The operand is in memory and its address is given directly by the address field of the instruction.

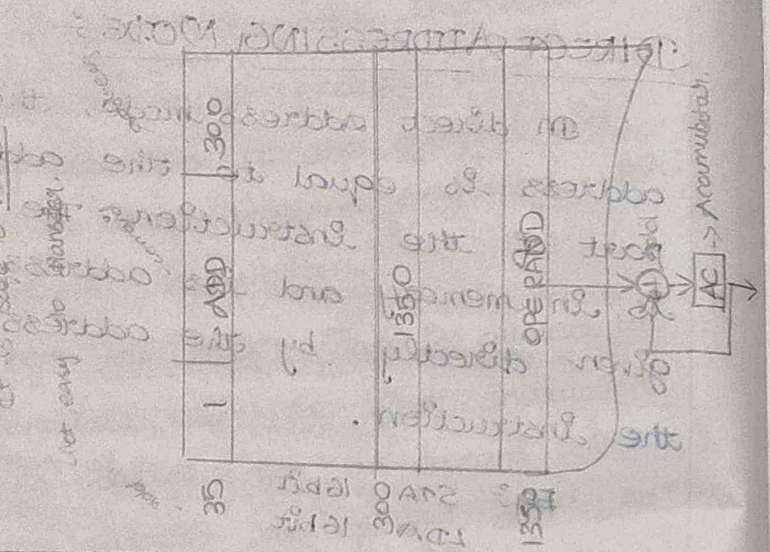
Eg:- STA 16bit → store to Accumulator.
LDA 16bit

INDIRECT ADDRESSING MODE :-

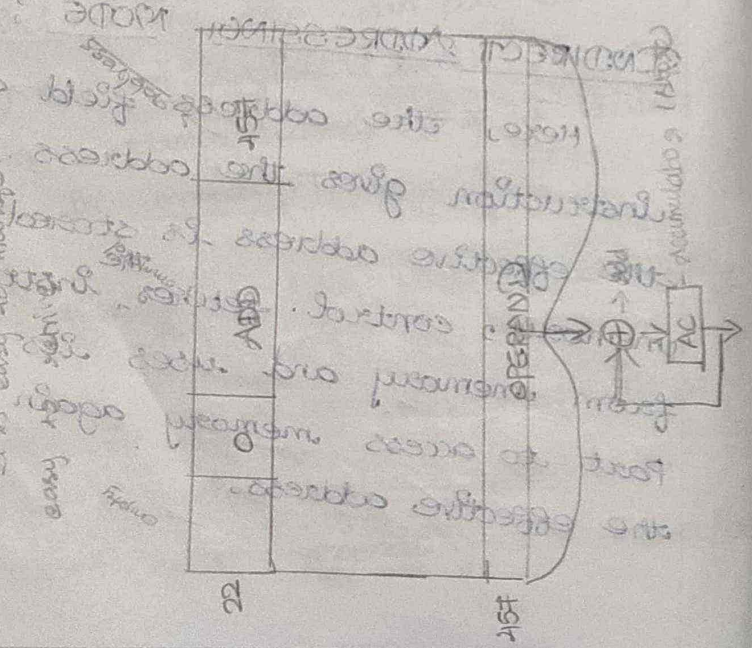
Here, the address field of the instruction gives the address where the effective address is stored in the memory, control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

Reffering to the Indirect addressing mode, the memory spends more bits when compared to the direct type.

INDIRECT ADDRESSING MODE.



DIRECT ADDRESSING MODE



RELATIVE ADDRESSING MODE :-

In this relative addressing mode, the content of the programme counter is added to address part inorder to obtain the effective address.

The address part is usually a signed number (two's complement).

which can be either positive or negative.

when this number is added to content of PC, the result produces an effective address whose position in memory is relative to the address of the next instruction.

Since, the shorter address field in the instruction format. since, the relative address can be specified with a smaller number of bits. compared, to the number of bits required for the memory address.

INDEXED ADDRESSING MODE :-

In this Indexed addressing mode the content of an Indexed register is added to the address part of the Instruction to obtain the effective address.

The Indexed register is special cpu register that contains an register that contain a Index and the Index value.

The address field of Instruction define the beginning address of data array in memory. Each operand is related to the Indexed beginning address.

Therefore, the distance between the beginning address and the operand address is related with the Indexed register.

Any Indexed address is the register not having address field is convert to Indirect address mode.

The Indexed addressing mode have a provision to Increment the address field related to the beginning address.

BASE REGISTER ADDRESSING MODE :-

In this base Register Addressing mode the content of a base register is added to the address part of the Instruction to obtain a effective address.

This is similar to the Indexed register mode the difference between two mode is in the way that they are used rather than they are computed.

This base register is assumed to hold a base address and address field of the Instruction. This register is used in compilers to facilitate the relocation Programme in memory.

When data or programmes change from one segment to other. the address value of Instruction must reflect this change of position.

With a base register the displacement value of Instruction do not have to change only the value of the base register requires updating to the new segment.

DATA TRANSFER INSTRUCTIONS :- (I) → Move to CMC

Data transfer instruction transfer or copy a data from register to register or register to memory or memory to register.

It also transfer an immediate data to register or memory.

This instructions do not affect any flags.

(1) MOVE INSTRUCTIONS :-

* Move between Register and Register.

Eg: MOV Rd, Rs

Here, Rd is the register that acts as a source designation and Rs is the source register.

Eg:-

```
mov Rd Rs
    ↓  ↓
mov B  A
```

Here, the contents are copied from Register A to Register B.

* Move between Register and Memory

Eg:- MOV M R

The letter M is used to represent memory, there are thousands of memory locations and each has distinct address.

Here, R is a general purpose Register (1 byte)

Eg:-

```
MOV M R
    ↓ ↓
MOV M A   → register
```

* Move Immediate Instructions :-

FORMAT :- MVI R, data 8

Eg:- MVI M, C

or

```
MVI R, data 8
    ↓    ↓
MVI A    C
```

This is a 2 byte Instruction.

* Load immediate Register pair :-

This instruction is used to load a 16 bit immediate data into a specified register pair.

FORMAT :- LXI RP, data 16

Eg:- LXI H, 2050

* Store Accumulator Direct

Store Accumulator Direct Instruction stores the contents of the Accumulator in the memory, whose address is specified in the Instruction itself.

Eg:- STA 2550

A
23

Memory address	Data
2550	67

A
23

Memory address	Data
2550	23

This instruction stores the content of the Accumulator in the memory, whose address is 2550.

The content of the Accumulator is not changed.

* Load Accumulator Direct :-

Load Accumulator direct loads the accumulator with a byte from the memory location whose address is specified in the Instruction itself.

Eg:- LDA 2150

This copies the contents of the memory location whose address 2150 into the Accumulator.

This is a three byte instruction.

No Instruction is available in 8085. to transfer data directly from one memory location to another.

(2) ARITHMETIC INSTRUCTIONS:-

* The arithmetic group of instructions are used to perform addition, subtraction, increment, decrement

a) ADD with register (or) Memory:-

The ADD instructions ADD r and ADD M are used to the contents of a specified register or memory to the contents of the Accumulator.

b) ADD immediate

The ADD immediate instruction (ADI data 8) adds the content of the accumulator with an 8 bit data. ie) included in the instruction itself.

c) ADD with carry :-

The ADD with carry instruction is used to ADD the contents of a specified register (or) memory along with carry flag to the contents of the accumulator.

Eg :- ADC R
ADC M

d) ADD immediate with carry :-

This instruction is of the form ACI data 8. This adds with carry. The contents of the accumulator with an 8-bit data, ie) included.

All types of ADD instructions affects flags.

b) ADD immediate

The ADD immediate instruction (ADI data 8) adds the content of the accumulator with an 8 bit data. ie) included in the instruction itself.

c) ADD with carry :-

The ADD with carry instruction is used to ADD the contents of a specified register (or) memory along with carry flag to the contents of the accumulator.

Eg :- ADC R
ADC M

d) ADD immediate with carry :-

This instruction is of the foam ACI data 8. This adds with carry the contents of the accumulator with an 8-bit data. ie) included.

All types of ADD instructions affects flags.

Subtraction :-

a) SUB with register or memory :-

The SUB instructions SUB R and SUB M are used to the contents of a specified register or memory to the contents of the accumulator.

b) SUB immediate :-

The SUB immediate instruction (SBI data 8) subtract the contents of the accumulator with an 8 bit data. ie) included in the instruction itself.

c) SUB with borrow :-

The SUB with borrow instruction is used to subtract the contents of a specified register (or) memory along with borrow flag to the contents of the accumulator.

Eg :- SBB R
SBB M

d) SUB immediate with borrow :-

This instruction is of the foam SBI data 8.

This SUB with borrow. the contents of the accumulator with an 8-bit data are included.

All types of SUB instructions affect flag.

3) Increment / Decrement Instructions :-

① Increment register / memory :-

Eg :- INR R
INR B
INR M

This Instruction increments the content of the register by 1.

[R] ← [R] + 1

② Increment register pair :-

A register pairs used to increment by 1 with respect to BC, DE, HL, and SP (16 bit)

Eg :- INX RP

Decrement :-

① Decrement register / memory :-

Eg :- DCR R
DCR B
DCR M

This Instruction decrements the content of the register by 1.

[R] ← [R] − 1

② Decrement Register pair :-

A register pairs used to decrement by 1 word to BC, DE, HL and SP (16 bit)

Eg :- DCX RP

4) Double ADD Instruction :-

Eg :- DAD RP

This Instruction ADDS the contents of the specified register pairs to the HL register pair.

LOGICAL INSTRUCTIONS :-

Logic instructions in 8085 are useful for performing various logical operations with the contents of the accumulator.

1) AND Instructions :-

a) AND WITH REGISTER / MEMORY :-

ANA r

The ANA r this instruction performs logical AND operation between the specified register and accumulator

It modifies the flags S, Z and P

CY flag is always reset and AC flag is set.

ANA M

b) AND IMMEDIATE :-

ANI data 8

This instruction performs logical AND operation between the contents of the accumulator and the immediate data.

which is given in the instruction itself.

It modifies the flag S, Z and P.

CY flag is always reset and AC flag is set.

Here, ANA r and ANI data 8, can be used for masking specific bits.

2) OR INSTRUCTIONS :-

a) OR WITH REGISTER / MEMORY :-

ORA r

The ORA r this instruction performs logical OR operation between the specified register and accumulator.

It modifies the flags S, Z and P

CY flag is always reset and AC flag is set.

ORA M

b) OR IMMEDIATE :-

ORI data 8

This instruction performs logical OR operation between the contents of the

accumulator and immediate data which is given in the instruction itself.

It modifies the flag S, Z and P CY flag is always reset and AC flag is set.

Here, ORA R and ORI data 8 can be used for masking specific bits.

iii) EX-OR INSTRUCTIONS :-

a) EX-OR WITH REGISTER / MEMORY :-

XRA R

This instruction performs logical EX-OR operation between the specified register and accumulator.

It modifies the flag S, Z and P. CY flag is always reset and AC flag is set.

XRA M

b) EX-OR IMMEDIATE :-

XRI data 8

This instruction performs logical EX-OR operation between the contents of the accumulator and immediate data which is given in the instruction itself.

It modifies the flag S, Z and P CY flag is always reset and AC flag is set.

Here, XRA R and XRI data 8 can be used for masking specific bits.

iv) COMPARE INSTRUCTIONS :-

a) compare with register / memory :-

CMP R

These instructions compares the contents of the register / memory with that of the accumulator.

CMP M

The comparison is performed by

subtracting the contents of the specified registers/ memory from that of the accumulator.

The contents of the register are not changed after the execution of the instruction.

b) COMPARE IMMEDIATE :-

CPI data 8

This instruction compare the immediate byte with the accumulator content by subtraction.

Here, S, P, Ac flags are modified.

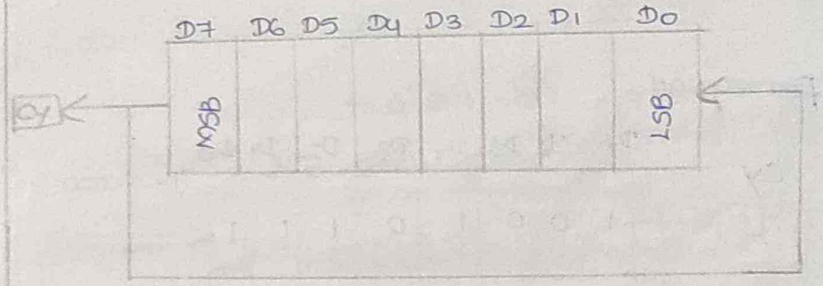
ROTATE INSTRUCTION :-

1) ROTATE ACCUMULATOR LEFT WITHOUT CARRY :- (RLC)

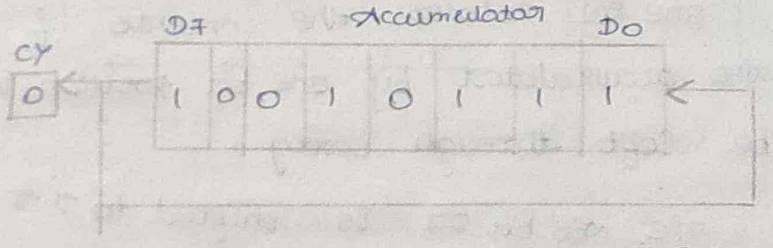
The instruction RLC rotates the contents of the accumulator by one bit position to be left.

The MSB bit D7 is shifted to the LSB D0

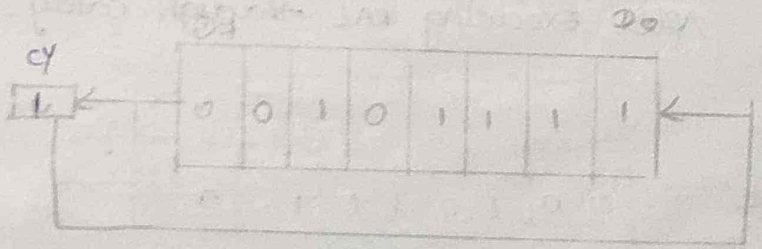
Also, the D7 bit comes the carry flag bit and modified.



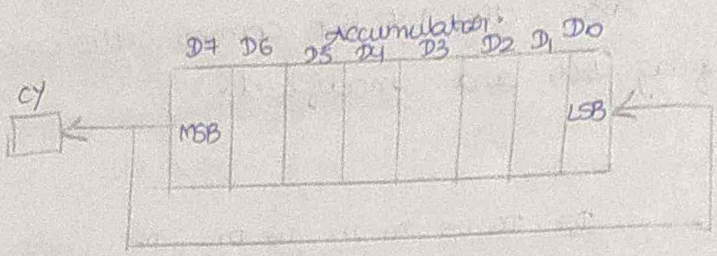
For eg: 97
(10010111)



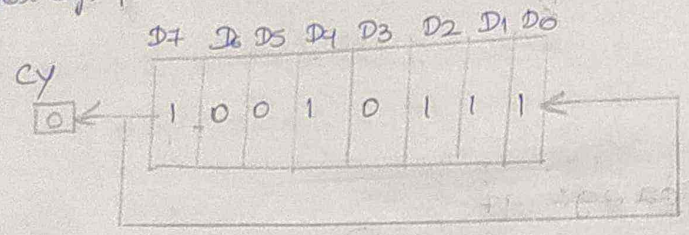
After executing RLC instruction



ii) ROTATE ACCUMULATOR LEFT THROUGH CARRY: (RAL)



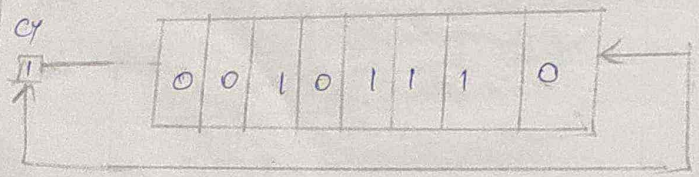
For eg: 97



The RAL Rotates the contents of the accumulator by one bit position to be left through carry.

The MSB bit D7 is shifted to the LSB bit D0.

After Executing RAL through carry.

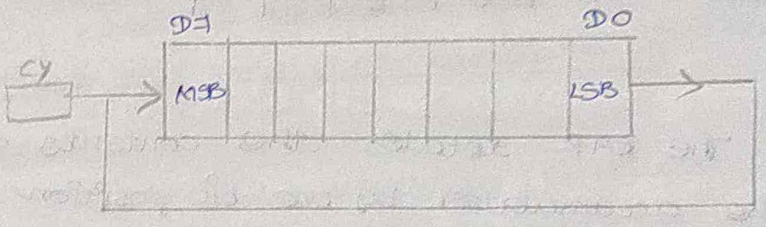


i) ROTATE ACCUMULATOR RIGHT WITHOUT CARRY:-

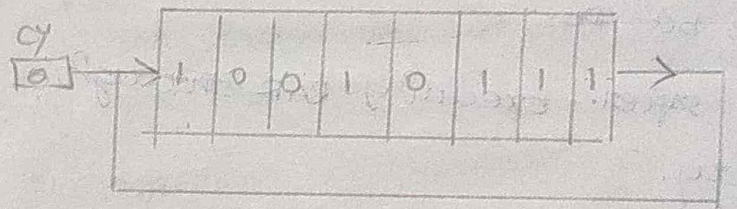
The RRC rotates the contents of the accumulator by one bit position to be right.

The MSB bit D7 is shifted to the LSB bit D0.

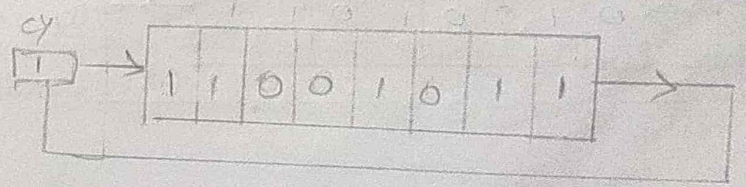
Also, the D7 bit comes the carry flag bit and modified.



For eg: 97

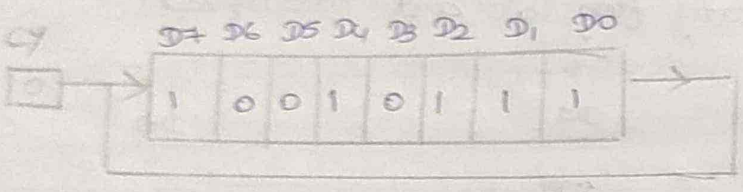


After executing RRC



ii) ROTATE ACCUMULATOR RIGHT THROUGH CARRY

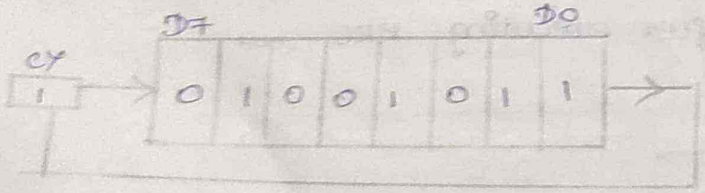
For eg: 97



The RAR rotates the contents of the accumulator by one bit position to be right with carry.

The MSB D7 is shifted to the LSB bit D0

After executing RAR through carry.



SPECIAL INSTRUCTIONS:

i) Decimal adjust accumulator instruction:

(DAA)

DAA is used to produce a correct BCD result when BCD addition is performed.

The auxiliary carry is used to get the correct BCD result. It is used to produce BCD result internally and not available for the user.

DAA is used to produce BCD result only after an addition or increment operation which affect the flag.

DAA instruction is applicable only for the data in the accumulator.

ii) COMPLEMENT ACCUMULATOR:

The CMA complements each bit in the accumulator. Here, flags are not affected.

iii) SET CARRY:

The Instruction STC sets the carry flag to 1

No flags are affected.

iv) COMPLEMENT CARRY:

The Instruction CMC complements the carry flag

DATA TRANSFER INSTRUCTION – 2

i) STORE ACCUMULATOR INDIRECT:

STAX B - register

The Instruction STAX B stores the content of the accumulator in a memory location whose address is [BC] from [A] - accumulator

register

Eg: STAX D

ii) LOAD ACCUMULATOR INDIRECT:

LDAX B

The Instruction LDAX B Load the content of the accumulator in a memory location whose address is [BC] from [A]

Eg: LDAX D

iii) STORE H AND L DIRECT:

The Instruction SHLD Address is used to store the content of H register and L register in two successive memory locations.

The contents of low register L is moved to a memory having a lower address and the contents of the high register H is moved to a memory having a higher address.

(Addr) ← (L)

(Addr+1) ← (H)

iv) LOAD H AND L DIRECT:

The LHLD Address is used to load the content of H register and L register in two successive memory locations.

The contents of low register L is moved to a memory having a

lower address and the contents of high register H is moved to a memory having a higher address.

(Address) ← (L)

(Address+1) ← (H)

v) EXCHANGE THE REGISTER PAIR HL and DE

The instruction XCHG

[H L] ⟷ [D E]

This instruction Exchanges the contents of HL pair with DE pair.

vi) COPY H and L REGISTERS to the STACK POINTER:

SPHL

This instruction SPHL loads the contents of H and L Registers into the stack pointer.

BRANCH INSTRUCTION:

8085 program counter sends out the address of the memory location. For a byte to be read or written into. In the mean time the program counter is automatically incremented and is ready with the address of the next memory location.

i) JUMP INSTRUCTIONS:

⇒ unconditional jump

⇒ conditional jump.

UNCONDITIONAL JUMP:

The unconditional jump instruction is of the form Jmp address 16.

Ex: Jmp 2550

This is a 3 byte instruction with the first byte containing the opcode (machine code) of the instruction and byte 2 and 3 containing the address.

Ex: Jmp START

Jmp LOOP1

CONDITIONAL JUMP:

The conditional jump instructions such as JZ or JC will cause the program counter to be loaded with the 16bit address given in the instruction only when the condition is true.

JZ - Jump if 0 flag is set.

JNZ - Jump if 0 flag is not set.

JC - Jump if carry flag is set.
JNC - Jump if carry flag is not set.
JPE - Jump if parity flag is set.
JPO - Jump if parity flag is not set.
JM - Jump when sign flag is set.
JP - Jump if sign flag is not set.

(iii) CALL AND RETURN INSTRUCTION

To call and return instruction are under branch instruction. First we should know that the substitutes are different from subprograms.

while writing a program a group of lines are repeated, then the memory space is large. This problem is solved in a elegant way.

The group of instruction which are to be repeated used must be stored in a seperate memory block. This block is group called subroutines.

From different points from the main program the subroutine is called.

Now the processor branches to execute the at the end of the subroutine, the processor has to return back to the main program.

TYPES:

i) unconditional call and return.

EX: CALL 2550
CALL Address 16

ii) conditional call and return.

CONDITIONAL CALL INSTRUCTIONS SUCH AS:

CZ - call if zero flag is set.
CNZ - call if zero flag is not set.
CC - call if carry flag is set.
CNC - call if carry flag is not set.
CPE - call if parity flag is set.
CPO - call if parity flag is not set.
CM - call when sign flag is set.
CP - call if sign flag is not set.

CONDITIONAL RETURN INSTRUCTION SUCH AS:

RZ - Return if zero flag is set.
RNZ - Return if zero flag is not set.
RC - Return if carry flag is set.
RNC - Return if carry flag is not set.
RPE - Return if parity flag is set
RPO - Return if parity flag is not set.

RM - Return if sign flag is set.
RP - Return if sign flag is not set.

iv) RESTART FUNCTION:

Restart instruction are special one byte instruction.

There are 8 restart instruction.

RST 0
RST 1
RST 2
RST 3
RST 4
RST 5
RST 6
RST 7

STACK AND STACK RELATED INSTRUCTIONS:

A small area of the read or write memory RAM can be called as stack.

When we use the program, the registers may have to use again and again. The content of a registers has to be saved for future use. Before using the same register in the other part of program.

To save the register PUSH instruction is used and to retrive the data or content POP instruction is used.

PUSH:

The PUSH instruction is used to PUSH or STORE a register pair on stack.

Eg: PUSH B

PUSH B pushes the data on content of BC register on the stack.

The stack pointer is decremented once and the contents of B register is stored in the memory (SP-1).

Next, If the instruction PUSH B is executed.

The contents of BC pair is saved in the stack and the value is determined.

POP:

The pop instruction is used to transfer two bytes from the top of the stack to the register pair is specified.

Eg: POP D

the registers are saved and retrieved on a last in first out basis (LIFO). This means the register pair that is pushed in last must be popped out first.

iii) IO AND MACHINE INSTRUCTIONS :

IO INSTRUCTIONS :

To communicate with the outside world the processor uses input ports and output ports.

INSTRUCTIONS:

1) IN
2) OUT

IN

The IN instruction is to input a byte from an input port to the accumulator.

OUT

The OUT instruction is used to output a byte from an output port to the accumulator.

MACHINE INSTRUCTION / MACHINE CONTROL INSTRUCTION :-

9) HALT INSTRUCTION:

The HLT Instruction stops the microprocessor and no further Instructions are executed.

UNIT-03

ALGORITHM:

STEP:1 Get the first number from the memory location 2050H to the accumulator.

STEP:2 save the first number in B register.

STEP:3 Get the second number from the memory location 2051H to the accumulator.

STEP:4 Add the number in A and B

STEP:5 Store the result which is in accumulator in the memory location 2052H

STEP:6 END.

EXAMPLE:

INPUT data 2050:03
2051:04

OUTPUT 2052:07

8-bit addition

PROGRAM;

```
LDA  2050H  ; Take the first number A
MOV  B,A    ; Transfer the number
              from A into B.
LDA  2051H  ; Take the second number
              to A
ADD  B      ; Add the two number in
              A and B.
STA  2052H  ; Store the result in
              memory.
HLT         ; End of program.
```

STEP 1: The numbers stored at memory locations 2050H and 2051H are added and the result is memory location 2052H

STEP 2: Register indirect addressing mode instructions like MOV A,M and MOV M,A are used.

ALGORITHIM:

STEP 1: Initialize the memory location 2050H with HL register.

STEP 2 : Get the first number from the memory location 2050H to the accumulator A.

STEP 3 : Increment the address by one. The memory address is 2051H

STEP 4 : Add the numbers in A and the content of the memory location 2051H.

STEP 5 : Increment the address by one. Now the address is 2052H.

STEP 6 : Store the result which is in accumulator in the memory location 2052H

STEP 7 : End.

2) 8-bit addition with carry :

ALGORITHM :

STEP 1 : Get the first number for the memory location 2050H in the accumulator.

STEP 2 : Save the first number in B register

STEP 3 : Get the second number from the memory location 2051H to the accumulator.

STEP 4 : Clear the C register to store carry.

STEP 5 : Add the number in A and B

STEP 6 : The carry produced which is saved in C register.

STEP 7 : Store the result in memory.

STEP 8 : Store the carry in memory.

STEP 9 : End.

PROGRAM :

```
STEP 1 : LDA 2050H ; Take the first
                     number in A.
         MOV B,A   ; Transfer the first
                     number into B
         LDA 2051H ; Take the second
                     number in A
         MVI C,00H ; Clear the C register
         ADD B     ; Add the two numbers
         INC GOTO  ; If the result is less
                     than FFH,
                   ; Jump to the location
                     named GOTO.
         INR C     ; Else, Increment C register
GOTO     STA 2052H ; Store the result in
                     memory.
```

HLT ; End of program.

EXAMPLE:

INPUT data 2050 : 89 (Addend)
2051 : C3 (Augend)

result 2052 : 4C
CARRY → 2053 : 01

ADDITION (16-bit, 32-bit and 64-bit addition):

Program, two multibyte numbers are added. Let us take two numbers each of 16 bits. Each of these two byte numbers require two memory locations. The address of the first number & second number are distributed with H and L register.

result is also stored in the memory in the place of either the first data or second

program can be used to perform 32 bit or 64-bit additions, by changing the counter to H or B.

PROGRAM:

```
LDA 2050H ; Take the first number (subtracted) in A
MOV B,A   ; Transfer the 1st number into B
LDA 2051H ; Take the second number (minuend) in A.
SUB B     ; subtract the first number from.
          ; the second number.
STA 2052H ; store the difference in memory
HLT       ; End of program
```

EXAMPLE:

INPUT data 2050 : 49 (subtrahend)
2051 : 85 (minuend)

difference 2052 : 3C

NOTE:

In the example given we have subtracted the smaller number from a bigger number obtained a positive result i.e) 85H - 49H = 3CH

HLT ; End of program.

EXAMPLE:

INPUT data 2050 : 89 (Addend)
2051 : C3 (Augend)

Result 2052 : HC
CARRY → 2053 : 01

ADDITION (16-bit, 32-bit and 64-bit addition):

program, two multibyte numbers are added. Let us take two numbers each of 16 bits. Each of these two byte numbers require two memory locations. The address of the first number & second number are initialized with HL and DE register.

result is also stored in the memory in the place of either the first data or second program can be used to perform 32-bit or 64-bit additions, by changing the counter to H or 8.

PROGRAM:

```
LDA 2050H ; Take the first number
            (subtracted) in A
MOV B,A   ; Transfer the 8-bit number
            into B
LDA 2051H ; Take the second number
            (Minuend) in A.
SUB B     ; subtract the first
            number from
          ; the second number.
STA 2052H ; Store the difference in
            memory
HLT       ; End of program
```

EXAMPLE:

INPUT data 2050 : 49 (subtraend)
2051 : 85 (Minuend)

Difference 2052 : 32

NOTE:

In the example given, we have subtracted the smaller number from a bigger number obtained a positive result ie) 85H - 49H = 3CH

on the other hand, if we subtract the bigger number from the smaller number, we will get a negative result. ie) (-3C)H the negative result will be stored in 2's complement from in the memory as C4H. In this case a occurs which is indicated by setting carry flag. In the next program, we get the differ-ent as well as the borrow.

3) 8-BIT SUBTRACTION WITH BORROW:

It is similar to add with carry program. A borrow (carry flag) occurs when a bigger number is subtracted from a smaller number.

ALGORITHM:

STEP 1: Get the first number (subtraend) from the memory to the accumulator

STEP 2: Store the first number in B register.

STEP 3: Get the second number (minuend) from the memory to the accumu-lator.

STEP 4: clear C register to store the borrow, if the 1st number is bigger than the second.

STEP 5: Subtract the subtraend from the minuend.

STEP 6: If the first number is bigger than the second number, a borrow is produced which is saved in C register.

STEP 7: Store the difference in memory.

STEP 8: Store the borrow in memory

STEP 9: End.

PROGRAM:

```
LDA  2050H  ; Take the subtraend in A
MOV  B,A    ; Transfer the subtraend
              into B
LDA  2051H  ; Take the minuend in A
MVI  C,00H  ; clear the C register.
SUB  B      ; subtract the subtraend
              from the minuend.
JNC  GOTO   ; If there is no borrow
            ; Jump to the location
            ; named GOTO
INR  C      ; Else, increment C
              register to
            ; store the borrow
```

STA 2050H ; store the difference in memory.

MOV A,C ; move the borrow in C to A

STA 2053H ; store the borrow in memory.

HLT ; End of program.

EXAMPLE :

INPUTdata : 2050 : 04
2051 : 03

RESULT : 2052 : 01
BORROW : 2053 : 01

4) 8-BIT DIVISION :

ALGORITHM :

STEP :1 Get the divisor from the memory location 2050H to the accumulator

STEP :2 save the divisor in B register.

STEP :3 Get the dividend from the memory location 2051H to the accumulator.

STEP4 Initialize the c register

STEP :5 Increment the C register.

STEP:6 Subtract the divisor from the dividend.

STEP :7 If the dividend is greater the divisor, then go to step5.

STEP :8 If the dividend is less than the divisor, and the contents of A and B to get the remainder.

STEP: 9 store the remainder in memory.

STEP ; 10 store the quotient in memory

STEP ; 11 End.

PROGRAM :

LDA 2050H ; Take the divisor to accumulator.

MOV B,A ; Transfer the divisor to B register.

LDA 2051H ; Take the dividend to accumulator.

MVI C, FFH ; Initialize C register to -1

HERE

INR , C ; Increment c register content by one.

```
SUB B   ; Subtract the divisor
          from the dividend.
JNC HERE ; If the dividend is
           greater than the divisor,
           then go to the location
           named HERE.
ADD B   ; Add the contents of
          A and B to get the
          remainder.
STA 2053H ; Store the remainder
            in memory.
MOV A,C  ; Transfer the quotient
           in C register to the
           accumulator.
STA 2052H ; Store the quotient
            in memory.
HLT      ; End of program.
```

EXAMPLE:

INPUT data 2050 : 04H
2051 : 0EH

RESULT 2052 : 03
2053 : 02

5) 8-bit MULTIPLICATION

ALGORITHM:

STEP : 1 Get the multiplier from the memory location 2050H to the accumulator.

STEP : 2 Save the multiplier to B register.

STEP : 3 Get the multiplicand from the memory location 2051H to the accumulator.

STEP : 4 Clear D refers to store the carry.

STEP : 5 Add the multiplier and the multiplicand.

STEP : 6 If there is no carry

STEP : 7 Decrement the B register.

STEP : 8 Store the lower byte in memory.

STEP : 9 Store the higher byte in memory.

STEP : 10 End.

PROGRAM;

LDA 2050H : Take the multiplier to accumulator.

MOV B,A : Transfer the multiplier to B register.

LDA 2051H : take the multiplicand to accumulator.

MOV C, A : Transfer the multiplicand to C register.

XRA A : clear the accumulator.

MOV D, A : clear D register to store the carry.

ADD C : Add the contents of A and C registers.

JNC GOTO : If there is no carry, then go to the location named GOTO.

INR D : Else, increment D register.

DCR B : Decrement the B register by one.

JNZ HERE : If the value in B register is not equal to zero, then jump to the location named HERE.

STA 2052H : store the lower byte of product in memory.

MOV A, D : Transfer the higher byte in memory.

STA 2053H : store the higher byte in memory.

HLT : End of program.

Example :

Input data : 2050 : 43
2051 : 07

Result : 2052 : 05
2053 : 01

6) 8-bit SUBRACTION :

ALGORITHM :-

STEP 1 : Get the first number from the memory location 2050H to the accumulator.

STEP 2 : Save the first number in B register.

STEP 3 : Get the second number from the memory location 2051H to the accumulator.

STEP 4 : Subtract the numbers in A and B.

STEP 5 : Store the difference which is in accumulator in the memory location 2052H.

STEP 6 : End.

EXAMPLE :-

LDA 2050H : Take the first number (subtrahend) in A.

MOV B, A : Transfer the first number into B.

LDA 2051H : Take the second number (minuend) in A.

SUB B : subtract the first number from the second number.

STA 2052 H : Store the difference in memory.

HLT : End of program.

EXAMPLE :

INPUT DATA : 2050 : 49 (subtrahend)

2051 : 85 (Minuend)

Difference : 2052 : 36

7) ASCENDING - DESCENDING :

ALGORITHM :

STEP 1 : Initialize HL pair as memory pointer.

STEP 2 : Get the count to D register.

STEP 3 : Load the array size to the HL pair

STEP 4 : Move the content of first number to A.

STEP 5 : Increment the memory by one.

STEP 6 : Move the second data from memory to B register.

STEP 7 : Compare the data in A register with the data in B register.

STEP 8 : If data in A register is smaller than the data in B register go to step 11.

STEP 9 : Else, "swap" the first data with the second data stored in memory.

STEP 10 : Decrement count in register D by one.

STEP 11 : If count in D register is not zero go to step 5.

STEP 12 : Decrement the count in C register by one.

STEP 13 : If count in C register is not zero go to step 3.

STEP 14 : End.

PROGRAM :

MVI C, 04H : Load C register with required count.

Loop 3 : MOV D, C : Copy the count in D register.

LXI H, 2050H : Initialize HL register pair with the starting address of the memory which stores the number.

Loop 2 : MOV A, M : Move the first number to accumulator.

INX H : Increment the memory address by one.

MOV B, M : Move the second number to B register.

CMP B : Compare the data in A and B registers.

JC Loop 1 : If data in A is smaller, then leave the smaller number in its position are continue further comparisons.

```
MOV M,A ; Else, 'swap' first data and second.
DCX H   ; data stored in memory.
MOV M,B ;
INX H   ; Restore memory address
LOOP1: DCR D; If comparisons are not over continue
JNZ LOOP2; in loop 2 and pick up the largest
   DCR C ; decrement the C register by one,
           to pick the largest from the
           remainder.
JNZ LOOP 3; data and continue till C register
           is equal to zero be in loop 3.
   HLT   ; End of program.
```

EXAMPLE :

INPUT DATA:	RESULT:
2050 : 2A	2050 : 03
2051 : FF	2051 : 1C
2052 : 03	2052 : 2A
2053 : E6	2053 : 2F
2054 : 32	2054 : 32
2055 : 1C	2055 : B6
2056 : D1	2056 : A0
2057 : 86	2057 : D1
2058 : A0	2058 : E6
2059 : 2F	2059 : FF

8) SQUARE OF A NUMBER USING LOOK UP TABLE:

ALGORITHM:

STEP1: Initialize HL pair to point look up table.
STEP2: Get the data.
STEP3: check whether the given input is less than 9.
STEP4: If yes go to next step as else halt the program.
STEP5: Add the desired address with the accumulator content.
STEP6: Store the result.

PROGRAM:

```
LXI H,5000 ; Initialize look up table address
LDA  5050  ; Get the data
CPI  0A    ; check input >9
JC AFTER   ; Error indication.
STA 5051;
   HLT
AFTER; MOV C,A ; Add the desired address.
       MVI B,00
       DAD B
       MOV A,M
       STA 5051 ; store the result
       HLT      ; terminate the program.
```

LOOKUP TABLE:

5000	01
5001	04
5002	09
5003	16
5004	25
5005	36
5006	49
5007	64
5008	81

RESULT:

INPUT DATA: 05H in memory location 5050
OUTPUT DATA: 25H in memory location 5051
INPUT DATA: 11H in memory location 5050
OUTPUT DATA: FFH (error indication) in memory location 5051.

9) LARGEST - SMALLEST:

ALGORITHM:

STEP1: Initialize HL pair which stores 10 numbers.
STEP2: Get the count to C register.
STEP3: Move the first data from memory to accumulator.
STEP4: Increment memory address in HL.
STEP5: Move the second data from memory to B register.
STEP6: Compare the data in A register with the data in B register.
STEP7: If data in A register is smaller then go to step 9.
STEP8: Else, 'swap' the first data with the second data stored in memory.
STEP9: Decrement count register c by one.
STEP10: If c register content is the not equal to zero go to step 3.
STEP11: End.

PROGRAM:

```
LXI H,2050H ; Initialize HL register pair with
              the starting address of the
              memory which stores 10 numbers.
MVI C,09H   ; Load c register with required count
LOOP2: MOV A,M ; Move the 1st number to accumulator.
INX H       ; Increment the memory address by one
MOV B,M     ; Move the second number to B
              register.
CMP B       ; compare the data in A and B
              registers.
JC LOOP1    ; If data in A smaller then leave the
              smaller number in its position and
              continue further comparisons.
MOV M,A     ; Else 'swap' first dat and second
DCX H       ; data stored in memory.
MOV M,B     ;
INX H       ; Restore memory address.
```

Loop1 : DCR ; If comparisons are not over.

JNZ Loop2 ; continue in loop 2.

HLT ; End of program.

EXAMPLE:

42H 21H 01H 1FH FFH 25H 52H 54H DAH ABH

10) MULTIBYTE ADDITION:

ALGORITHM:

STEP 1 : clear accumulator and carry flag.

STEP 2 : Get the count to C register with the number of bytes.

STEP 3 : Load HL register pair with memory location 2050H, where first data is stored.

STEP 4 : Load DE register pair in memory location 2060H, where second data is stored.

STEP 5 : Take the lower byte into the accumulator.

STEP 6 : Store the result in the memory.

STEP 7 : Increment the address by one

STEP 8 : Decrement the C register by one.

STEP 9 : If the C register content is zero, go to the location HERE.

STEP 10 : End.

PROGRAM:

XRA A ; clear the accumulator and carry flag.

MVI C,04H : Load the C register with the number of bytes C register is the counter.

LXI D, 2060H ; Load the HL register pair with memory address where the first data is stored.

LXI D, 2060H ; Load the DE register pair with memory address where the second data is stored.

LOOP: LDAX D ; Take the lower byte from the memory address 2060H into the accumulator.

ADC M ; Add with carry, the content of the accumulator with the byte in the memory address in 2050H

MOV M, A ; Store the result in the memory whose address is 2050H.

INX H ; Increment the address in HL register by one.

INX D ; Increment the address in DE register.

DCR C ; Decrement the C register content by one.

JNZ HERE ; If the content of the C register is not zero, then jump to the location named HERE.

HLT ; End of program.

11) MULTIBYTE SUBTRACTION:

ALGORITHM:

STEP1 : clear the accumulator and carry flag.
STEP2 : Get the count to C register with number of bytes.
STEP 3 : Load H2 register with memory location.
STEP 4 : 2050H, where first data is stored.
STEP 5 : Load DE register pair in memory location 2060H, where second data is stored.
STEP 6 : Take the lower byte into accumulator
STEP 7 : sub with borrow.
STEP 8 : Store the result in the memory.
STEP 9 : Increment the H2 register address by one.
STEP 10 : Increment the DE register address by one.
STEP 11 : Increment the c register content by one.
STEP 12 : If the content is not zero, go to step 5.
STEP 13 : End.

PROGRAM :

```
XRA  A ; clear the accumulator and carry flag.
MVI  C,02H; Load the c register with the number of bytes c register is the counter.
LXI  H,2050H ; Load the HL register pair with memory address where the first data is stored.
     Load the DE register pair with memory address where the second data is stored.
LXI  D,2060H ; Load the DE register pair with memory address where the second data is stored
LDAX D ; Take the lower byte from the memory address 2060H into the accumulator.
SBB  M ; Sub with borrow, the byte in the memory address 2050 from the content of accumulator.
MOV  M,A ; Store the result in the memory whose address is 2050H.
INX  H ; Increment the address in HL register by one.
INX  D ; Increment the address in DE register by one.
DCR  C ; Decrement the C register content by one.
JNZ  HERE ; If the content of C register is not zero, then Jump to the location named HERE.
HLT , End of program,
```

EXAMPLE:

INPUT DATA:	RESULT:
2050 : 03	2050 : EB
2051 : 73	2051 : CF
2052 : 21	2052 : 6D

6) Evaluate:

a) 89H + 14H

89
14
9D

89H + 14H = 9DH

b) 51H - 19H

4 5X 11
19
38

51H - 19H = 38H

5) program to store data byte 52H into memory location 5000H.