Microprocessor

The computer is a digital programmable machine. Its main components are CPU (central Processing Unit), memory, input device and output device. The CPU executes instruction. The input device is used to feed programs and data to the computer. The memory is a storage device. It stores programs, data and result. The output device, display or prints programs, data and results according to the instruction given to the computer.

The central processing unit built on a single IC is called microprocessor.

A digital computer, in which one microprocessor has been provided to act as a CPU is called microcomputer.

Evolution OF Microprocessor.

→ The first microprocessor Intel 4004, a 4 bit PMOS microprocessor was introduced in the year 1971 by Intel corporation, USA. Other company are also developed the microprocessor are Rock well International PPS-4, Toshiba's T3472, etc.

→ In 1972, Intel introduced, the first 8bit microprocessor. Intel 8008 which also uses PMOS technology. it's slow and not compatible with TTL logic.

→ In 1973, intel introduced a more powerful and faster 8 bit NMOS microprocessor, Intel 8080. The drawback of Intel 8080 was that it need three power supply.

→ In 1975, Intel developed an improved 8-bit NMOS microprocessor, Intel 8085 which was only one +5 V supply.

→ In 1978, Intel introduced a 16-bit microprocessor Intel 8086. The 80186 and 80188 are integrated microprocessor. Besides CPU they contains some additional components like programmable interrupt controller, two independent high speed DMA channels, three programmable 16 bit timer / counters, clock generator, programmable memory and peripheral chip select logic. Programmable wait state generator and local bus controller.

→ In 1980, personal computers were widely used. They used 16 bit microprocessors. Very popular personal computer PC and PC/XT used Intel 8088 microprocessor.

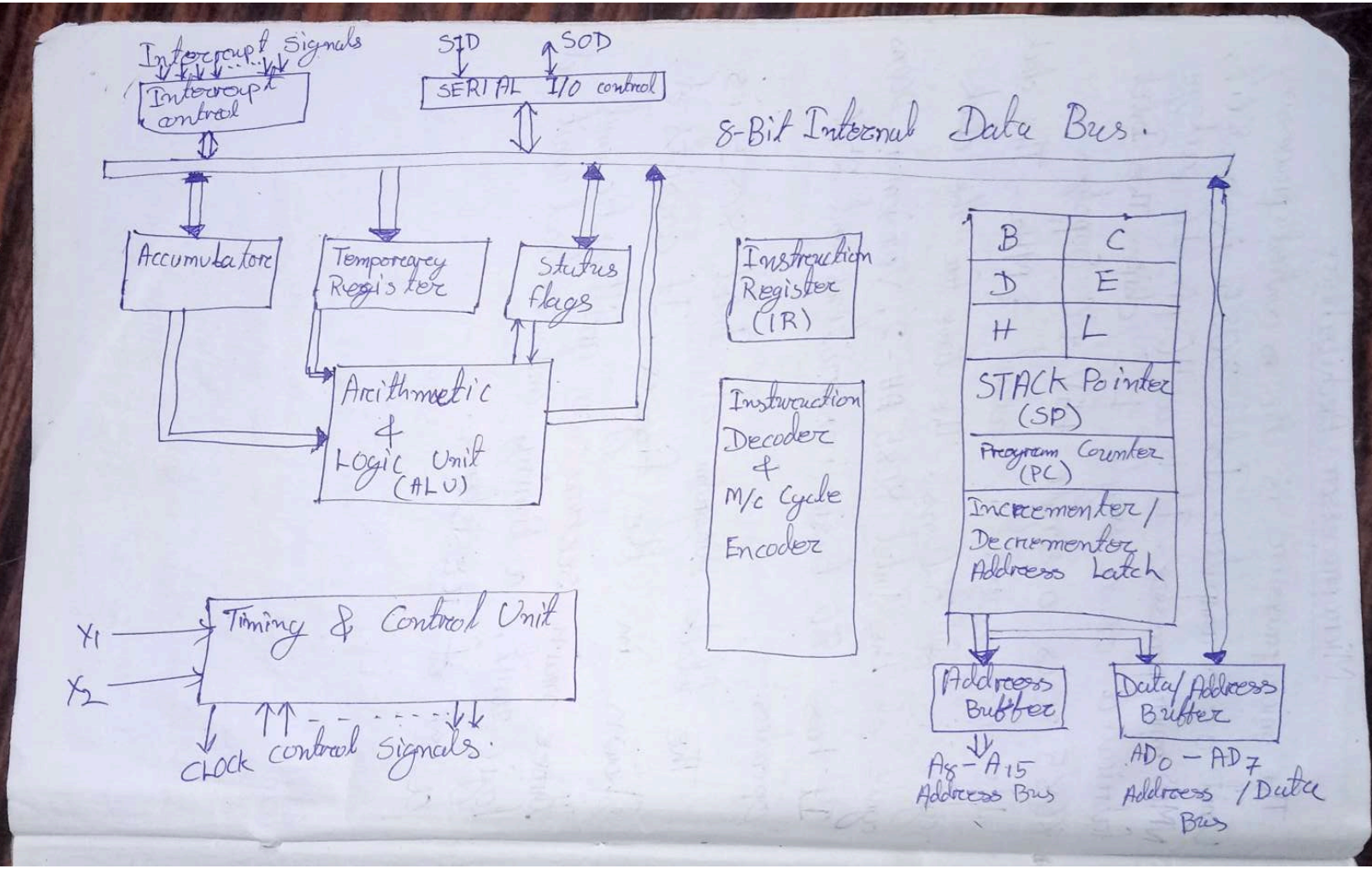
→ In 1985, Intel introduced a more powerful 32-bit microprocessor. Intel 80386 which become very popular and was widely used in desktop computer.

→ In 1998 Pentium II xeon uses second-level cache memory 1MB / 2MB. The celeron is a low cost 32 bit processor ~~ba~~ built on P6 architechture.

Microprocessor Architechture

The microprocessor is the central processing Unit of a computer. Intel 8085 is a 8 bit, NMOS microprocessor. It is a 40 pin IC package fabricated on a single LSI chip. The Intel 8085 uses a single +5V d.C Supply. Its clock speed is about 3 MHz. The clock cycle is of 320 ns. The time for the clock cycle of the Intel 8085 AH-2, Version is 200ns It has 80 basic instructions and 246 opcodes.

The block diagram of Intel 8085 is shown in the figure. It consists of three main sections: an arithmetic and logic unit, a timing and control unit and a set of resistores.


Interrupt Signals
Interrupt control
SID
SOD
SERIAL I/O control
8-Bit Internal Data Bus.
Accumulator
Temporary Register
Status flags
Arithmetic & Logic Unit (ALU)
Instruction Register (IR)
Instruction Decoder & M/c Cycle Encoder
B
C
D
E
H
L
STACK Pointer (SP)
Program Counter (PC)
Incrementer / Decrementer Address Latch
Timing & Control Unit
X1
X2
CLOCK
control Signals
Address Buffer
Data/Address Buffer
A8 - A15
Address Bus
AD0 - AD7
Address / Data Bus

ALU Arithematics and Logic Unit

The arithmetic and logic Unit, ALU, Performs the following arithmetic and logical operations.

1. Addition
2. Subtraction
3. Logical AND
4. Logical OR
5. Logical Exclusive OR
6. Complement (Logical NOT)
7. Increement (add 1)
8. Decreement (Subtract 1)
9. Left shift, Rotate Left, Rotate right.
10. Clear etc.

Registers

Registers are used by the microprocessor fore temporary storage and ~~main~~ manipulation of data and instructions. Data remain in the registers till they are sent to memory or I/O devices. Intel 8085 up has the following reesigster.

1. Six 8 bit general purpose registers They are B, C, D-E & H-L pairs.
2. All the Other registers are called Special purpose register.
3. One 8 bit accumulatore (ACC) i.e Register A.

(2) One 16-bit stack pointer (SP)
(3) One 16-bit program counter (PC)
(4) Instruction register.
(5) Temporary register.
Another a set of five flip-flops which serves as flags are status flags.
A status flag is a flip flop which indicates some condition which arises after the execution of an arithmetic are logical instruction.

General Purpose Register :- The 8085 Microprocessore contains Six 8-bit general prepose Registers. They are B-C, D-E & H-L. To hold 16-bit data a combination of two 8 bit register known as register-pair are used. The valid register paives are B-C, D-E & H-L.

ACCUMULATOR The accumulatore is an 8-bit reesistor associated with ALU. The register A in the 8085 is an accumulator. It is used to hold one of the operands of an arithmetic are logical operation. It serveres as one

input to the ALU. The other input to the ALU is either in memory or in general purpose registers. The final result is stored in ALU is also stored in Accumulator.

Program Counter (PC) :- It is a 16-bit special purpose register. It is used to hold the memory address of the next instruction to be executed. It keeps the track of memory address of the instruction in a program while they are being executed. The Microprocessor increases the content of the program counter during the execution of an instruction so that it points to the address of the next instruction in the program.

Stack Pointer (SP) :- It is a 16-bit special function register. The stack is a sequence of memory location set aside by a programmer to store or retrive the contents of accumulator, flags, program counter and general purpose registers during the execution of a program. The stack works on LIFO (Last in first out) principle, its operation is faster compared normal store or retrieve of memory location. The stack pointer (SP) controls addressing of the stack. The Stack Pointer holds the address of the top element of data stored in the stack. The stack is defind and the stack

pointer is initialized by the programmer at the beginning of a program which needs stack operation.

Instruction Register (IR) :- The instruction register holds the opcode (operation code or instruction code) of the instruction which is being decoded and executed.

Temporary Register :- It is an 8 bit register associated with the ALU. It holds data during an arithmatic / Logical operation. It is used by the micro-processor. It is not accessible to programmer.

Flags :- The Intel 8085 microprocessor contains five flip flops to serve as status flags. The flip flops are set or reset according to the conditions which arise during an arithmetic or logical operation.

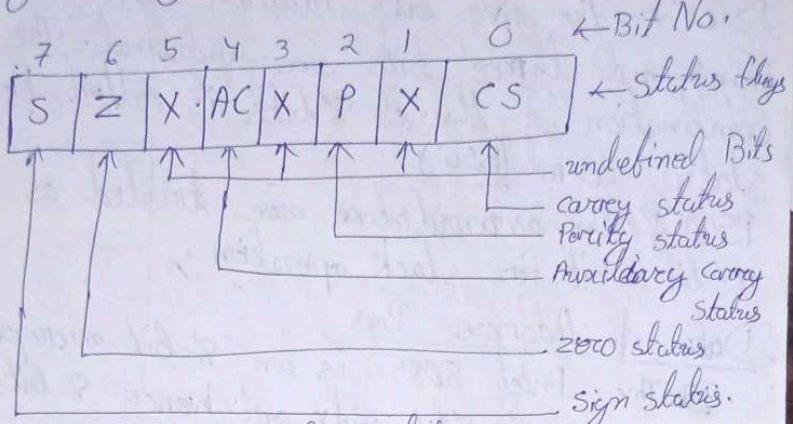
The five status flags of Intel 8085 are

① Carry flag (CS)
② Parity flag (P)

(3) Auxiliary Carry flags (AC)

(4) zero flags (Z)

(5) Sign flags (S)



Status flags of Intel 8085

Carry flag (CS) :- After the execution of an arithmetic instruction if a carry is produced, the carry flag CS is set to 1 otherwise 0.

Parity flag (P) :- The parity status flag P is set to 1 if the result of an arithmetic or logical operation contains even number of 1s. It is reset i.e it is 0. if the contains odd number of 1's.

Auxilliary Carry flags (AC) :- The auxilliary carry flag AC holds carry out of the bit number 3 to the bit number 4 resulting from the execution of an arithmetic operation.

```
  1100 1011
  1110 0001
 11011 0100
```

There is a carry from 3rd bit to 4th bit AC is set to 1.

zero flag (Z) :- The zero status flag (Z) is set to 1, if the result of an arithmetic or logical operation is negative. If the result is positive, the sign flag is set to 0.

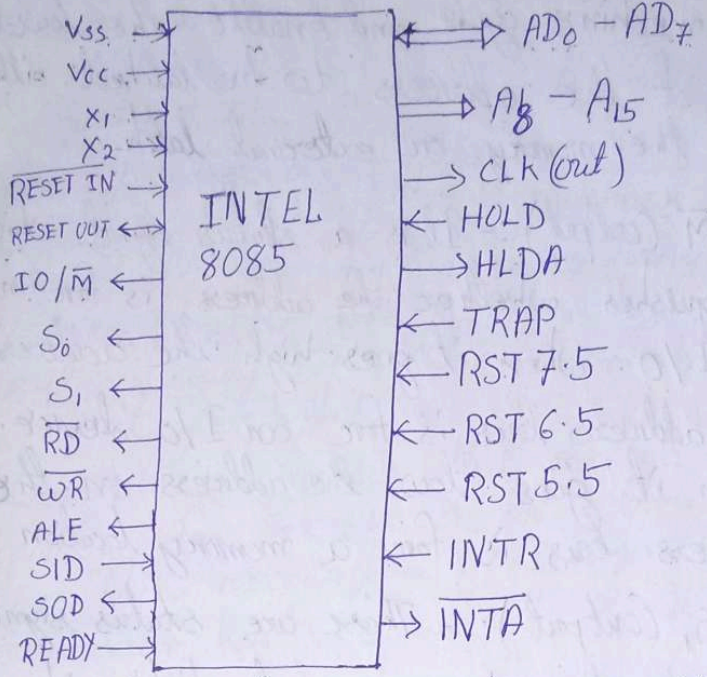
PSW :- The five bits indicate the five status flags and three bits are undefined. The combination of these 8 bits is called Program status word (PSW)

PSW & the accumulator are treated as a 16-bit unit for stack operation.

Data & Address Bus

The Intel 8085 is an 8-bit microprocessor. Its data bus is 8 bit wide and hence 8 bits of data can be transmitted in parallel from or to the microprocessor. The intel 8085 requires a 16 bit wide address bus as the memory address are of 16 bits.

Pin Configuration



The description of various pins are as follows.

$A_8 - A_{15}$ → These are address bus and are used for the most significant bits of the memory address are 8 bits of I/O address.

$AD_0 - AD_7$ (input / output) :- These are time multiplexed address / data bus. i.e they serve dual purpose. They are used for the least significant 8 bits of the memory address or I/O address during the first clock cycle of a machine cycle. Again they are used for data during second and thired clock cycle.

ALE (output) :- It is an address latch enable signal. It goes high during first clock cycle of a machine cycle and enable the lower 8 bits of the address to be latched either into the memory or external latch.

IO/$\overline{M}$ (output) :- It is a status signal which distinguishes whether the address is for memory or I/O. when it goes high the address on the address bus is for an I/O device. when it goes low the address on the address bus is for a memory location.

S_0, S_1 (output) :- These are status signal sent by the microprocessor to distinguish the various types of operation given below.

S_1	S_0	operation
0	0	HALT
0	1	write
1	0	READ
1	1	FETCH

$\overline{RD}$ (output) :- It is a signal to control READ operation, when it goes low the selected memory or I/O device is read.

$\overline{WR}$ (output) :- It is a signal to control write operation. When it goes low the data on the data bus is written into the selected memory or I/O location.

READY (input) :- It is used by the microprocessor to sense whether a peripheral is ready to transfer data or not. A slow peripheral may be connected to the microprocessor through READY line. If the ~~RDR~~ READY is high the peripheral is ready. If is low the microprocessor waits till it goes high.

HOLD (Input) :- It indicates that another device is requesting for the use of the address and data bus. Having received a HOLD request the ~~mp~~ microprocessor hold the use of the buses as soon as the current machine cycle is completed. Internal processing may continue. The processor again send an acknowledgement ~~the bus~~ after the removal of the HOLD signal.

HLDA (output) :- It is a signal for HOLD acknowledgment. It indicates that the HOLD request has been recived. After the removal of HOLD request the ~~HOLD~~ HLDA goes low. The cpu takes over the buses half ~~cl~~ cycle after the HLDA goes low.

INTR (Input) :- It is an interrupt request signal. Among interrupts it has lowest priority. when it goes high the program counter does not increment its content. The Up suspends its normal sequence of instructions. After completing the instruction at the last it attends the interrupt device.

INTA (output) :- It is an interrupt acknowledgement send by the microprocessor after INTR is Received.

RST 5.5, 6.5, 7.5 and TRAP (inpts) :-

These are interrupts, when an interrupt is recognised the next instruction is executed from a fixed location in the memory.

RST 7.5, RST 6.5 and RST 5.5 are the restart interrupts. They cause an internal restart to be automatically inserted. Each of them has a programmable mask. The TRAP has the highest priority among interrupts. It is a non maskable interrupt. It is unaffected by any mask or interrupt enable.

The order of priority of interrupt is

TRAP (Highest Priority)
RST 7.5
RST 6.5
RST 5.5
INTR (lower priority)

RESET IN (input) :- It resets the program counter to zero. It also resets interrupt enable and HLDA flipflops. It does not affect any other flag or register except the instruction register. The CPU is held in reset condition as long as RESET is applied.

RESET OUT (output) :- It indicates that the CPU is being reset.

X_1, X_2 (Input) :- These are terminals to be connected to an external crystal oscillator which drives an internal circuitry of the microprocessor to produce a suitable clock for the operation of microprocessor

CLK (output) :- It is a clock output for user which can be used for other digital ICs. Its frequency is some at which processor operates.

SID (input) :- It is data line for serial input. The data on this line is loaded into the 7th bit of the accumulator

RIM instruction is executed.

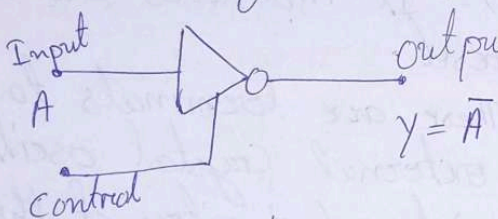
V_{CC} +5 Volts supply.

V_{SS} ground reference.

Tristate Logic Gate

In a tristate logic circuit there are three distinct output states namely logic 1, logic 0 & high impedance state.

The figure shows a tristate inverter with active high control.



The active high means that the control is effective when the signal applied to the control terminal is high. When the control signal is high the inverter operates as a normal inverter.

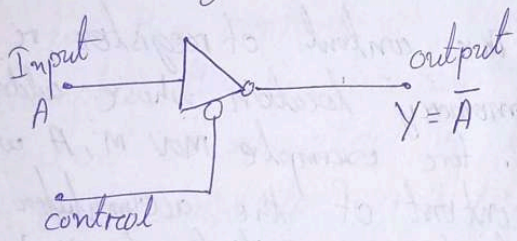
→ when input is high the output is low and when the input is low the output enters a third state in which the output impedance of the inverter gate becomes very high. In this state the inverter gate circuit

is effectively disconnected from the external circuit irrespective of the value of the input signal.

The effective ~~con~~ control signal may be either High or low, it depends on the design of a particular tristate gate.

The figure shows a tristate inverter with active low control. A small circle has been shown at the control terminal to indicate that the gate has an active low control.

In this case for the normal operation of the inverter the control signal is kept at 0 logic



RIM

In 8085 Instruction set, Read Interrupt Mask. It is a 1 Byte Multi purpose Instruction. ~~It is used fo~~ After the RIM instruction is executed the data on SID pin of 8085 gets loaded into this bit position.

Intel 8085 Instructions ch-2

An Instruction is a command given to the computer to perform a specified operation on given data.

Intel 8085 microprocessor is widely used in India. Some of the instructions are discride below.

MOV r1, r2 :- The content of reegister r2 is moved to reegister r1, fore example the instruction MOV A, B moves the content of register B to register A. The instruction MOV B, A moves the content of reegister A to reegister B.

MOV M, r :- The content of reegister r is moved to the memory location whose address is in H-L paire. fore example MOV M, A will transfore the content of the accumulator to the memory location specified by H-L paires

MVI r, data :- The data given just after an instruction are moved to the register r. fore example the instruction MVI A, 05 moves 05 to resister A.

LXI rp, data 16 bit :- 16 bit immediate data are moved to the reegister paire rp. fore example the instruction LXI H, 2400H moves 2400 to H-L paire.

LDA addr :- The content of the memory location whose address is specified with in the instruction itself is moved to accumulator. For example, the instruction LDA 2400H moves the content of the ~~mem~~ memory location 2400H to the accumulator.

STA addr :- The content of the accumulator is moved to the memory location whose address is specified within the instruction itself. For example, the instruction STA 2500H moves the content of accumulator to memory location 2500H.

ADD r :- The content of the register r is added to the content of the accumulator. For example, the instruction ADD B adds the content of reesistore B to the content of the accumulator and places the sum in the accumulator.

SUB r :- The content of register r is subtracted from the content of the accumulator ~~f~~ and the result is placed in the accumulator. for example the instruction SUB C ~~sd~~ subtracts the content of register C from the content of the accumulator and places the result in the accumulator.

RAL :- The content of the accumulatore is rotated left one bit through carrey.

INR rc :- The content of register rc is incremented by one.

IN Port-address :- This instruction transfers data from input device ore port. The data avaliable at the input port ore device is moved to the accumulatore.

OUT Port address :- This instruction transfers data from processore to the output port ore device. The content of the accumulatore is transferred to the output port ore device.

Opcode and Operands :-

Each instruction contains two parts :- operation code (opcode) and operand.

→ The firest part of an instruction which specifies the task to be perforemed by the amputer is called opcode.

→ The second part of the ~~ins~~ instruction is the data to be operated on and it is called operand.

→ The operand (ore data) given in the instruction may be in various forms such as 8 bit ore 16-bit data, 8-bit ore 16-bit address, internal registers ore a register ore memorey location.

Instruction Word Size.

A digital Computer understands instruction written in binary codes (Machine codes). The machine codes of all instructions are not of the same length.

According to the word size the Intel 8085 instructions are classified into the following three types:-

1. 1-byte instruction
2. 2-byte instruction
3. 3-byte instruction.

One-byte Instruction:- Examples of one-byte instruction are:-

MOV A, B :- Move the content of register B to register A. 78H is the opcode for MOV A, B Besides the operation to be performed the opcode also specifies the registers which contain operands (data). The opcode 78H can be written in the binary ~~fro~~ ~~from~~ form. as 0111 1000. The 1st two bits i.e 01 are for MOV operation; the next three bits 111 are the binary code for register A and the last three bits, 000 are the binary code for register B.

ADD B :- Add the content of register B to the content of the accumulator 80H is the opcode for the instruction ADD B.

RAL :- Rotate the content of the accumulator left by one bit. The opcode for RAL is 17H. In this instruction operand is not given, it is implied. The operand is in the accumulator. The instruction has to operate on the content of accumulator.

Two Byte Instruction :- In a two byte instruction the 1st byte of the instruction is its opcode and the 2nd byte is either data or address.

MVI B, 05 :- Move 05 to register B

06, 05 :- MVI B, 05 in the code form. The 1st byte 06 is the opcode for MVI B and the 2nd byte 05 is the data which is to be moved to register B.

IN 01 :- Read data at port B.

DB, 01 :- IN 01 in the code form.

DB is the opcode for the instruction IN and 01 is the address of a port. A two byte instruction is stored in two congucative memory location.

For Three Byte Instruction:-

In a three byte instruction the 1st byte of the instruction is its opcode and the 2nd and 3rd bytes are either 16-bit data or 16-bit address.

① LXI H, 2400H :- Load H-L Paire with 2400H

21, 00, 24 ; LXI H, 2400H in the code form.

The 1st byte 21 is the opcode fore the instruction LXI H. The 2nd byte 00 is 8 LSBs of the data (2400H), which is loaded into register L. The 3rd byte 24 is 8 MSBs of the data (2400H) which is loaded into register H.

② LDA 2500H ;- get the content of the memorey location 2500H into accumulator.

3A, 00, 25 :- LDA 2500H in the code form

The 1st byte 3A is the opcode fore the instruction LDA. The 2nd byte 00 is 8 LSBs of the address of the memorey location 2500H. The 3rd byte 25 is 8 MSBs of the address of the memorey location 2500H. In this instruction the data is the content of the memorey location ~~2500~~ 2500H. The data is to be loaded in to the accumulator.

A 3 byte instruction is stored in three ~~consecutive~~ consecutive memory locations.

Addressing Modes

Each instruction requirees certain data on which it has to operate. It has already been explained that there are various techniques to specify data for instructions. These techniques are called addressing modes. Intel 8085 uses the following addressing modes.

1. Direct addressing
2. Register addressing
3. Register indirect addressing
4. Immediate addressing.

Direct addressing

In this mode of addressing the address of operand (data) is given in the instruction it self.

Example

1. STA 2400H Store the content of accumulator in the memory location 2400H.

   32, 00, 24 The above instruction in the code form.

In this instruction 2400H is the memory address where data is to be stored. It is given in the instruction itself. The 2nd & 3rd bytes of the instruction specify the address of the memory location. Here it is understood that the source of the data is accumulator.

2- IN 02 Read data from the port C

DB, 02 Instruction in the code form.

In this instruction 02 is the address of the port C of an I/O port from where the data is to be read. Here it is implied that the destination is the accumulator. The 2nd byte of the instruction specifies the address of the port.

Register Addressing

In register addressing mode the operand is in one of the general purpose ~~reesig~~ register. The opcode specifies the address of the register in addition to the operation to be performed.

① MOV A, B Move the content of register B to register A.

78 The instruction in the code form

② ADD B Add the content of register B to the content of register A.

80 The instruction in the code form.

In Example, the opcode for MOV A, B is 78H. Besides the operation to be perform the opcode also specifies source and destination registers. The opcode 78H can be written in binary form as 01111000. The first two bits i.e 01 are for MOV operation, the next three bits 111 are the binary code for register A, the last three bits 000 are the binary code for register B.

In Example, the opcode for ADD B is 80H. In this instruction one of the operands is register B (its content is one of the data) which is indicated in the instruction itself In this type of instruction it is understood that the other operand is in the accumulator. The opcode 80H in the binary form is 1000 0000. The first five bits 10000 specify the operation to be performed ADD. The last three bits 000 are the binary code for register B for 8085 microprocessor.

Register Indirect Addressing

In this mode of Addressing the address of the operand is specified by a register pair. Examples are

① LXI H, 2500 H Load H-L pair with 2500H
MOV A, M Move the content of the memory location whose address is in H-L pair to the accumulator.
HLT Halt.

In the above program the instruction MOV A,M is an example of register indirect addressing for this instruction the operand is in the memory. The address of the memory is not directly given in the instruction. The address of the memory resides in H-L pair and this has already been specified by an earlier instruction in the program.

② LXI H, 2500H Load the H-L pair with 2500H
ADD M Add the content of the memory location whose address is in H-L pair to content of the accumulator.
HLT HAIT.

Immediate Addressing :-

In immediate addressing mode the operand is specified with in the instruction itself. examples are.

① MVI A, 05 Move 05, in register A.
3E, 05 The instruction in the code form

② ADI 06 Add 06 to the content of the accumula
C6, 06 The instruction in the code form.

③ LXI H, 2500 is an example of immediate addressing.
2500 is 16 bit data which is given in the instruction itself. It is to be loaded into H-L paire.

Implicit Addressing

There are certain instruction which operate on the content of the accumulator. Such instruction do ~~not~~ not recquirred the address of the operand.
Examples are :- CMA, RAI, RAR etc.

Intel 8085 Instructions

Some of Intel 8085 instructions are freequently some occasionally and some seldom used by the programmer. It is not necessary that one should learen all the instructions to understand simple program. The explanations of the most instructions are given in the subsequent sub sections.

Data Transfer Group | MOV r_1, r_2

Move data; Move the content of the one register to another.

$[r_1] \leftarrow [r_2]$ States: 4, flags: none, Addressing: register, Machine cycle: 1.

The content of register r_2 is moved to register r_1. For example, the instruction MOV A, B moves the content of register B to register A. The instruction MOV B, A moves the content of register A to register B. The time for the execution of this instruction is 4 clock period. One clock period is called state. No flag is affected.

Move r, M (move the content of memory to register)

$[r] \leftarrow [[H-L]]$ States: 7, flag none, Addressing: register indirect, Machine cycles: 2.

The content of the memory location, whose address is in H-L pair, is moved to register r.

Example

LXI H, 2000H	Load H-L pair by 2000H
MOV B, M	Move the content of the memory location 2000H to register B.
HLT	Halt.

In this example the instruction LXI H, 2000H loads H-L pair with 2000H. which is the address of a memory location. Then the instruction MOV B, M will move the content of the memory location 2000H to register B.

MOV M, r (Move the content of register to memory)

[[H-L]] ← [r] states: 7, flags: none

Addressing: reg. indirect, Machine cycles: 2

The content of register r is moved to the memory location addressed by H-L pair. for example MOV M, C moves the content of register C to memory location whose address is in H-L pair.

MVI r, data (Move immediate data to register)

[r] ← data states: 7, flags: none, Addressing: Immediate, Machine cycle: 2.

The 1st byte of the instruction is its oprode. The 2nd byte of instruction is the data which is moved to register r. for example, the instruction MVI A, 05 moves 05 to register A. In the code form it is written as ~~3E~~ @3E, 05. The oprode for MVI A is 3E and 05 is the data which is to be moved to register A.

MVI M, data (Move immediate data to memory)

[[H-L]] ← data States : 10 flags : none, Addressing : immediate or reg. indirect, Machine cycle: 3

The data is moved to memory location whose address is in H-L pair.

Example :-

```
LXI H, 2400H    Load H-L pair with 2400H
MVI M, 08       Move 08 to the memory location 2400H.
HLT             Halt.
```

LXI H, 2400H Loads H-L pair with 2400H which is the address of a memory location. Then the instruction MVI M, 08 will move 08 to memory location 2400H. In the code form it is written as 36, 08.

The opcode for MVI M is 36 and 08 is the dat which is to be moved to the memory location. 2400H.

LXI rp, data 16 (Load register pair immediate)

[rp] ← data 16 bits [rh] ← 8 MSBs, [rl] ← 8 LSBs of Data

states : 10, flags : none, Addressing : immediate, machine cycle : 3.

This instruction loads 16-bit immediate data into register pair rp. This instruction is for register pair ; only high order register is mentioned after the instruction. for example.

H in the instruction LXI H stands for H-L pair. Similarly LXI B, is for B-C pair.

LDA addr (Load Accumulator direct)

[A] ← [addr]. States: 13, flags: none, Addressing: direct, Machine cycle: 4.

The content of the memory location, whose address is specified by the 2nd and 3rd bytes of the instruction is loaded into the accumulator.

The instruction LDA 2400H will load the content of memory location 2400H into the accumulator.

STA Addr. (Store accumulator direct)

[addr] ← [A]. States: 13, flags: none. Addressing: direct, Machine cycle: 4.

The content of the accumulator is stored in the memory location whose address is specified. the 2nd and 3rd byte of instruction. STA 2000H will store the content of the accumulator in the memory location 2000H.

LHLD addr (Load H-L pair direct)

[L] ← [addr] [H] ← [addr + 1]. States: 16, flags: none Addressing: direct, Machine cycles: 5.

The content of the memory location, whose address is specified by the 2nd and 3rd

bytes of the instruction, is loaded into register L. The content of the next memory location is loaded into register H.

SHLD addr (Store H-L pair direct)

[addr] ← [L], [addr+1] ← H. States: 16, flags: none, Addressing: direct, Machine cycles: 5.

The content of register L is stored in the memory location whose address is specified by the 2nd and 3rd bytes of the instruction. The content of register H is stored in the next memory location.

LDAX rp (Load accumulator indirect)

[A] ← [[rp]]. States: 7, flags: none, Addressing: register indirect, Machine cycle: 2.

The content of memory location, whose address is in the register pair rp, is loaded into the accumulator. For example LDAX B will load the content of the memory location, whose address is in the B-C pair, into the accumulator. This instruction is used only for B-C and D-E register pairs

STAX rp (Store accumulator indirect)

[[rp]] ← [A]. States: 7, flags: none, Addressing: register indirect, machine cycles: 2.

The content of the accumulator is stored in the memory location whose address is in the register pair rep. for example STAX D will store the content of the accumulator in the memory location whose address is in D-E pair. This instruction is true only for register pairs B-C & D-E.

XCHG. (Exchange the contents of H-L with DE pair)

[H-L] ↔ [D-E]. states: 4, flags: none, Addressing: register, Machine cycles: 1

The contents of [H-L] pair are exchanged with contents of D-E pair.

Arithmetic Group

ADD r (Add register to accumulator)

[A] ← [A] + [r]. States: 4, flags: all, Addressing: reg. indirect, Machine cycle: 1.

The content of register r is added to the content of the accumulator and the sum is placed in the accumulator.

~~ADD M~~

ADD M (Add memory to accumulator)

[A] ← [A] + [[H-L]] . States: 7, flags: all,
Addressing: reg. indirect, Machine cycles: 2.
The content of the memory location addressed by H-L paire is added to the content of the accumulator. The sum is placed in the accumulator.

ADC re: (Add register with carrey to accumulator)

[A] ← [A] + [re] + [cs] states: 4, flags: all
Addressing: register, Machine cycles: 1
The content of register re and carrey status are added to the content of accumulator. The sum is placed in the accumulator.

ADC M (Add memory with carrey to accumulator)

[A] ← [A] + [[H-L]] + [cs] . States: 7, flags: all
Addressing: reg. indirect. Machine cycles: 2.
The content of the memory location addressed by H-L paire and carrey status are added to the content of the accumulator. The sum is placed in the accumulator.

ADI data (Add immediate data to accumulator)

[A] ← [A] + data . states: 7, flags: all, Addressing: immediate, Machine cycles: 2
The immediate data is added to the content of the accumulator. The 1st byte

of the instruction is its opcode. The 2nd byte of the instruction is data, and it is added to content of accumulator. The sum is placed in the accumulator.

ACI data (Add with carry immediate data to accumulator)

[A] ← [A] + data + [CS]. states: 7, flags: all Addressing: immediate Machine cycles: 2.

The 2nd byte of the instruction and the carry status are added to the content of the accumulator. The sum is placed in the accumulator.

DAD rep (Add register pair to H-L pair)

[H-L] ← [H-L] + [rep]. States: 10, flags: CS Addressing: register, machine cycles: 3

The contents of register pair rep are added to the contents of H-L pair and the result is placed in H-L pair. only carry flag is affected.

SUB re (subtract register from accumulator)

[A] ← [A] - [re]. states: 4, flags: all, Addressing: register, Machine cycles: 1.

The content of register re is subtracted from the content of the

accumulator and the result is placed in the accumulator.

SUB M (Subtract memory from accumulator)

[A] ← [A] - [[H-L]] States: 7, flags: all, Addressing: reg. indirect, Machine cycles: 2.

The content of the memory location addressing by H-L pair is subtracted from the content of the accumulator. The result is placed in the accumulator.

SBB r (Subtract register from accumulator with borrow)

[A] ← [A] - [r] - [CS] state: 4, flags: all Addressing: register. Machine cycles: 1.

The content of register r and carry status are subtracted from the content of the accumulator. The result is placed in the accumulator.

SBB M (Subtract memory from accumulator with borrow)

[A] ← [A] - [[H-L]] - [CS]. States: 7, flags: all, Addressing: reg. indirect. Machine cycles: 2.

The content of the memory location addressing by H-L pair and carry status are subtracted from the content of the accumulator. The result is placed in the accumulator.

SUI data (Subtract immediate data from accumulator)

[A] ← [A] - data. States: 7, flags: all, Addressing: immediate, Machine cycle: 2.

The 2nd byte of the instruction is data. It is subtracted from the content of the accumulator. The result is placed in the accumulator.

SBI data (Subtract immediate data from accumulator with borrow)

[A] ← [A] - data - [CS] states: 7, flags: all Addressing: immediate, Machine cycles: 2.

The data any carry status are subtracted from the content of the accumulator. The result is placed in the accumulator.

INR r (Increment register content)

[r] ← [r] + 1 states: 4, flags: all except ~~Carry~~ carry flags, Addressing: register, Machine cycle: 1.

The content of register r is incremented by one. All flags except CS are affected.

INR M (Increment memory content)

[[H-L]] ← [[H-L]] + 1 States: 10, flags: all except carry flag. Addressing: reg. indirect, Machine cycle: 3.

The content of the memory location addressing by H-L pair is incremented by one. All flags except CS are affected.

DCR r (Decrement register content)

[r] ← [r] - 1. States: 4, flags: all except carry flag, Addressing: register, Machine cycles: 1.

The content of register r is decrement by one. All the flags except CS are affected.

DCR M (Decrement memory content)

[[H-L]] ← [[H-L]] - 1 States: 10, flags: all except carry flags, Addressing: reg. indirect, Machine cycles: 3.

The content of the memory location address by H-L pair is decremented by one. All flags except CS are affected.

INX rp (Increment register pair)

[rp] ← [rp] + 1. States: 6. flags: none. Addressing: register, Machine cycles: 1.

The content of the register pair rp is increment by one. No flag is affected.

DCX rp (Decrement register pair)

[rp] ← [rp] - 1. States: 6, flags: none. Addressing: register. Machine cycles: 1.

The content of the register pair rp is decremented by one. No flag is affected.

DAA (Decimal adjust accumulator)

states: 4, flags: all, Machine cycle: 1.

The instruction DAA is used in the program after ADD, ADI, ACI, ADC, etc instructions. After the execution of ADD, ADC, etc instructions the result is in hexadecimal and it is placed in the accumulator.

The DAA instruction operates on this result and gives the final result in decimal system.

Logical Group :-

The instruction of this group perform AND, OR, Exclusive - OR operations, compare, rotate or take complement of data in register or memory.

ANA r (AND register with accumulator)

[A] ← [A] ^ [r]. States : 4, flags : all, Addressing : register, Machine cycles : 1.

The content of register r is ANDed with the content of the accumulator and the result is placed in the accumulator. All status flags are affected. The flags Carry status is cleared i.e it is set to 0. Auxiliary Carry flag AC is set to 1.

ANA M (AND memory with accumulator)

[A] ← [A] ^ [[H-L]]. States : 7, flags : all, Addressing : reg. indirect, Machine cycles : 2

The content of the memory location addressed by H-L pair is ANDed with the accumulator. The result is placed in the accumulator. All flags are affected. The CS flags is set to 0 and AC to 1.

ANI data (AND immediate data with accumulator)

[A] ← [A] ∩ [[H-L]] . States : 7, flags : all, Addressing : reg. indirect, Machine cycles : 2.

The 2nd byte of the instruction is data and it is ANDed with the content of the accumulator. The result is placed in the accumulator. The CS flag is set to 0 and AC to 1.

ORA r (OR register with Accumulator)

[A] ← [A] ∨ [r] . States : 4, flags : all, Addressing : register, Machine cycles : 1.

The content of register r is ORed with the content of the accumulator. The result is placed in the accumulator. All status flags are affected. Carry and auxiliary Carry are cleared. i.e the CS and AC flags are set to 0.

ORA M (OR memory with accumulator)

[A] ← [A] ∨ [[H-L]] . States : 7, flags : all Addressing : reg. immediate, Machine cycles : 2.

The 2nd byte of the instruction is data and it's ORed with content of the accumulator. The result is placed in the accumulator. The CS and AC flags are set to 0.

ORI data (OR immediate data with accumulator)

[A] ← [A] ∨ data. States: 7, flags: all, Addressing: immediate, Machine cycles: 2.

The 2nd byte of the instruction is data and it is ORed with the content of the accumulator. The result is placed in the accumulator. All status flags are affected. The CS and AC flags are set 0.

XRA r (Exclusive OR register with accumulator)

[A] ← [A] ∀ [r]. States: 4, flags: all, Addressing: register, Machine cycles: 1.

The content of register r is Exclusive-ORed with the content of the accumulator. The result is placed in the accumulator. All status flags are affected. The CS and AC flags are set to '0'.

XRAM (Exclusive-OR memory with accumulator)

[A] ← [A] ∀ [[H-L]] States: 7, flags: all, Addressing: reg. indirect, Machine cycles: 2.

The content of the memory location addressed by H-L pair is Exclusive-ORed with the content of the accumulator. The result is placed in the accumulator. All status ~~fed~~ flags are affected. The CS and AC flags are set to 0.

XRI data (Exclusive-OR immediate data with accumulator)

[A] ← [A] ∀ data. States: 7, flags: all, Addressing: immediate, Machine cycles: 2

The 2nd byte of the instruction is data and it is Exclusive-ORed with the content of the accumulator. The result is placed in the accumulator. All flags are affected. The CS and AC flags are set to '0'.

CMA (Complement the accumulator)

[A] ← [$\bar{A}$]. States: 4, flags: none, Machine cycles: 1 Addressing: implicit.

1's complement of the content of the accumulator is obtained, and the result is placed in the accumulator. To obtain the 1's complement of a binary number 0 is replaced by 1 and 1 by 0 The example one's complement of 1100 is 0011

CMC. (Complement the carrey status)

[CS] ← [$\overline{CS}$] states: 4, flags: CS, Machine cycle: 1

The CS flag is complemented other flags are not affected.

STC (set carrey Status)

[CS] ← 1. States: 4. flags: CS, Machine cycles: 1

The status flags CS is set to 1. other flags are not affected.

CMP r (Compare register with accumulator)

[A] ← [r], States: 4, flags: all, Addressing: register, Machine cycles: 1.

The content of register r is subtracted from the content of the accumulator and status flags are set according to the result of the subtraction. But the result is discarded. The content of the accumulator remains unchanged.

CMP M (compare memory with accumulator)

[A] ← [H-L]. States: 7, flags: all, Addressing: reg. indirect, Machine cycles: 2.

The content of the memory location addressed by H-L pair is subtracted from the content of the accumulator. and status flags are set according to the result of the subtraction. But the result is discarded. The content of the accumulator remains unchanged.

CPI data (Compare immediate data with accumulator)

[A] ← data. States: 7, flags: all, Addressing: immediate, Machine cycles: 2.

The 2nd byte of the instruction is data. and it is subtracted from the content

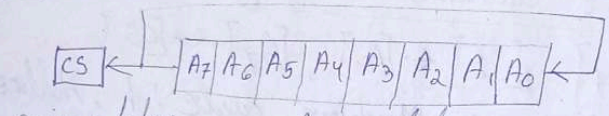
of the accumulator. The status flags are set according to the result of subtraction. But the result is discarded. The content of the accumulator remains unchanged.

RLC (Rotate accumulator left)

$[A_{n+1}] \leftarrow [A_n], [A_0] \leftarrow [A_7], [CS] \leftarrow [A_7]$.

States: 4, flags: CS, Machine cycles: 1, Addressing: implicit.

The content of the accumulator is rotated left by one bit. The seventh bit of the accumulator moved to carrey bit as wellas to the zero bit of the accumulator. only CS flag is affected.

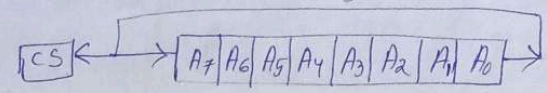


RRC (Rotate accumulator right)

$[A_7] \leftarrow [A_0], [CS] \leftarrow [A_0], [A_n] \leftarrow [A_{n+1}]$

States: 4 flags: CS Machine cycles: 1, Addressing: implicit.

The content of the accumulator is rotated right by one bit. The zero bit of the accumulator is moved to the seventh bit as wellas to carrey bit. Only CS flags is affected.

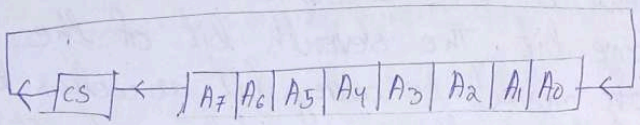


RAL (Rotate accumulator left through carry)

$[A_{n+1}] \leftarrow [A_n]$, $[CS] \leftarrow [A_7]$, $[A_0] \leftarrow [CS]$

States: 4, flags: CS, Machine cycles: 1, Addressing: implict.

The content of the accumulator is rotated left one bit through carry. The seventh bit of the accumulator is moved to carry and the carry bit is moved to the zero bit of the accumulator only carry flag is affected.

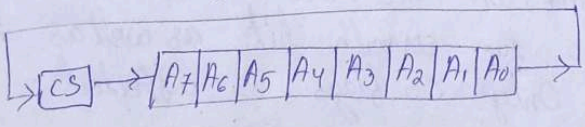


RAR (Rotate accumulator right through Carry)

$[A_n] \leftarrow [A_{n+1}]$, $[CS] \leftarrow [A_0]$, $[A_7] \leftarrow [CS]$

States: 4, flags: CS, Machine cycle: 1, Addressing: implicite.

The content of the accumulator is rotated right one bit through carry. The zero bit of the accumulator is moved to carry and the carry bit to the seventh bit of the accumulator. only CS flag is affected.



Branch Group

There are two types of branch instructions: conditional and unconditional. The conditional branch instruction transfer the program to the specified label when certain condition is statified. The unconditional branch instructions transfer the program to the specified ~~label~~ label unconditionally.

JMP addr (label) (unconditional jump: jump to the instruction specified by the address)

[PC] ← Label states: 10, flags: none, Addressing: immediate, Machine cycles: 3.

Byte 2nd and byte 3rd of the instruction give the address of the label where the program jumps the address of the label is the address of the memory location fore next instruction to be executed. The program jumps to the instruction specified by the address (label) unconditionally.

Conditional Jump addr (label)

After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled.

the program proceeds further in the normal sequence if the specified condition is not fulfilled.

JZ addre (label) (Jump if the result is zero)

[PC] ← address (label) jump if z=1.
states : 7/10, flags : none, Addressing : immediate, Machine cycles : 2/3.

The program jumps to the instruction specified by the address if the result is zero. Here the result after the execution of the preceding instruction is under consideration.

JNZ addre (label) (Jump if the result is not zero.)

[PC] ← address (label), jump if z=0. States: 7/10 flags: none, Addressing : immediate, Machine cycle: 2/3

The program jumps to the instruction specified by the address (label) if the result is non-zero (i.e the zero status z=0)

JC addre (label) (Jump if there is a carry)

[PC] ← address label, jump if CS=1,
States : 7/10, flags : none, Addressing : immediate, Machine cycles : 2/3.

The program jumps to the instruction specified by the address (label) if

there is a carry (i.e the carry status CS=1). Here the carry after the execution of the preceding instruction is under consideration.

INC addr (label) (Jump if there is no carry)

[PC] ← address (label) jump if CS=0
States: 7/10. Flags: none. Addressing: immediate. Machine cycles: 2/3.
The program jumps to the instruction specified by the address (label) if there is no carry (i.e the carry states CS=0)

JP addr (label) (Jump if the result is plus)

[PC] ← address (label), jump if S=0, States: 7/10
flags: none. Addressing: immediate. Machine cycles: 2/3.
The program jumps to the instruction specified by the address (label) if the result is plus.

JM addr (label) (Jump if the result is minus)

[PC] ← address label), jump if S=1, states: 7/10
flags: none, Addressing: immediate, Machine cycles: 2/3.
If the result is minus the program jumps to the instruction specified by

the address (label)

JPE addr (label) (Jump if even parity)

[PC] ← address (label), jump if even parity, the parity status P = 1, states: 7/10, flags: none, Addressing: immediate, Machine cycles: 2/3. If the result contains even number of 1's, the program jumps to the instruction specified by address (label)

JPO addr (label) (Jump if odd parity)

[PC] ← address (label), jump if odd parity status P = 0, states: 7/10, flags: none Addressing: immediate, Machine cycles: 2/3. If the result contains odd number of 1's, then program jumps to the instruction specified by the address (label).

Assembly Language Programs

Program-1

Get 05 is register A; then move it to register B.

Solution

memory Address	Machine codes	Mnemonics	operands	comments
2000	3E, 05	MVI	A, 05	→ Get 05 in register A.
2002	47	MOV	B, A	→ Transfer 05 from register A to B
2003	76	HLT	Halt	→ stop.

Program -2

Place 05 in the accumulator. Increment it by one store the result in the memory location 2550 H.

Sol^n

memory address	machine code	mnemonics	operands	comments
2000	3E, 05	MVI	A, 05	Get 05 in Accumulator.
2002	3 6	INR	A	Increment the content of Accumulator by one
2003	32, 50, 25	STA	2550 H	→ Store result in 2550H
2006	76	HLT	Halt	Stop

→ The instruction MVI A, 05 moves 05 to the accumulator. INR A increases the content

of the accumulator from 05 to 06. STA 2550 stores the content of the accumulator in the memory location 2550 H. After the execution of the above program the memory location 2550 H will contain 06.

Problem-3
Addition of two 8-bit Numbers: Sum 8 Bits.

Soln
The 1st number 49H is in the memory location 2501 H.
The 2nd number 56H is in the memory location 2502 H. The result is to be stored in the memory location 2503 H.

memory address	machine codes	mnemonics	operands	comments
2000	21, 01, 25	LXI	H, 2501H	→ Get address of 1st number in H-L pair
2003	7E	MOV	A, M	→ 1st number in accumulator
2004	23	INX	H	→ Increment content of HL pair
2005	86	ADD	M	→ Add 1st and 2nd no.
2006	32, 03, 25	STA	2503H	→ Store sum in 2503H
2009	76	HLT	Halt	→ Stop.

Data 2501 — 49H
2502 — 56H

The sum is stored in the memory location 2503 H 2503 — 9FH.

Problem-4

Subtraction of Two 8 bit numbers, and result is stored in 8 bit.

Soln

The 1st number 49H is in the memory location 2501H

The 2nd number 32H is in the memory location 2502H

The result is to be stored in the memory location 2503H.

memory address	machine codes	Mnemonics	operands	comments
2000	21, 01, 25	LXI	H, 2501H	→ Get address of 1st number in H-L pair.
2003	7E	MOV	A, M	→ 1st number in accumulator
2004	23	INX	H	→ content of H-L pair increases from 2501 to 2502H
2005	96	SUB	M	→ 1st number − 2nd no
2006	23	INX	H	→ content of H-L pair becomes 2503H
2007	77	MOV	M, A	→ store result in 2503H
2008	76	HLT	Halt	→ stop.

Data 2501 → 49 H

2502 → 32 H

Result 2503 → 17 H

Problem-5

find One's complement of An 8-bit 96H number.

Soln

The no. in the binary form is represented as

96H = 1001 0110

9 6

1's complement = 0110 1001

6 9

The number is placed in the memory location 2501 H. The result is stored in the memory location 2502 H.

memory Address	Machine codes	mnemonics	operands	comments
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator
2003	2F	CMA		Take it's complement
2004	32, 02, 25	STA	2502H	store result in 2502 H
2007	76	HLT	Halt	Stop.

Data 2501 - 96 H
Result 2502 - 69 H (1's complement of An 8bit no. - 96 H)

Problem-6
Find the 2's complement of An 8-bit number. 96.

Solution

96 = 1001 0110
1's complement → 0110 1001
+ 1
2's complement = 0110 1010 = 6AH.
6 A

The number is placed in the memory location 2501 H. The result is to be stored in the memory location 2502 H.

memory Address	machine codes	Mnemonics	operands	comments
2000	3A, 01, 25	LDA	2501H	→ Get data in accumulator
2003	2F	CMA		→ Take it's 1's complement
2004	3C	INR	A	→ Take 2's complement
2005	32, 02, 25	STA	2502H	→ store result in 2502H
2008	76	HLT	Halt	→ Stop.

Problem-7

Shift An 8-bit number LEFT By one Bit the number 65H.

Solution

The binary representation of 65 is given below.

65 = 0110 0101
6 5

shift 1 bit we have Result is → 1100 1010
C A

To shift a number left by one bit the no. is added to it self. If 65 is added to it self. If 65 is added to 65 the result is CA.

memory Address	machine codes	mnemonics	operands	comments
2000	3A, 01, 25	LDA	2501H	→ Get data in Accumulator.
2003	87	ADD	A	→ shift it left by one bit.
2004	32, 02, 25	STA	2502H	→ Store result in 2502H
2007	76	HLT	Halt	→ Stop.

Data 2501 → 65H

Result → 2502 → CAH.

Problem-8

To find Large of Two 8 bit Numbers, given 2 nos are 98H and 87H.

Sol^n

The 1st number 98H is placed in the memory 2501H.

The 2nd number 87H is placed in the memory 2502.

The result is stored in the memory 2503H.

memory Address	machine codes	Labels	Mnemonics	operands	comments
~~2000~~	~~21,01,25~~	~~LXI~~	~~H, 2501H~~		
2000	21,01,25		LXI	H, 2501H	Address of 1st no
2003	7E		MOV	A, M	1st no in Accumulator
2004	23		INX	H	Address of 2nd no.
2005	BE		CMP	M	compare 2nd no with 1st no. Is the 2nd no > 1st no?
2006	D2,0A,20		JNC	AHEAD	→ No, Large no is in the accumulator. Go to AHEAD.
2009	7E		MOV	A, M	→ Yes, get 2nd no. in Accumulator
200A	32,03,25	AHEAD	STA	2503H	→ store large number in 2503 H.
200D	76		HLT		Halt → STOP.

Data 2501 → 98H

2502 → 87H

Result → 98H.

Problem-9

To Find the largest Number in A DATA ARRAY.

Soln The number in a series are 98, 75 and 99. As there are three numbers in the series, the count = 03. The count is placed in the memory location 2500H. The numbers are placed in the memory location 2501 to 2503H. The result is to be stored in the memory location 2450H.

memory address	machine codes	Labels	Mnemonics	operands	Comments
2000	21,0025		LXI	H, 2500H	Address fore count in H-L pair.
2003	4E		MOV	C, M	count in register C.
2004	23		INX	H ⟶	Address of 1st no is H-L pair.
2005	7E		MOV	A, M ⟶	1st no in accumulator
2006	0D		DCR	C	⟶ Decrement count.
2007	23	LOOP	INX	H	→ Add of next No.
2008	BE		CMP	M	→ compare next no. with previous maximum. Is next no > previous no.
2009	D2,0D,20		JNC	AHEAD	→ No, large no is in accumulator. Go to the Lable AHEAD.
200C	7E		MOV	A, M	→ Yes, get large number in accumulator.
200D	0D	AHEAD	DCR	C	→ Decrement count.
200E	C2,07,20		JNZ	Loop	
2011	32,50,24		STA	2450H	→ Store result in 2450H
2014	76		HLT	Halt	→ STOP.

Data → 2500-03
2501-98
2502-75
2503-99

Result 2450-99.

Problem-10

8 bit Multiplication; produced 16-bit.

Soln

Given Multiply 84H by 56H.
In this case multiplicand is 84H. It is extended to 16-bits and stored in the two consecutive memory location 2501 and 2502. The multiplier is 56H is stored in 2503H.
The product is a 16 bit number and it is stored in 2504H and in 2505H.

memory Address	Machine codes	Label	Mnemonics	operands	comments
2000	2A, 01, 25		LHLD	2501 H	→ Get multiplicand in H-L pair
2003	EB		XCHG		→ Multiplier in Accumulator.
2004	3A, 03, 25		LDA	2503 H	→ Multiplier in Accumulator
2007	21, 00, 00		LXI	H, 0000	→ Initial value of product is 00 in H-L pair.
200A	0E, 08		MVI	C, 08	→ count 8 in register C.
200C	29	LOOP	DAD	H	→ shift partial product left by 1 bit.
200D	17		RAL		→ Rotate multiplier left one bit Is multiplier's bit=1?
200E	D2, 12, 20		JNC	AHEAD	→ No, go to AHEAD
2011	19		DAD	D	→ Product = product + Multiplicand.
2012	0D	AHEAD	DCR	C	→ Decrement count
2013	C2, 0C, 20		JNZ	loop	
2016	22, 04, 25		SHLD	2504H	→ store result
2019	76		HLT	Halt	→ stop.

Data

2501 → 84H, LSBs of multiplicand
2502 → 00H, MSBs of multiplicand
2503 → 56H, multiplier

Result → 2504 → 58H, LSBs of product
2505 → 2CH, MSBs of product.

Microprocessor Based System Development Aids :-

In Intel 8085 uses a 16-bit wide address bus for addressing memories and I/O devices. Using 16-bit wide address bus it can access 2^{16} = 64k bytes of memory and I/O devices. The 64k addresses are to be assigned to memories and I/O devices for their addressing. There are two schemes for the allocation of addresses to memories and input/output devices.

1. Memory mapped I/O scheme.
2. I/O Mapped I/O Scheme.

Memory mapped I/O scheme :- In memory mapped I/O scheme there is only one address space. Address space is defined as the set of all possible address that a microprocessor can genereate. Some address are assigned to memories and some address to I/O devices.

An I/O devices is also treated as a memory location and one address is assigned to it. The addresses for I/O devices are different from the addresses which have been assigned to memories.

The addresses which have not been assigned to memories can be assigned to I/O devices.

I/O Mapped I/O Scheme:-

In this scheme the address assigned to memory locations can also be assigned to I/O devices. Since the same address may be assigned to a memory location or an I/O device, the microprocessor must issue a signal to distinguish whether the address on the address bus is for a memory location or an I/O device.

EPROM

An EPROM is an erasable Programmable Read only Memory. The contents of EPROM can be erased by exposing it to high intensity short wave ultraviolet light of 2537 A° for 10 to 20 minutes. An EPROM eraser has a UV tube of 2537 A°, built in 60 minutes timer and auto shut off facility. When the set time is over the tray is pulled out. The timer automatically switches off the UV tube.

A Spring loaded timer is provided to set the time depending on the type of EPROM to be erased. The timer automatically turns off the light source after a present time. The eraser requirred a 230 V AC ± 10% supply for its operation.

RAM (Random Access Memory)

It is the internal memory of the CPU for storing data, program and Program result. It is a read/write memory which stores data until the machine is switched off, data is erased.

Access time in Ram is independent of addrees, that is, each storage location inside the memory is an easy to reach as other location and takes the same amount of time. Data in the Ram can be accessed randomly but it is very expensive.

Ram is volatile i.e data stored in it is lost when we switch off the computer or if there is a power failure. Hence a back-up Uninterruptible power system

(UPS) is often used with computer, RAM is small, both in terms of the its physical size and in the amount of data it can hold.