Microprocessor 8085

Addressing modes

Types of addressing modes –

In 8085 microprocessor there are 5 types of addressing modes:

1. **Immediate Addressing Mode –**
   In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.
   Examples:
   MVI B 45 (move the data 45H immediately to register B)
   LXI H 3050 (load the H-L pair with the operand 3050H immediately)
   JMP address (jump to the operand address immediately)
2. **Register Addressing Mode –**
   In register addressing mode, the data to be operated is available inside the register(s) and register(s) is(are) operands. Therefore the operation is performed within various registers of the microprocessor.
   Examples:
   MOV A, B (move the contents of register B to register A)
   ADD B (add contents of registers A and B and store the result in register A)
   INR A (increment the contents of register A by one)
3. **Direct Addressing Mode –**
   In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself.
   Examples:
   LDA 2050 (load the contents of memory location into accumulator A)
   LHLD address (load contents of 16-bit memory location into H-L register pair)
   IN 35 (read the data from port whose address is 01)
4. **Register Indirect Addressing Mode –**
   IN register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified b a register pair.
   Examples:
   MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)
   LDAX B (move contains of B-C register to the accumulator)
   LXIH 9570 (load immediate the H-L pair with the address of the location 9570)
5. **Implied/Implicit Addressing Mode –**
   In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.
   Examples:
   CMA (finds and stores the 1's complement of the contains of accumulator A in A)
   RRC (rotate accumulator A right by one bit)
   RLC (rotate accumulator A left by one bit)

UNIT II

8085 instruction set and classifications:

Data-transfer instructions

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination(Dt)	This instruction copies the contents of the source register into the destination register without any alteration. Example – MOV K, L
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. Example – MVI K, 55L
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. Example – LDA 2034K
LDAX	B/D Reg. pair	Load the accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. Example – LDAX K
LXI	Reg. pair, 16-bit data	Load the register pair immediate	The instruction loads 16-bit data in the register pair designated in the register or the memory.

			second byte specifies the low-order address and the third byte specifies the high-order address. Example – SHLD 3225K
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example – XCHG
SPHL	None	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register. The contents of the H register provide the high-order address and the contents of the L register provide the low-order address. Example – SPHL
XTHL	None	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1). Example – XTHL
PUSH	Reg. pair	Push the register pair onto the stack	The contents of the register pair designated in the operand are copied onto the stack in the

			Example – LXI K, 3225L
LHLD	16-bit address	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. Example – LHLD 3225K
STA	16-bit address	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example – STA 325K
STAX	16-bit address	Store the accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand. Example – STAX K
SHLD	16-bit address	Store H and L registers direct	The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. This is a 3-byte instruction, the

IN	**8-bit port address**	**Input data to accumulator from a port with 8-bit address**	**The contents of the input port designated in the operand are read and loaded into the accumulator.** **Example – IN5KL**

Arithmetic instructions

Opcode	Operand	Meaning	Explanation
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. **Example** – ADD K.
ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. **Example** – ADC K
ADI	8-bit data	Add the immediate to the accumulator	The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator. **Example** – ADI 55K
ACI	8-bit data	Add the	The 8-bit data and the Carry

			following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. Example – PUSH K
POP	Reg. pair	Pop off stack to the register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. Example – POPK
OUT	8-bit port address	Output the data from the accumulator to a port with 8bit address	The contents of the accumulator are copied into the I/O port specified by the operand. Example – OUT K9L

		immediate to the accumulator with carry	flag are added to the contents of the accumulator and the result is stored in the accumulator. **Example** – ACI 55K
LXI	Reg. pair, 16bit data	Load the register pair immediate	The instruction stores 16-bit data into the register pair designated in the operand. **Example** – LXI K, 3025M
DAD	Reg. pair	Add the register pair to H and L registers	The 16-bit data of the specified register pair are added to the contents of the HL register. **Example** – DAD K
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. **Example** – SUB K
SBB	R M	Subtract the source and borrow from the accumulator	The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. **Example** – SBB K
SUI	8-bit data	Subtract the immediate from	The 8-bit data is subtracted from the contents of the

		the accumulator	accumulator & the result is stored in the accumulator. **Example** – SUI 55K
SBI	8-bit data	Subtract the immediate from the accumulator with borrow	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. **Example** – XCHG
INR	R M	Increment the register or the memory by 1	The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. **Example** – INR K
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. **Example** – INX K
DCR	R M	Decrement the register or the memory by 1	The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. **Example** – DCR K
DCX	R	Decrement the	The contents of the designated

ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and the result is placed in the accumulator.
XRA	R M	Exclusive OR register or memory with the accumulator	The contents of the accumulator are Exclusive OR with M the contents of the register or memory, and the result is placed in the accumulator.
XRI	8-bit data	Exclusive OR immediate with the accumulator	The contents of the accumulator are Exclusive OR with the 8-bit data and the result is placed in the accumulator.
ORA	R M	Logical OR register or memory with the accumulator	The contents of the accumulator are logically OR with M the contents of the register or memory, and result is placed in the accumulator.
ORI	8-bit data	Logical OR immediate with the accumulator	The contents of the accumulator are logically OR with the 8-bit data and the result is placed in the accumulator.
RLC	None	Rotate the accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.
RRC	None	Rotate the accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.
RAL	None	Rotate the accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.
RAR	None	Rotate the accumulator right	Each binary bit of the accumulator is rotated right by one position through the Carry flag.

		through carry	Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0.
CMA	None	Complement accumulator	The contents of the accumulator are complemented. No flags are affected.
CMC	None	Complement carry	The Carry flag is complemented. No other flags are affected.
STC	None	Set Carry	Set Carry

Branching instructions

Opcode	Operand	Meaning	Explanation
JMP	**16-bit address**	Jump unconditionally	The program sequence is transferred to the memory address given in the operand.
RET	**None**	Return from subroutine unconditionally	The program sequence is transferred from the subroutine to the calling program.

Control instructions

Opcode	Operand	Meaning	Explanation
NOP	**None**	**No operation**	**No operation is performed, i.e., the instruction is fetched and decoded.**
HLT	**None**	**Halt and enter wait state**	**The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.**

DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.
RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.

LOOPING COUNTING INDEXING

- The programming technique used to instruct the microprocessor to repeat tasks is called looping.
- This task is accomplished by using jump instructions.

CLASSIFICATION OF LOOPS:

1.continuous loop
2.Unconditional loop

CONTINUOUS LOOP:

- repeats a task continuously.
- A continuous loop is set up by using the unconditional jump instruction
- A program with a continuous loop does not stop repeating the tasks until the system is reset.

CONDITIONAL LOOP:

- A conditional loop is set up by a conditional jump instructions.
- These instructions check flags(Z,CY,P,S) and repeat the tasks if the conditions are satisfied.
- These loops include counting and indexing.

CONDITIONAL LOOP AND COUNTER:

- A counter is a typical application of the conditional loop.
- A microprocessor needs a counter,flag to accomplish the looping task.
- Counter is set up by loading an appropriate count in a register.
- Counting is performed by either increment or decrement the counter.
- Loop is set up by a conditional jump instruction

- End of counting is indicated by a flag.

CONDITIONAL LOOP,COUNTER AND INDEXING:

- Another type of loop which includes counter and indexing .

INDEXING:

- pointing of referencing objects with sequential numbers.
- Data bytes are stored in memory locations and those data bytes are referred to by their memory locations.

Example:

- Steps to add ten bytes of data stored in memory locations starting ata given location and display the sum.
- The microprocessor needs

1. a counter to count 10 data bytes.
2. an index or a memory pointer to locate where data bytes are stored.
3. to transfer data from a memory location to the microprocessor(ALU)
4. to perform addition
5. registers for temporary storage of partial answers
6. a flag to indicate the completion of the stack
7. to store or output the result.