## Aim: Study of

**Theory:**

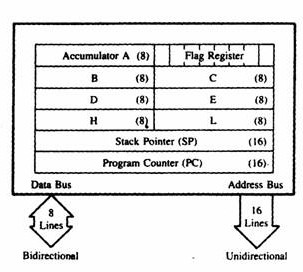
# Practical 2

## Programming model of 8085 microprocessor

1. **Addressing modes.**

## Instruction set classification.

1. **Instruction format**.
   1. **PROGRAMMING MODEL OF 8085 MICROPROCESSOR:**



* **REGISTERS:**
* The 8085 has six general purpose registers to store 8 bit data; these are identifies as B, C, D, E, H, L.
* They can be combined as register pairs - BC, DE and HL to perform some 16-bit operations.
* The programmer can use these registers to store or copy data into the registers by using data copy instructions.

## ACCUMULATOR:

* The 8085 has six general purpose registers to store 8 bit data; these are identifies as B, C, D, E, H, L.
* They can be combined as register pairs - BC, DE and HL to perform some 16-bit operations.
* The programmer can use these registers to store or copy data into the registers by using data copy instructions

## FLAGS:

* 8085 has five flag registers:-
* Sign Flag (S):
  + Sets or Resets based on the result stored in the accumulator.
  + If the result stored is positive, the flag resets else if the result stored is negative the flag is set.
* Zero Flag (Z):
  + Sets or Resets based on the result stored in the accumulator.
  + If the result stored is zero the flag is set else it is reset.
* Auxiliary Carry Flag (AC):
  + This flag is set if there is a carry from low nibble (lowest 4 bits) to high nibble(upper 4 bits) or a borrow from high nibble to low nibble, in the low order 8-bit portion of an addition or subtraction operation.
* Parity Flag (P):
  + This flag is set if there is even parity else it resets.
* Carry Flag (CY):
  + This flag is set if there is a carry bit else it resets.

## PROGRAM COUNTER:

* This 16-bit register deals with sequencing the execution of instructions this register is a memory pointer.
* Memory locations have 16 bit addresses and that is why this is a 16 bit register.
* The function of the PC is to point to the memory address from which the next byte is to be fetched.
* When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

## STACK POINTER:

* The stack pointer is also a 16-bit register used as a memory pointer.
* It points to a memory location in R/W memory called stack.
* The beginning of the stack is defined by loading 16-bit address in the stack pointer.
  1. **ADDRESSING MODES:**
* The way of specifying data to be operated by an instruction is called addressing mode.
* Types of addressing modes –

In 8085 microprocessor there are 5 types of addressing modes:

## Immediate Addressing Mode –

* + - * In immediate addressing mode the source operand is always data. If the data is 8bit, 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)

## 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)

## 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 35)

## Register Indirect Addressing Mode –

* + - * In register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a register pair.
      * Examples:

 MOV A, M (move the contents of the memory location pointed by the HL pair to the accumulator)

 LDAX B (move contents of B-C register to the accumulator)

 LXIH 9570 (load immediate the H-L pair with the address of the location 9570)

## 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)

* 1. **INSTRUCTION SET CLASSIFICATION:**
* An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
* The entire group of instructions that a microprocessor supports is called Instruction Set.
* 8085 has 246 instructions.
* Each instruction is represented by an 8-bit binary value.
* These 8-bits of binary value is called Op-Code or Instruction Byte.
* Classification of Instruction Set:

Data Transfer Instruction • Arithmetic Instructions • Logical Instructions • Branching Instructions • Control Instructions

## DATA TRANSFER INSTRUCTIONS:

* These instructions move data between registers, or between memory and registers.
* These instructions copy data from source to destination.
* While copying, the contents of source are not modified.

|  |  |  |  |
| --- | --- | --- | --- |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |

|  |  |  |  |
| --- | --- | --- | --- |
| MOV | Rd, Rs Rd, M M, Rs | This instruction copies the contents of the source register into the destination register. The contents of the source register are not altered.  If one of the operands is a memory  location, its location is  specified by the contents of the HL registers. | MOV B, C MOV B, M MOV M, C |
| MVI | Rd, Data M, Data | The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the H-L  registers. | MVI A, 57H MVI M, 57H |
| LXI | Reg. pair,  16-bit data | This instruction loads 16-bit data  in the register pair. | LXI H, 2034H |
| LDA | 16-bit address | The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents  of the source are not altered | LDA 2034H |
| LDAX | B/D Register pair | The contents of the designated register pair point to a memory location.  This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not  altered. | LDAX B |
| LHLD | 16-bit address | This instruction copies the contents of memory location pointed out by 16-bit address into register L.  It copies the contents of next  memory location into register H. | LHLD 2040H |
| STA | 16-bit address | The contents of accumulator are  copied into the memory location specified by the operand. | STA 2500H |

|  |  |  |  |
| --- | --- | --- | --- |
| STAX | Reg. pair | The contents of accumulator are copied into the memory location specified by the contents of the  register pair. | STAX B |
| SHLD | 16-bit address | The contents of register L are stored into memory location specified by the 16-bit address. The contents of register H are stored into the next memory location. | SHLD 2550H |
| XCHG | None | The contents of register H are exchanged with the contents of register D.  The contents of register L are exchanged with the contents of  register E. | XCHG |

## ARITHMETIC INSTRUCTIONS:

* These instructions perform the operations like: • Addition • Subtract • Increment
  + Decrement
* Addition: Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator. • The result (sum) is stored in the accumulator. • No two other 8-bit registers can be added directly. • Example: The contents of register B cannot be added directly to the contents of register C.
* Subtraction: Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator. • The result is stored in the accumulator. • Subtraction is performed in 2’s complement form.
  + If the result is negative, it is stored in 2’s complement form. • No two other 8- bit registers can be subtracted directly.
* Increment/Decrement: The 8-bit contents of a register or a memory location can be incremented or decremented by 1. • The 16-bit contents of a register pair can be incremented or decremented by 1. • Increment or decrement can be performed on any register or a memory location.

|  |  |  |  |
| --- | --- | --- | --- |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |
| ADD | R M | The contents of register or memory are added to the contents of accumulator. The result is stored in accumulator. If the operand  is memory location, its | ADD B ADD M |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | address is specified by H-L pair. All flags are modified to reflect the result of the addition |  |
| ADC | R M | The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to reflect the result of the  addition. | ADC B ADC M |
| ADI | 8-bit data | The 8-bit data is added to the contents of accumulator. The result is stored in accumulator. All flags are modified to reflect the result  of the addition. | ADI 45H |
| ACI | 8-bit data | The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator. The result is stored in accumulator. All flags are modified to reflect the result  of the addition. | ACI 45H |

|  |  |  |  |
| --- | --- | --- | --- |
| DAD | Reg. pair | The 16-bit contents of the register pair are added to the contents of H-L pair. The result is stored in H-L pair. If the result is larger than 16 bits, then CY is set. No other flags  are changed. | DAD B |

|  |  |  |  |
| --- | --- | --- | --- |
| SUB | R M | The contents of the register or memory location are subtracted from the contents of the accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to  reflect the result of subtraction. | SUB B SUB M |
| SBB | R M | The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator. The result is stored in accumulator. If the operand is memory location, its address is specified by H-L pair. All flags are modified to  reflect the result of subtraction. | SBB B SBB M |
| SUI | 8-bit data | The 8-bit data is subtracted from the contents of the accumulator. The result is stored in accumulator. All flags are modified to reflect the  result of subtraction. | SUI 45H |
| SBI | 8-bit data | The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator. The result is stored in accumulator. All flags are modified to reflect the  result of subtraction. | SBI 45H |
| INR | R M | The contents of register or memory location are incremented by 1. The result is stored in the same place.  If the operand is a memory location, its address is specified by the contents of H-  L pair. | INR B INR M |
| INX | R | The contents of register pair are incremented by 1. The | INX H |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | result is stored in the same place. |  |
| DCR | R M | The contents of register or memory location are decremented by 1. The result is stored in the same place. If the operand is a memory location, its address is specified by the  contents of H-L pair. | DCR B DCR M |
| DCX | R | The contents of register pair are  decremented by 1. The result is | DCX H |
|  |  | stored in the same place. |  |

## LOGICAL INSTRUCTIONS:

* These instructions perform logical operations on data stored in registers, memory and status flags.
* The logical operations are:
* AND • OR • XOR • Rotate • Compare • Complement
  + AND, OR, XOR: Any 8-bit data, or the contents of register, or memory location can logically have • AND operation • OR operation • XOR operation with the contents of accumulator. • The result is stored in accumulator.
  + ROTATE: Each bit in the accumulator can be shifted either left or right to the next position.
  + COMPARE: Any 8-bit data, or the contents of register, or memory location can be compares for:
* Equality • Greater Than • Less Than with the contents of accumulator. • The result is reflected in status flags.
  + COMPLEMENT: The contents of accumulator can be complemented. • Each 0 is replaced by 1 and each 1 is replaced by 0.

|  |  |  |  |
| --- | --- | --- | --- |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |

|  |  |  |  |
| --- | --- | --- | --- |
| CMP | R M | The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < (reg /mem): carry flag is set if (A) = (reg /mem): zero flag is set if (A) > (reg /mem): carry and  zero flags are reset. | CMP B CMP M |
| CPI | 8-bit data | The 8-bit data is compared with the contents of accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:  if (A) < data: carry flag is set if (A) = data: zero flag is set if (A) > data: carry  and zero flags are reset | CPI 89H |
| ANA | R M | The contents of the accumulator are logically AND ed with the contents  of register or memory. | ANA B ANA M |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | The result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of H-L pair. S, Z, P are modified to reflect the result of the operation. CY is  reset and AC is set. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| ANI | 8-bit data | The contents of the accumulator are logically ANDed with the 8-bit data. The result is placed in the accumulator. S, Z, P are modified to reflect the result.  CY is reset, AC is set. | ANI 86H |
| XRA | R M | The contents of the accumulator are XORed with the contents of the register or memory. The result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of H-L pair. S, Z, P are modified to reflect the result of the operation. CY and AC  are reset. | XRA B XRA M |
| ORA | R M | The contents of the accumulator are logically ORed with the contents of the register or memory. The result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of H-L pair. S, Z, P are modified to reflect the  result. CY and AC are reset. | ORA B ORA M |
| ORI | 8-bit data | The contents of the accumulator are logically ORed with the 8- bit data. The result is placed in the accumulator. S, Z, P are modified to reflect the result.  CY and AC are reset. | ORI 86H |

|  |  |  |  |
| --- | --- | --- | --- |
| XRI | 8-bit data | The contents of the accumulator are XORed with the 8-bit data.  The result is placed in the accumulator. S, Z, P are modified to reflect the result.  CY and AC are reset. | XRI 86H |
| RLC | None | Each binary bit of the accumulator is rotated left by one  Position. Bit D7 is placed in  the | RLC |
|  |  | position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not  affected. |  |
| RRC | None | 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.  S, Z, P, AC are not affected. | RRC |
| RAL | None | 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.  S, Z, P, AC are not affected. | RAL |

|  |  |  |  |
| --- | --- | --- | --- |
| RAR | None | Each binary bit of the accumulator is rotated right by one position through the Carry flag. 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. S, Z, P, AC are not  affected. | RAR |
| CMA | None | The contents of the accumulator are complemented. No flags are  affected. | CMA |
| CMC | None | The Carry flag is complemented. No other  flags are affected. | CMC |
| STC | None | The Carry flag is set to 1. No other flags are affected. | STC |

## BRANCHING INSTRUCTIONS:

* + The branching instruction alter the normal sequential flow.
  + These instructions alter either unconditionally or conditionally.

|  |  |  |  |
| --- | --- | --- | --- |
| **JUMP** | | | |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |
| JMP | 16-bit address | Jump unconditionally  The program sequence is transferred to the memory location specified by the 16-bit address given in the  operand. | JMP 2034H |
| Jx | 16-bit | Jump Conditionally | JZ 2034H |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | address | The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW. | | | |  |
|  | Instruction | Description | Status Flags |
| JC | Jump if Carry | CY=1 |
| JNC | Jump if No Carry | CY=0 |
| JP | Jump if Positive | S=0 |
| JM | Jump if Minus | S=1 |
| JZ | Jump if Zero | Z=1 |
| JNZ | Jump if No Zero | Z=0 |
| JPE | Jump if Parity Even | P=1 |
| JPO | Jump if Parity Odd | P=0 |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **CALL** | | | |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |
| CALL | 16-bit address | Call Unconditionally  The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program  counter) is pushed onto the stack. | CALL 2034H |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cx | 16-bit address | Call Conditionally  The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack. | | | | | CZ 2034H |
|  | Instruction | Description | Status Flags |  |
| CC | Call if Carry | CY=1 |
| CNC | Call if No Carry | CY=0 |
| CP | Call if Positive | S=0 |
|  |  |  | CM | Call if Minus | S=1 |  |  |
| CZ | Call if Zero | Z=1 |
| CNZ | Call if No Zero | Z=0 |
| CPE | Call if Parity Even | P=1 |
| CPO | Call if Parity Odd | P=0 |
|  | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **RETURN** | | | |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |
| RET | None | Return Unconditionally  The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program  execution begins at the new address. | RET |

|  |  |  |  |
| --- | --- | --- | --- |
| Rx | None | Return Conditionally  The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address. | CZ 2034H |

|  |  |  |
| --- | --- | --- |
| Instruction | Description | Status Flags |
| RC | Return if Carry | CY=1 |
| RNC | Return if No Carry | CY=0 |
| RP | Return if Positive | S=0 |
| RM | Return if Minus | S=1 |
| RZ | Return if Zero | Z=1 |
| RNZ | Return if No Zero | Z=0 |
| RPE | Return if Parity Even | P=1 |
| RPO | Return if Parity Odd | P=0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **RESTART** | | | |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |

|  |  |  |  |
| --- | --- | --- | --- |
| RST | 0-7 | Restart (Software Interrupts) The RST instruction jumps the control to one of eight memory locations depending upon the number. These are used as  software instructions in a program to transfer program execution to one of the eight locations. | RST 3 |

## CONTROL INSTRUCTIONS:

|  |  |
| --- | --- |
| Instruction | Restart Address |
| RST 0 | 0000 H |
| RST 1 | 0008 H |
| RST 2 | 0010 H |
| RST 3 | 0018 H |
| RST 4 | 0020 H |
| RST 5 | 0028 H |
| RST 6 | 0030 H |
| RST 7 | 0038 H |

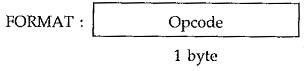
* + The control instructions control the operation of microprocessor.

|  |  |  |  |
| --- | --- | --- | --- |
| **INSTRUCTION** | **OPERAND** | **DESCRIPTION** | **EXAMPLE** |
| NOP | None | No operation is performed. The instruction is fetched and decoded but no operation is  executed. | NOP |
| HLT | None | The CPU finishes executing the current instruction and halts any further execution. An  interrupt or reset is | HLT |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | necessary to exit from the halt state. |  |
| DI | None | The interrupt enable flipflop is reset and all the interrupts except the TRAP are disabled. No  flags are affected. | DI |
| SIM | None | This is a multipurpose instruction and used to implement the 8085 interrupts 7.5,  6.5, 5.5, and | SIM |
|  |  | serial data output. The instruction interprets the accumulator  contents as follows. |  |
| EI | None | The interrupt enable flipflop is set and all interrupts are enabled. No flags are affected. This instruction is necessary to re- enable the interrupts  (except TRAP). | EI |
| RIM | None | This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5,  5.5 and read serial data input bit. The instruction loads eight bits in the  accumulator with the | RIM |

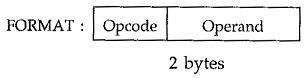
|  |  |  |  |
| --- | --- | --- | --- |
|  |  | following interpretations. |  |

* 1. **INSTRUCTION FORMAT:**
* The Instruction Format of 8085 set consists of one, two and three byte instructions. The first byte is always the op-code; in two-byte instructions the second byte is usually data; in three byte instructions the last two bytes present address or 16-bit data.
  + 1. **One Byte Instruction:**



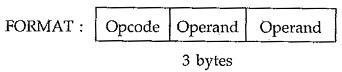
For Example: MOV A, B whose op-code is 78H which is one byte. This Instruction and Data Format of 8085 copies the contents of B register in A register.

* + 1. **Two Byte Instruction:**



For Example: MVI B, 02H. The op-code for this instruction is 06H and is always followed by a byte data (02H in this case). This instruction is a two byte instruction which copies immediate data into B register.

* + 1. **Three Byte Instruction:**



For Example: JMP 6200H. The op-code for this instruction is C3H and is always followed by 16 bit address (6200H in this case). This instruction is a three byte instruction which loads 16 bit address into program counter.