Instruction Format

- Instruction is a command to the microprocessor to perform a given task on a particular data
- Each instruction (instruction format) is composed of two parts-
  - **Opcode:** called operation code which indicates task to be performed,
  - **Operand:** the data to be operated on
    - Can be specified in different ways
    - May include an 8-bit or 16-bit data, an internal Register, a memory location , or 8-bit or 16-bit address
    - In some instructions, the operand is implicit.
- Types of instruction format
  - Fixed length(same numbers of bits for all types of instructions)
  - Variable length(variety of instruction format of different length)

Instruction Format

- 8085 uses 3 variety of variable length instruction format
  - **One Byte Instruction**: includes opcode and operand in same byte

    E.g. MOV A,B; A <- B

    ADD C; A <- A+C
  - **Two Bytes Instruction**: first byte specifies the opcode and second byte specifies the operand

    E.g. MVI B, 20H; B <- 20H
  - **Three Bytes Instruction**: first byte specifies the opcode and second and third byte specifies 2 bytes of data or 2 bytes(16-bit address)

    E.g. LXI H, 2050H; H <- 20H, L <- 50H

Addressing Modes

- To perform any operation, we have to give the corresponding instructions to the microprocessor.
- In each instruction, programmer has to specify 3 things:
  - Operation to be performed i.e. Opcode
  - Address of source of data i.e. Operands
  - Address of destination of result.
- The various formats for specifying the operands in an instruction are called addressing modes.
  - The operands may be source only, destination only or both of them.
  - The source operands can be immediate value, registers, input port, or a memory location.
  - The destination operands can be registers, output port, or a memory location.

Addressing Modes

- Microprocessor uses addressing mode techniques for the purpose of accommodating one or both of the following provisions.
  - To give programming versatility to the user by providing such facilities.
  - To reduce the number of bits in the addressing field of the instruction.
- Following are the five addressing modes supported by 8085 microprocessor
  i. Implied addressing mode
  ii. Immediate addressing mode
  iii. Register direct addressing mode
  iv. Register indirect addressing modes
  v. Direct (absolute) addressing mode

Addressing Modes

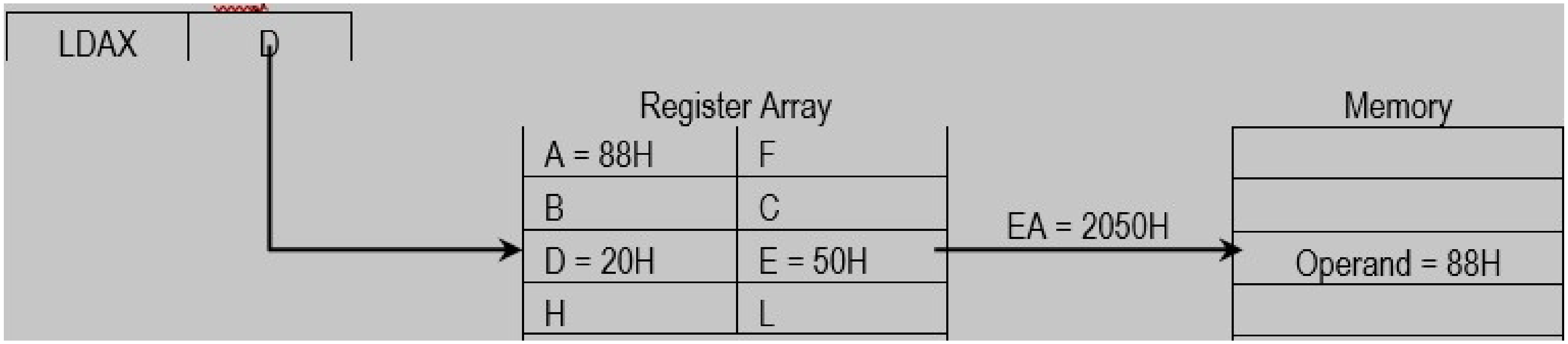
- **Implied Addressing Mode**
  - In this addressing mode, the operands are specified implicitly in the definition of an instruction.
    - E.g. RLC, SIM, STC etc.
- **Immediate Addressing Mode**
  - In this addressing mode, the operand is specified immediately after mnemonic in the instruction itself
  - The immediate mode instructions are mostly useful for initializing the registers to a constant value
    - E.g. MVI A, 29H; A <- 29H
      LXI B, C010H; B <- C0H, C <- 10H

Addressing Modes

- **Register Direct Addressing Mode**
  - In this addressing mode, the operands are the contents of registers itself given in the instruction
    - E.g. MOV H,C; H <- C
      INR A; A <- A+1
- **Register Indirect Addressing Mode**
  - In this addressing mode, the register specified in the register field contains the address of operand rather than the operand itself.
  - That means, the required operand is present in the memory location pointed by a value (address) present in the register field
    - E.g. LDAX D; A <- [DE]
      STAX B; [BC] <- A

Addressing Modes

- **Register Indirect Addressing Mode**

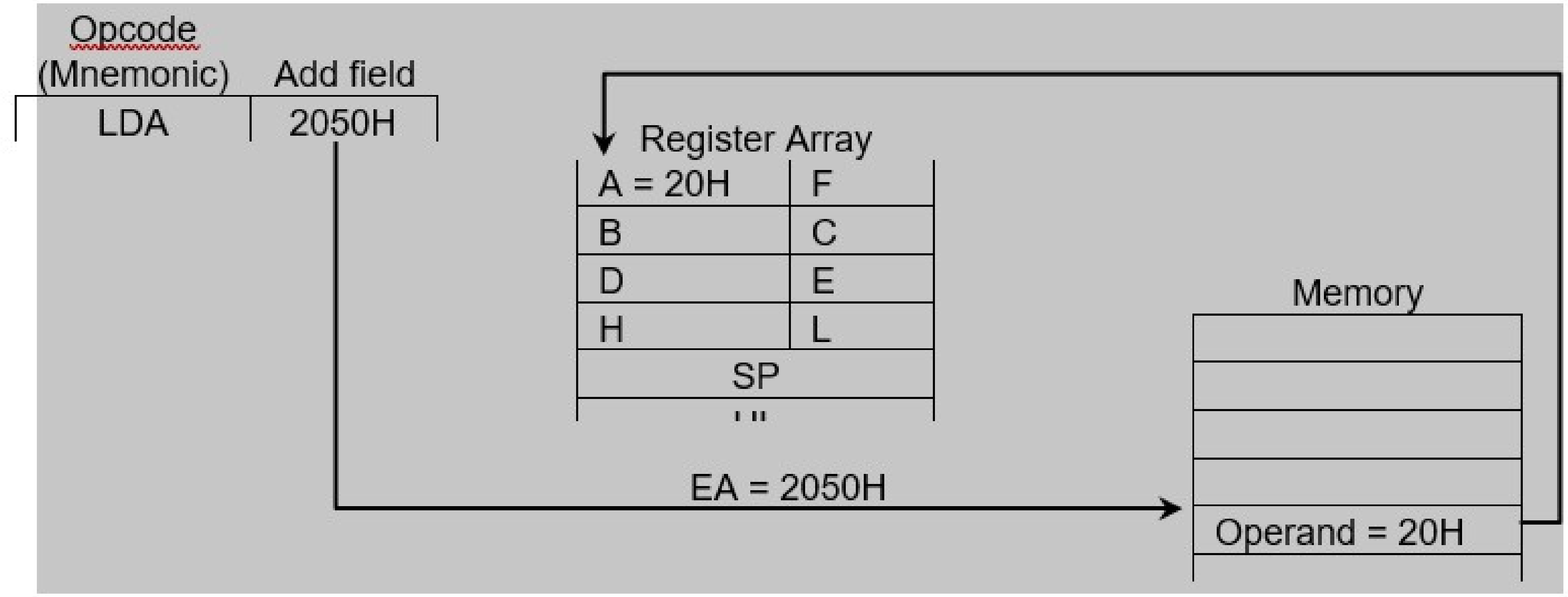


- **Direct (Absolute) Addressing Mode**
  - In this addressing mode, address of operand is specified in instruction itself.
    - E.g. LDA 2050H; A <- [2050]

      STA B040H; [B040] <- A

Addressing Modes

- **Direct (Absolute) Addressing Mode**



Instruction Set

- 8085 supports 246 instructions
- Each instruction is represented by 8-bit(1 byte) hex code called op-code
- Based on the function of instructions, instructions are divided into following groups
  - Data Transfer Instructions
  - Arithmetic Instructions
  - Logical Instructions
  - Branching Instructions
  - Machine Control Instructions
  - Miscellaneous Instructions

Data Transfer Instruction (doesn’t affect flags)

- **Load registers with immediate value**
  - Instruction: **MVI** reg, 8-bit value; reg <- value

    (**2 bytes instruction**)

    e.g. MVI B, 12H;   B <- 12H

    MVI E, 4DH;   E <- 4DH
- **Load register pair with immediate value**
  - Instruction: **LXI** reg_pair, 16-bit value; reg_pair <- value

    (**3 bytes instruction**)

    e.g. LXI B, 1050H; B <- 10H, C <- 50H

    LXI D, 30B1H; D <- 30H, E <- B1H

    LXI H, C100H; H <- C1, L <- 00H

Data Transfer Instruction (doesn’t affect flags)

- **Load memory with immediate value**
  - Instruction: **MVI** M, 8-bit value; [HL] <- value

    (M represents memory whose address is pointed by content of HL)

    (**2 bytes instruction**)

    e.g. LXI H, 8050H; H <- 80H, L <- 50H

    MVI M, 90H; [HL] <- 90H

$\boxed{90}$ HL= 8050H

- **Between register and register**
  - Instruction: **MOV** rd, rs; rd <- rs

    (**1 byte instruction**)

    e.g. MOV B, D; B <- D

    MOV L, A; L <- A

    MOV C, B; C <- B

Data Transfer Instruction (doesn’t affect flags)

- **Question (Classwork)**
  - Write a program in 8085 to load registers A with 33H, B with 3BH, C with 79H, L with 44H. Then copy the value of register C to register D.

```
MVI A, 33H
LXI B, 3B79H
MVI L, 44H
MOV D, C
```

  - Write a program in 8085 to load value DAH into the memory with address 4050H

```
LXI H, 4050H
MVI M, DAH
```

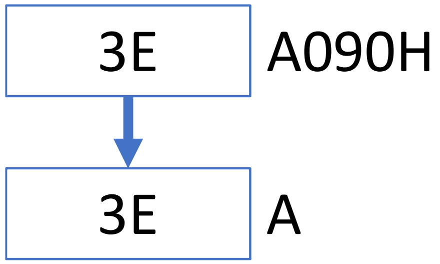
Data Transfer Instruction (doesn’t affect flags)

- **Between Accumulator and memory (direct)**
  - Instruction: **LDA** 16-bit address; A <- [address]

    - Loads the content of memory address given in instruction into accumulator

      (**3 bytes instruction**)

      e.g. LDA A090H; A <- [A090H]



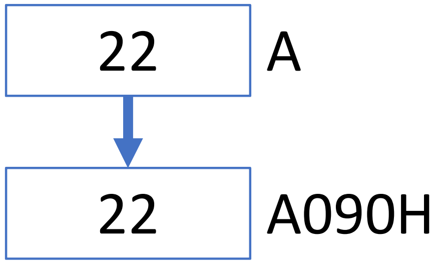
  - Instruction: **STA** 16-bit address; [address] <- A

    - Loads the content of accumulator into memory with address given in instruction

      (**3 bytes instruction**)

      e.g. MVI A, 22H; A <- 22H

      STA A090H; [A090H] <- A



Data Transfer Instruction (doesn't affect flags)

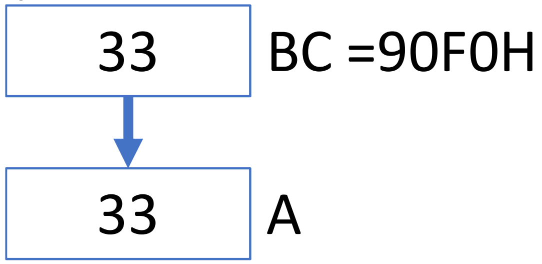
- **Between Accumulator and memory (indirect)**
  - Instruction: **LDAX** reg_pair; A <- [reg_pair]

    - Loads the content of memory address pointed by register pair into accumulator

    (**1 byte instruction**)

    e.g. LXI B, 90F0H; B<- 90H, C <- F0H

    LDAX B; A <- [BC]



  - Instruction: **STAX** reg_pair; [reg_pair] <- A

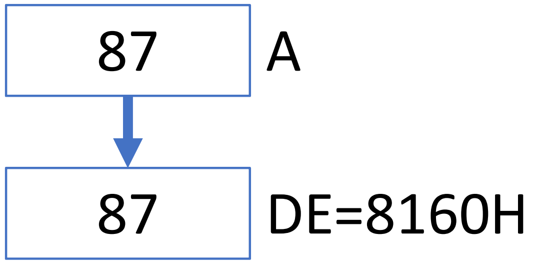
    - Loads the value of accumulator into memory with address pointed by register pair

    (**1 byte instruction**)

    e.g. MVI A, 87H; A <- 87H

    LXI D, 8160H; D <- 81H, E <- 60H

    STAX D; [DE] <- A



(Note: Register pair HL can't be used in LDAX and STAX instruction)

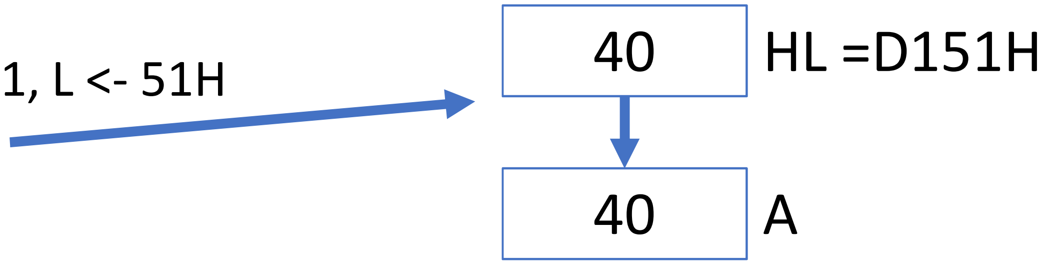
Data Transfer Instruction (doesn’t affect flags)

- **Memory to registers and vice versa**
  - Instruction: **MOV** reg, M; reg <- [HL]

    - Loads the content of memory address pointed by HL pair into register

      (1 byte instruction)

```
e.g.  LXI H, D151H;  H <- D1, L <- 51H
      MOV A, M;  A <- [HL]
      MOV C, M;  C <- [HL]
```

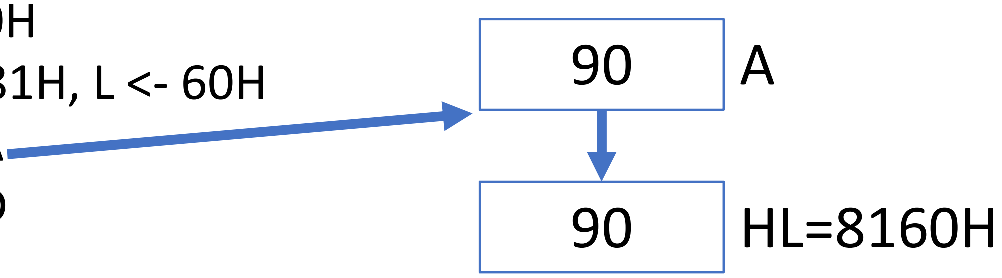


  - Instruction: **MOV** M**,** reg; [HL] <- reg

    - Loads the value of register into memory with address pointed by HL pair

      (1 byte instruction)

```
e.g.  MVI A, 90H;  A <- 90H
      LXI H, 8160H;  H <- 81H, L <- 60H
      MOV M,A;  [HL] <- A
      MOV M, D; [HL] <- D
```


90
A
90
HL=8160H