LAB 2

Aim

To Analyse Data Transfer, Arithmetic and Logical instructions in 8085.

Practice Assignment

1. Analyze the Data transfer instructions by checking the usage on the 8085 simulator.

- **LXI**
- **LHLD**
- **SHLD**
- **LDAX**
- **STAX**
- **XCHG**

Ans:

- **LXI** (Load register pair immediate)
    - **Syntax:** `LXI B OR D OR H, VALUE`
    - **Working:** This instruction loads a 16-bit immediate value into the specified register pair (e.g., HL).
    - **Example:**

```
LXI H, 1234H ; Load 1234H into HL register pair
```

- **LHLD** (Load H and L register direct)
    - **Syntax:** `LHLD Memory`
    - **Working:** This instruction loads the contents of the 16-bit memory location into the HL register pair.
    - **Operation:**
    - The lower byte (byte at the lower address) is loaded into L.
    - The higher byte (byte at the higher address) is loaded into H.
    - **Example:**

```
LHLD 2500H ; Load contents of memory location 2500H into L and 2501H into H
```

- **SHLD** (Store HL Direct)
    - **Syntax:** `SHLD Memory`
    - **Working:** It stores the contents of the HL register pair into two memory locations:
    - The lower byte (stored in register L) is stored at the specified memory address.

- The higher byte (stored in register H) is stored at the next memory address (i.e., the memory address is incremented by 1).
- **Example:**

```
SHLD 2600H ; Store contents of L in 2600H and H in 2601H
```

- **LDAX** (Load accumulator indirect)
  - **Syntax:** `LDAX B or LDAX D`
  - **Working:** This instruction loads the accumulator (A) with the data from the memory location pointed to by the specified register pair (e.g., BC or DE).
  - **Example:**

```
LDAX B ; Load accumulator with the contents of memory location pointed by BC
```

- **STAX** (Store accumulator indirect)
  - **Syntax:** `STAX B or STAX D`
  - **Working:** This instruction stores the contents of the accumulator (A) into the memory location pointed to by the specified register pair (e.g., BC or DE).
  - **Example:**

```
STAX D ; Store contents of accumulator in memory location pointed by DE
```

- **XCHG** (Exchange HL and DE)
  - **Syntax:** `XCHG`
  - **Working:** This instruction exchanges the contents of the HL and DE register pairs.
  - **Example:**

```
XCHG ; Exchange contents of HL and DE register pairs
```

2. Analyze the different Arithmetic and logical instructions by checking the usage on the 8085 simulator.

- **DAD** (Double Addition)
- **ANA, ANI** (Logical AND)
- **ORA, ORI** (Logical OR)
- **XRA, XRI** (Logical XOR)
- **INR, INX** (Increment)
- **DCR, DCX** (Decrement)
- **CMA** (Complement Accumulator)
- **CMP, CPI** (Compare)

- **STC** (Set Carry)

Ans:

- **DAD** (Double Addition)
  - **Syntax:** `DAD B or DAD D or DAD H or DAD SP`
  - **Working:** This instruction adds the contents of the specified register pair to the contents of the HL register pair and stores the result in the HL register pair.
  - **Example:**

```
DAD B ; Add contents of BC to HL
```

- **ANA, ANI** (Logical AND)
  - **Syntax:** `ANA R` or `ANI data`
  - **Working:** This instruction performs a bitwise AND operation between the accumulator and the specified register or immediate data.
  - **Example:**

```
ANA B ; AND accumulator with register B
ANI 0F0H ; AND accumulator with immediate data 0F0H
```

- **ORA, ORI** (Logical OR)
  - **Syntax:** `ORA R` or `ORI data`
  - **Working:** This instruction performs a bitwise OR operation between the accumulator and the specified register or immediate data.
  - **Example:**

```
ORA C ; OR accumulator with register C
ORI 0F0H ; OR accumulator with immediate data 0F0H
```

- **XRA, XRI** (Logical XOR)
  - **Syntax:** `XRA R` or `XRI data`
  - **Working:** This instruction performs a bitwise XOR operation between the accumulator and the specified register or immediate data.
  - **Example:**

```
XRA D ; XOR accumulator with register D
XRI 0F0H ; XOR accumulator with immediate data 0F0H
```

- **INR, INX** (Increment)

- **Syntax:** `INR R` or `INX RP`
- **Working:** `INR` increments the contents of the specified register by 1. `INX` increments the contents of the specified register pair by 1.
- **Example:**

```
INR A ; Increment accumulator by 1
INX H ; Increment HL register pair by 1
```

- **DCR, DCX** (Decrement)
  - **Syntax:** `DCR R` or `DCX RP`
  - **Working:** `DCR` decrements the contents of the specified register by 1. `DCX` decrements the contents of the specified register pair by 1.
  - **Example:**

```
DCR B ; Decrement register B by 1
DCX D ; Decrement DE register pair by 1
```

- **CMA** (Complement Accumulator)
  - **Syntax:** `CMA`
  - **Working:** This instruction complements (inverts) the contents of the accumulator.
  - **Example:**

```
CMA ; Complement the contents of the accumulator
```

- **CMP, CPI** (Compare)
  - **Syntax:** `CMP R` or `CPI data`
  - **Working:** `CMP` compares the contents of the accumulator with the contents of the specified register. `CPI` compares the contents of the accumulator with the immediate data.
  - If (A) < data: carry flag is set
  - If (A) = data: zero flag is set
  - If (A) > data: carry and zero flags are reset
  - **Example:**

```
CMP E ; Compare accumulator with register E
CPI 0F0H ; Compare accumulator with immediate data 0F0H
```

- **STC** (Set Carry)
  - **Syntax:** `STC`
  - **Working:** This instruction sets the carry flag to 1.

- **Example:**

```
STC ; Set the carry flag
```

LAB ASSIGNMENT

1. Write an assembly language program to perform the following operations: AND, OR, Complement, EX-OR. Store the results in the memory location (pointed by HL register-pair) starting from 2210H respectively and by incrementing HL.

```
MVI A, 0F0H ; Load immediate value F0H into accumulator
ANI 0F0H    ; AND immediate value F0H with accumulator
MOV B, A    ; Move the contents of accumulator to register B (AND result)

MVI A, 0F0H ; Load immediate value F0H into accumulator
ORI 0F0H    ; OR immediate value F0H with accumulator
MOV C, A    ; Move the contents of accumulator to register C (OR result)

MVI A, 0F0H ; Load immediate value F0H into accumulator
CMA         ; Complement the accumulator
MOV D, A    ; Move the contents of accumulator to register D (Complement result)

MVI A, 0F0H ; Load immediate value F0H into accumulator
XRI 0F0H    ; XOR immediate value F0H with accumulator
MOV E, A    ; Move the contents of accumulator to register E (XOR result)

LXI H, 2210H ; Load HL pair with the starting memory address 2210H
MOV M, B     ; Store AND result at memory location 2210H
INX H        ; Increment HL to point to next memory location
MOV M, C     ; Store OR result at memory location 2211H
INX H        ; Increment HL to point to next memory location
MOV M, D     ; Store Complement result at memory location 2212H
INX H        ; Increment HL to point to next memory location
MOV M, E     ; Store XOR result at memory location 2213H
```

2. Write an assembly language program to find 1’s complement and 2’s complement of a number.

- **Input:** (2234H) = 23H
- **Output:** (2235H) = DCH, (2236H) = DDH

```
MVI A, 23H   ; Load immediate value 23H into accumulator
STA 2234H    ; Store the value 23H into memory location 2234H

LDA 2234H    ; Load the value from memory location 2234H into accumulator
CMA          ; Complement the accumulator to get 1's complement
STA 2235H    ; Store the 1's complement result in memory location 2235H

LDA 2234H    ; Load the value from memory location 2234H into accumulator again
```

```
CMA          ; Complement the accumulator to get 1's complement
INR A        ; Increment the accumulator to get 2's complement
STA 2236H    ; Store the 2's complement result in memory location 2236H
```

3. Specify the Register contents and the Flag Status as the following instructions are executed having the following initial values A=XX, C=XX, S=0, Z=0, CY=0.

- **Program:**
  - `MVI A, 5EH`
  - `ADI A2H`
  - `MOV C, A`
  - `HLT`

4. Write an Assembly program to perform the following operations and verify the contents by step-by-step execution.

- **a.** Clear the Accumulator
- **b.** Add 47H
- **c.** Subtract 92H
- **d.** Add 64H

```
MVI A, 00H  ; Clear the accumulator
ADD A       ; Add 00H to accumulator (accumulator remains 00H)
ADI 47H     ; Add immediate value 47H to accumulator
SUI 92H     ; Subtract immediate value 92H from accumulator
ADI 64H     ; Add immediate value 64H to accumulator
```

5. In many embedded systems, hardware control registers are used to manage multiple devices or features, such as turning LEDs on/off or controlling motors. If you need to disable a specific feature represented by a particular bit in the control register, how would you achieve the same? Explain your answer. Write an 8085 assembly program that results in the value 0x3C (binary 00111100) in the B register when the Accumulator is initially loaded with the value 7FH.

To disable a specific feature represented by a particular bit in the control register, you can use the AND operation with a mask that has 0s in the positions of the bits you want to disable and 1s in the positions of the bits you want to keep unchanged. This will clear the bits corresponding to the features you want to disable while leaving the other bits unchanged.

```
MVI A, 7FH  ; Load the accumulator with the value 7FH
ANI 3CH     ; AND the accumulator with 3CH to disable specific features
MOV B, A    ; Move the result to register B
```