**Practical – 3**

**AIM: Write an assembly language code in GNUsim8085 to implement data transfer instruction.**

**THEORY:**

An instruction is a binary pattern designed a microprocessor to perform a specific function.

The entire group of instructions, called the instructions set, determines what functions the microprocessor can perform. The 8085 instructions can be classified to 5 functional categories data transfer copy instructions; arithmetic instructions; logical instructions; branching instructions; and machine control instructions;

One of the primary functions of the microprocessor is copying data from a register called the source, to another register, called the destination. In technical literature, the copying function is frequently labeled as the data transfer function, which is misleading in fact the contents of the source are not transferred but are copied in to the called the destination register without modifying the contents of the source

The various type of data transfer copy is listed below together with examples of each type.

|  |  |
| --- | --- |
| **Type** | **Examples** |
| Between register | Copy the content of register B in toregister D |
| Specific data byte to a register or a memory location | Load register B with data byte 32H |
| Between a memory location & a register | From the memory location 200H to register B |
| Between an I/O device & the accumulator | From an i/p keyboard to the accumulator |

**Notations:**

|  |  |
| --- | --- |
| R | 8085 8 bit register (A, B, C, D, E, H, L) |
| M | Memory register (location) |
| Rs | Register source |
| Rd | Register destination (A, B, C, D, E, H, L) |
| Rp | Register pair (BC, DE, HL, SP) |
| ( ) | contents of |

|  |
| --- |
| **MOV: Copy the data from one place to another.** |
| **MOV Rd, Rs**   * Copies the content of Rs to Rd (MOV 1-byte instruction)   **MOV M, Rs**   * **MOV M, r** will copy 8-bit value from the register r to the memory location as pointed by HL register pair. This instruction uses register addressing for specifying the data. * As “r” can have any one of the seven values −   r = A, B, C, D, E, H, or L   * Copies the content of register Rs to memory location pointed by HL Register * 2-byte instruction * Copy the data byte from the register in to memory specified by the address in HL register. * Thus, there are seven opcodes for this type of instruction. It occupies only 1-Byte in memory.  |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | MOV M, A | 1 | | MOV M, B | 1 | | MOV M, C | 1 | | MOV M, D | 1 | | MOV M, E | 1 | | MOV M, H | 1 | | MOV M, L | 1 |   **MOV Rd, M**   * Copies the content of memory location pointed by the HL register to the register Rd. * 2 byte instruction * Copy the data byte in to the register from the memory specified by the address in HL register. * **MOV r, M** is an instruction where the 8-bit data content of the memory location as pointed by HL register pair will be moved to the register r. Thus this is an instruction to load register r with the 8-bit value from a specified memory location whose 16-bit address is in HL register pair. * As r can have any of the seven values, there are seven opcodes for this type of instruction.   r = A, B, C, D, E, H, or L   |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | MOV A, M | 1 | | MOV B, M | 1 | | MOV C, M | 1 | | MOV D, M | 1 | | MOV E, M | 1 | | MOV H, M | 1 | | MOV L, M | 1 |   **Examples:**  MVI B, 10h  MOV A, B  MOV M, B  MOV C, M |
| **MVI: Move immediate data to a register or memory location.** |
| * **MVI** is a mnemonic, which actually means “Move Immediate”. With this instruction,we can load a register with an 8-bitsor 1-Bytevalue. * This instruction supports immediate addressing mode for specifying the data in the instruction. * In the instruction “d8” stands for any 8-bit data, and ‘r’ stands for any one of the registers e.g. A, B, C, D, E, H or L. So this r can replace any one of the seven registers. * As ‘r’ can have any of the seven register names, so there are seven opcodes for this type of instruction. It occupies 2-Bytes in the memory.  |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | MVI A, Data | 2 | | MVI B, Data | 2 | | MVI C, Data | 2 | | MVI D, Data | 2 | | MVI E, Data | 2 | | MVI H, Data | 2 | | MVI L, Data | 2 |   MVI Rd, #30H   * 30h is stored in register Rd   MVI M, #30H   * 30h is stored in memory location pointed by HL Reg * 2 byte instruction   **Examples:**  MVI B, 10H [Loads the 8 bits of the 2nd byte in to the register specified]  MVI M, 30H |
| **LDA: Load Accumulator**  (This instruction copies the data from a given 16 bit address to the Accumulator) |
| * **LDA** is a mnemonic that stands for LoaD Accumulator with the contents from memory. * In this instruction Accumulator will get initialized with 8-bit content from the 16-bit memory address as indicated in the instruction as a16. * This instruction uses absolute addressing for specifying the data. * It occupies 3-Bytes in the memory. * First Byte specifies the opcode, and the successive 2-Bytes provide the 16-bit address, i.e. 1-Byte each for each memory location. * 3 byte instruction   LDA 3000H   * Load the data byte from Memory into Accumulator specified by 16 bit address * Content of memory location 3000h is copied in accumulator   **Example:1**  start: nop  **LDA var1**  hlt  var1: db 04h  **Example:2** (Store 45H data to Specific Memory Location 000BH)  start: nop  lxi h, 000Bh  MVI M, 45H  **LDA 000Bh**  Hlt |
| **LXI: (Load register pair immediate)**  The instruction loads 16-bit data in the register pair designated in the operand. |
| LXI Rp, 16 bit   * 3 byte instruction * Load intermediate immediate 16 bit no. in a register pair * These instructions are used to load the **16-bit** address into the register pair. * We can use this instruction to load data from memory location using the memory address, which is stored in the register pair **rp**. * The rp can be BC, DE, HL or SP. * The LXI instructions and their Hex-codes are as follows.  |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | LXI B | 3 | | LXI D | 3 | | LXI H | 3 | | LXI SP | 3 |   Example:  LXI B,2050H |
| **STA**  **The content of accumulator is copied into the memory location.** |
| STA 16 BIT   * **STA** is a mnemonic that stands for STore Accumulator contents in memory. * In this instruction, Accumulator 8-bit content will be stored to a memory location whose 16-bit address is indicated in the instruction as a16. * This instruction uses absolute addressing for specifying the destination. * This instruction occupies 3-Bytes of memory. * First Byte is required for the opcode, and next successive 2-Bytes provide the 16-bit address divided into 8-bits each consecutively. * 3 byte instruction * Load the data byte from A into the memory specified by 16 bit address   STA 2060H  **Example:**  start: nop  MVI A, 45h  **STA 000Bh**  Hlt |
| **LDAX(Load 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.  The contents of either the register pair or the memory location are not altered. |
| LDAX Rp   * **LDAX** is a mnemonic that stands for LoaD Accumulator from memory pointed by eXtended register pair denoted as “rp” in the instruction. * This instruction uses register indirect addressing for specifying the data. * It occupies only 1-Byte in the memory i.e. 1 byte instruction * Copy the data byte in to A from the memory specified by the address in the register pair * This rp can be either BC register pair represented by B or DE register pair represented by D. * Note that LDAX H is not provided in 8085 instruction set.   + This is because; LDAX H is the same as MOV A, M in its function.  |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | LDAX B | 1 | | LDAX D | 1 |  * BC: 4050H * LDAX B : A <- Content of the memory 4050H   Example:  start: nop  MVI M, 45h  LDAX D  Hlt |
| **INR R** |
| INR is a mnemonic that stands for ‘Increment’ and ‘R’ stands for any of the following registers or memory location M pointed by HL pair.  R = A, B, C, D, E, H, L, or M   * 1 byte instruction * Increase the contents of register R by 1. * This instruction is used to add 1 with the contents of R. So the previous value in R will get increased by amount 1 only. * The result of increment will be stored in R updating its previous content. * All flags, except Cy flag, are affected depending on the result thus produced. * INR R( the content of register R is incremented by 1. * INR M( the content of memory location pointed by HL pair in incremented by 1)   Example:  start: nop  MVI B, 45h  INR B  Hlt |
| **DCR R** |
| DCR is a mnemonic, which stands for ‘Decrement’ and ‘R’ stands for any of the following registers, or memory location M pointed by HL pair.  R = A, B, C, D, E, H, L, or M   * 1 byte instruction * This instruction is used to decrease the content of register R. * Also we can say it will subtract 1 from the register R content. And the decremented value will be stored on to the register R itself. * As it is an arithmetic instruction, so all flags, except Cy flag, are affected depending on the result. * DCR R(content of register R is decremented by 1) * DCR M(the content of memory location pointed by the HL pointer is decremented by 1.) * Example:   start: nop  MVI B, 45h  DCR B  Hlt |
| **INX Rp** |
| **INX** is a mnemonic that stands for “Increment extended register” and **rp** stands for register pair. And it can be any one of the following register pairs.  rp = BC, DE, or HL   * 2 byte instruction * This instruction will be used to add 1 to the present content of the rp. And thus the result of the incremented content will remain stored in rp itself. * Though it is an arithmetic instruction, note that, flag bits are not at all affected by the execution of this instruction. * A register pair is generally used to store 16-bit memory address. * If flag bits got affected during increment of a memory address, then it may cause problems in many cases. * So as per design of 8085, flag bits are not getting affected by the execution of this instruction **INXrp**. * As rp can have any one of the three values, there are three opcodes for this type of instruction. It occupies only 1-Byte in memory.  |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | INX B | 1 | | INX D | 1 | | INX H | 1 |  * Example:   start: nop  LXI B, 4523h  INX B  Hlt |
| **DCX Rp** |
| **DCX** is a mnemonic that stands for “Decrement extended register” and rp stands for register pair. And it can be any one of the following register pairs −  rp = BC, DE, or HL   * 2 byte instruction * Decrease the contents of register pair * This instruction will be used to subtract 1 from the present content of the **rp**. * And thus the result of the decremented content will remain stored in rp itself. * Though, it is an arithmetic instruction, note that flags are not at all affected by the execution of this instruction. * A register pair is generally used to store 16-bit memory address. * If flag bits got affected during decrement of a memory address, then it may cause problems in many cases. * So as per design of 8085, flag bits are not getting affected by the execution of this instruction **DCXrp**. * Example:   start: nop  LXI B, 4523h  DCX B  Hlt |
| **XCHG** |
| * In 8085 Instruction set, there is one mnemonic **XCHG**, which stands for exchange. * This is an instruction to exchange contents of HL register pair with DE register pair. * This instruction uses implied addressing mode. * As it is1-Byte instruction, so It occupies only 1-Byte in the memory. * After execution of this instruction, the content between H and D registers and L and E registers will get swapped respectively.   Example:  **start: nop**  **LXI H, 2050H**  **LXI D, 1030H**  **XCHG**  **hlt** |
| **LHLD**  **Load H and L Register Direct** |
| * This instruction loads the contents of the 16- bit memory location into the HL register pair. * This instruction copies the contents of the memory location pointed by the 16-bit address in register L and copies the contents of the next memory location in register H. * **LHLD** is a mnemonic that stands for Load HL pair using Direct addressing from memory location whose 16-bit address is denoted as a16. So the previous content of HL register pair will get updated with the new 16-bits value. As HL pair has to be updated, so data comes from two consecutive memory locations starting at the address a16 and also from next address location. This instruction uses absolute addressing mode for specifying the data in the instruction. * It occupies 3-Bytes in the memory.   **Example: 1**  start: nop  LXI H, 000Ah  MVI M, 45h  LHLD 000Ah  hlt  **Example: 2**  start: nop  MVI M, 45h  MVI A, 10h  STA 0001h  LHLD 0000h  Hlt |
| **SHLD**  **Store H and L Registers Direct** |
| * The contents of register L are stored into 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. * The contents of registers HL are not altered. * This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. * **SHLD** is a mnemonic, which stands for Store HLpair using Direct addressing in memory location whose 16-bit address is denoted as a16. As HL pair has to be stored, so it has to be stored in two consecutive locations starting at the address a16. We know that H and L are 8-bit registers. So their contents will be stored in two consecutive memory locations as each memory location can hold 8-bits of data. This instruction uses absolute addressing mode for specifying the destination. * It occupies 3-Bytes in the memory.   **Example:**  **start: nop**  **LXI H, 000Ah**  **MVI M, 45h**  **LHLD 000Ah**  **SHLD 000Bh**  **hlt** |
| **STAX**  **Store accumulator indirect** |
| * The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. * **STAX** is a mnemonic that stands for SToreAccumulator contents in memory pointed by eXtended register denoted as “rp”.Hererp stands for register pair. This instruction uses register indirect addressing for specifying the destination. So using this instruction, the current content of Accumulator will be written to the memory location as pointed by 16-bit address as stored in the register pair. It occupies only 1-Byte in memory.   **STAX B**(the content of accumulator is stored into the memory location specified by the BC register pair.)  **Example:**  start: nop  MVI A, 45h  LXI B, 000Ah  STAX B  hlt |
| **SPHL** |
| * This instruction copies H and L register 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 highorder address and the contents of the L register provide the low-order address. * The contents of the H and L registers are not altered. * **SPHL** is an instruction with the help of which Stack Pointer will get initialized with the contents of register pair HL. It is an indirect way of initializing the stack pointer. But it is not a very common and regularly usable instruction as well. It occupies only 1-Byte in memory, compared to the other instruction LXI SP instruction, which is 3-Bytes long used for initializing SP on the other hand. Due to this advantage, SPHL can be useful when SP is required to get initialized to a specific value a number of times in a program.   **Example:**  start: nop  LXI H, 000Ah  SPHL  hlt |

**PROGRAMS:-**

(1) Write an ALP to load 35H in register B & copy it in register C

|  |  |  |
| --- | --- | --- |
| **Address** | **M/C Code** | **Mnemonics** |
| 2000 | 06 35 | MVIB,35H |
| 2002 | 48 | MOV C,B |
| 2003 | 76 | HLT |

(2) Write an ALP to copy the content of memory location 2060H in register C

|  |  |  |
| --- | --- | --- |
| **Address** | **M/C Code** | **Mnemonics** |
| 2000 | 21,60,20 | LXI H,2060H |
| 2003 | 4E | MVI C,M |
| 2005 | 76 | HLT |

(3) Write an ALP to set memory locations 3000H to all its 1’s

|  |  |  |
| --- | --- | --- |
| **Address** | **M/C Code** | **Mnemonics** |
| 2000 | 3E FF | MVI A,FFH |
| 2002 | 32,00,30 | STA 30 00H |
| 2005 | 76 | HLT |

(4) Write an ALP to interchange the content of 2060H to 2080H

|  |  |  |
| --- | --- | --- |
| **Address** | **M/C Code** | **Mnemonics** |
| 2000 | 21,60,20 | LXI H,2060H |
| 2003 | 11 80 20 | LXI D,2080H |
| 2006 | 46 | MOV B,M |
| 2007 | 1A | LDAX D |
| 2008 | 77 | MOV M,A |
| 2009 | 78 | MOV A,B |
| 200A | 12 | STAX D |
| 200B | 76 | HLT |