# Digital Electronics & Microprocessors Laboratory (EC2P006)

## EXPERIMENT-8

*Shorya Sharma*
*19EE01017*

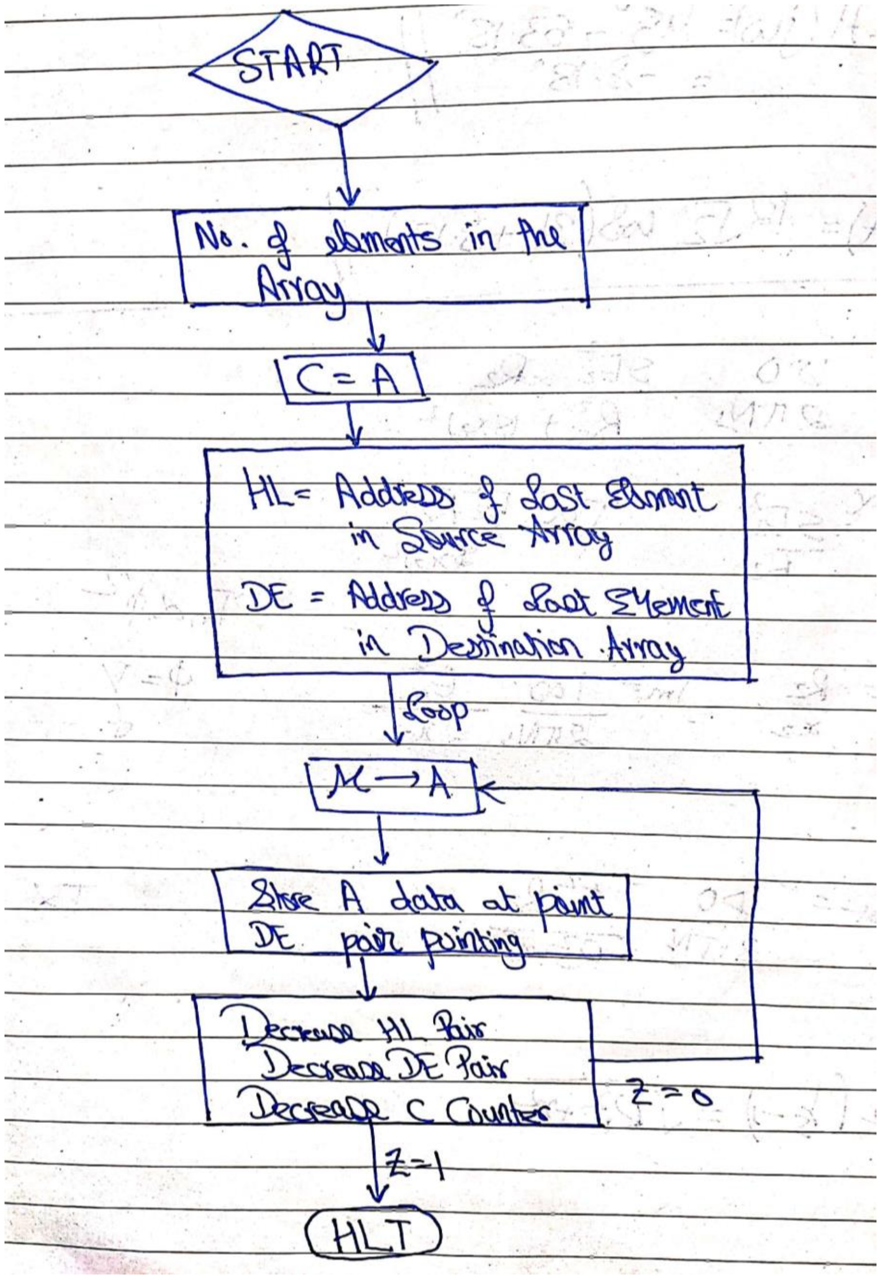
## AIM:

An assembly program which transfers the data from the source to the destination locations, where the source and destination memory locations are happened to overlap.

## ALGORITHM

1. Store the number of elements in the block of data in the accumulator.
2. Now copy the data of the accumulator in the C registor.
3. Now store the address of last element of source array in the HL registor and addressof last element pf destination array in the DE registor pair.
4. Move the value of memory to the accumulator
5. Store the data of the accumulator at the address DE registor pair is pointing.
6. Decrement the address stored in DE and HL registor pair.
7. Also decrease the counter C.
8. Check whether the counter C is zero if it is zero then end the program else continue with steps 4,5,6 and 7.

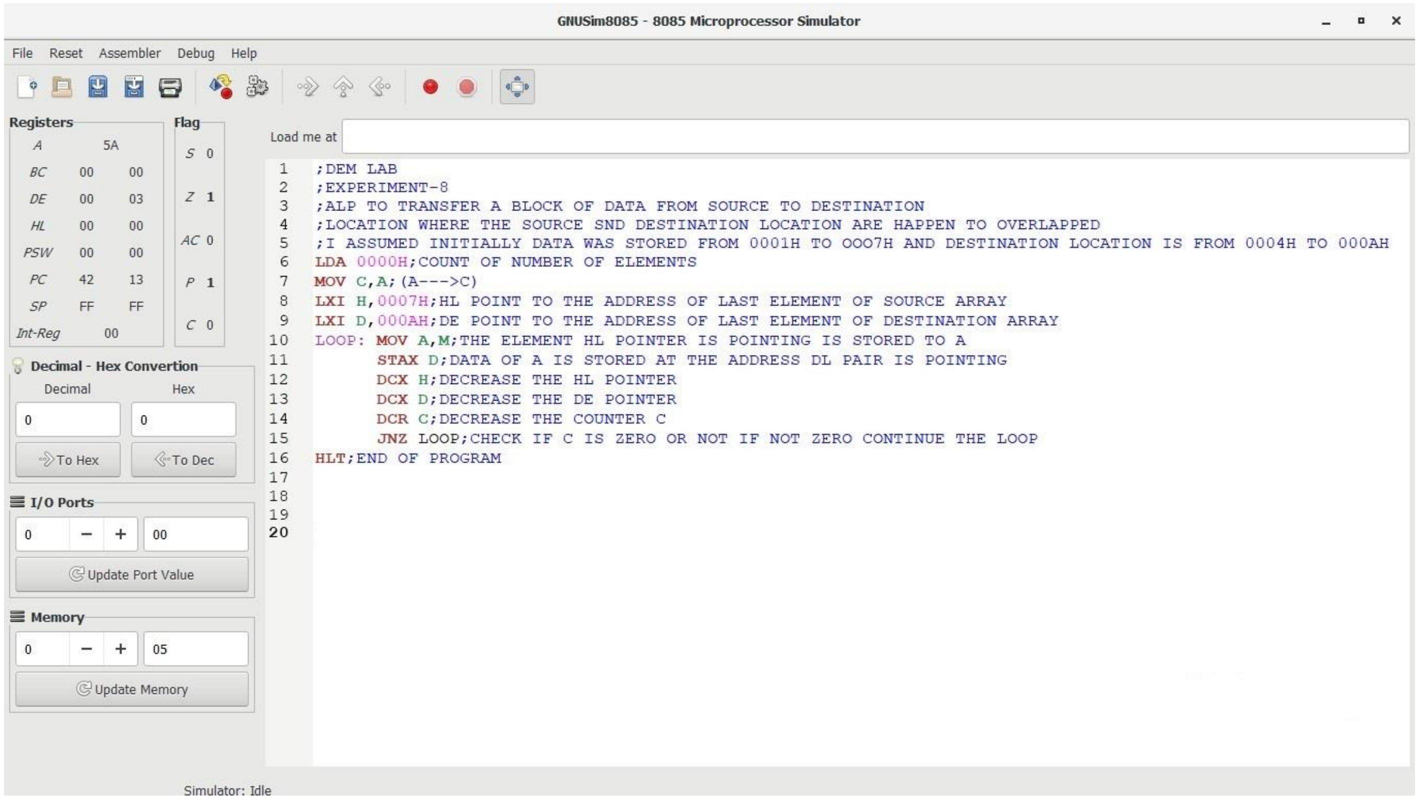
## FLOWCHART


START
No. of elements in the Array
C = A
HL = Address of Last Element in Source Array
DE = Address of Last Element in Destination Array
Loop
M → A
Store A data at point DE pair pointing
Decrease HL Pair
Decrease DE Pair
Decrease C Counter
Z = 0
Z = 1
HLT

## PROGRAM:

| Memory | Opcode | Label | Mnemonics | OPERANDS | Comments |
|---|---|---|---|---|---|
| 1000 | 3A | | LDA | A,05H | [A] = 05H |
| 1003 | 4F | | MOV | C,A | [C]<-[A] |
| 1004 | 21 | | LXI | H,0007H | [HL]<-0007H |
| 1007 | 21 | | LXI | D,000AH | [DE] <- 000AH |
| 1010 | 7E | LOOP | MOV | A,M | [A] <-  [M] |
| 1011 | 12 | | STAX | D | ACCUMULATOR DATA STORED AT ADDRESS POINTED BY DE PAIR |
| 1012 | 2B | | DCX | H | [HL] = [HL]-1 |
| 1013 | 1B | | DCX | D | [DE]=[DE]-1 |
| 1014 | 0D | | DCR | C | [C] = [C]-1 |
| 1015 | C2 | | JNZ | LOOP | Jump if Cy=0 |
| 1018 | 76 | | HLT | | TERMINATE |

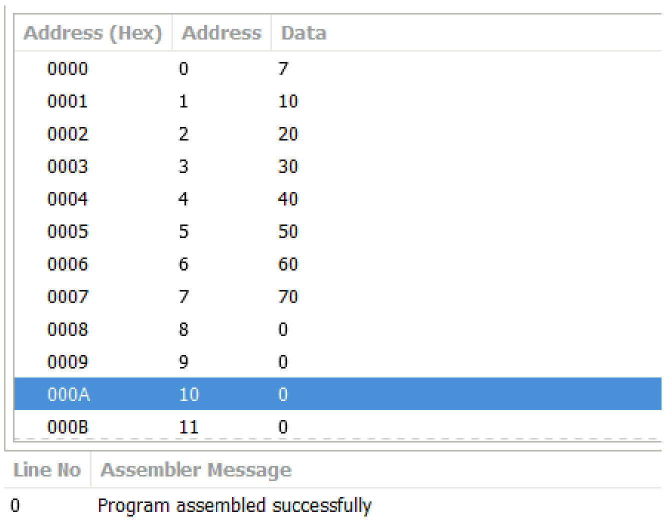
## CODE:



```
;DEM LAB
;EXPERIMENT-8
;ALP TO TRANSFER A BLOCK OF DATA FROM SOURCE TO DESTINATION
;LOCATION WHERE THE SOURCE SND DESTINATION LOCATION ARE HAPPEN TO OVERLAPPED
;I ASSUMED INITIALLY DATA WAS STORED FROM 0001H TO OOO7H AND DESTINATION LOCATION IS FROM 0004H TO 000AH
LDA 0000H;COUNT OF NUMBER OF ELEMENTS
MOV C,A;(A--->C)
LXI H,0007H;HL POINT TO THE ADDRESS OF LAST ELEMENT OF SOURCE ARRAY
LXI D,000AH;DE POINT TO THE ADDRESS OF LAST ELEMENT OF DESTINATION ARRAY
LOOP: MOV A,M;THE ELEMENT HL POINTER IS POINTING IS STORED TO A
      STAX D;DATA OF A IS STORED AT THE ADDRESS DL PAIR IS POINTING
      DCX H;DECREASE THE HL POINTER
      DCX D;DECREASE THE DE POINTER
      DCR C;DECREASE THE COUNTER C
      JNZ LOOP;CHECK IF C IS ZERO OR NOT IF NOT ZERO CONTINUE THE LOOP
HLT;END OF PROGRAM
```

## OBSERVATION:

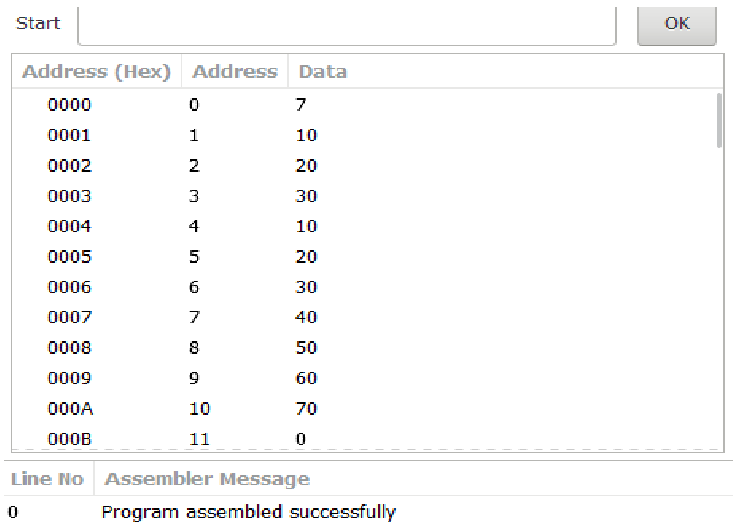
Before the data is transferred from source ie 0001H to 0007H



| Address (Hex) | Address | Data |
|---|---|---|
| 0000 | 0 | 7 |
| 0001 | 1 | 10 |
| 0002 | 2 | 20 |
| 0003 | 3 | 30 |
| 0004 | 4 | 40 |
| 0005 | 5 | 50 |
| 0006 | 6 | 60 |
| 0007 | 7 | 70 |
| 0008 | 8 | 0 |
| 0009 | 9 | 0 |
| 000A | 10 | 0 |
| 000B | 11 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

After the data is transferred to the destination i.e., from 0004H to 000AH



| Address (Hex) | Address | Data |
|---|---|---|
| 0000 | 0 | 7 |
| 0001 | 1 | 10 |
| 0002 | 2 | 20 |
| 0003 | 3 | 30 |
| 0004 | 4 | 10 |
| 0005 | 5 | 20 |
| 0006 | 6 | 30 |
| 0007 | 7 | 40 |
| 0008 | 8 | 50 |
| 0009 | 9 | 60 |
| 000A | 10 | 70 |
| 000B | 11 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

So the code successfully transferred the source data to the destination memory block.

## DISCUSSION

We are using memory location 8008H as initial location and then moving backward as the locations are overlapped. The Loop repeats itself until the counter is zero transferring all the data to other assigned memory. If the locations were overlapping in the opposite fashion, then, we would have run the loop forward to accomplish the transfer.

## CONCLUSION

We learnt how to use 16-bit registers to transfer the data quickly and dealing with overlapping addresses. Iterations using labels is not the only way, one could be using Stack and use stack pointer to send the data to required destination.

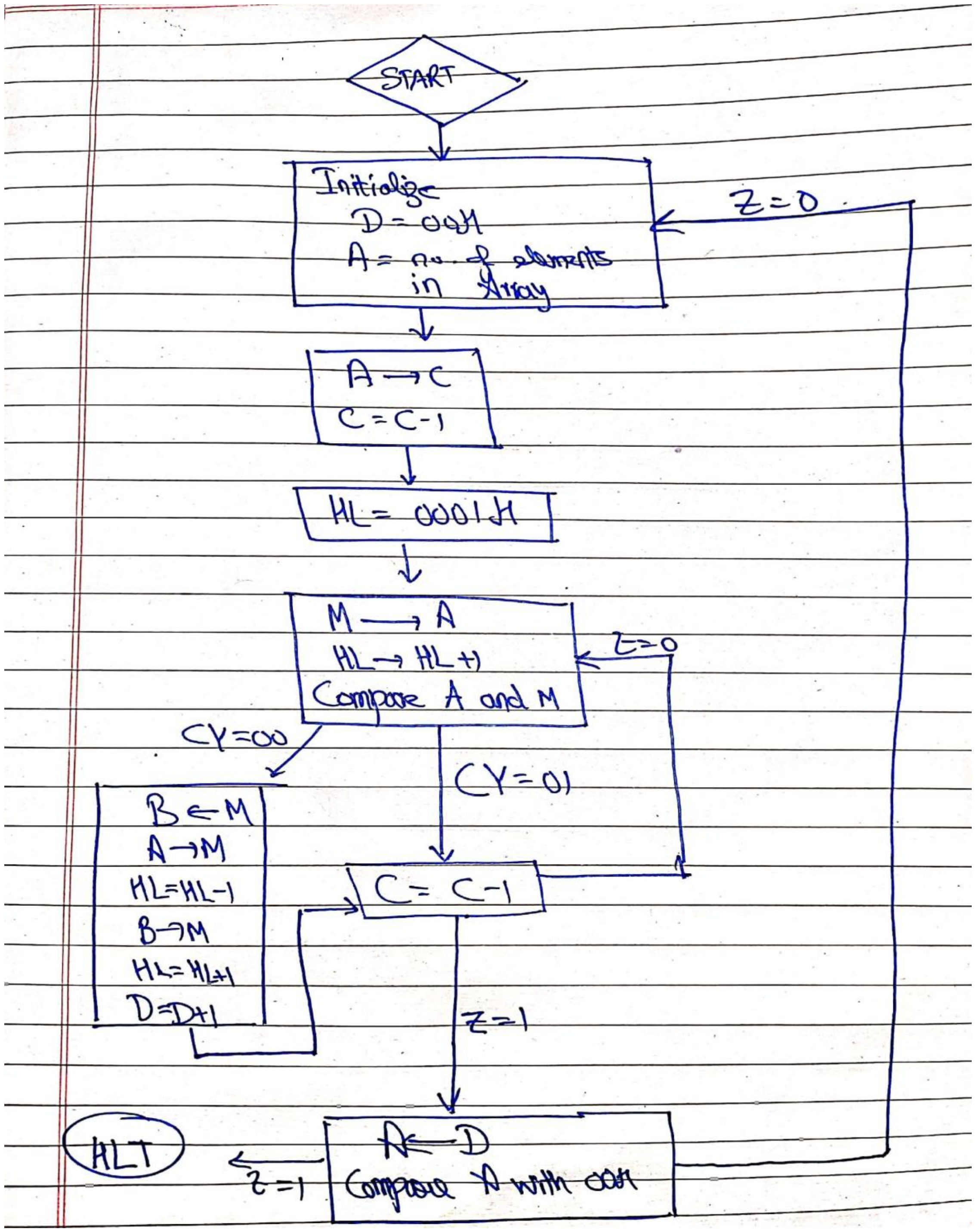
## AIM:

Write an assembly language program that arrange a series of numbersin Ascending order.

## ALGORITHM

1. Initialize the D registor to 00H store the number of the element in thearray in accumulator.
2. Move the data store in accumulator to the C registor.
3. As there will be C-1 comparison in each cycle so decrement C by one.
4. Store the address of starting element in the HL registor pair HL□0001H
5. Store the data of memory in the accumulator and then increment HLpair so that it points to the next element in the array.
6. Then compare the Accumulator A and the Memory and increase thenumber of the swaps counter i.e., D by 1
7. Then decrease the counter C.
8. If the C is not zero then repeat steps 5,6 and 7. Else if C is zero then moveD data into the Accumulator and then compare it with 00H
9. if zero flag is 0 then start the program from beginning else end theprogram.

## FLOWCHART


START
Initialize
D = 00H
A = no. of elements in Array
Z = 0
A → C
C = C-1
HL = 0001H
M ⟶ A
HL → HL+1
Compare A and M
Z=0
CY=00
CY=01
B ← M
A → M
HL=HL-1
B → M
HL=HL+1
D=D+1
C = C-1
Z=1
A ← D
Compare A with 00H
Z=1
HLT

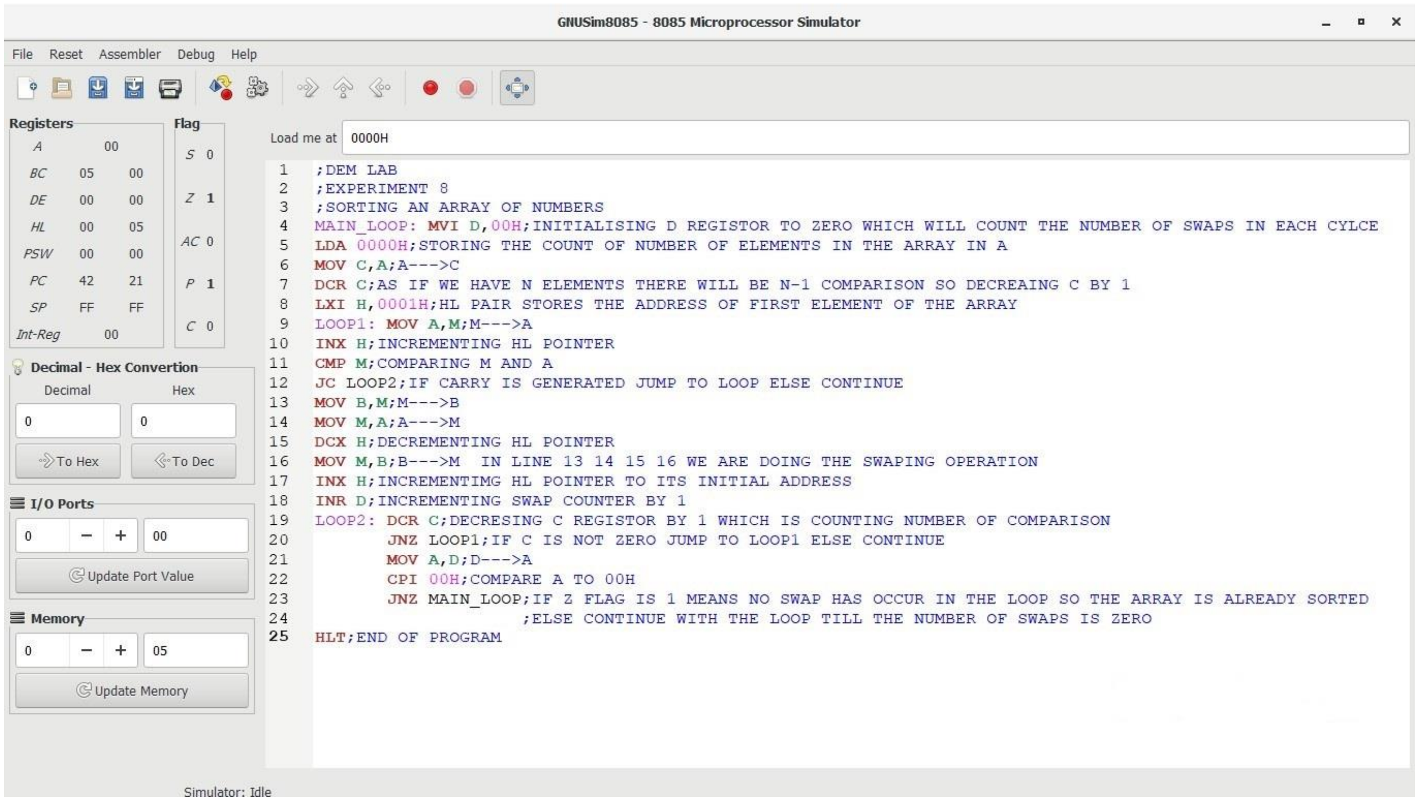
## PROGRAM

PROGRAM START AT MEMORY ADDRESS 07D0

| Memory | Opcode | Label | Mnemonics | OPERANDS | Comments |
|---|---|---|---|---|---|
| 1000 | 16 | MAIN_LOOP | MVI | D,00H | [D]<-00H |
| 1002 | 3A | | LDA | 0000H | LOAD A WITH DATAAT ADDRESS 0000H |
| 1005 | 4F | | MOV | C,A | [C]<-[A] |
| 1006 | 0D | | DCR | C | [C]= [C]-1 |
| 1007 | 7E | LOOP1 | MOV | A,M | [A] <- [M] |
| 1008 | 23 | | INX | H | INCREMENT THE HL REGISTOR PAIR BY ONE |
| 1009 | BD | | CMP | M | [A] AND [M] ARE COMPARED |
| 1010 | DA | | JC | LOOP2 | JUMP IF CARRY |
| 1013 | 46 | | MOV | B,M | [B] <- [M] |
| 1014 | 77 | | MOV | M,A | [M]<-[A] |
| 1015 | 2B | | DCX | H | [HL] <- [HL]-1 |
| 1016 | 70 | | MOV | M,B | [M] <- [B] |
| 1017 | 23 | | INX | H | [H-L] <- [H-L]+1 |
| 1018 | 15 | | DCR | D | [D]= [D]-1 |
| 1019 | 0D | LOOP2 | DCR | C | [C]=[C]-1 |
| 1020 | C2 | | JNZ | LOOP1 | JUMP IF C IS 0 |
| 1023 | 78 | | MOV | A,B | [A]<-[B] |
| 1024 | FE | | CPI | 00H | COMPARE A WITH 00H |

| | | | | | |
|---|---|---|---|---|---|
| 1026 | C2 | | JNZ | MAIN_LOOP | JUMP IF A IS NOT EQUAL TO 00H |
| 1029 | 76 | | HLT | | TERMINATE |

## CODE



Registers

| | | |
|---|---|---|
| A | 00 | |
| BC | 05 | 00 |
| DE | 00 | 00 |
| HL | 00 | 05 |
| PSW | 00 | 00 |
| PC | 42 | 21 |
| SP | FF | FF |
| Int-Reg | 00 | |

Flag

| | |
|---|---|
| S | 0 |
| Z | 1 |
| AC | 0 |
| P | 1 |
| C | 0 |

Decimal: 0 Hex: 0

I/O Ports: 0 00

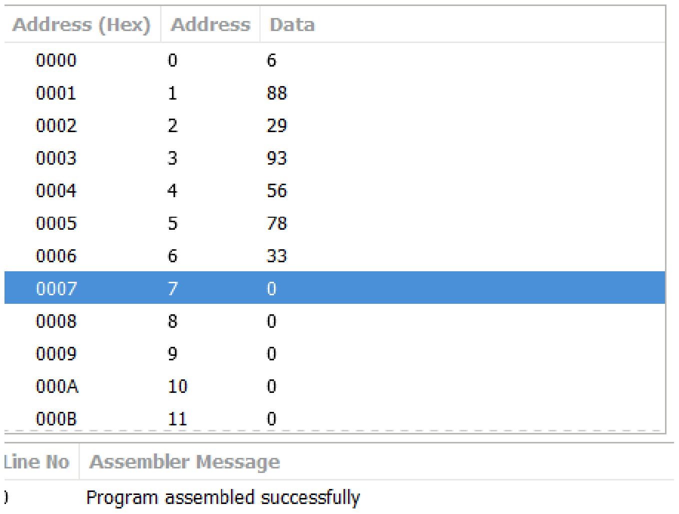
Memory: 0 05

Load me at 0000H

```
1   ;DEM LAB
2   ;EXPERIMENT 8
3   ;SORTING AN ARRAY OF NUMBERS
4   MAIN_LOOP: MVI D,00H;INITIALISING D REGISTOR TO ZERO WHICH WILL COUNT THE NUMBER OF SWAPS IN EACH CYLCE
5   LDA 0000H;STORING THE COUNT OF NUMBER OF ELEMENTS IN THE ARRAY IN A
6   MOV C,A;A--->C
7   DCR C;AS IF WE HAVE N ELEMENTS THERE WILL BE N-1 COMPARISON SO DECREAING C BY 1
8   LXI H,0001H;HL PAIR STORES THE ADDRESS OF FIRST ELEMENT OF THE ARRAY
9   LOOP1: MOV A,M;M--->A
10  INX H;INCREMENTING HL POINTER
11  CMP M;COMPARING M AND A
12  JC LOOP2;IF CARRY IS GENERATED JUMP TO LOOP ELSE CONTINUE
13  MOV B,M;M--->B
14  MOV M,A;A--->M
15  DCX H;DECREMENTING HL POINTER
16  MOV M,B;B--->M  IN LINE 13 14 15 16 WE ARE DOING THE SWAPING OPERATION
17  INX H;INCREMENTIMG HL POINTER TO ITS INITIAL ADDRESS
18  INR D;INCREMENTING SWAP COUNTER BY 1
19  LOOP2: DCR C;DECRESING C REGISTOR BY 1 WHICH IS COUNTING NUMBER OF COMPARISON
20         JNZ LOOP1;IF C IS NOT ZERO JUMP TO LOOP1 ELSE CONTINUE
21         MOV A,D;D--->A
22         CPI 00H;COMPARE A TO 00H
23         JNZ MAIN_LOOP;IF Z FLAG IS 1 MEANS NO SWAP HAS OCCUR IN THE LOOP SO THE ARRAY IS ALREADY SORTED
24                      ;ELSE CONTINUE WITH THE LOOP TILL THE NUMBER OF SWAPS IS ZERO
25  HLT;END OF PROGRAM
```

## OBSERVATION

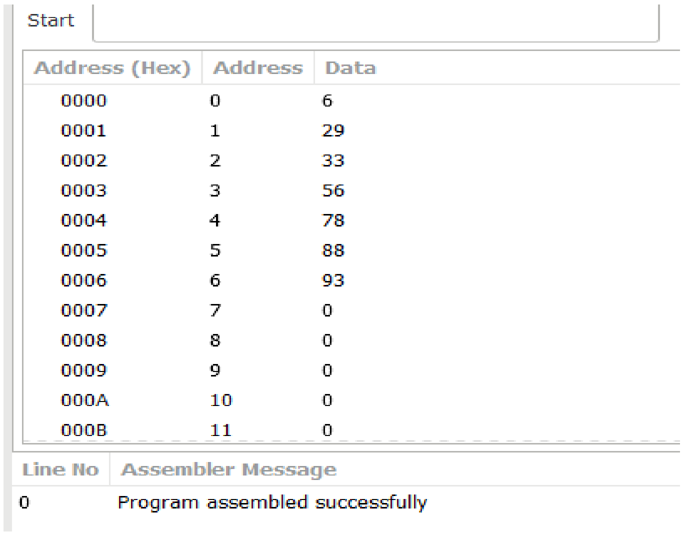
Before sorting in Ascending order



| Address (Hex) | Address | Data |
|---|---|---|
| 0000 | 0 | 6 |
| 0001 | 1 | 88 |
| 0002 | 2 | 29 |
| 0003 | 3 | 93 |
| 0004 | 4 | 56 |
| 0005 | 5 | 78 |
| 0006 | 6 | 33 |
| 0007 | 7 | 0 |
| 0008 | 8 | 0 |
| 0009 | 9 | 0 |
| 000A | 10 | 0 |
| 000B | 11 | 0 |

| Line No | Assembler Message |
|---|---|
| ) | Program assembled successfully |

After sorting



| Address (Hex) | Address | Data |
|---|---|---|
| 0000 | 0 | 6 |
| 0001 | 1 | 29 |
| 0002 | 2 | 33 |
| 0003 | 3 | 56 |
| 0004 | 4 | 78 |
| 0005 | 5 | 88 |
| 0006 | 6 | 93 |
| 0007 | 7 | 0 |
| 0008 | 8 | 0 |
| 0009 | 9 | 0 |
| 000A | 10 | 0 |
| 000B | 11 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

## DISCUSSION

We used Bubble Sorting as shown in Flow chart. It was all about Loading the data and running the loop N times denoted by B and iterations be N-1 times denoted by C. Shuffling was made between accumulator, memory and B register and at the end we increased address of H for getting ready for next shuffle.

At last, we checked if the carry is zero then terminated the loop if flag was down and repeated the loop otherwise.

## CONCLUSION

Learnt how to sort the data in 8085 up. We used three labels in single code for the first time and it was very difficult to code unless the flow chart was drawn. The clarity of loops has been increased through this experiment.