

EXP NO: 11

AIM: To find the smallest number from an array using 8085 processor.

ALGORITHM:

- 1) Load the address of the first element of the array in HL pair.
- 2) Move the count to B register.
- 3) Increment the pointer.
- 4) Get the first data in A register.
- 5) Decrement the count.
- 6) Increment the pointer.
- 7) Compare the content of memory addressed by HL pair with that of A register.
- 8) If carry=1, go to step 10 or if carry=0 go to step 9
- 9) Move the content of memory addressed by HL to A register.
- 10) Decrement the count.

PROGRAM:

```
LXI H,2050
MOV C,M
DCR C
INX H
MOV A,M
LOOP1: INX H
CMP M
JC LOOP
MOV A,M
LOOP: DCR C
JNZ LOOP1
STA 2058
HLT
```

INPUT :

Load me at:

```
1 LXI H,2050
2 MOV C,M
3 DCR C
4 INX H
5 MOV A,M
6 LOOP1: INX H
7 CMP M
8 JC LOOP
9 MOV A,M
10 LOOP: DCR C
11 JNZ LOOP1
12 STA 2058
13 HLT
14
```

Start: OK

Address (Hex)	Address	Data
0802	2050	12
0803	2051	34
0804	2052	56
0805	2053	78
0806	2054	1
0807	2055	0
0808	2056	0
0809	2057	0
080A	2058	0
080B	2059	0
080C	2060	0
080D	2061	0

Line No Assembler Message

OUTPUT:

pad me at

```
1  LOOP: LXI H, 3200
2  MVI D, 00
3  MVI C, 05
4  LOOP1: MOV A, M
5  INX H
6  CMP M
7  JC LOOP2
8  MOV B, M
9  MOV M, A
10 DCX H
11 MOV M, B
12 INX H
13 MVI D, 01
14 LOOP2: DCR C
15 JNZ LOOP1
16 MOV A, D
17 RRC
18 JC LOOP
19 HLT
20
```

Start 3200 OK

Address (Hex)	Address	Data
0C80	3200	0
0C81	3201	1
0C82	3202	2
0C83	3203	34
0C84	3204	56
0C85	3205	78
0C86	3206	0
0C87	3207	0
0C88	3208	0
0C89	3209	0
0C8A	3210	0
0C8B	3211	0

Line No Assembler Message

0 Program assembled successfully

RESULT: Thus the program was executed successfully using 8085 processor simulator.

EXP NO: 12

AIM: To compute ascending order of an array using 8085 processor.

ALGORITHM:

- 1) Initialize HL pair as memory pointer.
- 2) Get the count at memory and load it into C register
- 3) Copy it in D register (for bubble sort (N-1)) times required.
- 4) Get the first value in A register.
- 5) Compare it with the value at next location.
- 6) If they are out of order, exchange the contents of A register and memory.
- 7) Decrement D register content by 1
- 8) Repeat step 5 and 7 till the value in D register become zero.
- 9) Decrement the C register content by 1.
- 10) Repeat steps 3 to 9 till the value in C register becomes zero.

PROGRAM:

```
LOOP: LXI H, 3500
MVI D, 00
MVI C, 05
LOOP1: MOV A, M
INX H
```

```

CMP M
JC LOOP2
MOV B,M
MOV M,A
DCX H
MOV M,B
INX H
MVI D,01
LOOP2: DCR C
JNZ LOOP1
MOV A,D
RRC
JC LOOP
HLT

```

INPUT :

Load me at

```

1  LOOP: LXI H,3200
2  MVI D,00
3  MVI C,05
4  LOOP1: MOV A,M
5  INX H
6  CMP M
7  JC LOOP2
8  MOV B,M
9  MOV M,A
10 DCX H
11 MOV M,B
12 INX H
13 MVI D,01
14 LOOP2: DCR C
15 JNZ LOOP1
16 MOV A,D
17 RRC
18 JC LOOP
19 HLT
20

```

Start 3200 OK

Address (Hex)	Address	Data
0C80	3200	34
0C81	3201	56
0C82	3202	2
0C83	3203	1
0C84	3204	78
0C85	3205	0
0C86	3206	0
0C87	3207	0
0C88	3208	0
0C89	3209	0
0C8A	3210	0
0C8B	3211	0
0C8C	3212	0

Line No Assembler Message

OUTPUT:

Load me at

```

1  LOOP: LXI H,3200
2  MVI D,00
3  MVI C,05
4  LOOP1: MOV A,M
5  INX H
6  CMP M
7  JC LOOP2
8  MOV B,M
9  MOV M,A
10 DCX H
11 MOV M,B
12 INX H
13 MVI D,01
14 LOOP2: DCR C
15 JNZ LOOP1
16 MOV A,D
17 RRC
18 JC LOOP
19 HLT
20

```

Start 3200 OK

Address (Hex)	Address	Data
0C80	3200	0
0C81	3201	1
0C82	3202	2
0C83	3203	34
0C84	3204	56
0C85	3205	78
0C86	3206	0
0C87	3207	0
0C88	3208	0
0C89	3209	0
0C8A	3210	0
0C8B	3211	0

Line No Assembler Message

0 Program assembled successfully

RESULT: Thus the program was executed successfully using 8085 processor simulator.

EXP NO: 13

AIM: To compute descending order of an array using 8085 processor.

ALGORITHM:

- 1) Initialize HL pair as memory pointer.
- 2) Get the count at memory and load it into C register
- 3) Copy it in D register (for bubble sort (N-1)) times required.
- 4) Get the first value in A register.
- 5) Compare it with the value at next location.
- 6) If they are out of order, exchange the contents of A register and memory.
- 7) Decrement D register content by 1.
- 8) Repeat step 5 and 7 till the value in D register become zero.
- 9) Decrement the C register content by 1.
- 10) Repeat steps 3 to 9 till the value in C register becomes zero.

PROGRAM:

```
LOOP: LXI H,3500
MVI D,00
MVI C,05
LOOP1: MOV A,M
INX H
CMP M
JNC LOOP2
MOV B,M
MOV M,A
DCX H
MOV M,B
INX H
MVI D,01
LOOP2: DCR C
JNZ LOOP1
MOV A,D
RRC
JC LOOP
HLT
```

INPUT :

The screenshot shows an 8085 assembler simulator interface. On the left, the assembly code is listed with line numbers 1 through 20. The code is: 1: LOOP: LXI H,3500; 2: MVI D,00; 3: MVI C,05; 4: LOOP1: MOV A,M; 5: INX H; 6: CMP M; 7: JNC LOOP2; 8: MOV B,M; 9: MOV M,A; 10: DCX H; 11: MOV M,B; 12: INX H; 13: MVI D,01; 14: LOOP2: DCR C; 15: JNZ LOOP1; 16: MOV A,D; 17: RRC; 18: JC LOOP; 19: HLT; 20: . The 'Load me at' field is set to 3500. On the right, the 'Memory' window displays a table of memory addresses and data. The table has columns for Address (Hex), Address, and Data. The data shown is: 0DAC 3500 12; 0DAD 3501 15; 0DAE 3502 34; 0DAF 3503 1; 0DB0 3504 0; 0DB1 3505 0; 0DB2 3506 0; 0DB3 3507 0; 0DB4 3508 0; 0DB5 3509 0; 0DB6 3510 0; 0DB7 3511 0. At the bottom, the 'Assembler Message' window shows the message 'Program assembled successfully'.

Address (Hex)	Address	Data
0DAC	3500	12
0DAD	3501	15
0DAE	3502	34
0DAF	3503	1
0DB0	3504	0
0DB1	3505	0
0DB2	3506	0
0DB3	3507	0
0DB4	3508	0
0DB5	3509	0
0DB6	3510	0
0DB7	3511	0

Line No Assembler Message
0 Program assembled successfully

OUTPUT:

The screenshot displays an 8085 processor simulator interface. On the left, the assembly code is listed with line numbers 1 through 20. The code implements a loop that adds the contents of memory to the accumulator (A) until the counter (C) reaches zero. The memory dump on the right shows the state of memory starting at address 3500. The data at addresses 3500 through 3511 is as follows:

Address (Hex)	Address	Data
0DAC	3500	34
0DAD	3501	15
0DAE	3502	12
0DAF	3503	1
0DB0	3504	0
0DB1	3505	0
0DB2	3506	0
0DB3	3507	0
0DB4	3508	0
0DB5	3509	0
0DB6	3510	0
0DB7	3511	0

Below the memory dump, the assembler message indicates that the program was assembled successfully.

RESULT: Thus the program was executed successfully using 8085 processor simulator.

EXP NO: 14

AIM: To compute addition of N numbers using 8085 processor.

ALGORITHM:

- 1) Load the base address of the array in HL register pair.
- 2) Load the memory with data to be added.
- 3) Take it as count.
- 4) Initialize the accumulator with 00.
- 5) Add content of accumulator with content of memory.
- 6) Decrement count.
- 7) Load count value to memory location.
- 8) Repeat step 5.
- 9) Check whether count has become 0.
- 10) Halt.

PROGRAM:

```
LXI H,8000
MOV C,M
XRA A
MOV B,A
LOOP: INX H
ADD M
JNC SKIP
INR B
SKIP: DCR C
JNZ LOOP
INX H
MOV M,A
INX H
MOV M,B
HLT
```

INPUT & OUTPUT:

LXI H,8000
MOV C,M
XRA A
MOV B,A
LOOP: INX H
ADD M
JNC SKIP
INR B
SKIP: DCR C
JNZ LOOP
INX H
MOV M,A
INX H
MOV M,B
HLT

Start8000OK

Address (Hex)	Address	Data
1F40	8000	0
1F41	8001	2
1F42	8002	4
1F43	8003	0
1F44	8004	6
1F45	8005	0
1F46	8006	0
1F47	8007	0
1F48	8008	0
1F49	8009	0
1F4A	8010	0
1F4B	8011	0

RESULT: Thus the program was executed successfully using 8085 processor simulator.

AIM: To compute swapping of numbers using 8085 processor.

ALGORITHM:

- 1) Load a 8-bit number from memory location into accumulator.
- 2) Move value of accumulator into register H.
- 3) Load a 8-bit number from next memory location into accumulator.
- 4) Move value of accumulator into register D.
- 5) Exchange both the registers pairs.
- 6) Halt

PROGRAM:

```
LDA 2001  
MOV B,A  
LDA 2002  
STA 2001  
MOV A,B  
STA 2002  
HLT
```

INPUT :

LDA 2001
MOV B, A
LDA 2002
STA 2001
MOV A, B
STA 2002
HLT

Start2001OK

Address (Hex)	Address	Data
07D1	2001	10
07D2	2002	20
07D3	2003	0
07D4	2004	0
07D5	2005	0
07D6	2006	0
07D7	2007	0
07D8	2008	0
07D9	2009	0
07DA	2010	0
07DB	2011	0
07DC	2012	0

Line No Assembler Message

OUTPUT:

The screenshot displays the 8085 processor simulator interface. On the left, the assembly code is listed with line numbers 1 through 9. The code is as follows:

```
1 LDA 2001
2 MOV B,A
3 LDA 2002
4 STA 2001
5 MOV A,B
6 STA 2002
7 HLT
8
9
```

On the right, the 'Memory' tab is selected, showing a memory dump. The 'Start' address is 2001. The memory dump table is as follows:

Address (Hex)	Address	Data
07D1	2001	20
07D2	2002	10
07D3	2003	0
07D4	2004	0
07D5	2005	0
07D6	2006	0
07D7	2007	0
07D8	2008	0
07D9	2009	0
07DA	2010	0
07DB	2011	0
07DC	2012	0

Below the memory dump, the 'Assembler Message' window shows the following message:

```
Line No Assembler Message
0 Program assembled successfully
```

RESULT: Thus the program was executed successfully using 8085 processor simulator.

EXP NO: 16

AIM: To compute square of number using 8085 processor.

ALGORITHM:

- 1) Load the base address of the array in HL register pair.
- 2) Assign accumulator as 0.
- 3) Load the content of memory location specified into register.
- 4) Add content of memory location with accumulator and decrement register content by 01.
- 5) Check if register holds 00, if so store the value of accumulator in memory location.

PROGRAM:

```
LXI H,8000
XRA A
MOV B,M
LOOP: ADD M
DCR B
JNZ LOOP
STA 8001
HLT
```

INPUT & OUTPUT:

The screenshot displays an 8085 processor simulator interface. On the left, a text area contains assembly code with line numbers 1 through 9. The code is as follows:

```
1 LXI H, 8100
2 XRA A
3 MOV B, M
4 LOOP: ADD M
5 DCR B
6 JNZ LOOP
7 STA 8101
8 HLT
9
```

On the right, there are two panels. The top panel, titled 'Memory', shows a memory dump starting at address 8100. It lists addresses from 1FA4 to 1FAF in hexadecimal, with corresponding data values. The data at 8100 is 5, and at 8101 is 25. All other addresses contain 0.

Address (Hex)	Address	Data
1FA4	8100	5
1FA5	8101	25
1FA6	8102	0
1FA7	8103	0
1FA8	8104	0
1FA9	8105	0
1FAA	8106	0
1FAB	8107	0
1FAC	8108	0
1FAD	8109	0
1FAE	8110	0
1FAF	8111	0

The bottom panel, titled 'Assembler Message', shows a single message: '0 Program assembled successfully'.

RESULT: Thus the program was executed successfully using 8085 processor simulator.

ONEs AND TWOs COMPLEMENT

EXP NO: 17

AIM: To compute one's and two's complement using 8085 processor.

ALGORITHM:

- 1) Load the base address of the array in a register pair.
- 2) Move the data from memory location into accumulator.
- 3) Convert all ones into zeros and zeros into ones.
- 4) Add 01 to the accumulator content.
- 5) Store the results of one's and two's complement.

PROGRAM:

```
LDA 3000
CMA
STA 3001
ADI 01
STA 3002
HLT
```


INPUT & OUTPUT:

The screenshot shows the 8085 processor simulator interface. On the left, the assembly code is displayed with line numbers 1 through 7. The code is as follows:

```
1 LDA 3000
2 CMA
3 STA 3001
4 ADI 01
5 STA 3002
6 HLT
7
```

On the right, the 'Memory' tab is selected, showing a memory dump starting at address 3000. The dump is as follows:

Address (Hex)	Address	Data
0BB8	3000	5
0BB9	3001	250
0BBA	3002	251
0BBB	3003	0
0BBC	3004	0
0BBD	3005	0
0BBE	3006	0
0BBF	3007	0
0BC0	3008	0
0BC1	3009	0
0BC2	3010	0
0BC3	3011	0

Below the memory dump, the 'Assembler Message' section shows the following message:

```
Line No Assembler Message
0 Program assembled successfully
```

RESULT: Thus the program was executed successfully using 8085 processor simulator.

AIM: To compute rotation of given data in left without carry using 8085 processor.

ALGORITHM:

- 1) Load the base address of the array in HL register pair.
- 2) Move the data from memory location into accumulator.
- 3) Shift left the accumulator content for four times.
- 4) Store the result in the specified location.

PROGRAM:

```
MVI A,02
```

```
RLC
```

```
RLC
```

```
RLC
```

```
RLC
```

```
STA 2000
```

```
HLT
```

INPUT & OUTPUT:

The screenshot shows the 8085 processor simulator interface. On the left, the assembly code is displayed with line numbers 1 through 8. The code is as follows:

```
1 MVI A, 02
2 RLC
3 RLC
4 RLC
5 RLC
6 STA 2700
7 HLT
8
```

On the right, the 'Memory' tab is selected, showing a memory dump starting at address 2700. The dump is as follows:

Address (Hex)	Address	Data
0A8C	2700	32
0A8D	2701	0
0A8E	2702	0
0A8F	2703	0
0A90	2704	0
0A91	2705	0
0A92	2706	0
0A93	2707	0
0A94	2708	0
0A95	2709	0
0A96	2710	0
0A97	2711	0

Below the memory dump, the 'Assembler Message' section shows the following message:

```
Line No Assembler Message
0 Program assembled successfully
```

RESULT: Thus the program was executed successfully using 8085 processor simulator.

EXP NO: 19

AIM: To compute rotation of given data in right without carry using 8085 processor.

ALGORITHM:

- 1) Load the base address of the array in HL register pair.
- 2) Move the data from memory location into accumulator.
- 3) Shift right the accumulator content for four times left.
- 4) Store the result in the specified location.

PROGRAM:

MVI A,03

RRC

RRC

RRC

RRC

STA 2000

HLT

INPUT & OUTPUT:

The screenshot displays the 8085 processor simulator interface. On the left, the assembly code is loaded into a text area:

```
Load me at
1 MVI A,03
2 RRC
3 RRC
4 RRC
5 RRC
6 STA 2800
7 HLT
8
```

On the right, the Memory window is open, showing a memory dump starting at address 2800. The data at 2800 is 48, and all other addresses from 2801 to 2811 contain 0.

Address (Hex)	Address	Data
0AF0	2800	48
0AF1	2801	0
0AF2	2802	0
0AF3	2803	0
0AF4	2804	0
0AF5	2805	0
0AF6	2806	0
0AF7	2807	0
0AF8	2808	0
0AF9	2809	0
0AFA	2810	0
0AFB	2811	0

Below the memory dump, the Assembler Message window shows:

Line No	Assembler Message
0	Program assembled successfully

RESULT: Thus the program was executed successfully using 8085 processor simulator.

EXP NO: 20

AIM: To compute various logical operations using 8085 processor.

ALGORITHM:

- 1) Load data to accumulator.
- 2) Load another data in register.
- 3) Perform logical operations like AND, OR and XOR (Use ANA, ORA, XRA) with the accumulator content.
- 4) Store the result in specified memory location.

PROGRAM:

AND OPERATION:

MVI A,06

MVI B,04

ANA B

STA 2500
HLT

OR OPERATION:

MVI A,07
MVI B,06
ORA B
STA 2000
HLT

XOR OPERATION:

MVI A,03
MVI B,08
XRA B
STA 2000
HLT

INPUT & OUTPUT:

AND OPERATION:

The screenshot displays an 8085 assembly simulator interface. On the left, a code editor shows five lines of assembly code: `1 MVI A,06`, `2 MVI B,04`, `3 ANA B`, `4 STA 2900`, and `5 HLT`. The code is color-coded: `MVI` is blue, `ANA` is red, `STA` is green, and `HLT` is red. A 'Load me at' field is empty. On the right, the 'Memory' tab is active, showing a table of memory addresses from 0B54 to 0B5F. Address 2900 contains the value 4, while all other addresses contain 0. Below the memory table, the 'Assembler Message' section shows '0 Program assembled successfully'.

Address (Hex)	Address	Data
0B54	2900	4
0B55	2901	0
0B56	2902	0
0B57	2903	0
0B58	2904	0
0B59	2905	0
0B5A	2906	0
0B5B	2907	0
0B5C	2908	0
0B5D	2909	0
0B5E	2910	0
0B5F	2911	0

Line No	Assembler Message
0	Program assembled successfully

OR OPERATION:

Load me at

0

1

2

3

4

5

6

```

1  MVI A,07
2  MVI B,06
3  ORA B
4  STA 1900
5  HLT

```

Data

Stack

Keypad

Memory

I/O Ports

Start

1900

OK

Address (Hex)	Address	Data
076C	1900	7
076D	1901	0
076E	1902	0
076F	1903	0
0770	1904	0
0771	1905	0
0772	1906	0
0773	1907	0
0774	1908	0
0775	1909	0
0776	1910	0
0777	1911	0

Line No

Assembler Message

0

Program assembled successfully

XOR OPERATION:

Load me at

0

1

2

3

4

5

6

```

1  MVI A,03
2  MVI B,08
3  XRA B
4  STA 1901
5  HLT

```

Data

Stack

Keypad

Memory

I/O Ports

Start

1901

OK

Address (Hex)	Address	Data
076D	1901	11
076E	1902	0
076F	1903	0
0770	1904	0
0771	1905	0
0772	1906	0
0773	1907	0
0774	1908	0
0775	1909	0
0776	1910	0
0777	1911	0
0778	1912	0

Line No

Assembler Message

0

Program assembled successfully

RESULT: Thus the program was executed successfully using 8085 processor simulator.