8-BIT ADDITION

EXP NO: 1

AIM:
To write an assembly language program to implement 8-bit addition using 8085 processor.

ALGORITHM:
1) Start the program by loading the first data into the accumulator.

2) Move the data to a register.

3) Get the second data and load it into the accumulator.

4) Add the two register contents.

5) Check for carry.

6) Store the value of sum and carry in the memory location.

7) Halt.

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050	Load accumulator with first number in the address 8085
MOV B, A	Move the data from accumulator to'B' register
LDA 8051	Load accumulator with second number in the address 8051
ADD B	Add the data of 'B' register with accumulator
STA 8052	Store the data (Output) of the accumulator in address 8052
HLT	Halt

INPUT:

ADDRESS	DATA
8050	1
8051	2

OUTPUT:

ADDRESS	DATA
8052	3

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

8-BIT SUBTRACTION

EXP NO: 2

AIM: To write an assembly language program to implement 8-bit subtraction using 8085 processor.

ALGORITHM:

1) Start the program by loading the first data into the accumulator.

2) Move the data to a register.

3) Get the second data and load it into the accumulator.

4) Subtract the two register contents.

5) Check for borrow.

6) Store the difference and borrow in the memory location.

7) Halt.

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8000	Load accumulator with the first number in the address
MOV B, A	Move the data from accumulator to B register
LDA 8001	Load accumulator with the second number in the address
SUB B	Subtract the data B register with accumulator
STA 8002	Store the data (Output) of the accumulator in the address
RST 1	HALT

Alternate

INPUT:

ADDRESS	DATA
8000	4
8001	5

OUTPUT:

ADDRESS	DATA
8002	1

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

8-BIT MULTIPLICATION

EXP NO: 3

AIM: To write an assembly language program to implement 8-bit multiplication using 8085 processor.

ALGORITHM:

1) Start the program by loading a register pair with the address of memory location.

2) Move the data to a register.

3) Get the second data and load it into the accumulator.

4) Add the two register contents.

5) Increment the value of the carry.

6) Check whether the repeated addition is over.

7) Store the value of product and the carry in the memory location.

8) Halt.

PROGRAM:

MNEMONICS	EXPLANATION
LDA 2200	Load the accumulator with the first number in the address 8500
MOV E,A	Move the data from accumulator to 'E' register
MVI D,00	Move the immediate value 00 into
LDA 2201	Load the accumulator number in the address 2201
MOV C,A	Move the data from accumulator to 'C' register
LXI H,0000	Load the immediate value 0000 into the HL register pair
BACK: DAD D	Back : Label for the loop D ADD: Add the value in register D
DCR C	Decrement register E by 1
JNZ BACK	In register E, is not 0,jump back to the beginning of the loop
SHLD 2202	Store the value on the HL register pair at memory address 2202
HLT	HALT

INPUT:

ADDRESS	DATA
2200	4
2201	2

OUTPUT:

ADDRESS	DATA
2202	8

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

8-BIT DIVISION

EXP NO: 4

AIM: To write an assembly language program to implement 8-bit division using 8085 processor.

ALGORITHM:

1) Start the program by loading a register pair with the address of memory location.

2) Move the data to a register.

3) Get the second data and load it into the accumulator.

4) Subtract the two register contents.

5) Increment the value of the carry.

6) Check whether the repeated subtraction is over.

7) Store the value of quotient and the reminder in the memory location.

8) Halt.

PROGRAM:

MNEMONICS	EXPLANATION
START: NOP	It is often used for code alignment
LDA 8500	Load the accumulator with first number in address 8500
MOV B, A	Move data from accumulator to 'B' register
LDA 8501	Load the accumulator with second number in the address 8501
MVI C,00 H	Move the immediate value 00 into register 'C'
LOOP:CMP B	Loop : Label for loop CMP B: Compare the value in accumulator (A) with (B)
JC LOOP1	If the carrying is (A<B) jump to label loop 1
SUB B	Subtract the value in register (B) from the accumulator (A)
INR C	Increment register C by 1
JMP LOOP	Jump back to the beginning of the loop
Loop1: STA 8502	Store the data in accumulator 8502
MOV A, C	Move the data from `C' register to accumulator
STA 8503	Store the data in the accumulator in 8503
RST 1	Typically transfer control to a predefined interrup service routine
HLT	HALT

INPUT:

ADDRESS	DATA
8500	2
8501	6

OUTPUT:

ADDRESS	DATA
8502	0
8503	3

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

16-BIT ADDITION

EXP NO: 5

AIM:-

To write an assembly language program to implement 16-bit addition using 8085 processor.

ALGORITHM:-

1) Start the program by loading a register pair with address of 1st number.

2) Copy the data to another register pair.

3) Load the second number to the first register pair.

4) Add the two register pair contents.

5) Check for carry.

6) Store the value of sum and carry in memory locations.

7) Terminate the program.

PROGRAM:-

MNEMONICS	Explanation
LDA 3050	Load the content of the memory location 3050H into the accumulator (A).
MOV B,A	Move the content of the accumulator (A) into register B. Now, register B contains the value from 3050H.
LDA 3051	Load the content of the memory location 3051H into the accumulator (A).
ADD B	Add the content of register B (value from 3050H) to the accumulator (A).
STA 3052	Store the result of the addition (from the accumulator) into memory location 3052.
LDA 3053	Load the content of the memory location 3053H into the accumulator (A).
MOV B,A	Move the content of the accumulator (A) into register B.
LDA 3054	Load the content of the memory location 3054H into the accumulator (A).
ADC B	Add the content of register B to the accumulator (A)
STA 3055	Store the result of the addition into memory location 3055.
HLT	Halt the execution of the program.

INPUT:-

Address	Data
3050	2

3051	3
3053	5
3054	5

OUTPUT:-

Address	Data
3052	5
3055	10

RESULT:-

Thus the program was executed successfully using 8085 processor simulator.

16-BIT SUBTRACTION

EXP NO: 6

AIM:-

To write an assembly language program to implement 16-bit subtraction using 8085 processor.

ALGORITHM:-

1) Start the program by loading a register pair with address of 1st number.

2) Copy the data to another register pair.

3) Load the second number to the first register pair.

4) sub the two register pair contents.

5) Check for carry.

6) Store the value of sum and carry in memory locations.

7) End.

PROGRAM:-

MNEMONICS	Explanation
LHLD 2050	Load the 16-bit data from memory locations 2050H and 2051H into the HL register pair.
XCHG	Exchange the contents of the HL and DE register pairs.
LHLD 2052	Load the 16-bit data from memory locations 2052H and 2053H into the HL register pair.
MVI C,00	Move the immediate value 00H into register C.
MOV A, E	Move the content of register E into the accumulator (A).
SUB L	Subtract the content of register L from the accumulator (A).
STA 2054	Store the result of the subtraction into memory location 2054H.
MOV A, D	Move the content of register D into the accumulator (A).
SUB H	Subtract the content of register H from the accumulator (A).
STA 2055	Store the result of the subtraction into memory location 2055H.
HLT	Halt the execution of the program.

INPUT:-

Address	Data
2050	2
2052	3

OUTPUT:-

Address	Data
2054	1
2055	1

RESULT:-

Thus the program was executed successfully using 8085 processor simulator.

16-BIT SUBTRACTION

EXP NO: 6

AIM:-

To write an assembly language program to implement 16-bit subtraction using 8085 processor.

ALGORITHM:-

1) Start the program by loading a register pair with address of 1st number.

2) Copy the data to another register pair.

3) Load the second number to the first register pair.

4) sub the two register pair contents.

5) Check for carry.

6) Store the value of sum and carry in memory locations.

7) End.

PROGRAM:-

```
LHLD 2050
XCHG
LHLD 2052
MOV A, L
SUB E
STA 2054
MOV A, H
SBB D
STA 2055
HLT
```

INPUT:-

Address	Data
2050	34
2051	12

2052	78
2053	56

OUTPUT:-

Address	Data
2054	44
2055	44

RESULT:-

Thus the program was executed successfully using 8085 processor simulator.

16-BIT MULTIPLICATION

EXP NO: 7

AIM:-

To write an assembly language program to implement 16-bit multiplication using 8085 processor.

ALGORITHM:-

1) Load the first data in HL pair.

2) Move content of HL pair to stack pointer.

3) Load the second data in HL pair and move it to DE.

4) Make H register as 00H and L register as 00H.

5) ADD HL pair and stack pointer.

6) Check for carry if carry increment it by 1 else move to next step.

7) Then move E to A and perform OR operation with accumulator and register D.

8) The value of operation is zero, then store the value else go to step

PROGRAM:-

MNEMONICS	Explanation
LHLD 2050	Loads the contents of memory location 2050H and 2051H into the HL register pair.
SPHL	Sets the Stack Pointer (SP) to the value in the HL register pair.
LHLD 2052	Loads the contents of memory location 2052H and 2053H into the HL register pair.
XCHG	Exchanges the contents of the HL and DE register pairs.
LXI H,0000H	Loads the value 0000H into the HL register pair.
LXI B,0000H	Loads the value 0000H into the BC register pair.
AGAIN: DAD SP	Marks the beginning of a loop, Adds the contents of the SP register pair to the HL register pair.
JNC START	Jumps to the START label if the carry flag is not set (i.e., no overflow occurred in the previous operation).
INX B	Increments the BC register pair by 1.
START: DCX D	Marks the start of another loop, Decrements the DE register pair by 1.
MOV A,E	Moves the contents of register E into register A.
ORA D	Performs a logical OR operation between A and D.
JNZ AGAIN	Jumps back to the AGAIN label if the Zero flag is not set (i.e., if the result of the OR operation is non-zero).
SHLD 2054	Stores the contents of the HL register pair into memory locations 2054H and 2055H.
MOV L,C	Moves the contents of register C into register L.
MOV H,B	Moves the contents of register B into register H.
SHLD 2055	Stores the contents of the HL register pair into memory locations 2055H and 2056H.
HLT	Halts the program.

INPUT:-

Address	Data
2050	10
2052	5

OUTPUT:-

Address	Data
2054	50
2055	0

RESULT:-

Thus the program was executed successfully using 8085 processor simulator.

16-BIT DIVISION

EXP NO: 8

AIM:-

To write an assembly language program to implement 16-bit division using 8085 processor.

ALGORITHM:-

1) Read dividend (16 bit)

2) Read divisor

3) count <- 8

4) Left shift dividend

5) Subtract divisor from upper 8-bits of dividend

6) If CS = 1 go to 9

7) Restore dividend

8) Increment lower 8-bits of dividend

9) count <- count - 1

10) If count = 0 go to 5

11) Store upper 8-bit dividend as remainder and lower 8-bit as quotient

12) Stop

PROGRAM:-

```
LDA 8500
MOV B,A
LDA 8501
MVI C,00
LOOP: CMP B
JC LOOP1
SUB B
INR C
JMP LOOP
LOOP1: STA 8502
MOV A,C
STA 8503
HLT
```

INPUT:-

Address	Data
8500	2
8501	21

OUTPUT:-

Address	Data
8502	1 (Rem)
8503	10 (Quo)

RESULT:-

Thus the program was executed successfully using 8085 processor simulator

16-BIT ADDITION

EXP NO: 9

AIM :- To write an assembly language program to implement 16-Bit addition using 8086 processor.

ALGORITHM:-

1-Start the program by loading a register pair with address of 1st number.

2-Copy the data to another register pair.

3-Load the second number to the first register.

4-Add the two register pair contents.

5-Check for carry.

6- Store the value of sum and carry in memory location. Result stored in AX.

7-Terminate the program.

PROGRAM :

MNEMONICS	EXPLANATION
MOV AX, [1100H]	Loads the 16-Bit value from memory address into AX register.
MOV BX, [1102H]	Loads the 16-Bit value from memory address into the BX register.
ADD AX, BX	Adds the value in BX to AX and stores in AX.
MOV [1200H], AX	Moves the 16-Bit value in the AX register into memory address.
HLT	Halts the program execution.

INPUT :-

REGISTER	MEMORY	DATA
AX	32	1100
BX	45	1102

OUTPUT :-

REGISTER	MEMORY	DATA
AX	77	1200

RESULT :- Thus the program was executed successfully using 8086 process simulator.

16 BIT SUBTRACTION

EXP NO: 10

AIM:

To write an assembly language program to implement 16 bit subtraction using 8086 processor.

ALGORITHM:

1] Start the program by loading a register pair with address of first number.

2] Copy the data to another register pair.

3] Laod the second number to first register pair.

4] Subtract the two register pair contacts.

5] Check for borrow.

6] Store the value of difference and borrow in memory location.

7] End.

PROGRAM:

MNEMONICS	**EXPLANATION**
MOV AX, [1100H]	Move the accumulator(A) to [1100]
MOV BX, [1102H]	Move the base(b) to [1102]
SUB AX, BX	Subtract accumulator(A) and base (B)
MOV [1200H], AX	Move [1200] to accumulator (A)
HLT	Halt the program

INPUT:

ADDRESS	**DATA**
1100	30
1102	15

OUTPUT:

ADDRESS	**DATA**
1200	15

RESULT:

Thus the program was executed successfully using 8086 processor simulator.

16-bit multiplication

EXP NO: 11

Aim: To write an assembly language program to implement 16-bit multiplication on 8086 processer.

ALGORITHM:

1. Load the first data in HL pair
2. Move content of HL pair to stack pointer
3. Load the second data in the HL pair and move it to DE
4. Make H register as OH and L register OH
5. Add HL pair and stack pointer
6. Check for carry if carry increment by 1 else move to next step
7. Then move E to A and perform or operation with accumulation and register D
8. The value of operation is zero the solve the value else go to step 3

PROGRAM:

MNEMONICS	EXPLAINATION
MOV AX, [1100 H]	Move the accumulation [A] to [1100]
MOV BX, [1102H]	Move the base [B] to [1102]
MUL BX	Multiply base [B]
MOV [1200H], AX	move [1200]to accumulator [A]
MOV [1202H], DX	move the [1202] to direction [D]
HLT	Halt the program

INPUT:

ADDRESS	DATA
1100	20
1102	3

OUT PUT:

ADDRESS	DATA
1200	60

RESULT:

Thus the program was executed successfully using 8086 emulator.

16 BIT DIVISION

EXP NO: 12

AIM:

To write an assemble language program to implement 16 bit divided using 8086 processor.

ALGORITHM:

1] Read dividend (16) bit.

2] Read divisor.

3] Count <-8.

4] Left shift dividend.

5] Subtract divisor from upper 8 bits of dividend.

6] If cs=1 go to 9.

7] Restore dividend.

8] Increment lower 8 bits of dividend.

9] Count <- count -1.

10] If count =0 go to 5.

11] Store upper 8 bit dividend as remainder and lower 8 bit as quotient.

12] Stop.

PROGRAM:

MNEMONICS	**EXPLANATION**
MOV AX, [1100H]	Move the accumulator (A) to [1100]
MOV BX, [1102H]	Move the base (B) to [1102]
DIV BX	Divide by base (B)
MOV [1200H], AX	Move [1200] to accumulator (A)
MOV [1202H], DX	Move [1202] to direction (D)
HLT	Halt the program

INPUT:

ADDRESS	**DATA**
1100	10
1102	10

OUTPUT:

ADDRESS	**DATA**

1200	1
1202	0

RESULT:

Thus the program was executed successfully using 8086 processor simulator.

Greatest of 2 numbers

EXP NO: 13

AIM:-

To write an Assembly Language Program to find the smallest number in an array using 8085 Microprocessor in GNUSim8085.

ALGORITHM:-

1. Initialize the count
2. Get the input numbers
3. Compare the content of Accumulator(A) with HL pair for all input numbers
4. Stores the smallest number in the output register
5. End the program

PROGRAM:-

MNEMONICS	EXPLANATION
LDA 2050	Load the content of the memory location 2050H into the accumulator (A).
MOV B,A	Move the content of the accumulator (A) into register B. At this point, B contains the value from 2050H.
LDA 2051	Load the content of the memory location 2051H into the accumulator (A).
CMP B	Compare the content of register B with the accumulator (A).
JNC STORE	Jump to the label STORE if there is **no carry**, which means A >= B.
MOV A,B	If there **is a carry** (i.e., A < B), move the content of B (the smaller value) into the accumulator.
STORE: STA 2052	Store the content of the accumulator (A) into the memory location 2052H.
HLT	Halt the execution of the program.

Input

Address	Data
2050	29
2051	22

Output

Address	Data
2052	29

RESULT:

Thus the Assembly Language Program to find the smallest number in an array using 8085 Microprocessor in GNUSim is performed.

Smallest of 2 numbers

EXP NO: 14

AIM:-

To write an Assembly Language Program to find the smallest number in an array using 8085 Microprocessor in GNUSim 8085.

ALGORITHM:-

1. Initialize the count
2. Get the input numbers
3. Compare the content of Accumulator(A) with HL pair for all input numbers
4. Stores the smallest number in the output register
5. End the program

PROGRAM:-

MNEMONICS	EXPLANATION

LDA 2050	Load the content of the memory location 2050H into the accumulator (A).
MOV B,A	Move the content of the accumulator (A) into register B. At this point, B contains the value from 2050H.
LDA 2051	Load the content of the memory location 2051H into the accumulator (A).
CMP B	Compare the content of register B with the accumulator (A).
JC STORE	Jump to the label STORE if there is **no carry**, which means A >= B.
MOV A,B	If there **is a carry** (i.e., A < B), move the content of B (the smaller value) into the accumulator.
STORE: STA 2052	Store the content of the accumulator (A) into the memory location 2052H.
HLT	Halt the execution of the program.

Input

Address	Data
2050	29
2051	22

Output

Address	Data
2053	29

RESULT:

Thus the Assembly Language Program to find the smallest number in an array using 8085 Microprocessor in GNUSim is performed.

SWAPING OF TWO 8-BIT DATA

EXP NO: 15

AIM:

To Write an assembly language program to swap two 8-bit data using 8085 processor.

ALGORITHM:

1. Load the contents of memory address 1100 into accumulator A.

2. Move the contents of accumulator A into register B.

3. Load the contents of memory address 1101 into accumulator A.

4. Move the contents of accumulator A into register C.

5. Store the contents of accumulator A (which is the original value at 1101) into memory address 1102.

6. Move the contents of register B (which is the original value at 1100) into accumulator A.

7. Store the contents of accumulator A into memory address 1103.

PROGRAM:

```
LDA 1100     ; Load the contents of memory address 1100 into accumulator A
```

MOV B, A ; Move the contents of accumulator A into register B

LDA 1101 ; Load the contents of memory address 1101 into accumulator A

MOV C, A ; Move the contents of accumulator A into register C

STA 1102 ; Store the contents of accumulator A into memory address 1102

MOV A, B ; Move the contents of register B into accumulator A

STA 1103 ; Store the contents of accumulator A into memory address 1103

HLT ; Halt the program execution

INPUT:

ADDRESS	DATA
1100	6
1101	4

OUTPUT:

ADDRESS	DATA
1102	4
1103	6

RESULT:

Thus the program was executed successfully using 8085 processor simulator.

1's COMPLIMENT

EXP NO: 16

AIM :

To write assembly language to find 1's COMPLIMENT by using 8085 microprocessor Simulator

ALGORITHM:

1. Loads the value from memory address 8000 into accumulator A.

2. Complements the bits of the value in accumulator A using the CMA (Complement Accumulator) instruction. This means that all 1s become 0s and all 0s become 1s.

3. Stores the complemented value into memory address 8001.

4. Halts the program execution.

PROGRAM:

Mnemonics	**Explanation**
LDA 8000	Load the contents of memory address 8000 into accumulator A
CMA	Complement the contents of accumulator A (i.e., flip all bits)
STA 8001	Store the complemented value into memory address 8001
HLT	Halt the program execution

INPUT:

ADDRESS	DATA
8000	6

OUTPUT:

ADDRESS	DATA
8001	249

RESULT: THIS PROGRAM WAS EXECUTED SUCCESSFULLY BY USING 8085 MICROPROCESSOR SIMULATOR

2'S COMPLEMENT

EXP NO: 17

AIM:

To write an assembly language program to find 2's complement of 8-bit number

ALGORITHM:

1) Start with the binary number:
2) If the number is positive, simply write its binary equivalent.
3) If the number is negative, begin with the binary equivalent of its positive value.
4) Invert all the bits (1's complement):
5) Flip every 0 to 1 and every 1 to 0
6) Add 1 to the result:
7) Add 1 to the least significant bit (rightmost bit) of the inverted number.
8) The final result is the 2's complement representation of the number.

PROGRAM:

LDA 3000	LOAD THE ACCUMULATER WITH THE CONTENT OF MEMORY LOACTION 3000
CMA	COMPLEMENTARY ADDITIVE

STA 3001	STORE DATA OF AACCUMULATOR IN ADDRESS 3000
ADI 3002	ADD THE IMMEDTIE VALUE TO THE CONTENT OF REGISTER AT MERORY ADDRESS 3002
HLT	HALT

INPUT:

ADDRESS	**DATA**
3000	**8**

OUTPUT

ADDRESS	**DATA**
3001	**247**
3002	**0**

ESULT: Thus the PROGRAM WAS EXECUTED SUCCESSFULLY USING 8085 PROCESSOR SIMULATER

ODD OR EVEN – 8085 MICROPROCESSOR

EXP NO: 18

AIM:

To write an assembly language program to find the number is odd or even using 8085 Microprocessor in GNUSim8085

ALGORITHM:-

1. **I**nitialize the number in the accumulator
2. Perform the AND operation with accumulator by 01
3. If the result is '0', it means it is even number (indicates as 22)
4. If the result is non zero , it means the given number is odd (indicates as 11)
5. Stores the out put in the register
6. End the program

Program :-

MNEMONICS	EXPLANATION
LDA 8050	Load the accumulator with the content of memory Location 8050H
ANI 01	Logical and operation with accumulator and immediate value 01
JZ LOOP1	Jump to loop1 if the result of the AND operation is zero

MVI A,11	Move immediate value 11 into the accumulator (odd number)
JMP LOOP2	Jump to loop2
LOOP1: MVI A,22	Move immediate value 22 into the accumulator (even number)
LOOP2: STA 8051	Store the accumulator content at memory location 8051
HLT	Halt the process

Input :

Address	Data
8050	20

Address	Data
8050	19

Output:

Address	Data
8051	22

Address	Data
8051	11

RESULT:-

Thus the assembly Language Program to find the ODD OR EVEN is performed using 2050H Microprocessor in GNUSim8085

POSITIVE AND NEGATIVE

EXP.NO : 19

AIM:

To write an assembly language program to find the number is POSITIVE AND NEGATIVE using 8085 Microprocessor in GNUSim

ALGORITHM:-

1. **I**nitialize the number in the accumulator
2. Perform the AND operation with accumulator by 01
3. If the result is '0', it means it is even number (indicates as 22)
4. If the result is non zero , it means the given number is odd (indicates as 11)
5. Stores the out put in the register
6. End the program

Program :-

```
LDA 8050H
ANI 80H
JZ POS
MVI A,11
JMP STO
POS: MVI A,22
STO: STA 8051H
HLT
```

OUT PUT :-

Input :

Address	Data
8050H	2

Address	Data
8050H	80H (Input should be given in the memory window)

Output:

Address	Data
8051H	22

Address	Data
8051H	11

RESULT:-

Thus the assembly Langua6g36e5 885250Program to find the positive or negative is performed using 2050H Microprocessor in GNUSim

ASCENDING ORDER – 8085 MICROPROCESSOR

EXP NO: 20

AIM:

To write an assembly language program to find the ascending order of numbers using 8085 Microprocessor in GNUSim8085

ALGORITHM:-

1. **I**nitialize the count
2. Get the input numbers
3. compare content accumulator [A] with HL pair for all input numbers
4. stores the ascending numbers in the output registers
5. end the program

Program :-

MNEMONICS	EXPLANATION
LXI H,8000	Load H and L register with addresss 8000
MOV C,M	Mov the content of memory at HL to register C
DCR C	Decrement the value in register C
LOOP3: MOV D,C	Loop3 mov the content of register c to D
LXI H,8001	Load Hand L register with address 8001
LOOP2: MOV A,M	Loop2 mov the content of memory at HL to register A
INX H	Increment HL
CMP M	Compare the memory
JC LOOP1	Jump if carry in loop1
MOV B,M	Mov the content from memory at HL to register B
MOV M,A	Mov the content of register A to memory at HL

DCX H	Decrement the content HL
MOV M,B	Mov the content of register B to memory at HL
INX H	Increment HL
LOOP1: DCR D	Loop1 decrease the value in register D
JNZ LOOP2	Jump to loop2 if the zero flag is not set
DCR C	Decrease the value of register C
JNZ LOOP3	Jump to loop3 if the zero flag is not set
HLT	Halt the process

Input :

Address	Data
8000	3
8001	4
8002	18

Output:

Address	Data
8001	3
8002	4
8003	18

RESULT: Thus the assembly Language Program to find the Ascending order of numbers is performed using 8085 Microprocessor in GNUSim8085

DESCENDING ORDER

EXP NO: 21

AIM:-

To write an assembly language program to implement descending order using 8085 processor.

ALGORITHM:-

1) Load the number of elements in the array (N) into a register.

2) Use nested loops:

Outer loop: Decrease the range of comparison in each iteration.

Inner loop: Compare adjacent elements and swap if needed.

3) Repeat until the array is sorted in descending order.

PROGRAM:-

```
LXI H,8050
MOV C,M
DCR C
LOOP3: MOV D,C
LXI H,8051
LOOP2: MOV A,M
INX H
CMP M
JNC LOOP1
MOV B,M
MOV M,A
DCX H
MOV M,B
INX H
LOOP1: DCR D
JNZ LOOP2
DCR C
```

JNZ LOOP3

HLT

INPUT:-

Address	Data
8051	9
8052	8
8053	6
8054	5
8055	2

OUTPUT:-

Address	Data
8051	9
8052	8
8053	6
8054	5
8055	2

RESULT:-

Thus the program was executed successfully using 8085 processor simulator.

LARGEST NUMBER IN AN ARRAY

EXP NO: 22

AIM:

To write an Assembly Language Program to find the largest number in an array using 8085 Microprocessor in GNUSim.

ALGORITHM:

1. Initialize the count
2. Get the input numbers
3. Compare the content of Accumulator(A) with HL pair for all input numbers
4. Stores the largest number in the output register
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Load H and L registers with address 8050
MOV C, M	Move the content of memory at HL to register C
INX H	Increment HL
MOV B, M	Move the content of memory at HL to register B
DCR C	Decrement the value in register C
LOOP: INX H	LOOP: Increment HL
MOV A, M	Move the content of memory at HL to register A
CMP B	Compare A and B
JC SKIP	Jump to SKIP if the carry flag is set (A < B)
MOV B, A	Move the content of register A to register B
SKIP: DCR C	Decrement the value in register C
JNZ LOOP	Jump to LOOP if the zero flag is not set (C != 0)
LXI H, 8500	Load H and L registers with address 8500
MOV M, B	Move the content of register B to the memory at HL
HLT	Halt the microprocessor

Input

Address	Data
8050 (Counter)	5

Address	Data
8051	5
8052	2
8053	6
8054	8
8055	9

Output:

Address	Data
8500	9

RESULT:

Thus the Assembly Language Program to find the largest number in an array using 8085 Microprocessor in GNUSim is performed.

SMALLEST NUMBER IN AN ARRAY

EXP NO: 23

AIM:

To write an Assembly Language Program to find the smallest number in an array using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the count
2. Get the input numbers
3. Compare the content of Accumulator(A) with HL pair for all input numbers
4. Stores the smallest number in the output register
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Load H and L registers with address 8000
MOV C, M	Move the content of memory at HL to register C
INX H	Increment HL
MOV B, M	Move the content of memory at HL to register B
DCR C	Decrement the value in register C
LOOP: INX H	LOOP: Increment HL
MOV A, M	Move the content of memory at HL to register A
CMP B	Compare A and B
JNC SKIP	Jump to SKIP if the carry flag is set (A < B)
MOV B, A	Move the content of register A to register B
SKIP: DCR C	Decrement the value in register C
JNZ LOOP	Jump to LOOP if the zero flag is not set (C != 0)
LXI H,8500	Load H and L registers with address 8500
MOV M, B	Move the content of register B to the memory at HL
HLT	Halt the microprocessor

Input

Address	Data
8050 (Counter)	5

Address	Data
8051	2
8052	4
8053	7
8054	5
8055	9

Output:

Address	Data
8500	2

RESULT:

Thus the Assembly Language Program to find the smallest number in an array using 8085 Microprocessor in GNUSim is performed.

LCM – 8085 MICROPROCESSOR

EXP NO: 24

AIM:

To write an assembly language program to find the LCM of numbers using 8085 Microprocessor in GNUSim8085

SOFTWARE USED:-

GNUSim8085

ALGORITHM:-

1. start the program.
2. Load A into the accumulator.
3. Move A to R1.
4. Load B into the accumulator.
5. Move B to R2.
6. Call the GCD subroutine (the GCD subroutine is already implemented using the Euclidean algorithm).
7. Compute Product:
8. Multiply A and B to get the product.
9. Store this product temporarily.
10. Divide the Product by GCD to get the LCM.
11. Store the LCM at a memory location (e.g., 6009).
12. Halt the program

PROGRAM:-

MNEMONICS	EXPLANATION
LXI H,8000	Load H and L register with addresss 8000
MOV C,M	Mov the content of memory at HL to register C
MVI B,00	Mov immediately value 00 into the B register
INX H	Increment HL
MOV B,M	Mov the content of memory at HL to register B
CMA	Complement accumulator
MOV E,A	Mov the content of register A to register E
MVI D,00FH	Mov immediately the value 00 into the register D
MOV A,B	Mov the content of register B to register A
CMA	Complement accumulator
MOV D,A	Mov the content of register A to register D

INX D		Increment the register pair DE
LXI H,0000		Load H and L register with address 0000
NEXT: DAD B		Next double add the content of register pair to HL pair
SHLD 8010		Store HL direct into the memory location 8010 and 8011
LOOP: DAD D		double add
JNC SKIP		Jump to skip if the carry flag set [A<B]
MOV A,H		Mov the content of register H to register to A
ORA L		OR operation between the accumulator and L register
JZ EXIT		If the result is zero then jump to the exit
JMP LOOP		Jump to loop if the zero flag is not set
SKIP: LHLD 8010		Load from the memory location of 8010 into HL register pair
JMP NEXT		Jump unconditionally to the next label
EXIT: LHLD 8010		Load the Hl pair stored at the memory location 8010
HLT		Halt the process

Input :

Address	Data
8000	60
8001	45

Output:

Address	Data
8011	180

RESULT: Thus the assembly Language Program to find the LCM of numbers is performed using 8085 Microprocessor in GNUSim8085

GCD

EXP NO: 25

AIM:
To write an assembly language program to implement GCD using 8085 processor.

ALGORITHM:

1) Start the program by loading the first number (A) into the accumulator.
2) Move the first number (A) to a register (R1) to store it temporarily.
3) Get the second number (B) and load it into the accumulator.
4) Compare if B is greater than 0 (i.e., check if the divisor is non-zero).
5) Perform division of A by B and calculate the remainder.
6) If the remainder is 0, the GCD is B. Store the result in memory (at a designated location).
7) If the remainder is not 0, move B to register R1 and load the remainder into the accumulator.
8) Repeat the steps from step 4 (looping back) until the remainder becomes 0.
9) Store the result (GCD) when the loop terminates and the remainder is 0.
10) Halt the program after completing the process.

Program

MNEMONICS	**EXPLANATION**
LXI H,6000	Load H and L register with address 6000
MOV A,M	Move the content of memory at HL to register A
INX H	Increment of HL
MOV B,M	Move the content of memory at HL to register B
LOOP: CMP B	CMP B compares the contents of register A (accumulator) with register B.
JZ STORE	If A equals B, it means we have found the GCD, and the program jumps to STORE to save the result.
JC EXG	If **A** is smaller than **B** (carry is set), the program jumps to **EXG**, where the values of A and B are exchanged.
SUB B	Subtracts **B** from **A**, and the loop repeats the process.
JMP LOOP	Jump back to the LOOP label and repeat the process.
EXG: MOV C,B	The value of **B** is saved into **C**.
MOV B,A	The value of **A** is then moved into **B**.
MOV A,C	Finally, the value of **C** (which holds the original value of **B**) is moved into **A**.
JMP LOOP	Jump back to the LOOP label and repeat the process
STORE: STA 6009	Store the data (output) of the accumulator in address 6009
HLT	Halt

INPUT:

ADDRESS	**DATA**
6000	38

6001	19

OUTPUT:

ADDRESS	DATA
6009	19

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

Factorial

EXP NO: 26

AIM :

To Write an assembly language program to find factorial of n in the given

ALGORITHM:

1. Load the address 8050H into the HL register pair.
2. Move the value from the memory location (8050H) into the B register.
3. Load the value 01H into the D register to serve as an accumulator for the factorial result.
4. Call the subroutine MUL to multiply the current value of D (partial factorial) by B.
5. Decrement the B register to move to the next value in the factorial computation.
6. Check if B is zero. If not, jump back to the label FACT.
7. Increment the HL register to point to the next memory location (8051H).
8. Store the result from the D register at the memory location pointed to by HL.
9. Halt the program.
10. Move the current value of B into the E register (as a multiplier).
11. Clear the A register (set it to 0) to use as a running total for the multiplication.
12. Perform repeated addition (ADD D) E times to compute the product.
13. Decrement the E register after each addition and check if E is zero.
14. When E becomes zero, move the result from A to D and return.

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Loads the 16-bit address 8050H into the H register pair (HL).
MOV B,M	Moves the contents of the memory location pointed to by HL into the B register.
MVI D,01H	Moves the immediate value 01H into the D register.
FACT: CALL MUL	Calls the MUL subroutine
DCR B	Decrements the B register
JNZ FACT	Jumps back to the FACT label if B is not zero.
INX H	Increments the H register pair (HL).
MOV M,D	Moves the contents of the D register into the memory location pointed to by HL
HLT	Halts the program.
MUL: MOV E,B	Moves the contents of the B register into the E register.
XRA A	Clears the A register by performing a bitwise XOR with itself.
ML: ADD D	Adds the contents of the D register to the A register.
DCR E	Decrements the E register.
JNZ ML	Jumps back to the ML label if E is not zero.

MOV D,A	Moves the contents of the A register into the D register
RET	Returns from the MUL subroutine.
HLT	halt

INPUT:

ADDRESS	DATA
8050	5

OUTPUT:

ADDRESS	DATA
8051	120

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

DECIMAL TO HEXA DECIMAL

EXP NO: 27

AIM: Write a program to convert Decimal number to Hexadecimal number

SOFTWARE : GNUSIM 8085

ALGORITHM:

1. Initialize Registers:
2. Store the decimal number in a register (e.g., register B).
3. Perform repeated division of the decimal number by 16 to obtain the hexadecimal digits.
4. Store the quotient in a register (e.g., B or C).
5. Store the remainder (hex digit) separately.
6. If the remainder is greater than 9, convert it to its corresponding ASCII representation for A-F (e.g., add 7 to the remainder).
7. Store the hexadecimal digits (remainders) in reverse order in memory.
8. If the quotient is zero, the conversion is complete. Otherwise, repeat the division step with the quotient as the new dividend.
9. Use the stored hexadecimal digits to display the result.

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,2050	loads the immediate 16-bit value 2050 into the HL register pair.
MOV A,M	moves the content of the memory AT HL TO register A
ADD A	Add the content of register A to the accumlactor
MOV B, A	MOVE the content of the memory at hl to register B
ADD A	Add the content of register A to the accumlactor
ADD A	Add the content of register A to the accumlactor
ADD B	Add the content of register B to the accumlactor
INX H	This opcode corresponds to **incrementing the HL register pair**.
ADD M	Add the content of register Mto the accumlactor
INX H	This opcode corresponds to **incrementing the HL register pair**.
MOV M,A	MOVE the content of the memory at hi to register m
HLT	Halt

INPUT:

ADDRESS	DATA
2050	34

OUTPUT:

ADDRESS	DATA
2051	0
2052	84

RESULT: Thus the PROGRAM WAS EXECUTED SUCCESSFULLY USING 8085 PROCESSOR SIMULATER