Exp.No:1

8 BIT ADDITION

AIM:

To write an Assembly Language Program to perform addition of two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 without carry and with carry using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize two 8-bit data in registers
2. Add the data in registers
3. Store the results in some other register
4. End the program

PROGRAM:

1) ADDITION WITHOUT CARRY

MNEMONICS	EXPLANATION
LDA 8050	Load Accumulator with the first number in the address 8050
MOV B,A	Move the data from Accumulator to 'B' Register
LDA 8051	Load Accumulator with the 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 the address 8052
HLT	Halt

2) ADDITION WITH CARRY

MNEMONICS	EXPLANATION
MVI C,00	Clear the 'C' register
LDA 8050	Load Accumulator with the first number in the address 8050
MOV B,A	Move the data from Accumulator to 'B' Register

LDA 8051	Load Accumulator with the second number in the address 8051
ADD B	Add the data of ‘B’ Register with Accumulator
JNC LOOP	Jump if No Carry in the addition
INR C	Increment ‘C’ regsiter
LOOP: STA 8052	Store the data (output) of the Accumulator in the address 8052
MOV A,C	Move Carry in ‘C’ register to the Accumulator
STA 8053	Store the Carry output of the Accumulator in the address 8053
HLT	Halt

OUTPUT:

WITHOUT CARRY:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	

WITH CARRY:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	
8053	

RESULT:

Thus the Assembly Language Program to perform addition of two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 without carry and with carry is performed using 8085 Microprocessor in GNUSim.

Exp.No:2

8 BIT SUBTRACTION

AIM:

To write an Assembly Language Program to perform subtraction of two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize two 8-bit data in registers
2. Sub the data in registers
3. Store the results in some other registers
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
MVI C,00	Clear the 'C' register
LDA 8050	Load Accumulator with the first number in the address 8050
MOV B,A	Move the data from Accumulator to 'B' Register
LDA 8051	Load Accumulator with the second number in the address 8051
SUB B	Add the data of 'B' Register with Accumulator
JNC LOOP	Jump if No Carry in the addition
CMA	Complement the data of Accumulator (1's Complement)
INR A	Increment Accumulator (2's Complemet)
INR C	Increment 'C' regsiter
LOOP: STA 8052	Store the data (output) of the Accumulator in the address 8052
MOV A,C	Move Carry in 'C' register to the Accumulator
STA 8053	Store the Carry output of the Accumulator in the address 8053
HLT	Halt

OUTPUT:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	
8053	

RESULT:

Thus the Assembly Language Program to perform subtraction of two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 is performed using 8085 Microprocessor in GNUSim.

Exp.No:3

8 BIT MULTIPLICATION

AIM:

To write an Assembly Language Program to perform multiplication of two 8-bit data using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the Multiplicand data in B register and set Multiplier data in C register as Counter.
2. Add the contents of Accumulator and B register
3. Repeat the addition until C=0.
4. Store the results in the registers
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050	Load the accumulator with the data at memory address 8050.
MOV B, A	Move the content of the accumulator to register B.
LDA 8051	Load the accumulator with the data at memory address 8051.
MOV C, A	Move the content of the accumulator to register C.
XRA A	XOR the accumulator with itself, effectively clearing its content.
LOOP: ADD B	Add the content of register B to the accumulator in a loop.
DCR C	Decrement the content of register C.
JNZ LOOP	Jump to the label LOOP if the zero flag is not set (if C is not zero).
STA 8052	Store the content of the accumulator at memory address 8052.
HLT	Halt the execution of the program.

OUTPUT:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	

RESULT:

Thus the Assembly Language Program to perform multiplication of two 8-bit data is performed using 8085 Microprocessor in GNUSim.

Exp.No:4

8 BIT DIVISION

AIM:

To write an Assembly Language Program to perform division of two 8-bit data using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the data in B register and A register
2. Clear the Counter register C.
3. Compare A and B register, if A>B, subtract B from A and increment C register.
4. C register increments until A<B.
5. If A<B, Stores the value in the accumulator (A), which now holds the remainder.
6. C register stores the quotient.
7. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8500	Loads the value from memory address 8050 into the accumulator (A).
MOV B, A	Moves the value from the accumulator (A) to register B.
LDA 8501	Loads the value from memory address 8051 into the accumulator (A).
MVI C,00	Initializes register C with the value 00.
LOOP: CMP B	Compares the value in the accumulator (A) with the value in register B.
JC LOOP1	If the value in the accumulator is less than the value in register B, jump to the label LOOP1.
SUB B	Subtracts the value in register B from the value in the accumulator (A).
INR C	Increments the value in register C by 1.
JMP LOOP	Jumps back to the beginning of the loop.
LOOP1: STA 8502	Stores the value in the accumulator (A), which now holds the remainder, in memory address 8052.
MOV A, C	Moves the value from register C, which holds the quotient, to the accumulator (A).
STA 8503	Stores the value in the accumulator (A), which now holds the

	quotient, in memory address 8053.
HLT	Halt the execution of the program.

OUTPUT:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	
8053	

RESULT:

Thus the Assembly Language Program to perform division of two 8-bit data is performed using 8085 Microprocessor in GNUSim.

Exp.No:5

GREATEST OF TWO NUMBERS

AIM:

To write an Assembly Language Program to find the greatest of two 8-bit numbers using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the numbers in the registers.
2. Compare the Accumulator and B register.
3. If A>B, carry flag is 0.Otherwise it set to 1.
4. Store the results in the registers
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050	Load the accumulator (A) register with the content of memory location 8050H.
MOV B,A	Move the content of the accumulator (A) to the B register.
LDA 8051	Load the accumulator (A) register with the content of memory location 8051H.
CMP B	Compare the content of the B register with the accumulator (A) without modifying any register.
JNC STORE	Jump to the label STORE if no carry flag is set after the comparison.
MOV A,B	Move the content of the B register back to the accumulator (A).
STORE: STA 8052	Store the content of the accumulator (A) into memory location 8052H.
HLT	Halt the execution.

OUTPUT:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	

RESULT:

Thus the Assembly Language Program to find the greatest of 8-bit data is performed using 8085 Microprocessor in GNUSim.

Exp.No:6

SMALLEST OF TWO NUMBERS

AIM:

To write an Assembly Language Program to find the smallest of two 8-bit numbers using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the numbers in the registers.
2. Compare the Accumulator and B register.
3. If A<B, carry flag is set to 1. Otherwise it set to 0.
4. Store the results in the registers
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050	Load the content of memory location 8050 into register A
MOV B,A	Move the value from register A to register B
LDA 8051H	Load the content of memory location 8051H into register A
CMP B	Compare the content of register B with the content of register A
JNC STORE	Jump to the STORE label if no carry flag is set after the comparison
STORE: MOV A,B	If the jump condition is not met, move the value from register B to register A
STA 8052H	Store the value in register A into memory location 8052H
HLT	Halt the execution.

OUTPUT:

Input:

Address	Data
8050	
8051	

Output:

Address	Data
8052	

RESULT:

Thus the Assembly Language Program to find the greatest of 8-bit data is performed using 8085 Microprocessor in GNUSim.

Exp.No:7

1's COMPLEMENT

AIM:

To write an Assembly Language Program to find 1's complement of 8-bit number using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the number in the Accumulator.
2. Complement the contents of the Accumulator.
3. Store the result in the register
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050	Load the content of memory location 8050 into register A
CMA	Complement the contents of the Accumulator
STA 8051	Store the value in register A into memory location 8051
HLT	Halt the execution.

OUTPUT:

Input:

Address	Data
8050	

Output:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the greatest of 8-bit data is performed using 8085 Microprocessor in GNUSim.

Exp.No:8

2's COMPLEMENT

AIM:

To write an Assembly Language Program to find 2's complement of 8-bit number using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the number in theAccumulator.
2. Complement the contents of the Accumulator.
3. Store the result in the register
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050H	Load the content of memory location 8050 into register A
CMA	Complement the contents of the Accumulator
STA 8051H	Store the value (1's Complement) in register A into memory location 8051
INR A	Increment A
STA 8052H	Store the value (2's Complement) in register A into memory location 8052
HLT	Halt the execution.

OUTPUT:

Input:

Address	Data
8050	

Output:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the greatest of 8-bit data is performed using 8085 Microprocessor in GNUSim.

Exp.No:9

ODD OR EVEN NUMBER

AIM:

To write an Assembly Language Program to find the number is ODD or EVEN using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize 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 output in the registers
6. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050H	Load the accumulator with the content of memory location 8050
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 8051H	Store the accumulator content at memory location 8051H
HLT	Halt the processor

OUTPUT:

Input:

Address	Data
8050	

Output:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the given number is ODD or EVEN is performed using 8085 Microprocessor in GNUSim.

Exp.No:10

POSITIVE OR NEGATIVE NUMBER

AIM:

To write an Assembly Language Program to find the number is positive or negative using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the number in the Accumulator.
2. Perform the AND operation with Accumulator by 80.
3. If the result is '0' , it means it is Negative number (indicates as 11)
4. If the result is non-zero , it means the given number is positive number (indicates as 22)
5. Stores the output in the registers
6. End the program

PROGRAM:

MNEMONICS	EXPLANATION
MVI A,9AH	Move immediate value 9AH into the accumulator
ANI 80H	Logical AND operation with accumulator and immediate value 80
JZ NEG	Jump to NEG if the result of the AND operation is zero
MVI A,22	Move immediate value 22 into the accumulator [Positive number]
JMP STO	Jump to STO
NEG: MVI A,11	Move immediate value 11 into the accumulator [Negative number]
STO: STA 8501H	Store the accumulator content at memory location 8501
HLT	Halt the processor

OUTPUT:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the positive or negative number is performed using 8085 Microprocessor in GNUSim.

Exp.No:11

ASCENDING ORDER

AIM:

To write an Assembly Language Program to arrange the given numbers in ascending order 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 output in the registers
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Load HL register pair with the address 8050H
MOV C,M	Move the content of the memory location pointed by HL to register C
DCR C	Decrement the value in register C
LOOP3: MOV D,C	Move the content of register C to register D
LXI H,8051	Load HL register pair with the address 8051H
LOOP2: MOV A,M	Move the content of the memory location pointed by HL to register A
INX H	Increment the HL pair
CMP M	Compare the content of the memory location pointed by HL with A
JC LOOP1	Jump to LOOP1 if A is less than the content of the memory
MOV B,M	Move the content of the memory location pointed by HL to register B
MOV M,A	Move the content of A to the memory location pointed by HL
DCX H	Decrement the HL pair
MOV M,B	Move the content of B to the memory location pointed by HL
INX H	Increment the HL pair
LOOP1: DCR D	Decrement the value in register D
JNZ LOOP2	Jump to LOOP2 if D is not zero
DCR C	Decrement the value in register C
JNZ LOOP3	Jump to LOOP3 if C is not zero
HLT	Halt the program

Input

Address	Data
8051	
8052	
8053	
8054	
8055	

Output:

Address	Data
8051	
8052	
8053	
8054	
8055	

RESULT:

Thus the Assembly Language Program to arrange the given numbers in ascending order using 8085 Microprocessor in GNUSim.

Exp.No:12

DESCENDING ORDER

AIM:

To write an Assembly Language Program to arrange the given numbers in descending order 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 output in the registers
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Load HL register pair with the address 8050H
MOV C,M	Move the content of the memory location pointed by HL to register C
DCR C	Decrement the value in register C
LOOP3: MOV D,C	Move the content of register C to register D
LXI H,8051	Load HL register pair with the address 8051H
LOOP2: MOV A,M	Move the content of the memory location pointed by HL to register A
INX H	Increment the HL pair
CMP M	Compare the content of the memory location pointed by HL with A
JNC LOOP1	Jump to LOOP1 if no carry (A is not less than the content of the memory)
MOV B,M	Move the content of the memory location pointed by HL to register B
MOV M,A	Move the content of A to the memory location pointed by HL
DCX H	Decrement the HL pair
MOV M,B	Move the content of B to the memory location pointed by HL
INX H	Increment the HL pair
LOOP1: DCR D	Decrement the value in register D
JNZ LOOP2	Jump to LOOP2 if D is not zero
DCR C	Decrement the value in register C
JNZ LOOP3	Jump to LOOP3 if C is not zero
HLT	Halt the program

Input

Address	Data
8051	
8052	
8053	
8054	
8055	

Output:

Address	Data
8051	
8052	
8053	
8054	
8055	

RESULT:

Thus the Assembly Language Program to arrange the given numbers in descending order using 8085 Microprocessor in GNUSim.

Exp.No:13

SMALLEST NUMBER IN AN ARRAY

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

Input

Address	Data
8051	
8052	
8053	
8054	
8055	

Output:

Address	Data
8500	

RESULT:

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

Exp.No:14

LARGEST NUMBER IN AN ARRAY

AIM:

To write an Assembly Language Program to find the largest 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 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	
8052	
8053	
8054	
8055	

Output:

Address	Data
8500	

RESULT:

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

Exp.No:15

PRIME NUMBER

AIM:

To write an Assembly Language Program to find the given number is prime number or not using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the count
2. Get the input numbers
3. Apply the logic to find the prime number
4. Stores the output in the output register
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Load H and L registers with address 8050
MOV A,M	Move the content of memory at HL to register A
MVI C,00	Move immediate 00 to register C
MOV D,A	Move the content of register A to register D
MOV E,A	Move the content of register A to register E
L2: MOV B,D	Move the content of register D to register B
L1: CMP B	Compare A and B
JC LABEL	Jump to LABEL if the carry flag is set (A < B)
SUB B	Subtract B from A
JNZ L1	Jump to L1 if the result is not zero
LABEL: CPI 00	Compare immediate 00 with A
JNZ SKIP	Jump to SKIP if the zero flag is not set (A != 00)
INR C	Increment the content of register C
SKIP: MOV A,E	Move the content of register E to register A
DCR D	Decrement the content of register D
JNZ L2	Jump to L2 if the zero flag is not set (D != 0)
MOV A,C	Move the content of register C to register A
STA 8051	Store the content of register A to the memory at address 8051
HLT	Halt the microprocessor

Input

Address	Data
8050	

Output:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the given number is prime number or not using 8085 Microprocessor in GNUSim is performed.

Exp.No:16

FACTORIAL

AIM:

To write an Assembly Language Program to find the factorial of the given number using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Initialize the count
2. Get the input numbers
3. Apply the logic to find the factorial of the given input
4. Stores the output in the register
5. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H, 8050	Load H and L registers with address 8050
MOV B,M	Move the content of memory at HL to register B
MVI D, 01H	Move immediate 01H to register D
FACT: CALL MUL	Call the subroutine MUL
DCR B	Decrement the value in register B
JNZ FACT	Jump to FACT if the zero flag is not set (B != 0)
INX H	Increment HL
MOV M,D	Move the content of register D to the memory at HL
HLT	Halt the microprocessor
MUL: MOV E,B	Move the content of register B to register E
XRA A	XOR A with itself (clear A)
ML: ADD D	Add the content of register D to A
DCR E	Decrement the value in register E
JNZ ML	Jump to ML if the zero flag is not set (E != 0)
MOV D,A	Move the content of register A to register D
RET	Return from the subroutine
HLT	Halt the microprocessor

Input

Address	Data
8050	

Output:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the factorial of the given number using 8085 Microprocessor in GNUSim is performed.

Exp.No:17

GCD of 2 NUMBERS

AIM:

To write an Assembly Language Program to find the GCD of two numbers using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Get the input numbers
2. Apply the logic to find determine the GCD of the given inputs
3. Stores the output in the register
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H,8050	Load the address 8050H into the HL register pair
MOV A, M	Move the contents of the memory location pointed to by HL into the accumulator (A)
INX H	Increment the HL register pair to point to the next memory location
MOV B, M	Move the contents of the memory location pointed to by the updated HL into the B register
LOOP: CMP B	This label marks the beginning of a loop. Compare the contents of the accumulator (A) with the contents of the B register
JZ STORE	Jump to the label STORE if the zero flag is set, indicating that A and B are equal
JC EXG	Jump to the label EXG if the carry flag is set, indicating that A is less than B
SUB B	Subtract the contents of the B register from the accumulator (A)
JMP LOOP	Jump back to the LOOP label to repeat the loop
EXG: MOV C,B	This label is used for exchange. It swaps the contents of A and B. Move the contents of the B register into the C register
MOV B, A	Move the contents of the A register into the B register
MOV A, C	Move the contents of the C register into the A register
JMP LOOP	Jump back to the LOOP label to repeat the loop
STORE: STA 8500	This label is reached when A and B are equal. It marks the end of the loop. Store the contents of the accumulator (GCD) at memory location 8500
HLT	Halt the program

Input

Address	Data
8050	
8051	

Output:

Address	Data
8051	

RESULT:

Thus the Assembly Language Program to find the GCD of the given number using 8085 Microprocessor in GNUSim is performed.

Exp.No:18

LCM of 2 NUMBERS

AIM:

To write an Assembly Language Program to find the LCM of two numbers using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Get the input numbers
2. Apply the logic to find determine the LCM of the given inputs
3. Stores the output in the register
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H, 8000	Load the address 8000H into the HL register pair
MOV C, M	Move the contents of the memory location pointed to by HL into the C register
MVI B, 00	Initialize B register to 00
INX H	Increment the HL register pair to point to the next memory location
MOV A, M	Move the contents of the memory location pointed to by the updated HL into the A register
CMA	Take the one's complement of A register
MOV E, A	Move the result into the E register
MVI D, 0FFH	Initialize D register to FF
MOV A, B	Move the contents of the B register into the A register
CMA	Take the one's complement of A register
MOV D, A	Move the result into the D register
INX D	Increment the D register
LXI H, 0000	Initialize HL register pair to 0000H
NEXT: DAD B	Add the contents of B register to HL
SHLD 8050	Store the result at memory location 8050H
LOOP: DAD D	Add the contents of D register to HL
JNC SKIP	Jump to SKIP if there is no carry
MOV A, H	Move the contents of H register into A register
ORA L	Perform logical OR with the contents of L register
JZ EXIT	Jump to EXIT if the result is zero
JMP LOOP	Jump back to LOOP to continue the loop
SKIP: LHLD 8050	Load the contents of memory location 8050H into HL
JMP NEXT	Jump back to NEXT to continue the loop
EXIT: LHLD 8050	Load the contents of memory location 8050H into HL
HLT	Halt the program

Input

Address	Data
8000	
8001	

Output:

Address	Data
8050	

RESULT:

Thus the Assembly Language Program to find the LCM of the given number using 8085 Microprocessor in GNUSim is performed.

Exp.No:19

SWAPPING OF TWO 8 BIT DATA

AIM:

To write an Assembly Language Program to swap two 8 bit data using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Get the two 8 bit data
2. Swap the data
3. Stores the outputs in the register
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LDA 8050	Load the content of memory location 8050 into the accumulator (A)
MOV B, A	Move the content of the accumulator (A) to register B
LDA 8051	Load the content of memory location 8051 into the accumulator (A)
STA 8050	Store the content of the accumulator (A) at memory location 8050
MOV A, B	Move the content of register B to the accumulator (A)
STA 8051	Store the content of the accumulator (A) at memory location 8051
HLT	Halt the program

Input

Address	Data
8050	
8051	

Output:

Address	Data
8050	
8051	

RESULT:

Thus the Assembly Language Program to swap the two 8 bit data using 8085 Microprocessor in GNUSim is performed.

Exp.No:20

DECIMAL TO HEXADECIMAL CONVERSION

AIM:

To write an Assembly Language Program to convert a Decimal number into Hexadecimal number using 8085 Microprocessor in GNUSim.

SOFTWARE USED:

GNUSim8085

ALGORITHM:

1. Get the input decimal number
2. Apply the logic to convert it into hexadecimal.
3. Stores the output in the register
4. End the program

PROGRAM:

MNEMONICS	EXPLANATION
LXI H, 2050	Load the HL register pair with the address 2050H
MOV A, M	Move the content of memory location addressed by HL into register A
ADD A	Add the value of A to itself (A + A)
MOV B, A	Move the result to register B
ADD A	Add the value of A to itself (A + A)
ADD A	Add the value of A to itself (A + A)
ADD B	Add the value of B to A (A + B)
INX H	Increment the HL register pair to point to the next memory location
ADD M	Add the content of the new memory location to A
INX H	Increment the HL register pair again
MOV M, A	Move the result (A) back to the memory location pointed to by HL
HLT	Halt the execution of the program

Input

Address	Data
2050	
2051	

Output:

Address	Data
2052	

RESULT:

Thus the Assembly Language Program to convert a Decimal number into Hexadecimal number using 8085 Microprocessor in GNUSim is performed.

Date :
24/01/2024.

HALF ADDER

EXP. NO : 25

Aim :

To design and implement the two bit Half adder logisim simulator.

Procedure :

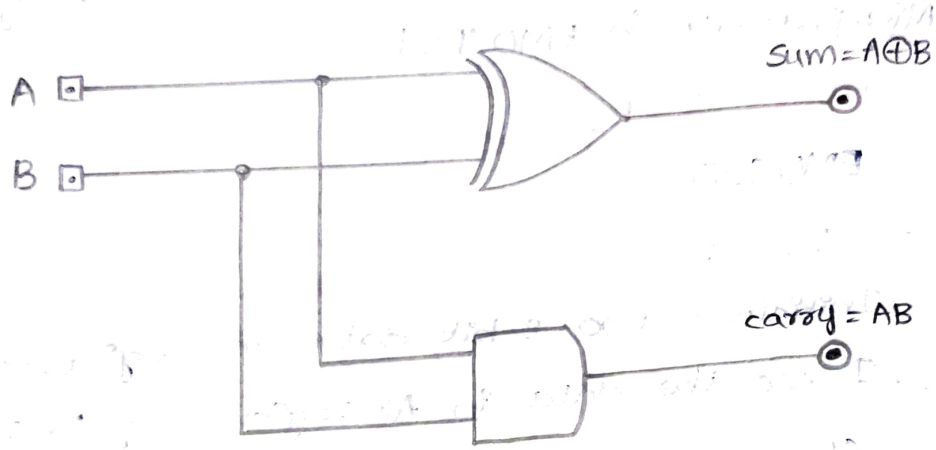
1. Pick and place the necessary canvas.
2. Insert 2 inputs into the canvas.
3. connect inputs into the XOR gate and AND gate.
4. Insert 2 outputs into the canvas.
5. Make the connections using the connecting wires.
6. Verify the truth Table.

Result :

The half adder has been deserved successfully using logisim simulator.

Barani 27/1/24

CIRCUIT DIAGRAM:



TRUTH TABLE:

A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

o/p required

2-BIT HALF ADDER USING NAND GATES

Date: 24/01/2024.

24

EXP. NO: 26.

Aim:

To design and implement 2-bit half adder using NAND gate by logisim simulator.

Procedure:

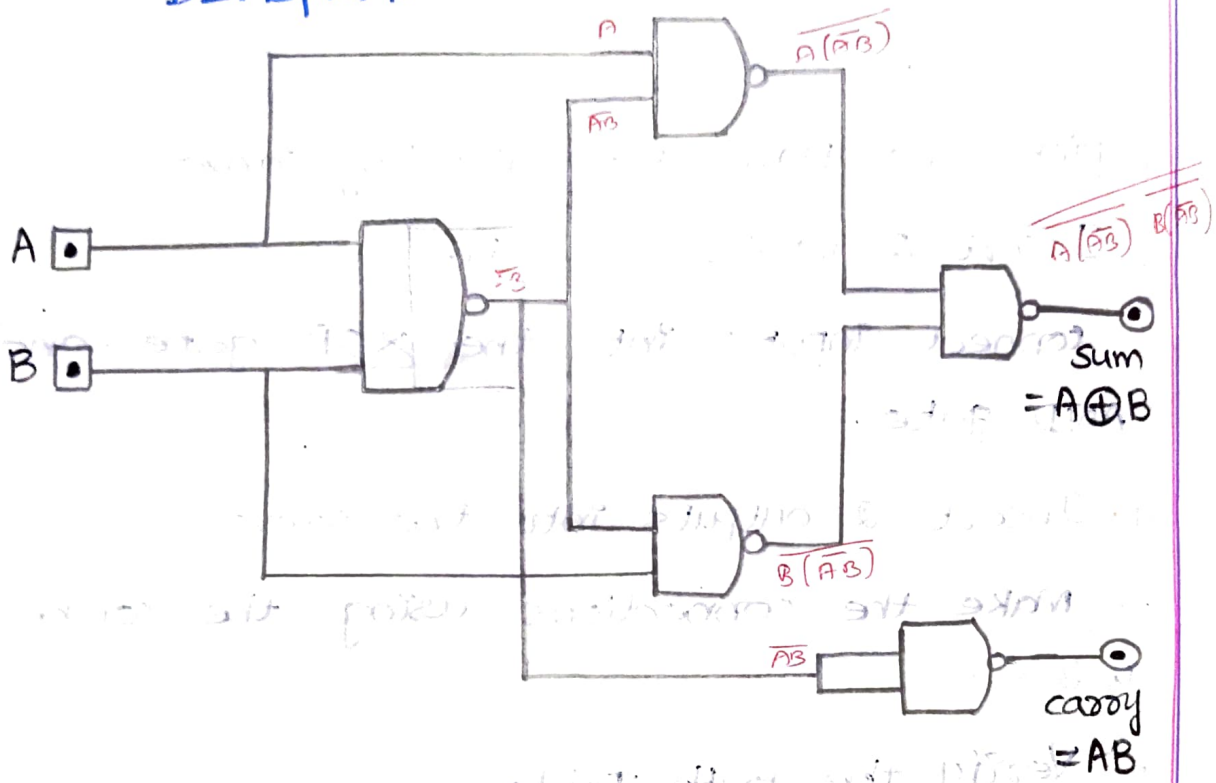
1. Insert two inputs A, B.
2. Inputs into the canvas.
3. label the inputs (A and B) by setting the attribute table.
4. Insert two outputs into the canvas.
5. Insert five NAND gates from gates tab logisim.
6. label the outputs.

Result:

Thus the half adder is designed using NAND gates and simulated.

Balaram 29/Thy

CIRCUIT DIAGRAM:



TRUTH TABLE:

A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2- BIT HALF ADDER USING NOR GATES.

Date: 24/01/2024.

EXP. NO: 27

Aim:

To design and implement 2-bit Half Adder using NOR gate by logisim simulator.

Procedure:

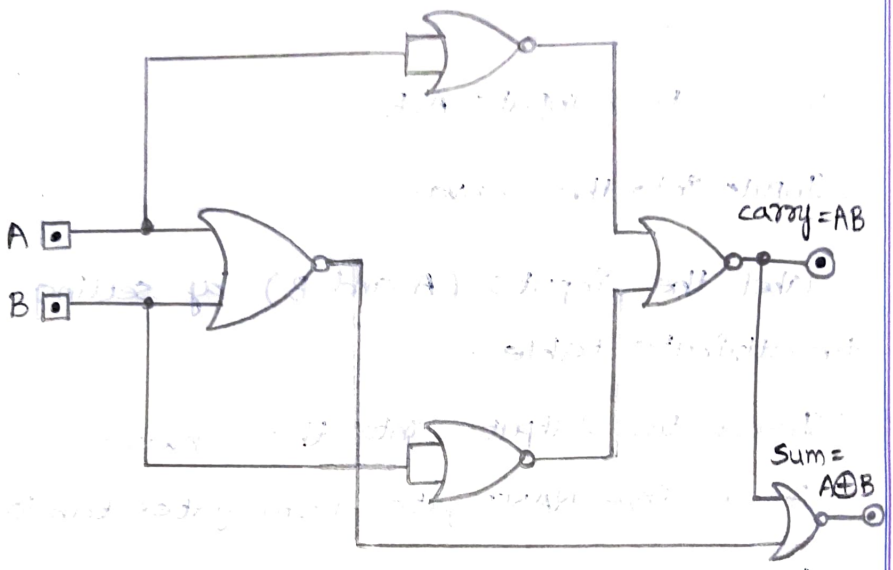
1. Insert two inputs A, B.
2. Inputs into the canvas
3. label the inputs (A & B) by setting the attribute table.
4. Insert two outputs into the canvas.
5. Insert five NOR gates from gates tab logisim.
6. Label the outputs.

Result:

Thus the half adder is designed using NOR gates and simulated.

Balamurugan

CIRCUIT DIAGRAM:



TRUTH TABLE:

A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

o/p varied

Date :
24/01/2024

FULL ADDER

EXP. NO : 28

Aim :

To design and implement the three bit full adder logisim simulator.

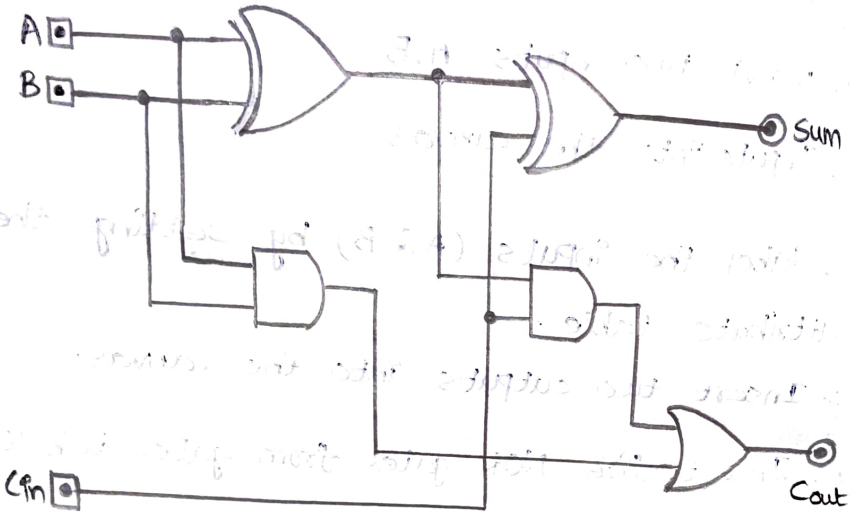
Procedure :

1. Pick and place the necessary canvas.
2. Insert 3 inputs into the canvas.
3. connect inputs into the XOR gate ; AND gate and OR gate.
4. Insert 2 outputs into the canvas.
5. Make the connections using the connecting wires.
6. Verify the truth Table.

Result :

The full adder has been deserved Successfully using logisim simulator.

CIRCUIT DIAGRAM:



TRUTH TABLE:

A	B	Cin	Sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

o/p verified

Date :
24/01/2024.

3-BIT FULL ADDER USING NAND GATES.

EXP. NO : 29.

Aim :

To design and implement 3-bit full adder using NAND gates by logisim simulator.

Procedure :

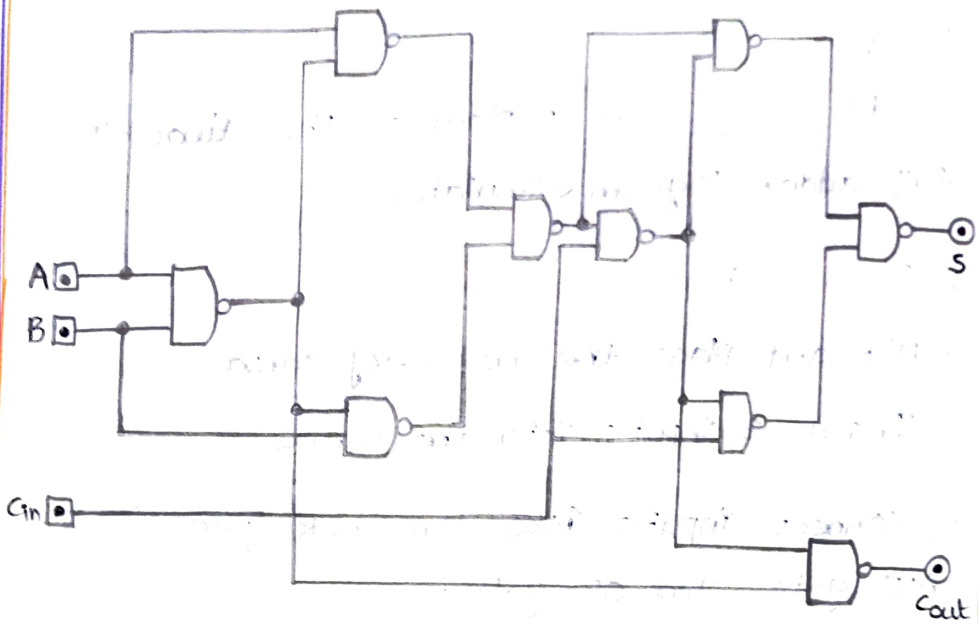
1. Insert three inputs A, B, Cin.
2. Inputs into the canvas.
3. Label the inputs (A, B & Cin) by setting the attribute table.
4. Insert two outputs into the canvas.
5. Insert nine NAND gates from gates tab logisim.
6. Label the outputs.

Result :

Thus the full adder is designed using NAND gates and simulated.

Balaji 27/1/24

CIRCUIT DIAGRAM:



TRUTH TABLE:

A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

8/1 verified

Date:
24/01/2024.

3-BIT FULL ADDER USING NOR GATES.

EXP. NO: 30

Aim:

To design and implement 3-bit full adder using NOR gate by logisim simulator.

Procedure:

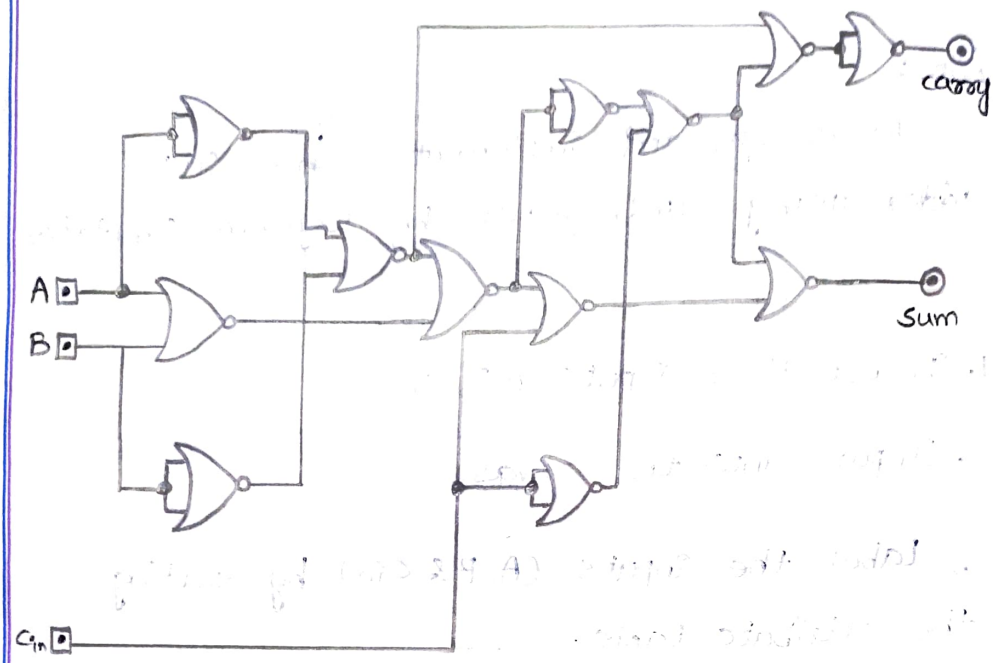
1. Insert three inputs A, B, Cin.
2. Inputs into the canvas
3. label the inputs (A, B & Cin) by setting the attribute table.
4. Insert two outputs into the canvas.
5. Insert twelve NOR gates from gates tab logisim.
6. label the outputs.

Result:

Thus the full adder is designed using NOR gates and simulated.

Bala 27/1/24

CIRCUIT DIAGRAM:



TRUTH TABLE:

A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

i/p verified

Date: 27/01/2024

4-BIT CARRY LOOK AHEAD ADDER.

EXP. NO: 31

Aim: To design and simulate the 4 bit carry look Ahead adder circuit using logisim.

Procedure:
1. Pick and place the necessary canvas.
2. Insert the inputs into the canvas.
3. Make the connections according to the circuit diagram.
4. obtain the outputs by giving respective inputs.

Theory: create two signals. (1) P (carry propagator). (2) G (carry generator).

* carry propagator (P) is propagated to the next level.
* carry generator (G) is used to generate the output carry regardless of input carry

$P_i = A_i \oplus B_i \rightarrow (1)$

$G_i = A_i \cdot B_i \rightarrow (2)$

$Sum = P_i \oplus C_i \rightarrow (3)$

$C_{i+1} = G_i + (P_i \cdot C_i) \rightarrow (4)$

* For designing 4 bit CLA we need $i = 0,1,2,3$.

Subtitute $i = 0,1,2,3$ in

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1$

$= G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

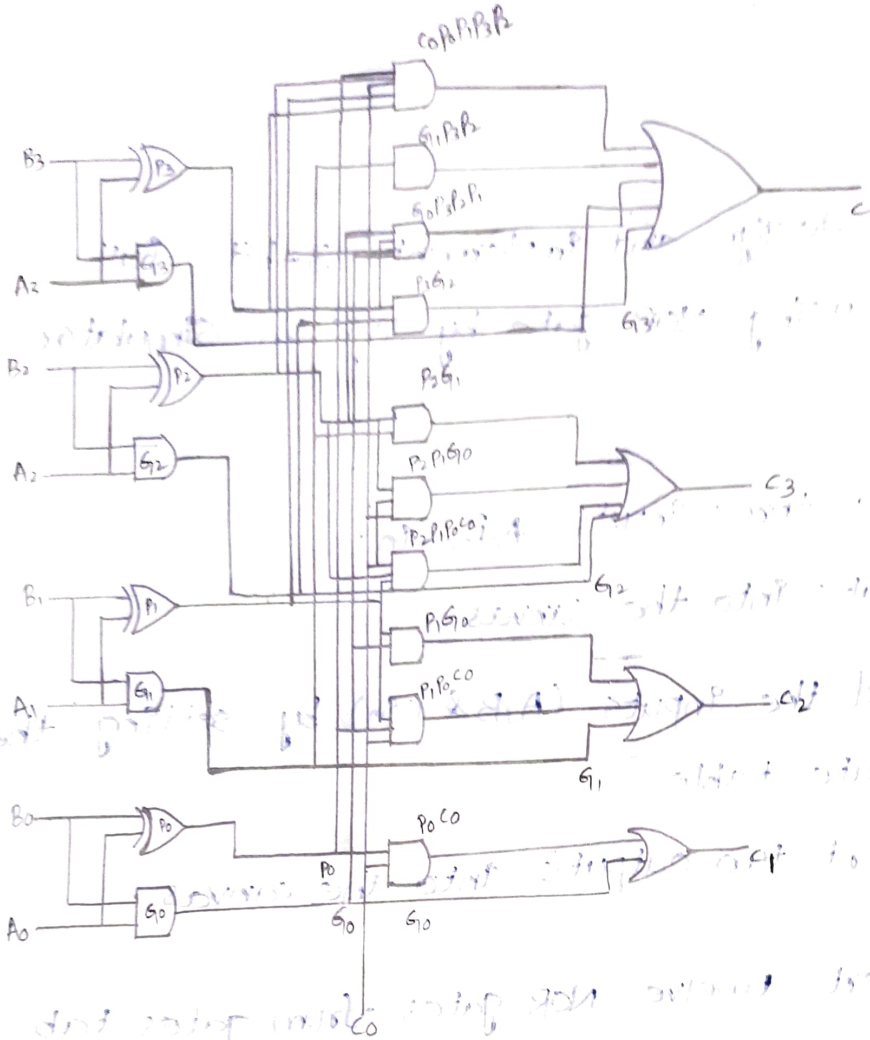
$C_4 = G_3 + P_3 C_3$

$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

Result: 30/1/24

The 4-bit carry look Ahead Adder circuit is successfully simulated by using logisim.

CIRCUIT DIAGRAM:



OUTPUT VERIFICATION:

Inputs								Output			
A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	C_4	C_3	C_2	C_1
1	1	1	1	1	1	1	1	1	1	1	1

O/P verified

Date:
24/01/2024

16 BIT MULTIPLICATION

EXP. NO: 23

Aim:

To write an Asembly language Program to Perform multiplication of two 16-bit data using 8086 Microprocessor in EMU Tool.

Software Used:

EMU 8086.

Algorithm:

1. Initialize two 8-bit data in registers.
2. Multiply the data in BX register with AX register.
3. Store the results in some other register.
4. End the program.

Program:

MNEMONICS	EXPLANATION
MOV AX, [1100H]	Move the 16-bit value at memory address 1100H into the AX register.
MOV BX, [1102H]	Move the 16-bit value at memory address 1102H into the BX register.
MUL BX	Multiply the contents of registers BX with AX and store the result in AX.
MOV [1200H], AX	Move the value in AX to the memory address 1200H.
MOV [1202H], DX	Move the value in DX to the memory address 1202H.
HLT	Halt the processor.

Result: 27/1/24

Thus the Assembly language program to Multiply two 16 bit numbers is performed using 8086 Microprocessor in EMU 8086 Tool.

INPUT:

Address	Data
1100	3
1102	2

OUTPUT:

Address	Data
1200	6
1202	0

o/p verified

Date :
24/01/2024

16 BIT DIVISION

80

EXP. NO: 24

Aim:

To write an Assembly language program to perform Division of two 16-bit data using 8086 Microprocessor in EMU Tool.

Software Used:

EMU 8086

Algorithm:

1. Initialize two 8-bit data in registers.
2. Divide the data in AX register by BX register.
3. Store the results in some other register.
4. End the program.

Program:

MNEMONICS	EXPLANATION
MOV AX, [1100H]	Move the 16-bit data value at memory address 1100H into AX Register
MOV BX, [1102H]	Move the 16-bit value at memory address 1102H into the BX register.
DIV BX	Divide the contents of registers AX by BX and store the result in AX.
MOV [1200H], AX	Move the value in AX to the memory address 1200H.
MOV [1202H], DX	Move the value in DX to the memory address 1202H.
HLT	Halt the processor.

Result:

Thus the Assembly language program to divide 16 bit numbers is performed using 8086 Microprocessor in EMU 8086 Tool.

Ra 24/1/24

INPUT:

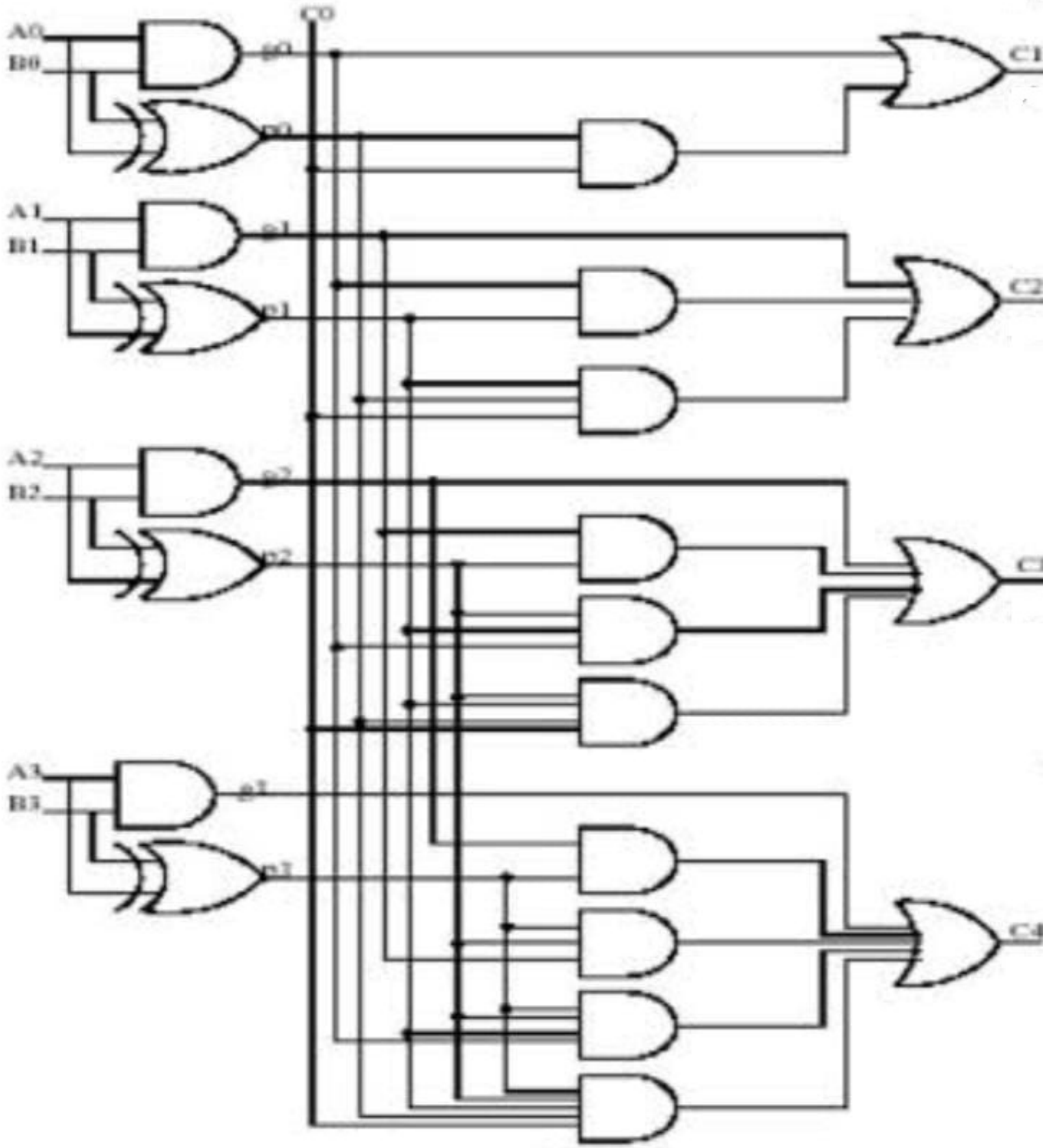
Address	Data
1100	8
1102	2

OUTPUT:

Address	Data
1200	4
1202	0

O/P verified

4-Bit Carry Look Ahead Adder



osé State

Date :
24/01/2024

16 BIT SUBTRACTION

EXP.NO : 22

Aim :

To write an Asembly language program to perform subtraction of two 16-bit data using 8086 Microprocessor in EMU Tool.

Software Used :

EMU 8086.

Algorithm :

1. Initialize two 8-bit data in registers.
2. Subtract the data in Bx register from Ax register.
3. Store the results in some other register.
4. End the program

Program :

MNEMONICS	EXPLANATION
MOV AX, [1100H]	Move the 16-bit value at memory address 1100H into the Ax register.
MOV BX, [1102H]	Move the 16-bit value at memory address 1102H into the Bx register.
SUB AX, BX	Subtract the contents of registers BX from Ax and store the result in AX.
MOV [1200H], AX	Move the value in Ax to the memory address 1200H.
HLT	Halt the processor.

Result :

24/1/24

Thus the Assembly language program to Subtract two 16 bit numbers is performed using 8086 Microprocessor in EMU 8086 Tool.

INPUT:

Address	Data.
1100	5
1102	3

OUTPUT:

Address	Data.
1200	2

o/p verified

Date :
24/01/2024

16 BIT ADDITION

EXP. NO : 21

Aim:

To write an Assembly language program to Perform addition of two 16-bit data using 8086 Microprocessor in EMU Tool.

Software Used:

EMU 8086

Algorithm:

1. Initialize two 8 bit-data in registers.
2. Add the data in registers.
3. Store the results in some other register.
4. End the program

Program:

MNEMONICS	EXPLANATION
MOV Ax, [1100H]	Move the 16-bit value at memory address 1100H into the AX register.
MOV Bx, [1102H]	Move the 16-bit value at memory address 1102H into the BX register.
ADD Ax, BX	Add the contents of registers AX and BX and store result in AX.
MOV [1200H], AX	Move the value in Ax to the memory address 1200H.
HLT	Halt the Processor.

Result:

Thus the Assembly language program to add two 16 bit numbers is performed using 8086 Microprocessor in EMU 8086 Tool.

Rajalakshmi 7/1/24

INPUT:

Address	Data
1100	5
1102	4

OUTPUT:

Address	Data
1200	9

O/P verified