**Practical 1**

**AIM: Implement a program for Binary to Decimal Conversion and Decimal to**

**Binary Conversion.**

**PROGRAM:**

#include <stdio.h>

#include <math.h>

int binaryToDecimal(long long binaryNumber)

{

int decimalNumber = 0, i = 0, remainder;

while (binaryNumber != 0)

{

remainder = binaryNumber % 10;

binaryNumber /= 10;

decimalNumber += remainder \* pow(2, i);

++i;

}

return decimalNumber;

}

long long decimalToBinary(int decimalNumber)

{

long long binaryNumber = 0;

int remainder, i = 1;

while (decimalNumber != 0)

{

remainder = decimalNumber % 2;

decimalNumber /= 2;

binaryNumber += remainder \* i;

i \*= 10;

}

return binaryNumber;

}

int main()

{

int choice;

printf("Choose conversion:\n");

printf("1. Binary to Decimal\n");

printf("2. Decimal to Binary\n");

scanf("%d", &choice);

if (choice == 1)

{

long long binaryNumber;

printf("Enter a binary number: ");

scanf("%lld", &binaryNumber);

printf("Decimal equivalent: %d\n", binaryToDecimal(binaryNumber));

}

else if (choice == 2)

{

int decimalNumber;

printf("Enter a decimal number: ");

scanf("%d", &decimalNumber);

printf("Binary equivalent: %lld\n", decimalToBinary(decimalNumber));

}

else

{

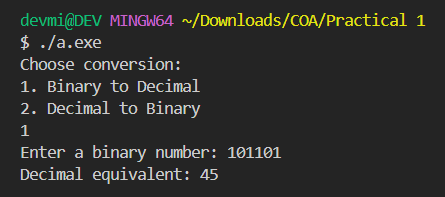
printf("Invalid choice\n");

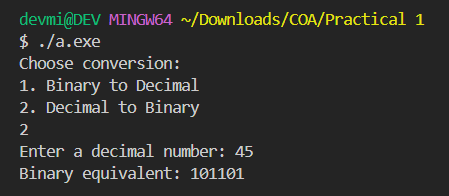
}

return 0;

}

**OUTPUT:**





**Practical 2**

**AIM: Analyze basic architecture of 8085 and write the working of 8085**

**GNUsim8085.**

**PROGRAM:**

import java.util.Arrays;

import java.util.Scanner;

public class Second {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] arr = new int[10];

System.out.println("Enter 10 Digits");

for(int i = 0; i < 10; i++){

while(true){

try {

arr[i] = scanner.nextInt();

break;

} catch (Exception err) {

System.out.println("Invalid Input " + err.getMessage());

scanner.nextLine();

}

}

}

Arrays.sort(arr);

for(int i = 9; i >= 0; i--){

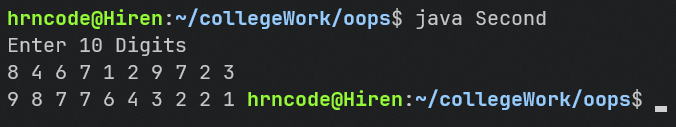
System.out.println(arr[i]);

}

}

}

**OUTPUT:**



**Practical 3**

**AIM: Write an assembly language code in GNUsim8085 to implement data transfer instruction. (HINT : MOV, MVI, LXI, LDAX, LDA, STA, STAX)**

**PROGRAM:**

jmp start

start: NOP

LXI H, 2000H ; Load immediate value 2000H into register pair HL

MVI A, 45H ; Move immediate value 45H into accumulator

STA 2002H ; Store the value in accumulator (45H) into memory address 2002H

MOV B, A ; Move the value in accumulator (45H) to register B

MOV A, M ; Move the contents of memory address pointed by HL to register A

MOV M, A ; Move the contents of register A to memory address pointed by HL

MOV A, B ; Move the contents of register B to accumulator (Corrected)

LXI D, 2005H ; Load immediate value 2005H into register pair DE

LDAX D ; Load the data from memory address pointed to by DE (2005H) into accumulator

STAX B ; Store the value in accumulator to memory address pointed to by BC

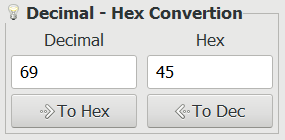
MVI A, 3AH ; Move immediate value 3AH into accumulator

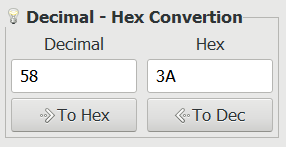
STAX D ; Store the value in accumulator to memory address pointed to by DE

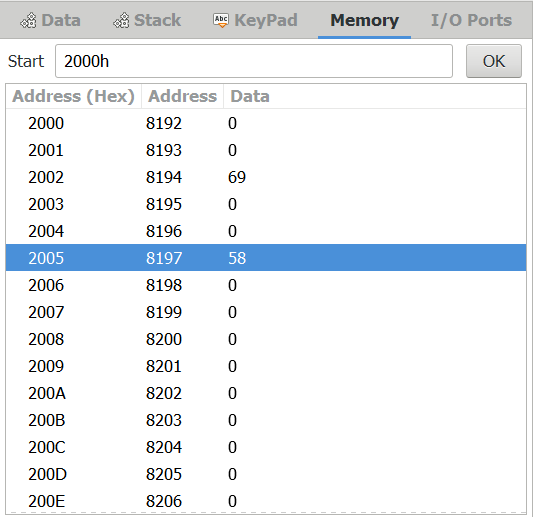
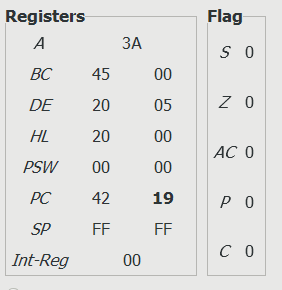
LDA 2005H ; Load the data from memory address 2005H into accumulator

HLT

**OUTPUT:**





**Practical 4**

**AIM: Write an assembly language code in GNUsim8085 to perform various**

**arithmetic operations.**

**PROGRAM:**

**a) Addition of two 8-bit numbers and also maintain carry registers**

start: NOP

MVI A, 1FH ; Load maximum value (255) into accumulator

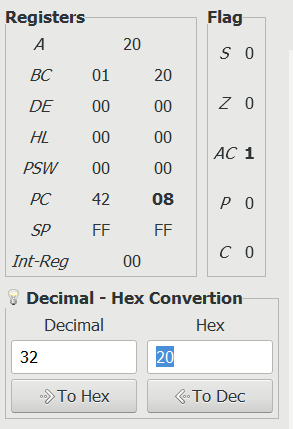
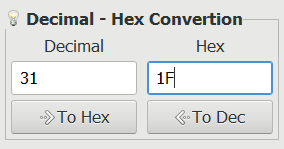
MVI B, 01H ; Load 1 into register B (for simplicity, adding 1)

ADD B ; Add the content of register B to accumulator

MOV C, A ; Move the result to register C

HLT

**OUTPUT:**

**b) Addition of two 16-bit numbers (with and without carry)**

**With Carry:**

**START: NOP**

LXI B, 0FFFH ; Load first number (0x3412) into BC pair

LXI D, 00FFH ; Load second number (0x7856) into DE pair

MOV A, C ; Move least significant byte to A

ADD E ; Add least significant byte to A

MOV C, A ; Store result back into C

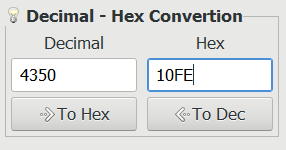
MOV A, B ; Move most significant byte of first number to A

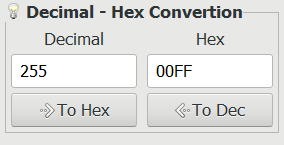
ADC D ; Add with carry (from previous addition) to A

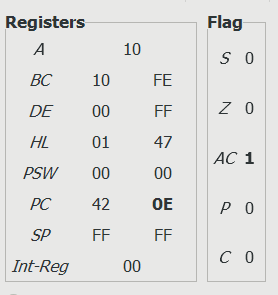
MOV B, A ; Store result back into B

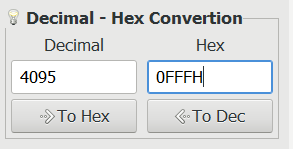
HLT

**OUTPUT:**









**Without Carry:**

START: NOP ; No operation, simply a placeholder label

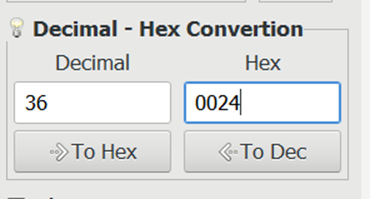
LXI H, 0123H ; Load immediate value 0123H into the HL register pair

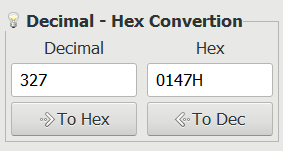
LXI D, 0024H ; Load immediate value 0024H into the DE register pair

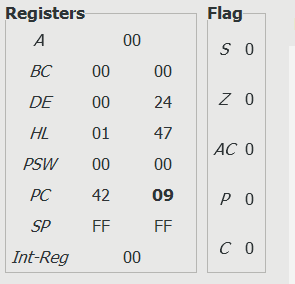
DAD D ; Add the value stored in DE to the value stored in HL,

HLT ; Halt execution

**OUTPUT:**







**c) Subtraction of two 8-bit numbers**

jmp start

start: nop

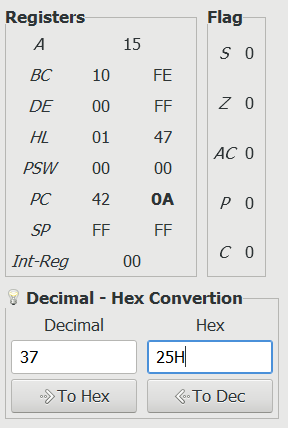
MVI A, 25H ; Load the first number

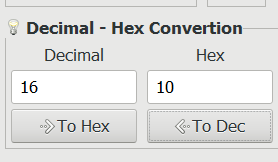
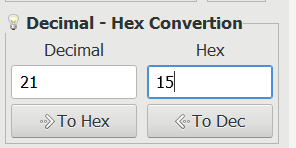
MVI B, 10H ; Load the second number

SUB B

Hlt

**OUTPUT:**





**d) Subtraction of two 16-bit numbers (with and without borrow)**

**With Borrow:**

START: NOP

LXI H, 7111H

LXI D, 5622H

MOV A, L

SUB E

MOV L, A

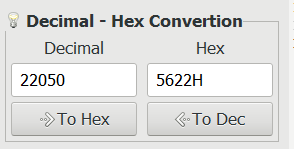
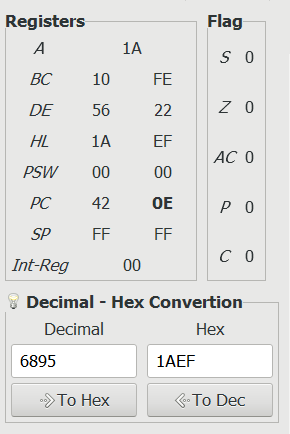
MOV A, H

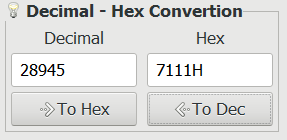
SBB D ; SUB WITH BORROW

MOV H, A

HLT

**OUTPUT:**





**Without Borrow:**

START: NOP ; Start of the program

LXI H, 2222H ; Load the first 16-bit number

LXI D, 1111H ; Load the second 16-bit number

MOV A, L

SUB E ; Subtract the least significant byte of the second number from A

MOV L, A

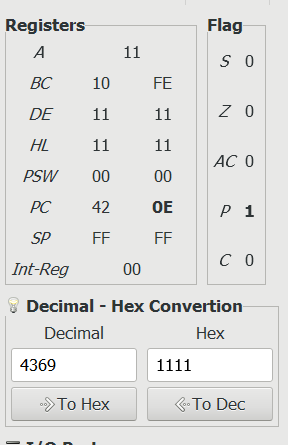
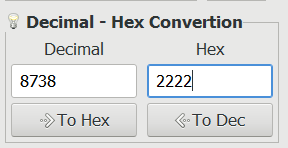
MOV A, H

SUB D ; Subtract the most significant byte of the second number along with borrow

MOV H, A

HLT

**OUTPUT:**

**e) Multiplication of 8-bit numbers**

jmp start

start: nop

XRA A

MVI B, 5H

MVI C, 3H

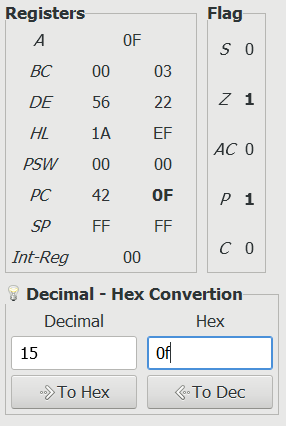
MUL: ADD C

DCR B

JNZ MUL

Hlt

**OUTPUT:**

****

**f) Division of 16 bit by 8-bit number**

**OUTPUT:**