VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Microprocessor and Interfacing

CSE2006

Lab Assignment 2

Slot: L3+L4

Name: Kulvir Singh

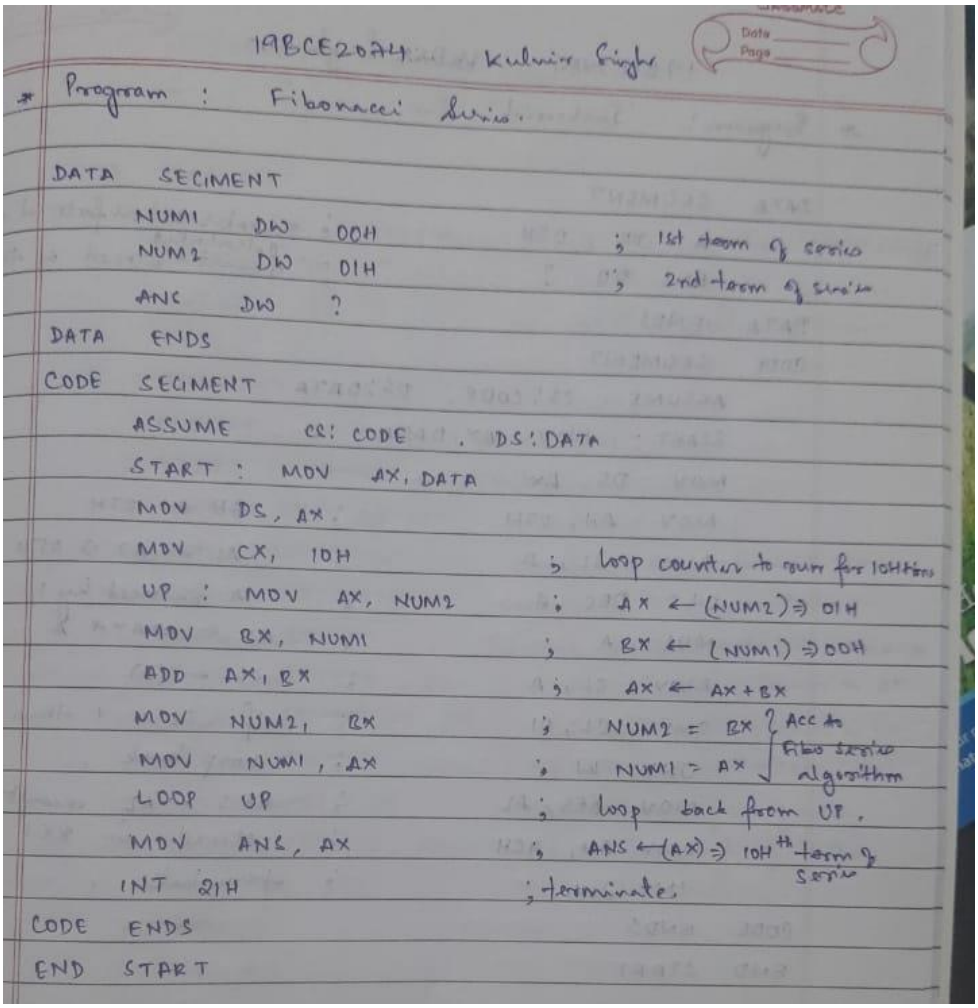
Register Number: 19BCE2074

1)Fibonacci Series

Aim :
Write a program in 8086 Assembly Language to get the Fibonacci Series term

Requirements :
8086 EMU - An emulator to run the 8086 Assembly Language Code
Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

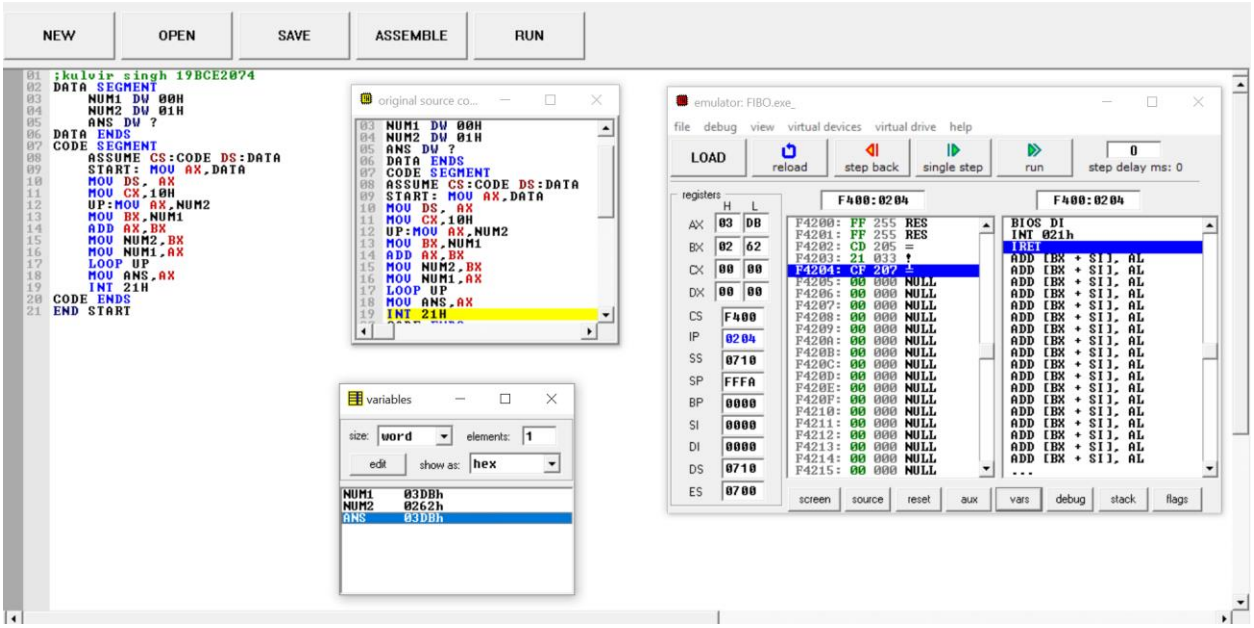


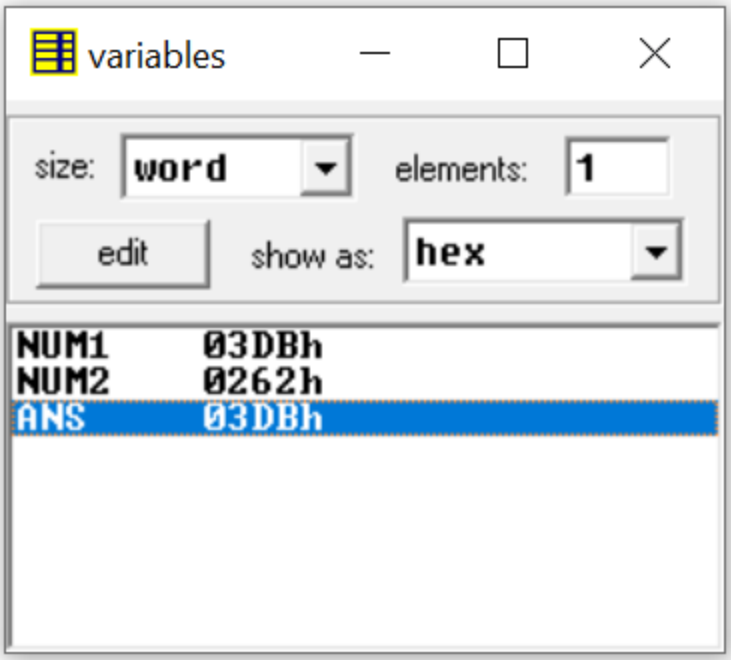
19BCE2074 Kulvir Singh

* Program : Fibonacci Series.

```
DATA    SEGMENT
        NUM1    DW    00H       ; 1st term of series
        NUM2    DW    01H       ; 2nd term of series
        ANS     DW    ?
DATA    ENDS
CODE    SEGMENT
        ASSUME    CS: CODE , DS: DATA
        START :   MOV   AX, DATA
        MOV     DS, AX
        MOV     CX, 10H         ; loop counter to run for 10H times
        UP :    MOV   AX, NUM2  ; AX ← (NUM2) ⇒ 01H
        MOV     BX, NUM1        ; BX ← (NUM1) ⇒ 00H
        ADD     AX, BX          ; AX ← AX + BX
        MOV     NUM2, BX        ; NUM2 = BX  } Acc to
        MOV     NUM1, AX        ; NUM1 = AX  } Fibo series algorithm
        LOOP    UP              ; loop back from UP.
        MOV     ANS, AX         ; ANS ← (AX) ⇒ 10H th term of series
        INT     21H             ; terminate.
CODE    ENDS
END     START
```

Screenshots :





Inference :

The program can successfully get the Fibonacci series terms as seen in the output screenshots

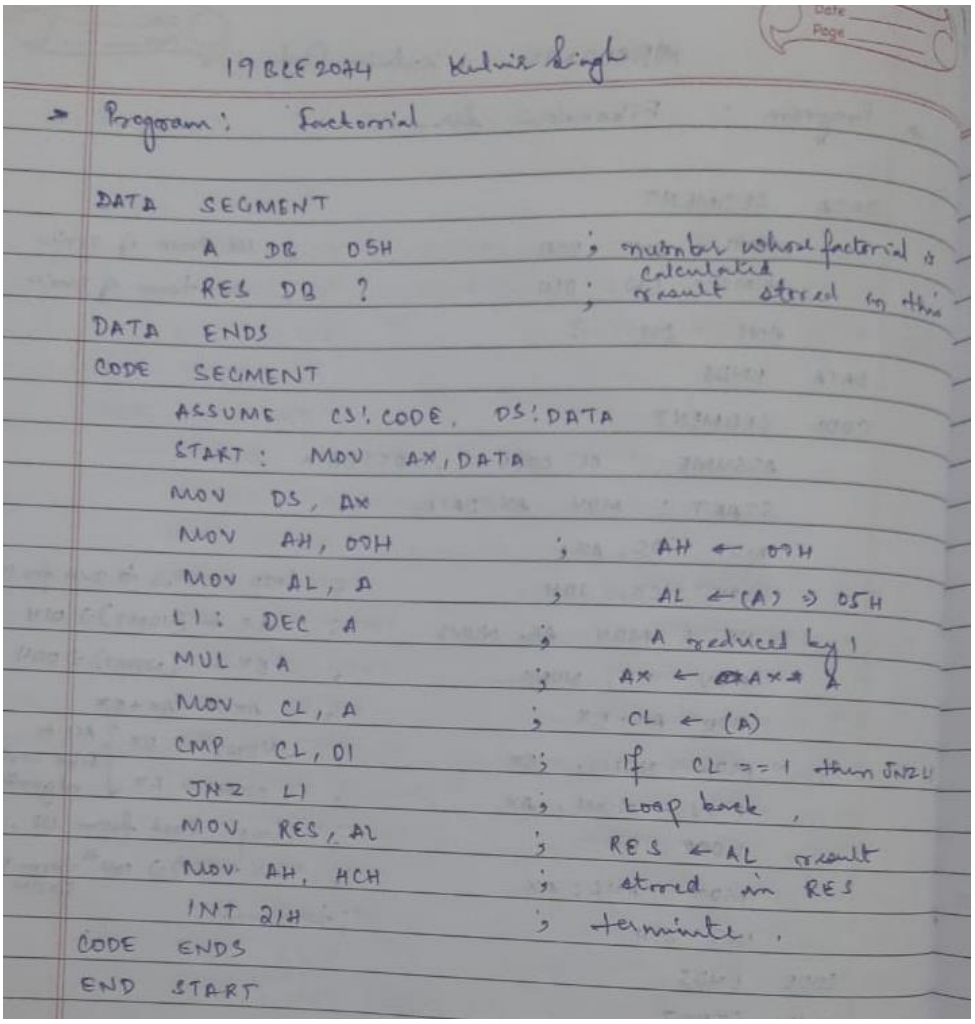
---

2)Factorial

Aim :
Write a program in 8086 Assembly Language to get the Factorial of a number

Requirements :
8086 EMU - An emulator to run the 8086 Assembly Language Code
Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

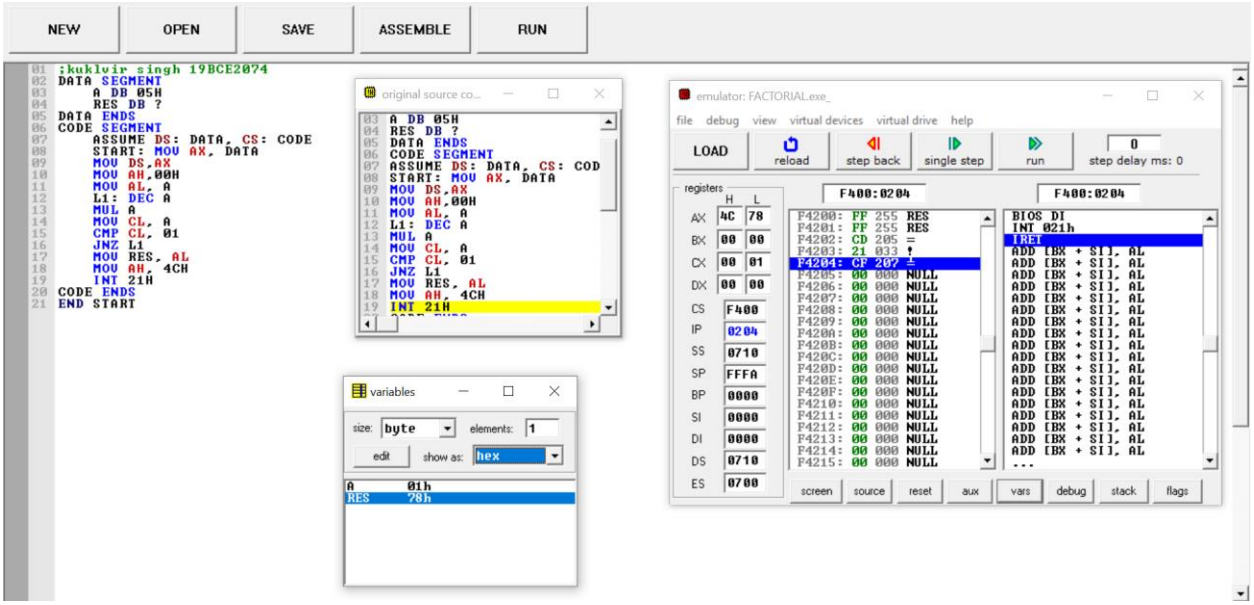


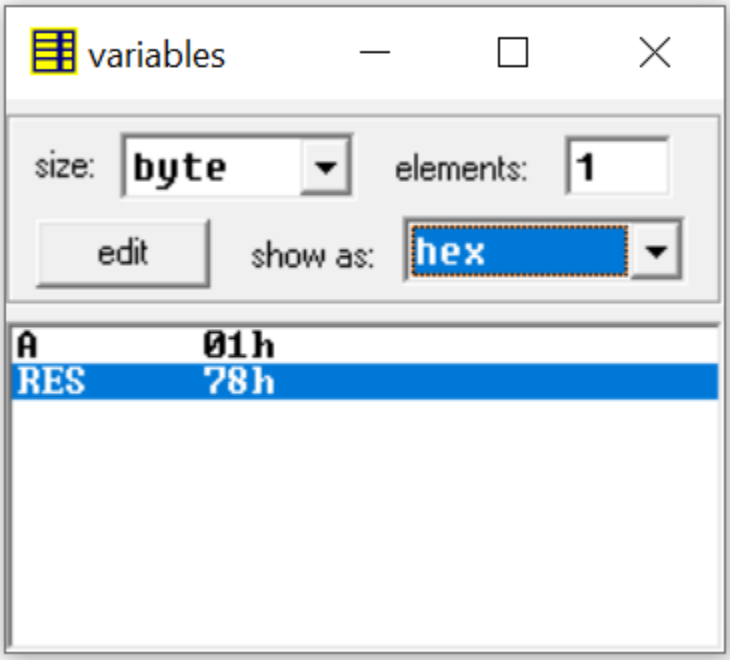
19 BCE 2074 Kulvir Singh

Program: Factorial

```
DATA    SEGMENT
        A   DB   05H          ; number whose factorial is
                               calculated
        RES DB   ?           ; result stored in this
DATA    ENDS
CODE    SEGMENT
        ASSUME   CS: CODE, DS: DATA
        START:   MOV  AX, DATA
        MOV  DS, AX
        MOV  AH, 00H         ;  AH ← 00H
        MOV  AL, A           ;  AL ← (A) ⇒ 05H
        L1:  DEC  A          ;  A reduced by 1
        MUL  A               ;  AX ← AX * A
        MOV  CL, A           ;  CL ← (A)
        CMP  CL, 01          ;  if  CL == 1 then JNZ L1
        JNZ  L1              ;  Loop back,
        MOV  RES, AL         ;  RES ← AL result
        MOV  AH, 4CH         ;  stored in RES
        INT  21H             ;  terminate.
CODE    ENDS
END     START
```

Screenshots :





Inference :

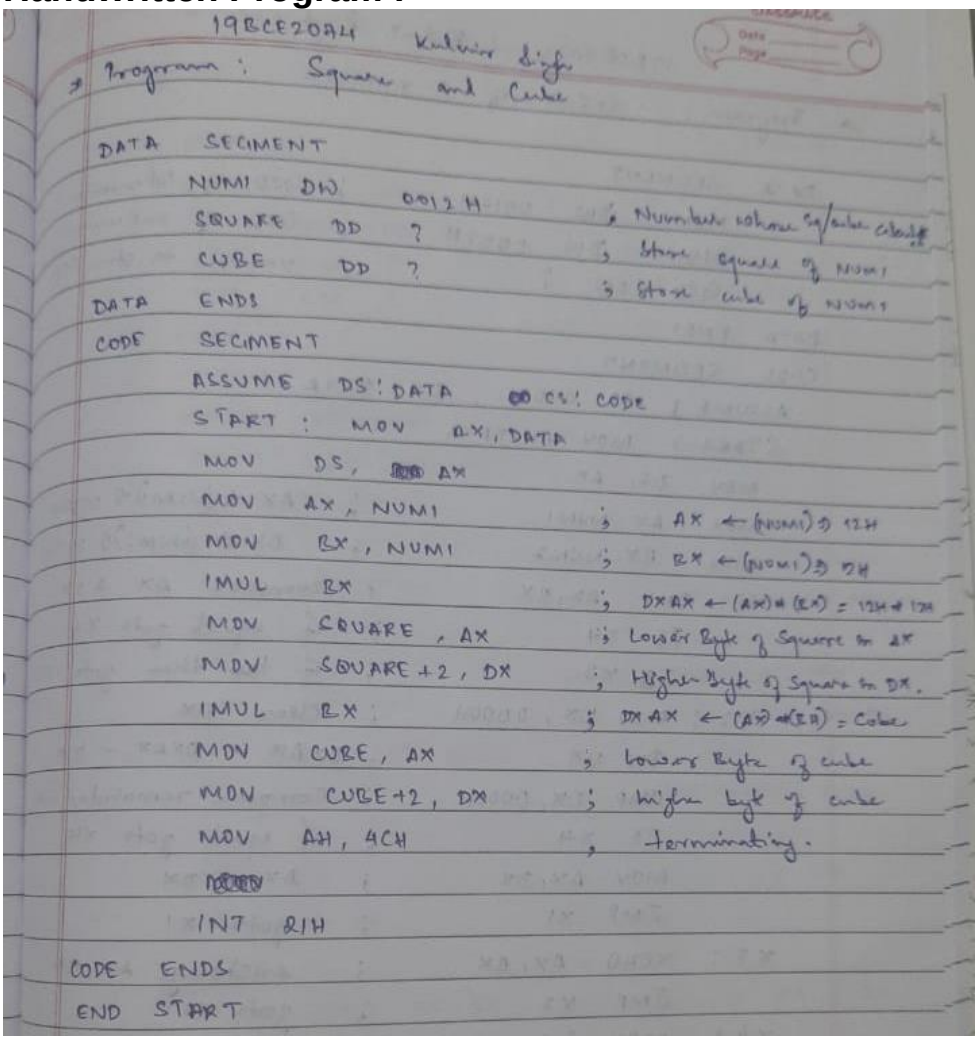
The program stores 120 as the factorial of 5 in res variable and hence is accurate.

---

3)Square and Cube

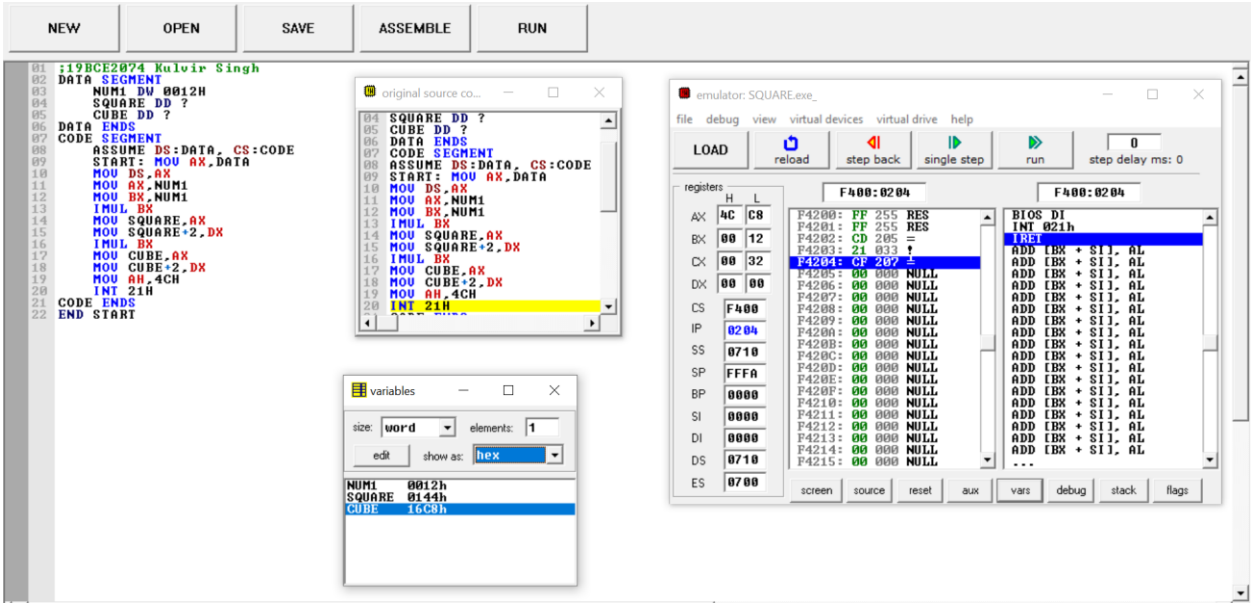
Aim :
Write a program in 8086 Assembly Language to get the Square and Cube of a number

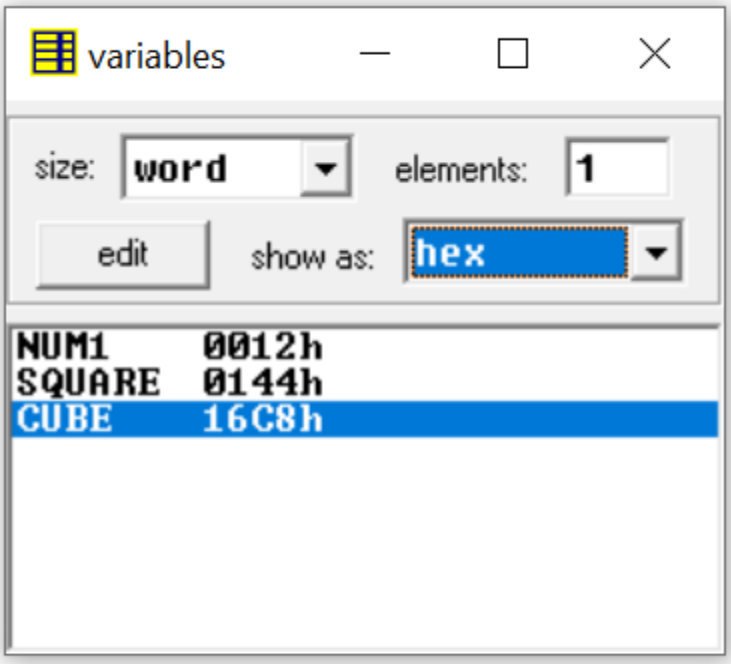
Requirements :
8086 EMU - An emulator to run the 8086 Assembly Language Code
Operating System - Any valid operating system that can execute the emulator

Handwritten Program :


```
19BCE2024  Kulvir Singh
Program : Square and Cube

DATA    SEGMENT
        NUM1    DW      0012 H          ; Number whose sq/cube calculate
        SQUARE  DD      ?               ; Store square of NUM1
        CUBE    DD      ?               ; Store cube of NUM1
DATA    ENDS
CODE    SEGMENT
        ASSUME  DS: DATA  CS: CODE
        START : MOV     AX, DATA
        MOV     DS,     AX
        MOV     AX, NUM1                ;  AX ← (NUM1) ⇒ 12H
        MOV     BX, NUM1                ;  BX ← (NUM1) ⇒ 12H
        IMUL    BX                      ;  DXAX ← (AX) * (BX) = 12H * 12H
        MOV     SQUARE, AX              ; Lower Byte of Square in AX
        MOV     SQUARE+2, DX            ; Higher Byte of Square in DX.
        IMUL    BX                      ;  DX AX ← (AX) * (BX) = Cube
        MOV     CUBE, AX                ; lower Byte of cube
        MOV     CUBE+2, DX              ; higher byt of cube
        MOV     AH, 4CH                 ,  terminating.
        INT     21H
CODE    ENDS
END     START
```

Screenshots :





Inference :

The program stores the accurate square and cube values of 0012H in the respective variables

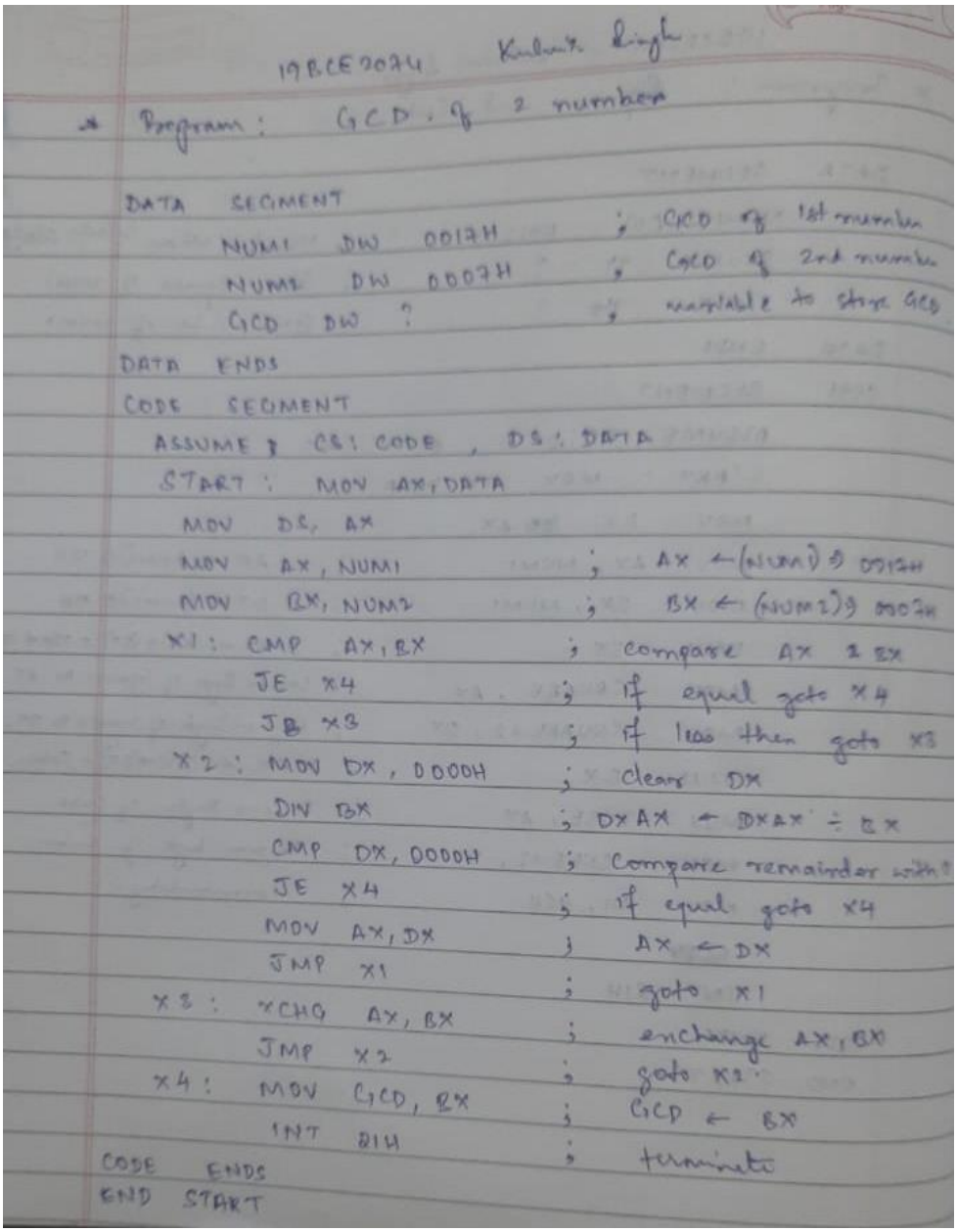
---

4)GCD – greatest common divisor

Aim :
Write a program in 8086 Assembly Language to get the GCD of a number

Requirements :
8086 EMU - An emulator to run the 8086 Assembly Language Code
Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

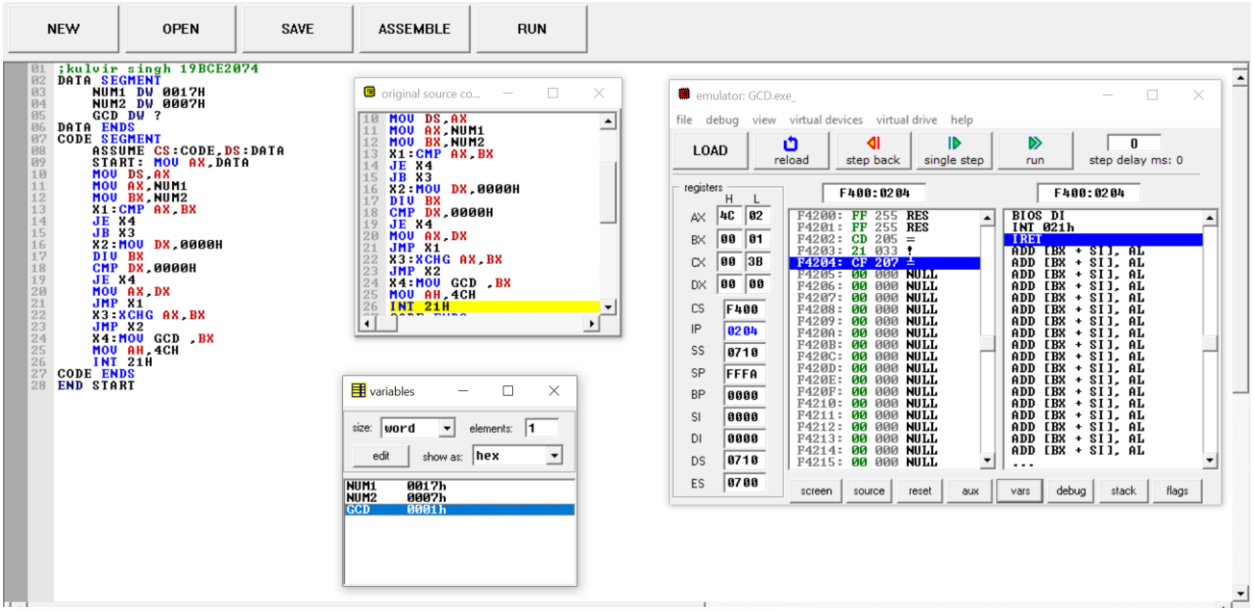


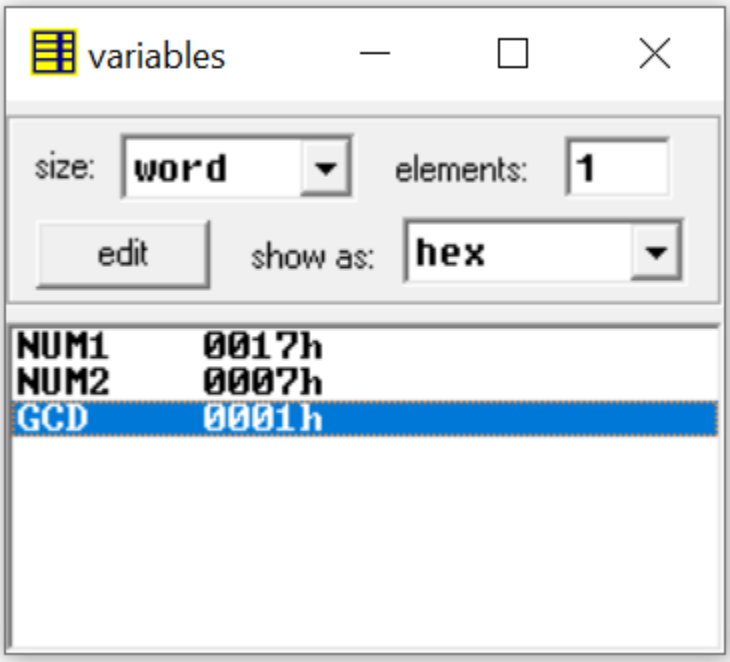
19BCE2074 Kulwant Singh

Program : GCD of 2 number

```
DATA SEGMENT
     NUM1 DW 0012H   ; GCD of 1st number
     NUM2 DW 0007H   ; GCD of 2nd number
     GCD DW ?        ; variable to store GCD
DATA ENDS
CODE SEGMENT
  ASSUME CS: CODE , DS: DATA
  START : MOV AX, DATA
    MOV DS, AX
    MOV AX, NUM1      ; AX ← (NUM1) 0012H
    MOV BX, NUM2      ; BX ← (NUM2) 0007H
  X1 : CMP AX, BX     ; compare AX & BX
       JE X4          ; if equal goto X4
       JB X3          ; if less then goto X3
  X2 : MOV DX, 0000H  ; clear DX
       DIV BX         ; DX AX ← DX AX ÷ BX
       CMP DX, 0000H  ; Compare remainder with 0
       JE X4          ; if equal goto X4
       MOV AX, DX     ; AX ← DX
       JMP X1         ; goto X1
  X3 : XCHG AX, BX    ; exchange AX, BX
       JMP X2         ; goto X2
  X4 : MOV GCD, BX    ; GCD ← BX
       INT 21H        ; terminate
CODE ENDS
END START
```

Screenshots :





Inference :

The program stores the accurate GCD 23 and 7 in GCD variable.

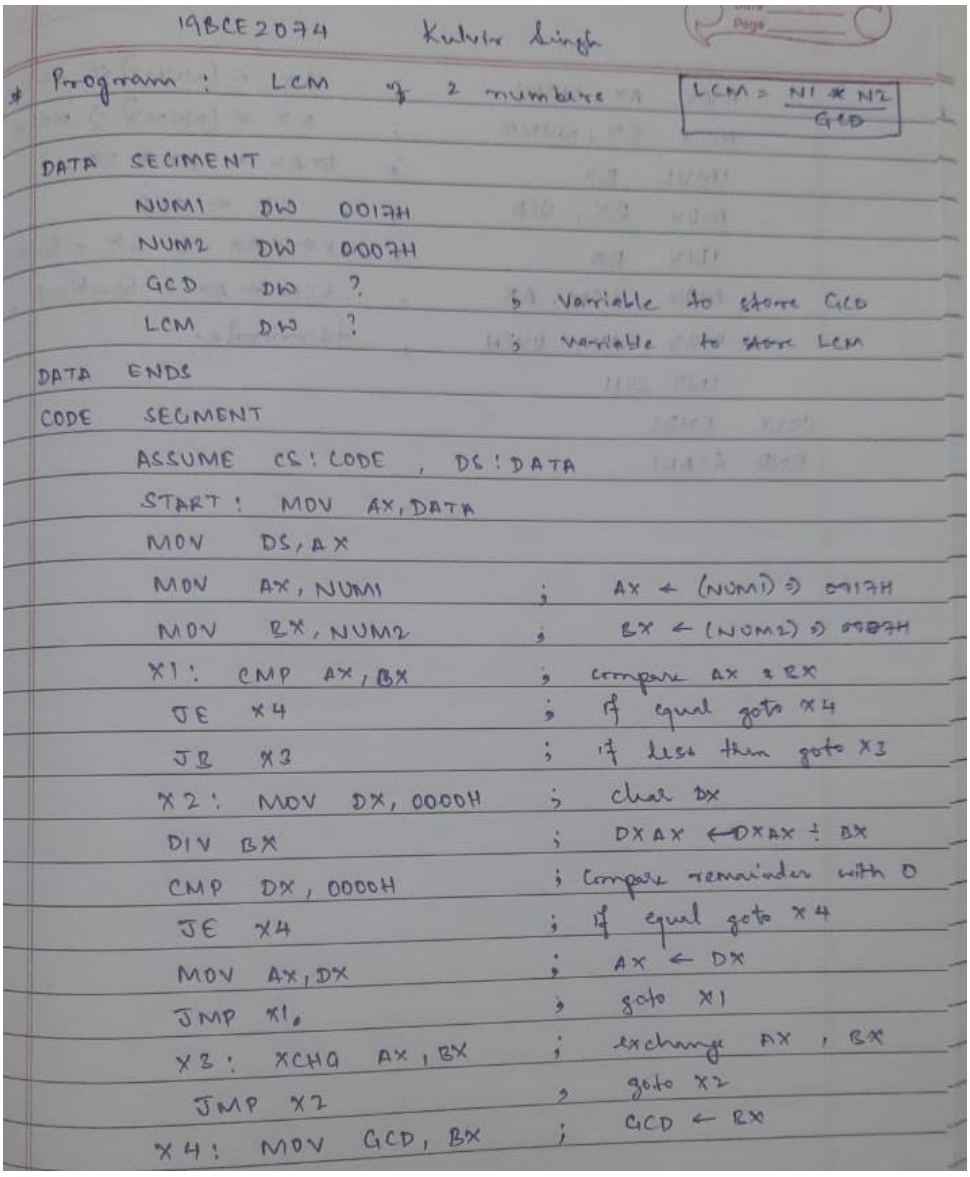
---

5)LCM – least common divisor

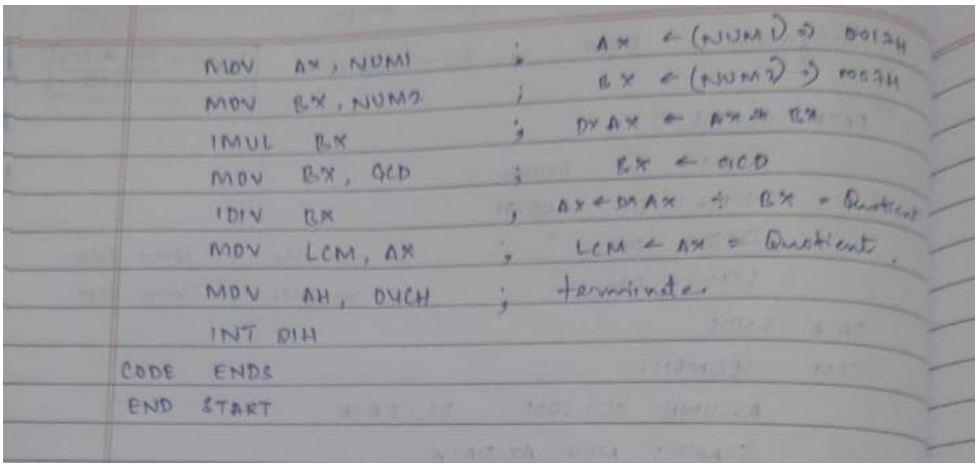
Aim :
Write a program in 8086 Assembly Language to get the LCM of a number

Requirements :
8086 EMU - An emulator to run the 8086 Assembly Language Code
Operating System - Any valid operating system that can execute the emulator

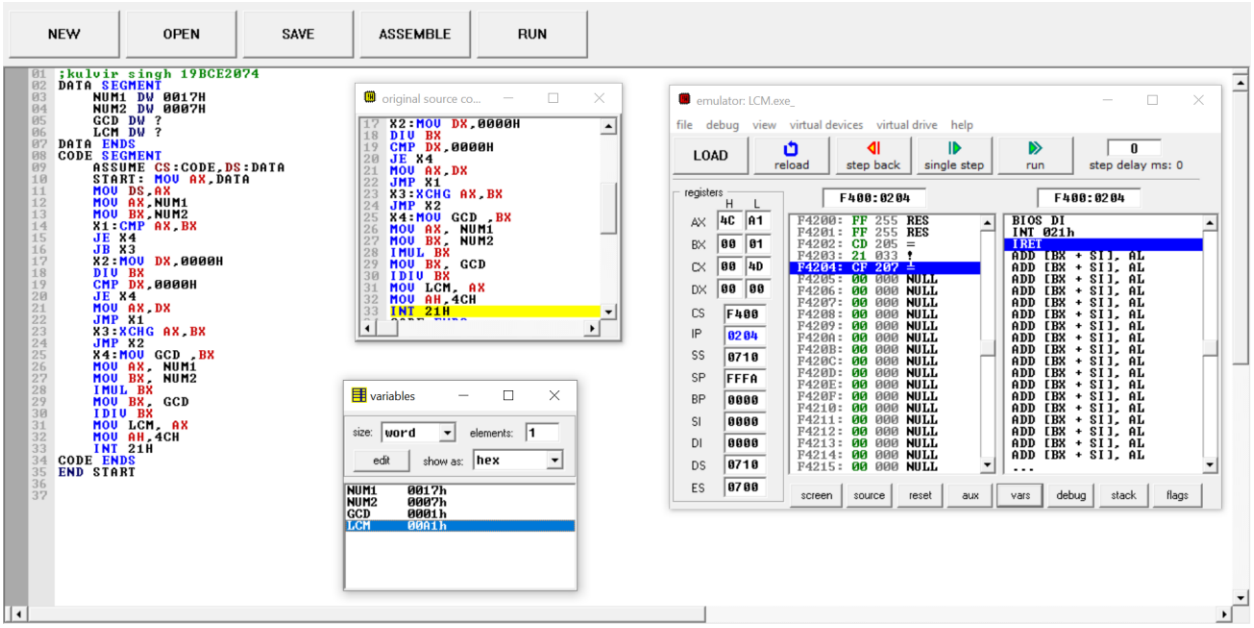
Handwritten Program :

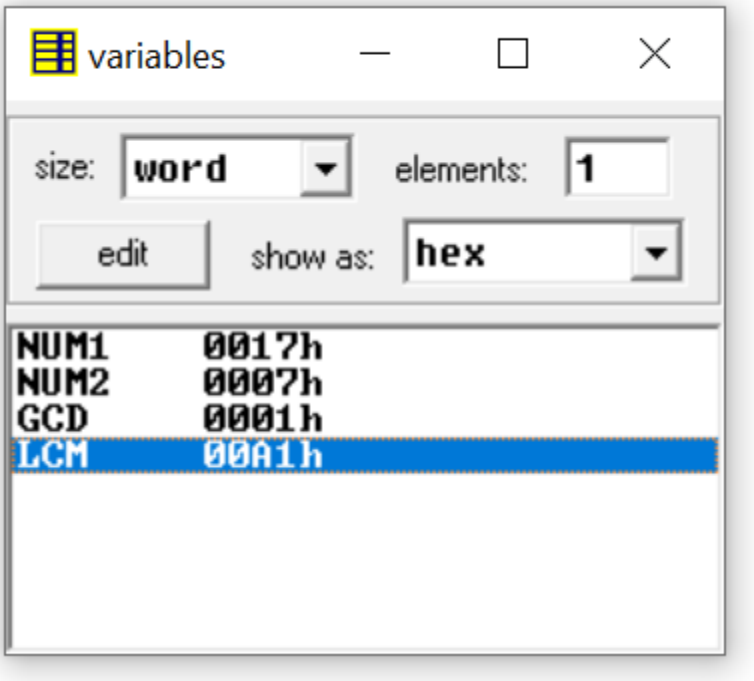


```
19BCE2074        Kulvir Singh

* Program : LCM of 2 numbers        LCM = N1 * N2 / GCD

DATA SEGMENT
    NUM1  DW  0017H
    NUM2  DW  0007H
    GCD   DW  ?        ; Variable to store GCD
    LCM   DW  ?        ; Variable to store LCM
DATA ENDS
CODE SEGMENT
    ASSUME CS: CODE , DS: DATA
    START: MOV AX, DATA
    MOV DS, AX
    MOV AX, NUM1       ; AX ← (NUM1) => 0017H
    MOV BX, NUM2       ; BX ← (NUM2) => 0007H
    X1: CMP AX, BX     ; compare AX & BX
    JE X4              ; if equal goto X4
    JB X3              ; if less than goto X3
    X2: MOV DX, 0000H  ; clear DX
    DIV BX             ; DXAX ← DXAX ÷ BX
    CMP DX, 0000H      ; Compare remainder with 0
    JE X4              ; if equal goto X4
    MOV AX, DX         ; AX ← DX
    JMP X1,            ; goto X1
    X3: XCHG AX, BX    ; exchange AX , BX
    JMP X2             ; goto X2
    X4: MOV GCD, BX    ; GCD ← BX
```



Screenshots :





Inference :
The program stores the accurate LCM 23 and 7 in GCD variable.
