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

Microprocessor and Interfacing
CSE2006

Lab Assignment 3

Slot: L3+L4

Name: Kulvir Singh

Register Number: 19BCE2074

1)ASCII to BCD Conversion

Aim :

Write a program in 8086 Assembly Language to convert the ASCII to BCD number

Requirements :

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

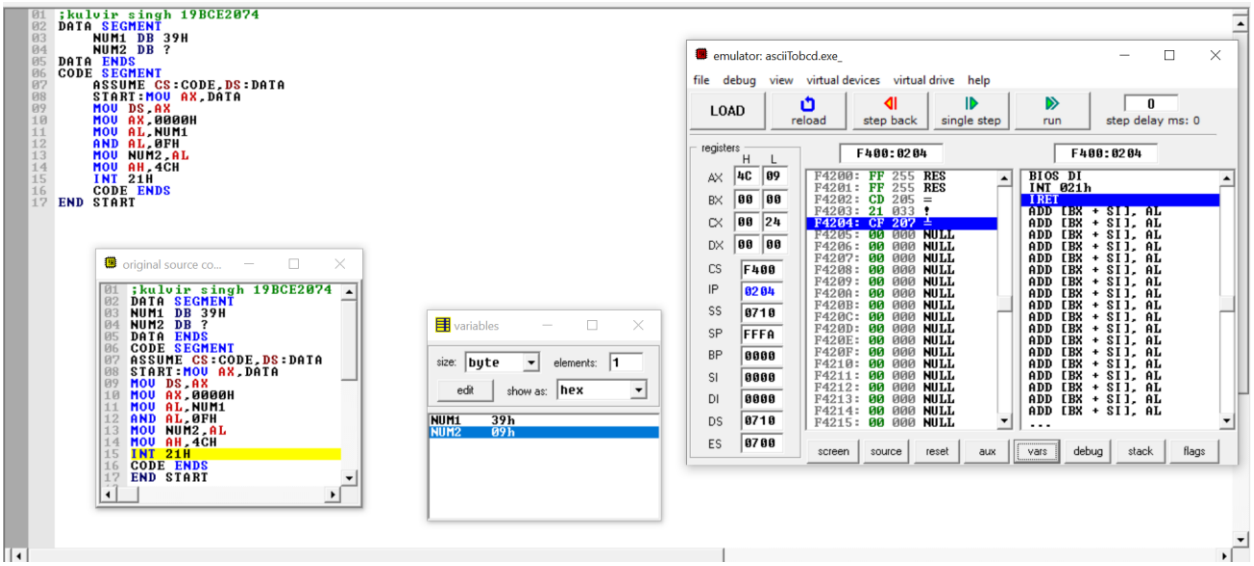
Handwritten Program :

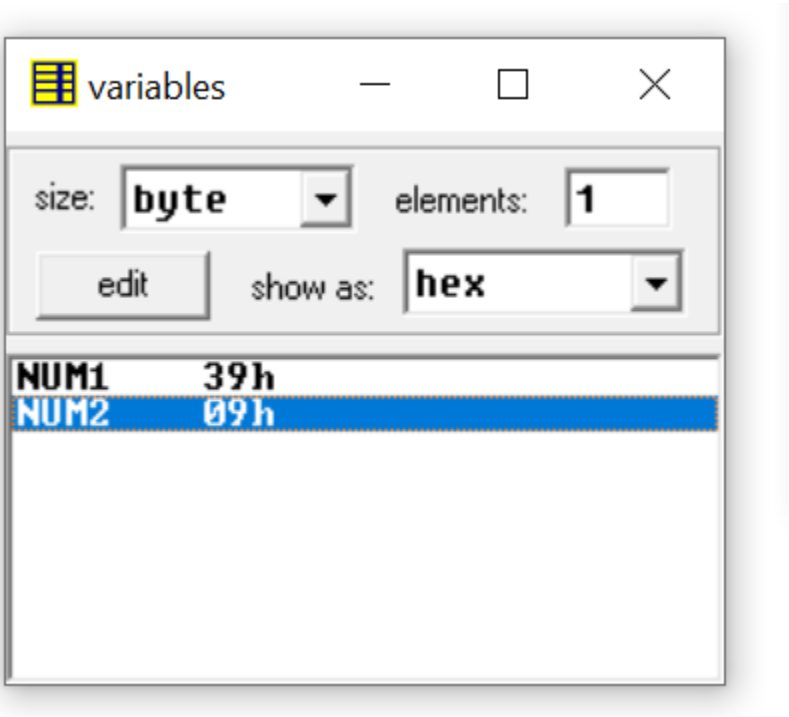
19BCE2074 Kulwin Singh

* Program : ASCII TO BCD Conversion.

```
DATA   SEGMENT
       NUM1   DB   39H      ; ASCII value to be converted
       NUM2   DB   ?
DATA   ENDS
CODE   SEGMENT
       ASSUME   CS:CODE , DS:DATA
       START :  MOV  AX, DATA
       MOV   DS, AX
       MOV   AX, 0000H      ; clear AX (AH and AL)
       MOV   AL, NUM1       ; AL ← NUM1 → 39H
       AND   AL, 0F H       ; logical operation get units value
       MOV   NUM2, AL       ; store result
       INT   21H            ; terminate
CODE   ENDS
END    START
```

Screenshots :





Inference :

The program can successfully change ascii value 39h to bcd value 9 which is accurate.

2)BCD to ASCII Conversion

Aim :
Write a program in 8086 Assembly Language to convert the BCD to ASCII number

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

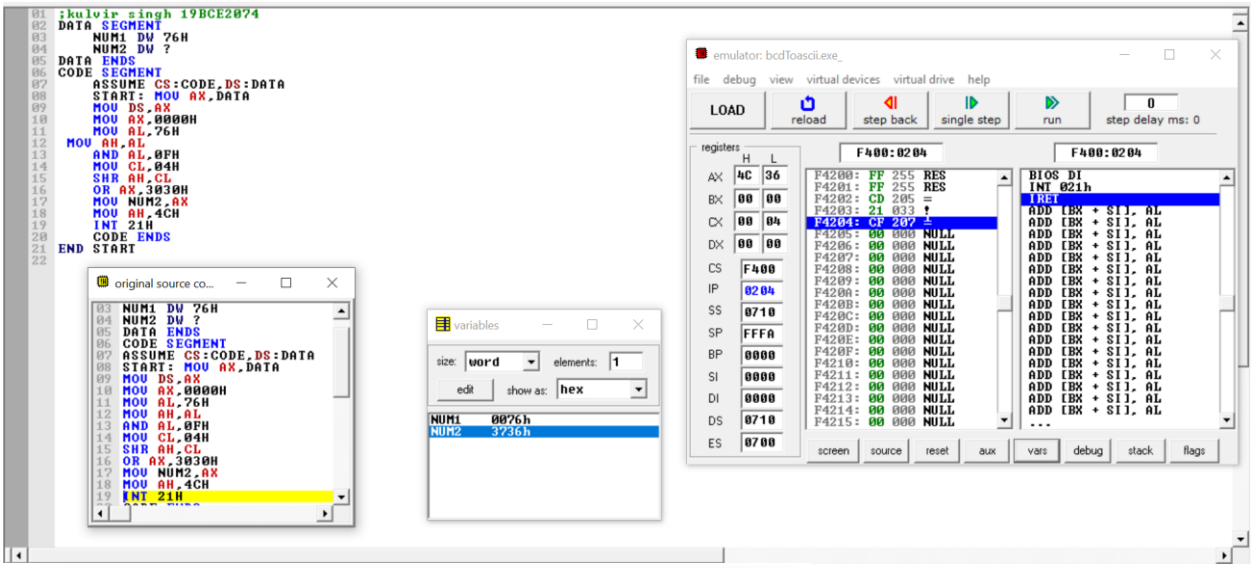
Handwritten Program :

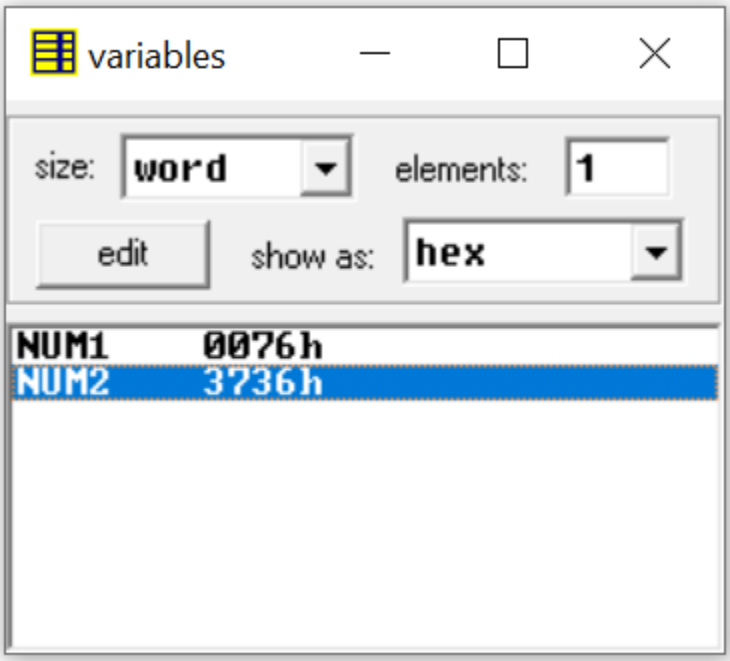
19BCE2074 Kulwin Singh

* Program : BCD To ASCII Conversion.

```
DATA SEGMENT
    NUM1  DW  76H      ; BCD value to be convert
    NUM2  DW  ?        ; store result .
DATA ENDS
CODE SEGMENT
    ASSUME  CS : CODE , DS: DATA
    START:  MOV AX, DATA
    MOV     DS, AX
    MOV     AX , 0000H     ; clear AX (AL and AH)
    MOV     AL, 76H        ; AL ← 76H (Num1)
    MOV     AH, AL         ; AH ← AL ⇒ 76H
    AND     AL, 0F H       ; logical operator AL ⇒ 6H.
    MOV     CL, 04H        ; CL ← 04H
    SHR     AH, CL         ; shift and rotate AH & CL
    OR      AX, 3030H      ; OR logical operator to get ASC
    MOV     NUM2, AX       ; store result
    INT 21H                ; terminate
CODE ENDS
END START
```

Screenshots :





Inference :

The program can successfully change 76 a bcd number to equivalent ascii 3736 which is accurate.

3)BCD to BINARY Conversion

Aim :
Write a program in 8086 Assembly Language to convert the BCD to Binary number

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

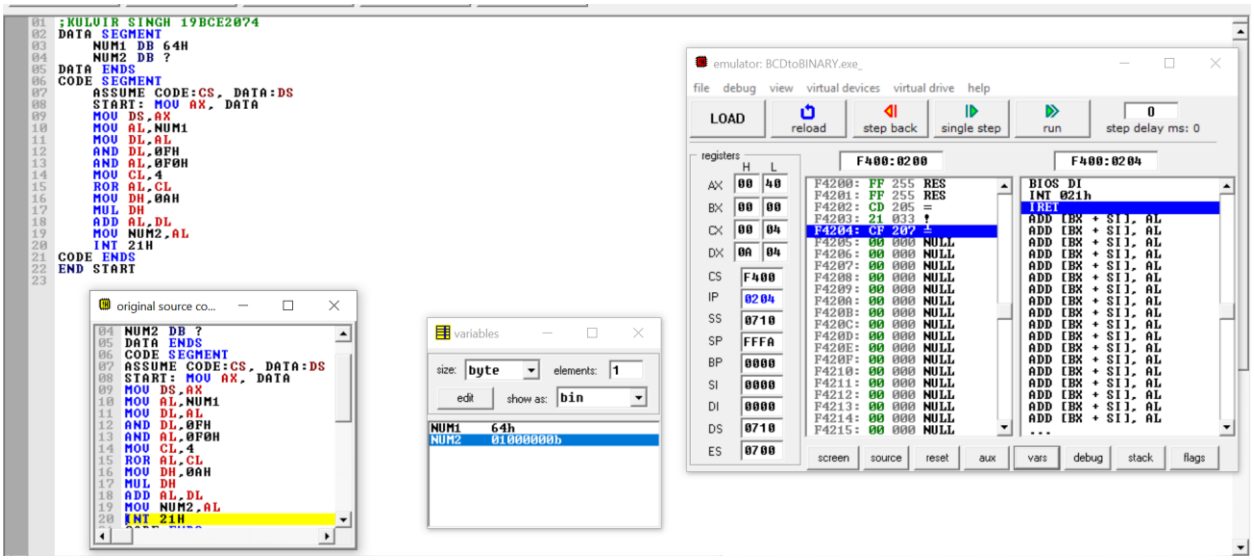
Handwritten Program :

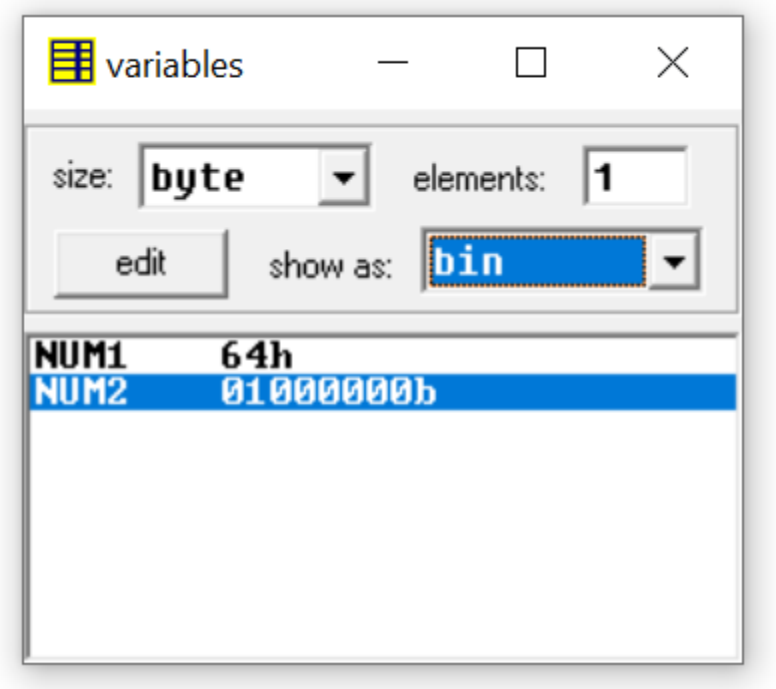
19BCE2074 Kulwin Singh

* Program : BCD TO BINARY Conversion.

```
DATA    SEGMENT
        NUM1    DB      64H     ; Number to be converted
        NUM2    DB      ?       ; Result to be stored.
DATA    ENDS
CODE    SEGMENT
        ASSUME CODE: CS, DATA: DS
        START:  MOV     AX, DATA
        MOV     DS, AX
        MOV     AL, NUM1        ; AL ← NUM1 ⇒ 64H
        MOV     DL, AL          ; DL ← AL
        AND     DL, 0FH         ; Mask upper nibble
        AND     AL, 0F0H        ; Mask lower nibble
        MOV     CL, 4           ; Rotate upper nibble
        ROR     DL, CL          ; to lower nibble
        MOV     DH, 0AH         ; Set multiplier as 0AH
        MUL     DH              ; Multiply tens digits by 0AH
        ADD     AL, DL          ; Get sum of unit & product
        MOV     NUM2, AL        ; Mov answer to NUM2
        INT 21H                 ; terminate.
CODE    ENDS
END     START
```

Screenshots :





Inference :

The program can successfully change 64 a bcd number to equivalent binary 01000000b which is accurate.

4)BINARY to BCD Conversion

Aim :
Write a program in 8086 Assembly Language to convert the BCD to Binary number

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

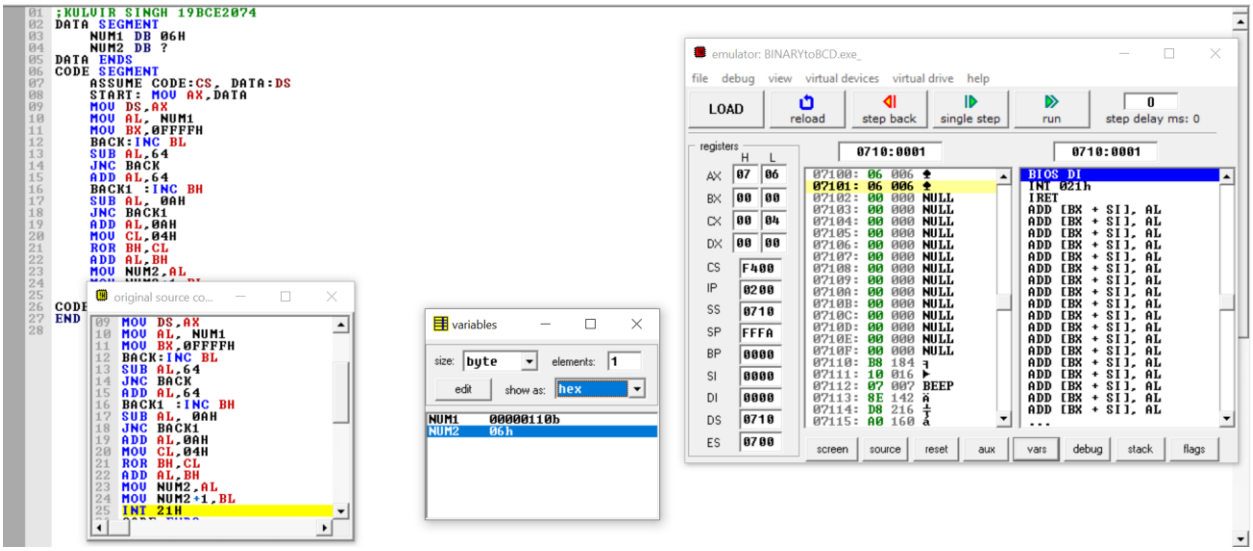
Handwritten Program :

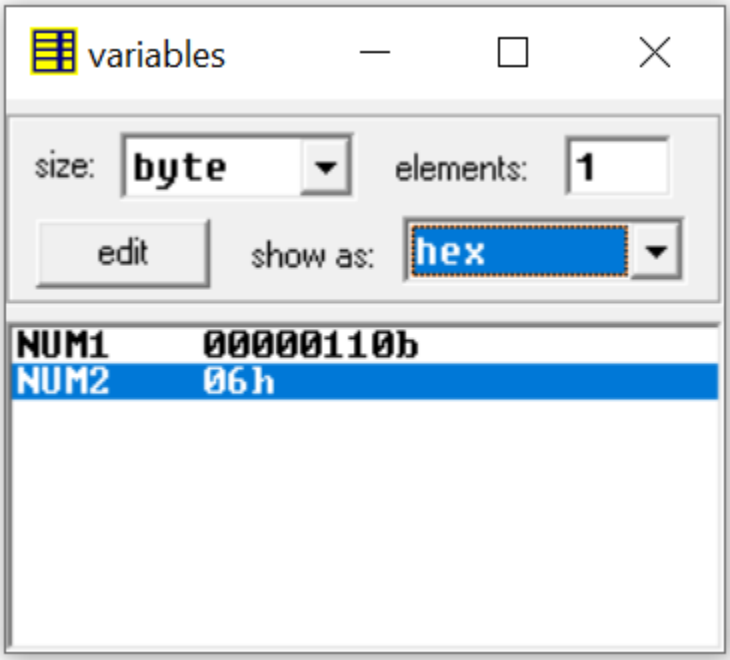
19BCE2024 Kishan Singh

Program : BINARY TO BCD Conversion

```
DATA   SEGMENT
       NUM1   DB   0AH      ; BINARY number to be converted
       NUM2   DB   ?        ; Result to be stored in
DATA   ENDS
CODE   SEGMENT
       ASSUME   CODE:CS , DATA:DS
       START:   MOV  AX, DATA
       MOV   DS, AX
       MOV   AL, NUM1       ; AL ← NUM1 D 06H
       MOV   BX, 0FFFFH     ; BX ← 0FFFFH
BACK:  INC   BL             ; BL ← BL+1
       SUB   AL, 64         ; AL ← AL - 64
       JNC   BACK           ; if carry present Jump back
       ADD   AL, 64         ; AL ← AL+64
BACK1: INC   BH             ; BH ← BH+1
       SUB   AL, 0AH        ; AL ← AL -0AH
       JNC   BACK1          ; if carry present Jump back
       ADD   AL, 0AH        ; AL ← AL+0AH
       MOV   CL, 04H        ; CL ← 04H
       ROR   BH, CL         ; rotate BH, CL
       ADD   AL, BH         ; AL ← AL + BH
       MOV   NUM2, AL       ; NUM2 ← AL
       MOV   NUM2+1, BL     ; [NUM2+1] ← AL
CODE   ENDS                 ; terminate .
END    START
```

Screenshots :





Inference :

The program can successfully change 00000110b a binary number to equivalent bcd 6 which is accurate.

5)Smallest and Largest Number in an array

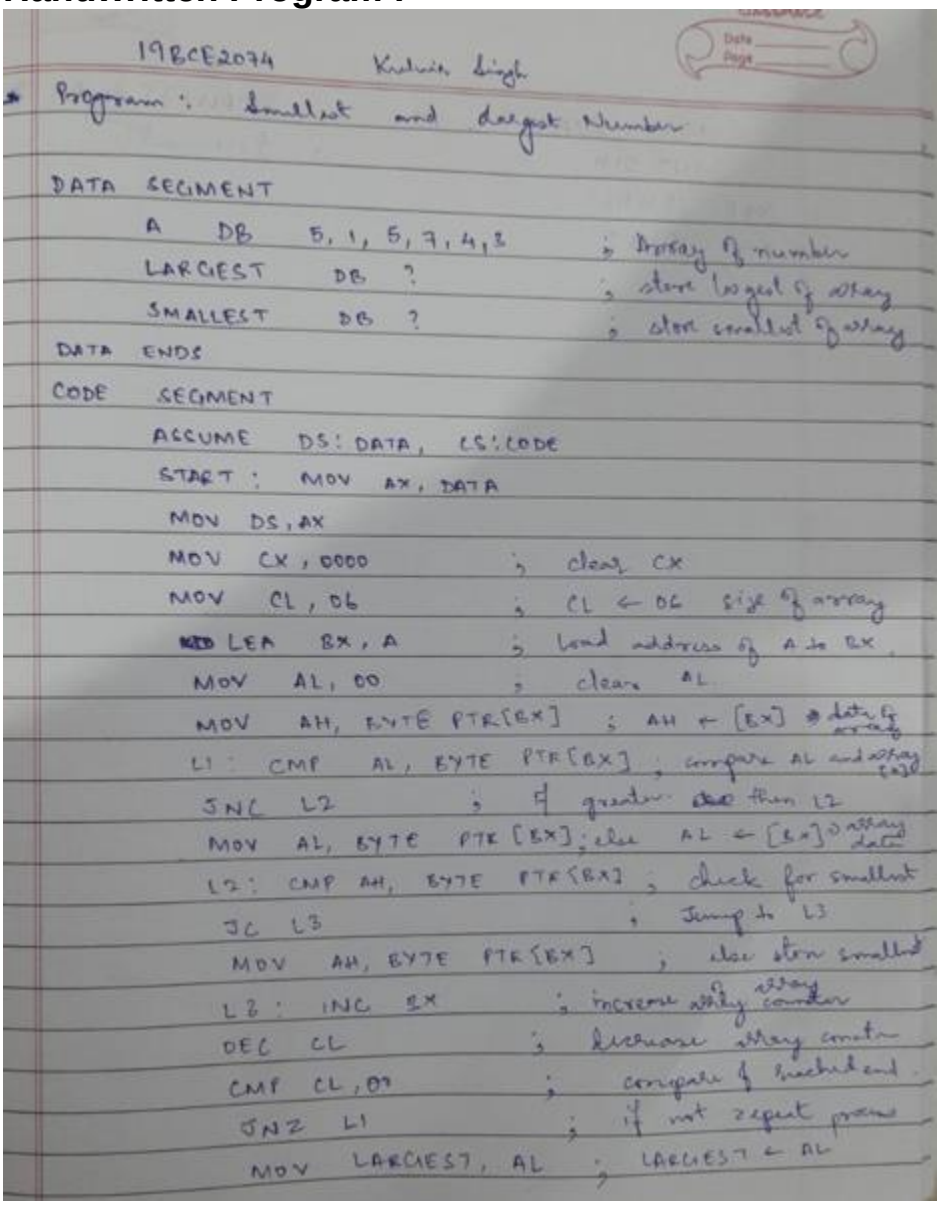
Aim :

Write a program in 8086 Assembly Language to find the smallest and largest number in an array.

Requirements :

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

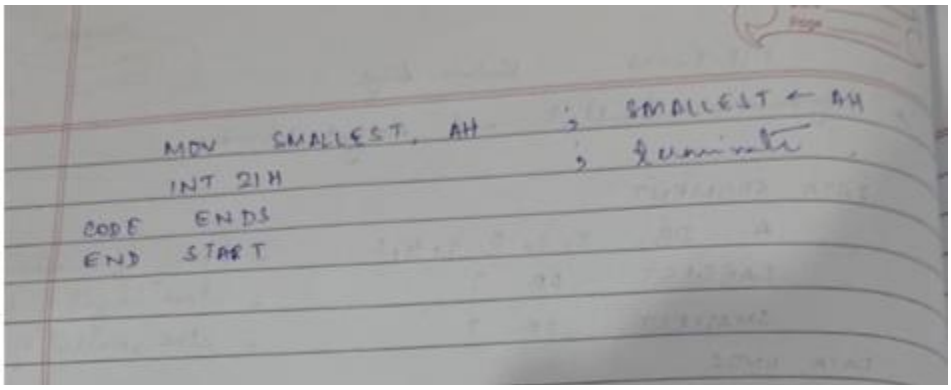
Handwritten Program :



19BCE2074 Kulwin Singh

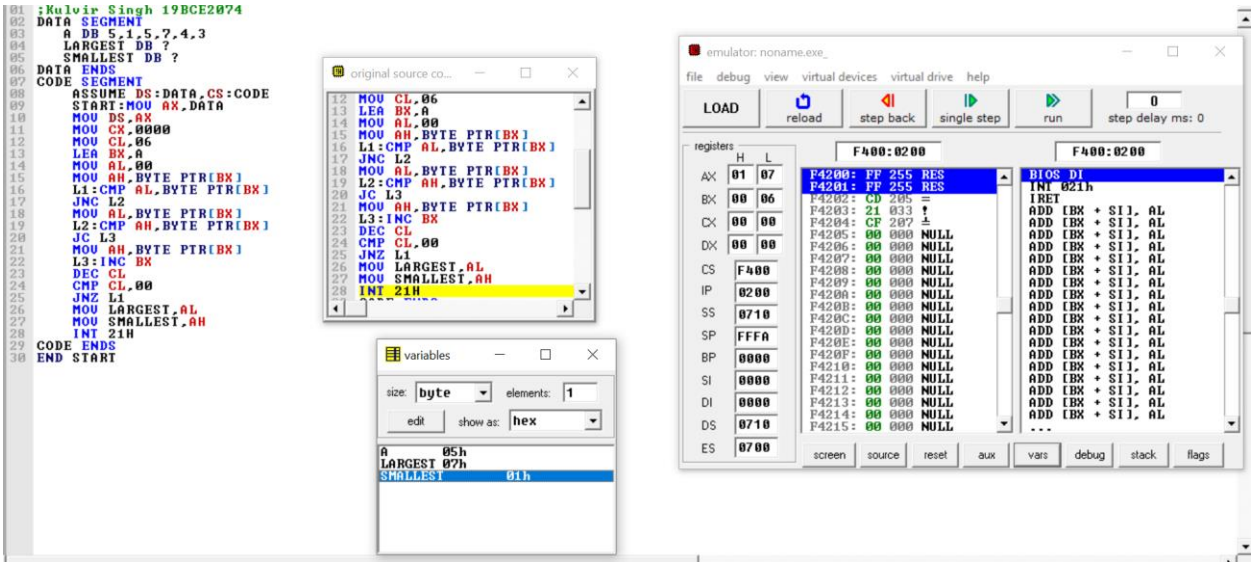
* Program : Smallest and largest Number

```
DATA SEGMENT
    A        DB   5, 1, 5, 7, 4, 3      ; Array of number
    LARGEST  DB   ?                     ; store largest of array
    SMALLEST DB   ?                     ; store smallest of array
DATA ENDS
CODE SEGMENT
    ASSUME   DS: DATA, CS: CODE
    START :  MOV  AX, DATA
     MOV  DS, AX
     MOV  CX, 0000          ; clear CX
     MOV  CL, 06            ; CL ← 06  size of array
     LEA  BX, A             ; load address of A to BX
     MOV  AL, 00            ; clear AL
     MOV  AH, BYTE PTR[BX]  ; AH ← [BX] data of array
     L1 : CMP  AL, BYTE PTR[BX]  ; compare AL and array [BX]
      JNC  L2               ; if greater then L2
      MOV  AL, BYTE PTR [BX]; else AL ← [BX] array data
      L2 : CMP AH, BYTE PTR[BX]  ; check for smallest
       JC  L3               ; Jump to L3
       MOV  AH, BYTE PTR[BX]  ; else store smallest
       L3 : INC BX          ; increase array counter
       DEC  CL              ; decrease array counter
       CMP  CL, 00          ; compare if reached end
       JNZ  L1              ; if not repeat process
       MOV  LARGEST, AL     ; LARGEST ← AL
```



```
MOV   SMALLEST, AH      ; SMALLEST ← AH
INT 21H                 ; terminate
CODE  ENDS
END   START
```

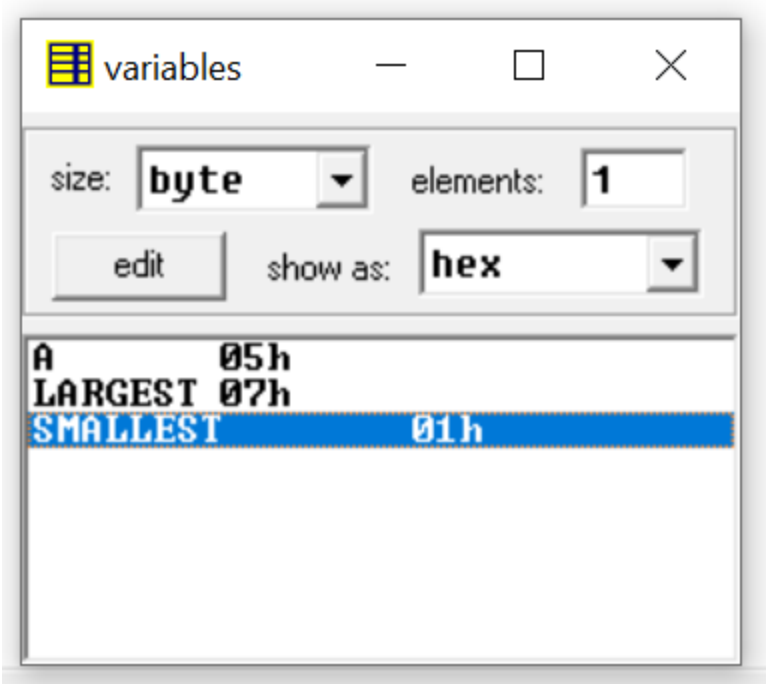
Screenshots :



```
;Kulvir Singh 19BCE2074
DATA SEGMENT
   A DB 5,1,5,7,4,3
   LARGEST DB ?
   SMALLEST DB ?
DATA ENDS
CODE SEGMENT
    ASSUME DS:DATA,CS:CODE
    START:MOV AX,DATA
    MOV DS,AX
    MOV CX,0000
    MOV CL,06
    LEA BX,A
    MOV AL,00
    MOV AH,BYTE PTR[BX]
    L1:CMP AL,BYTE PTR[BX]
    JNC L2
    MOV AL,BYTE PTR[BX]
    L2:CMP AH,BYTE PTR[BX]
    JC L3
    MOV AH,BYTE PTR[BX]
    L3:INC BX
    DEC CL
    CMP CL,00
    JNZ L1
    MOV LARGEST,AL
    MOV SMALLEST,AH
    INT 21H
CODE ENDS
END START
```

```
MOV CL,06
LEA BX,A
MOV AL,00
MOV AH,BYTE PTR[BX]
L1:CMP AL,BYTE PTR[BX]
JNC L2
MOV AL,BYTE PTR[BX]
L2:CMP AH,BYTE PTR[BX]
JC L3
MOV AH,BYTE PTR[BX]
L3:INC BX
DEC CL
CMP CL,00
JNZ L1
MOV LARGEST,AL
MOV SMALLEST,AH
INT 21H
```

```
A        05h
LARGEST  07h
SMALLEST 01h
```



Inference :

The program can successfully find the largest value 7 and smallest value 1 from the array 5,1,5,7,4,3