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

Microprocessor and Interfacing
CSE2006

Lab Assignment 4

Slot: L3+L4

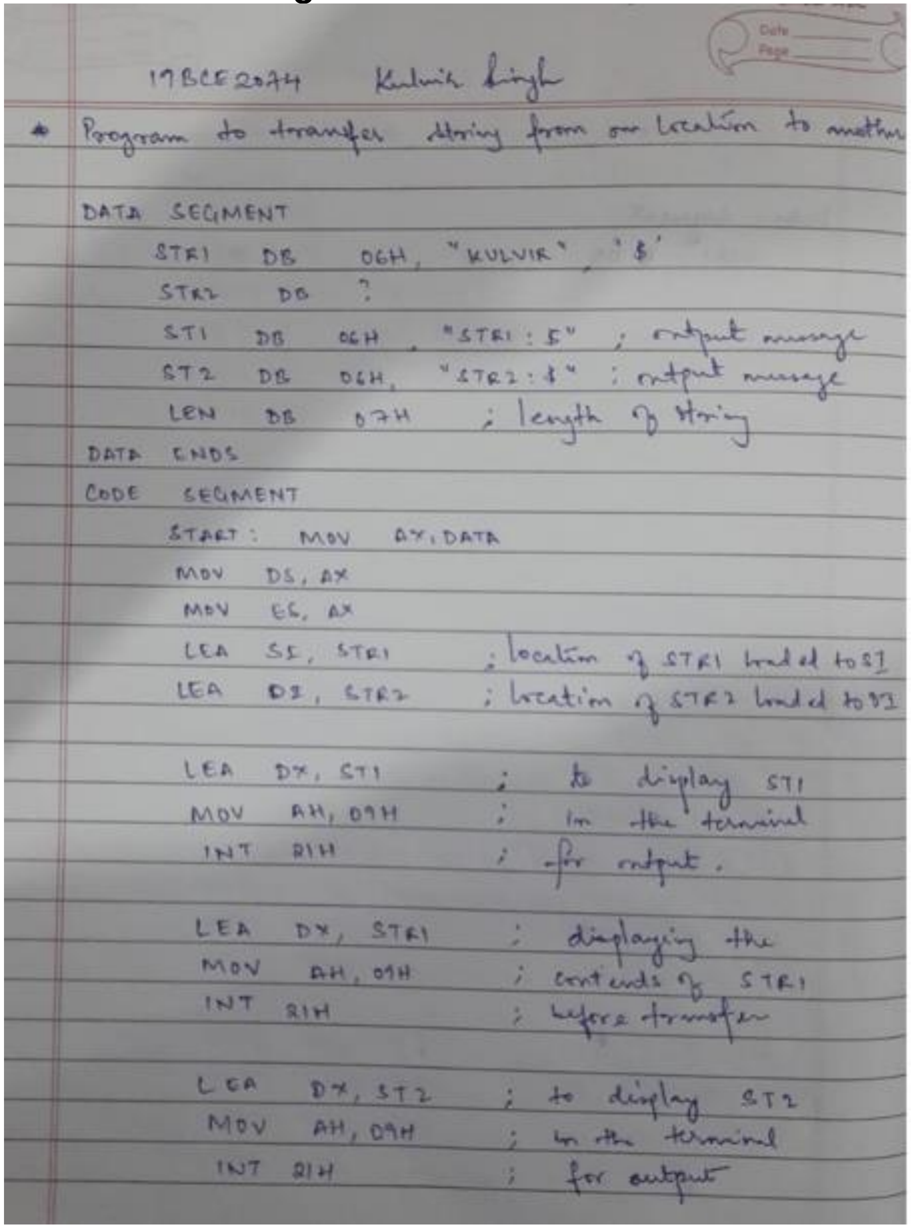
Name: Kulvir Singh

Register Number: 19BCE2074

1)String Transfer from One Location to Another Location

Aim :
Write a program in 8086 Assembly Language to transfer a string from one location to another.

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

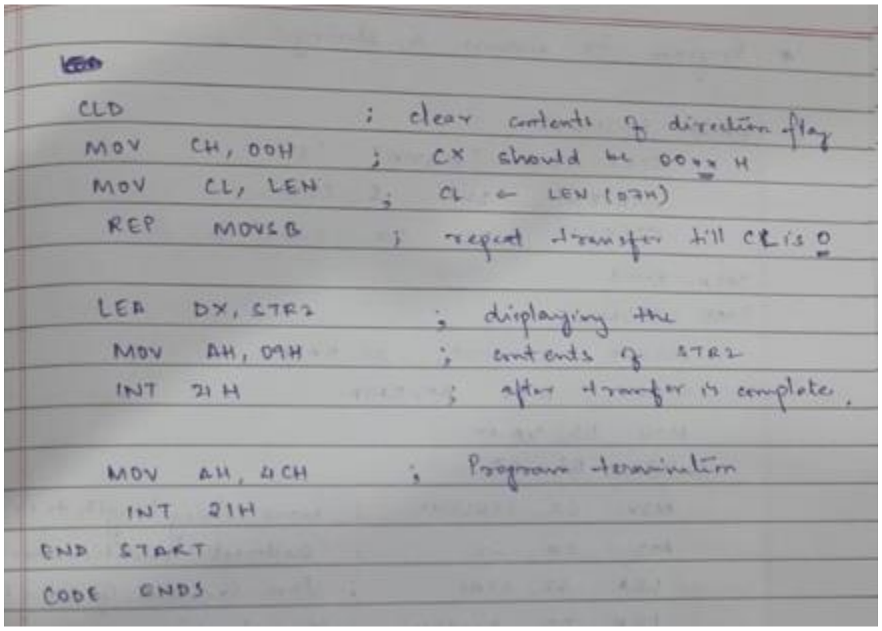
Handwritten Program :


Date ______
Page ______

17BCE2074 Kulvir Singh

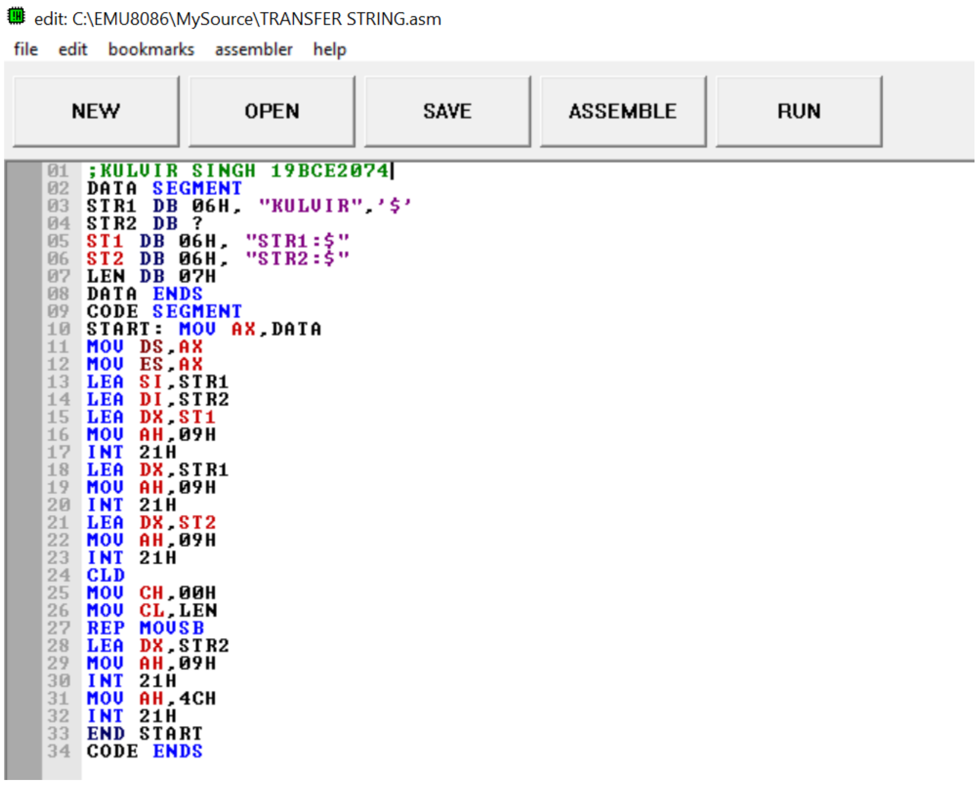
* Program to transfer string from one location to another

```
DATA SEGMENT
    STR1 DB 06H, "KULVIR", '$'
    STR2 DB ?
    ST1  DB 06H , "STR1 : $"  ; output message
    ST2  DB 06H, "STR2 : $"   ; output message
    LEN  DB 07H               ; length of string
DATA ENDS
CODE SEGMENT
    START : MOV AX, DATA
    MOV DS, AX
    MOV ES, AX
    LEA SI, STR1              ; location of STR1 loaded to SI
    LEA DI, STR2              ; location of STR2 loaded to DI

    LEA DX, ST1               ; to display ST1
    MOV AH, 09H               ; in the terminal
    INT 21H                   ; for output.

    LEA DX, STR1              ; displaying the
    MOV AH, 09H               ; contents of STR1
    INT 21H                   ; before transfer

    LEA DX, ST2               ; to display ST2
    MOV AH, 09H               ; in the terminal
    INT 21H                   ; for output
```

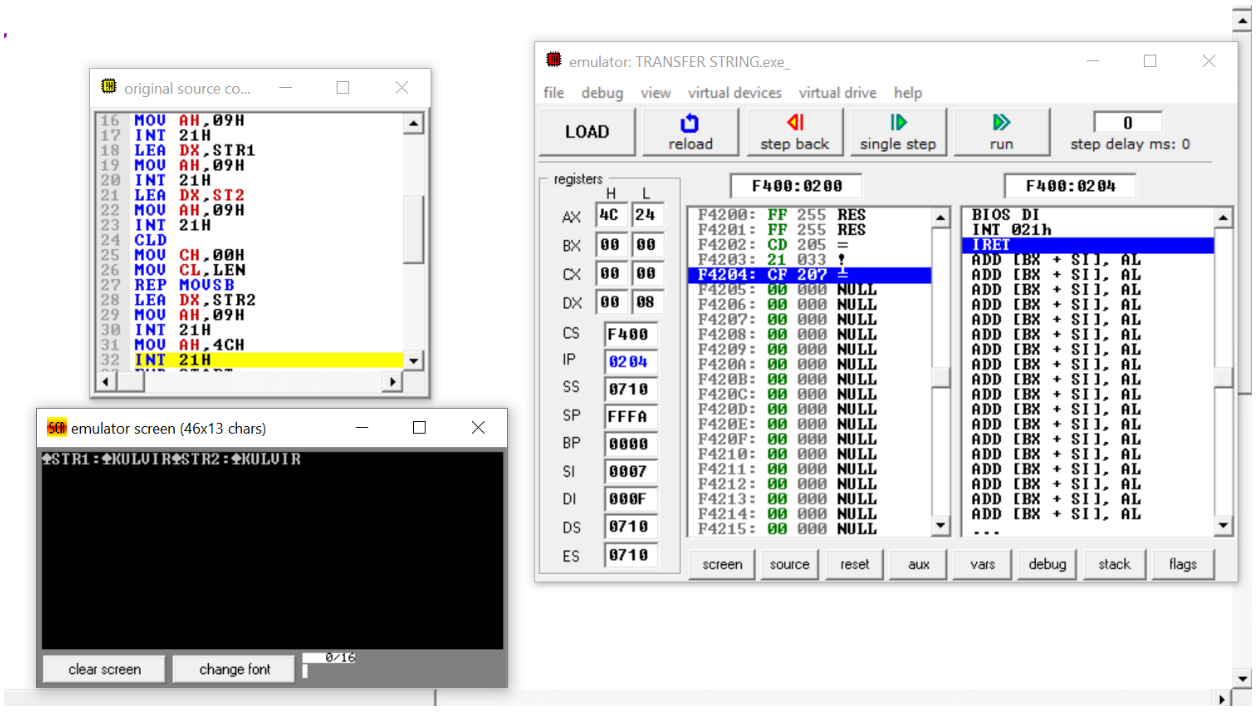


```
CLD                 ; clear contents of direction flag
MOV   CH, 00H       ; CX should be 00xx H
MOV   CL, LEN       ; CL ← LEN (07H)
REP   MOVSB         ; repeat transfer till CL is 0

LEA   DX, STR2      ; displaying the
MOV   AH, 09H       ; contents of STR2
INT   21 H          ; after transfer is complete.

MOV   AH, 4CH       ; Program termination
INT   21H
END   START
CODE  ENDS
```

Screenshots :


edit: C:\EMU8086\MySource\TRANSFER STRING.asm
file edit bookmarks assembler help
NEW
OPEN
SAVE
ASSEMBLE
RUN

```
;KULVIR SINGH 19BCE2074
DATA SEGMENT
STR1 DB 06H, "KULVIR",'$'
STR2 DB ?
ST1 DB 06H, "STR1:$"
ST2 DB 06H, "STR2:$"
LEN DB 07H
DATA ENDS
CODE SEGMENT
START: MOV AX,DATA
MOV DS,AX
MOV ES,AX
LEA SI,STR1
LEA DI,STR2
LEA DX,ST1
MOV AH,09H
INT 21H
LEA DX,STR1
MOV AH,09H
INT 21H
LEA DX,ST2
MOV AH,09H
INT 21H
CLD
MOV CH,00H
MOV CL,LEN
REP MOVSB
LEA DX,STR2
MOV AH,09H
INT 21H
MOV AH,4CH
INT 21H
END START
CODE ENDS
```



```
MOV  AH,09H
INT  21H
LEA  DX,STR1
MOV  AH,09H
INT  21H
LEA  DX,ST2
MOV  AH,09H
INT  21H
CLD
MOV  CH,00H
MOV  CL,LEN
REP  MOVSB
LEA  DX,STR2
MOV  AH,09H
INT  21H
MOV  AH,4CH
INT  21H
```

```
♠STR1:♠KULVIR♠STR2:♠KULVIR
```

```
     H   L
AX   4C  24
BX   00  00
CX   00  00
DX   00  08
CS   F400
IP   0204
SS   0710
SP   FFFA
BP   0000
SI   0007
DI   000F
DS   0710
ES   0710
```

```
F400:0200

F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204

BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

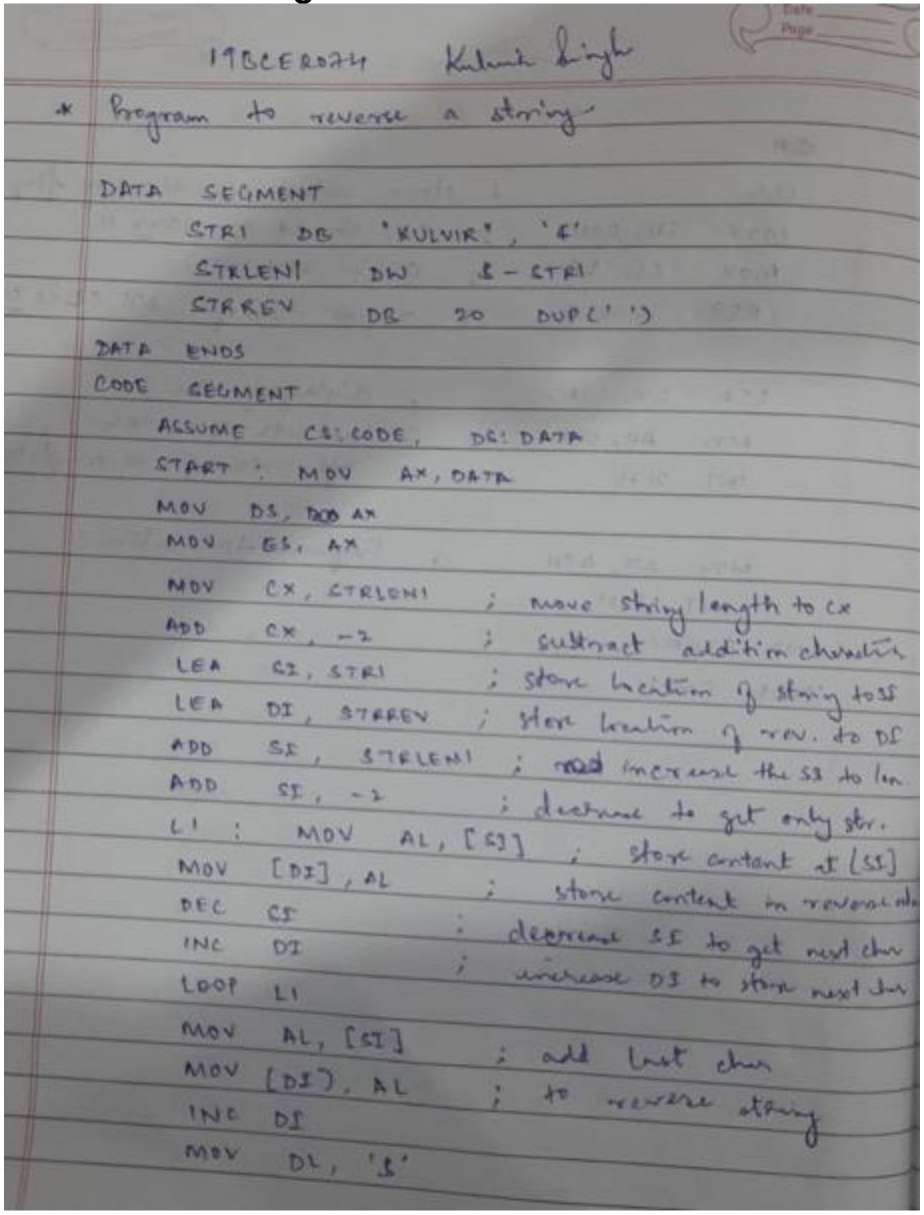
Inference :

The program can successfully transfer a block of string from one location to another a visible in the output terminal

2)Reverse a String

Aim :
Write a program in 8086 Assembly Language to reverse a string.

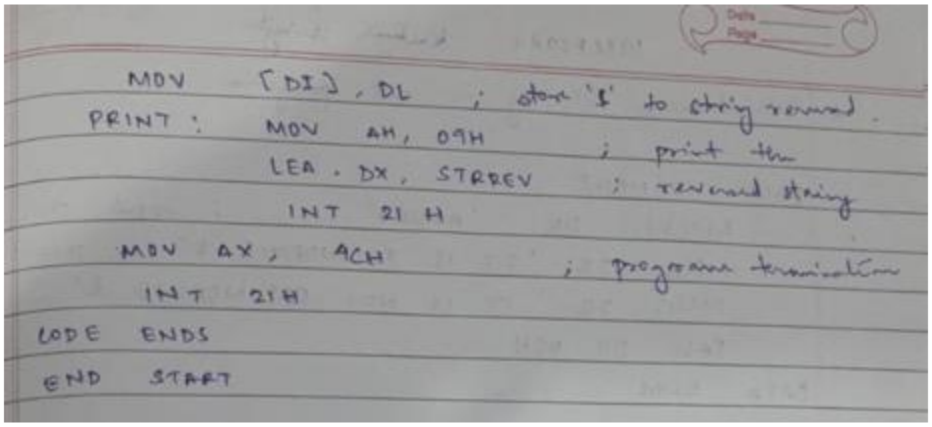
Requirements :
8086 EMU - An emulator to run the 8086 Assembly Language Code
Operating System - Any valid operating system that can execute the emulator

Handwritten Program :


19BCE2074 Kulvir Singh

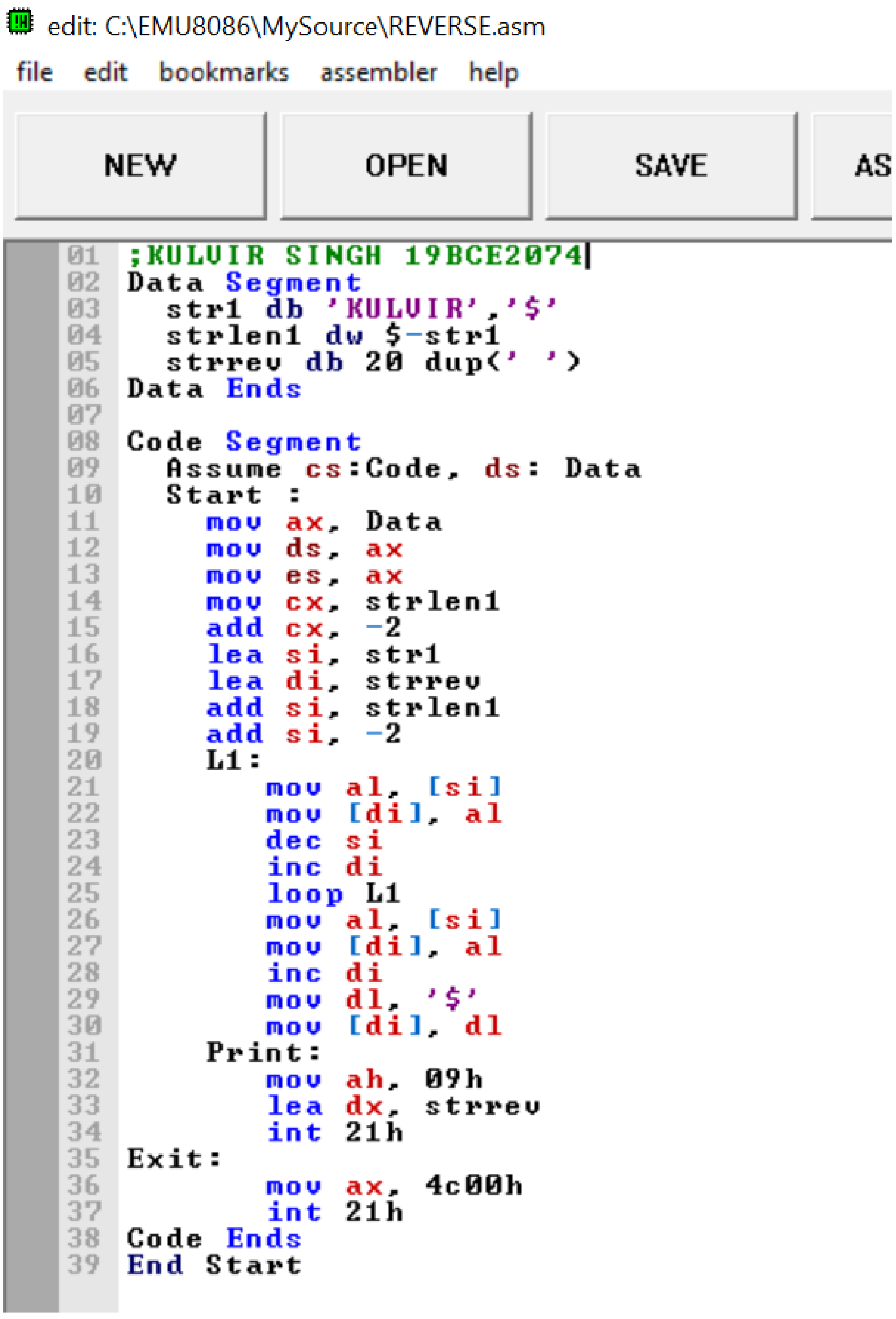
* Program to reverse a string.

```
DATA SEGMENT
        STR1    DB   'KULVIR', '$'
        STRLEN1     DW    $ - STR1
        STRREV      DB   20   DUP(' ')
DATA ENDS
CODE SEGMENT
     ASSUME   CS: CODE,   DS: DATA
     START :  MOV   AX, DATA
     MOV   DS, ~~DS~~ AX
     MOV   ES, AX
     MOV   CX, STRLEN1      ; move string length to cx
     ADD   CX, -2           ; subtract addition character
     LEA   SI, STR1         ; store location of string to SI
     LEA   DI, STRREV       ; store location of rev. to DI
     ADD   SI, STRLEN1      ; ~~read~~ increase the SI to len
     ADD   SI, -2           ; decrease to get only str.
     L1 :  MOV  AL, [SI]    ; store content at [SI]
     MOV   [DI], AL         ; store content in reverse order
     DEC   SI               ; decrease SI to get next char
     INC   DI               ; increase DI to store next char
     LOOP  L1
     MOV   AL, [SI]         ; add last char
     MOV   [DI], AL         ; to reverse string
     INC   DI
     MOV   DL, '$'
```

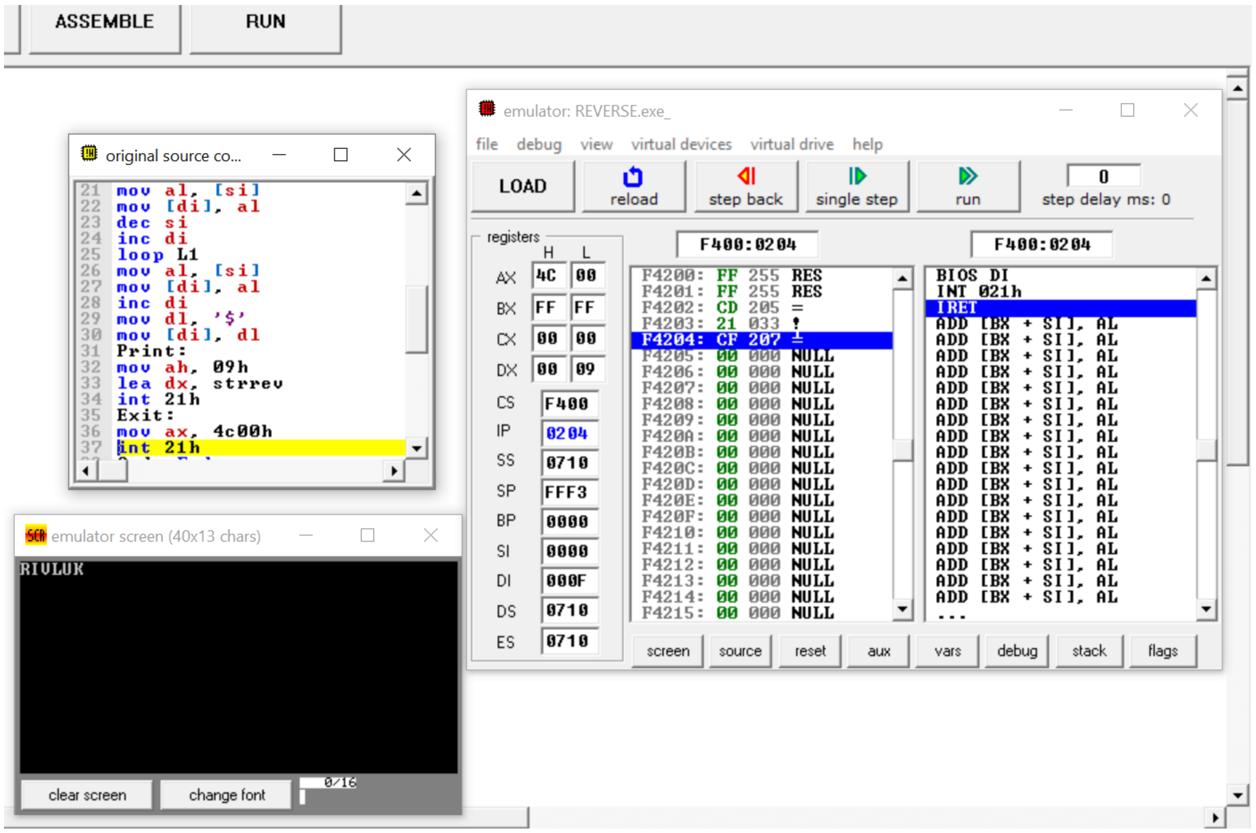


```
    MOV    [DI] , DL      ; store '$' to string reversed .
PRINT :    MOV   AH, 09H        ; print the
           LEA . DX , STRREV     ; reversed string
            INT  21 H
    MOV   AX ,  4CH            ; programme termination
     INT   21H
CODE  ENDS
END   START
```

Screenshots :


edit: C:\EMU8086\MySource\REVERSE.asm
file edit bookmarks assembler help
NEW
OPEN
SAVE
AS

```
;KULVIR SINGH 19BCE2074
Data Segment
  str1 db 'KULVIR','$'
  strlen1 dw $-str1
  strrev db 20 dup(' ')
Data Ends

Code Segment
  Assume cs:Code, ds: Data
  Start :
    mov ax, Data
    mov ds, ax
    mov es, ax
    mov cx, strlen1
    add cx, -2
    lea si, str1
    lea di, strrev
    add si, strlen1
    add si, -2
    L1:
       mov al, [si]
       mov [di], al
       dec si
       inc di
       loop L1
       mov al, [si]
       mov [di], al
       inc di
       mov dl, '$'
       mov [di], dl
    Print:
       mov ah, 09h
       lea dx, strrev
       int 21h
Exit:
       mov ax, 4c00h
       int 21h
Code Ends
End Start
```



```
mov al, [si]
mov [di], al
dec si
inc di
loop L1
mov al, [si]
mov [di], al
inc di
mov dl, '$'
mov [di], dl
Print:
mov ah, 09h
lea dx, strrev
int 21h
Exit:
mov ax, 4c00h
int 21h
```

```
RIVLUK
```

	H	L
AX	4C	00
BX	FF	FF
CX	00	00
DX	00	09
CS	F400	
IP	0204	
SS	0710	
SP	FFF3	
BP	0000	
SI	0000	
DI	000F	
DS	0710	
ES	0710	

```
F400:0204

F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204

BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

Inference :

The program can successfully reverse a string as seen in the output terminal

3)Check Palindrome String

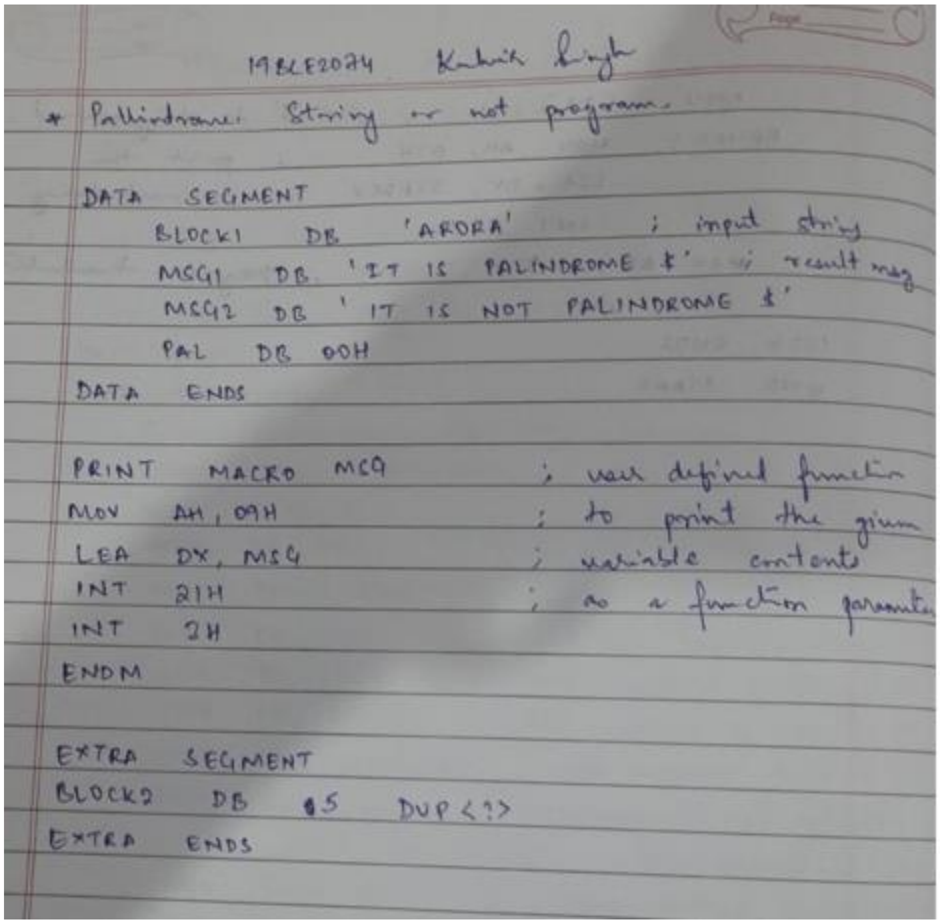
Aim :

Write a program in 8086 Assembly Language to check if a string is palindrome or not.

Requirements :

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

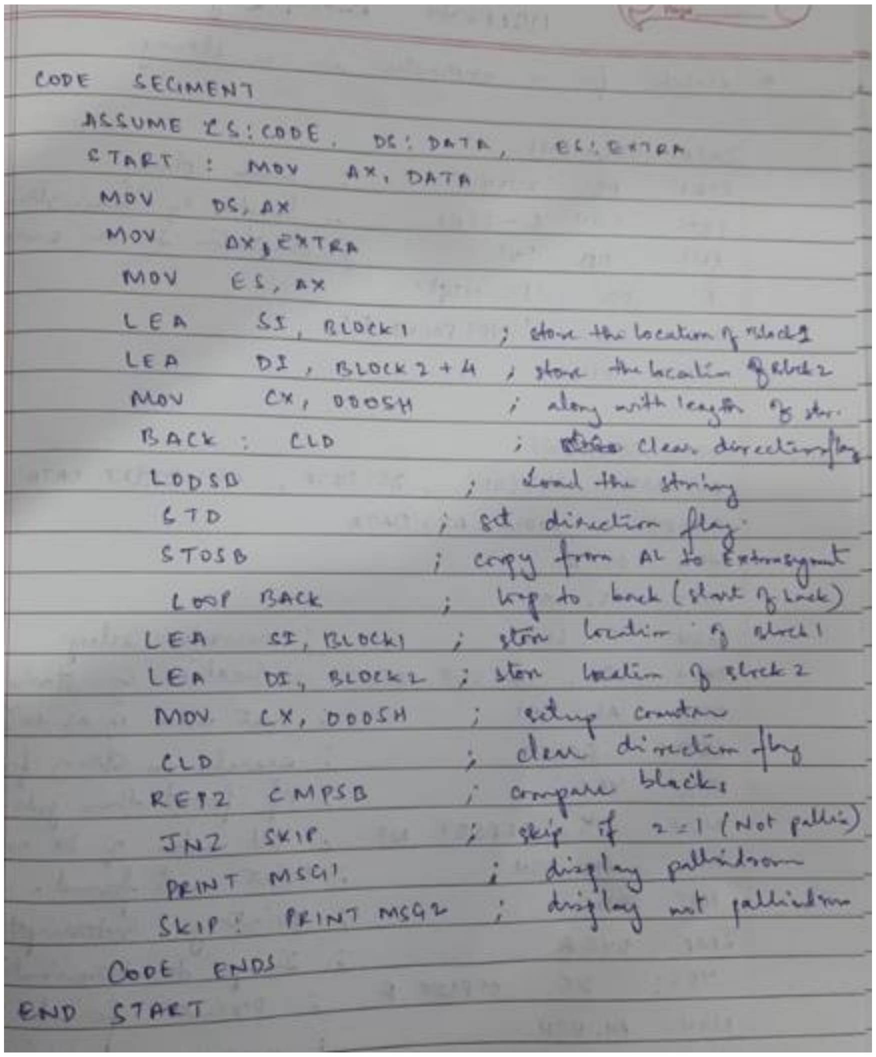
Handwritten Program :



19BCE2074 Kushik Singh

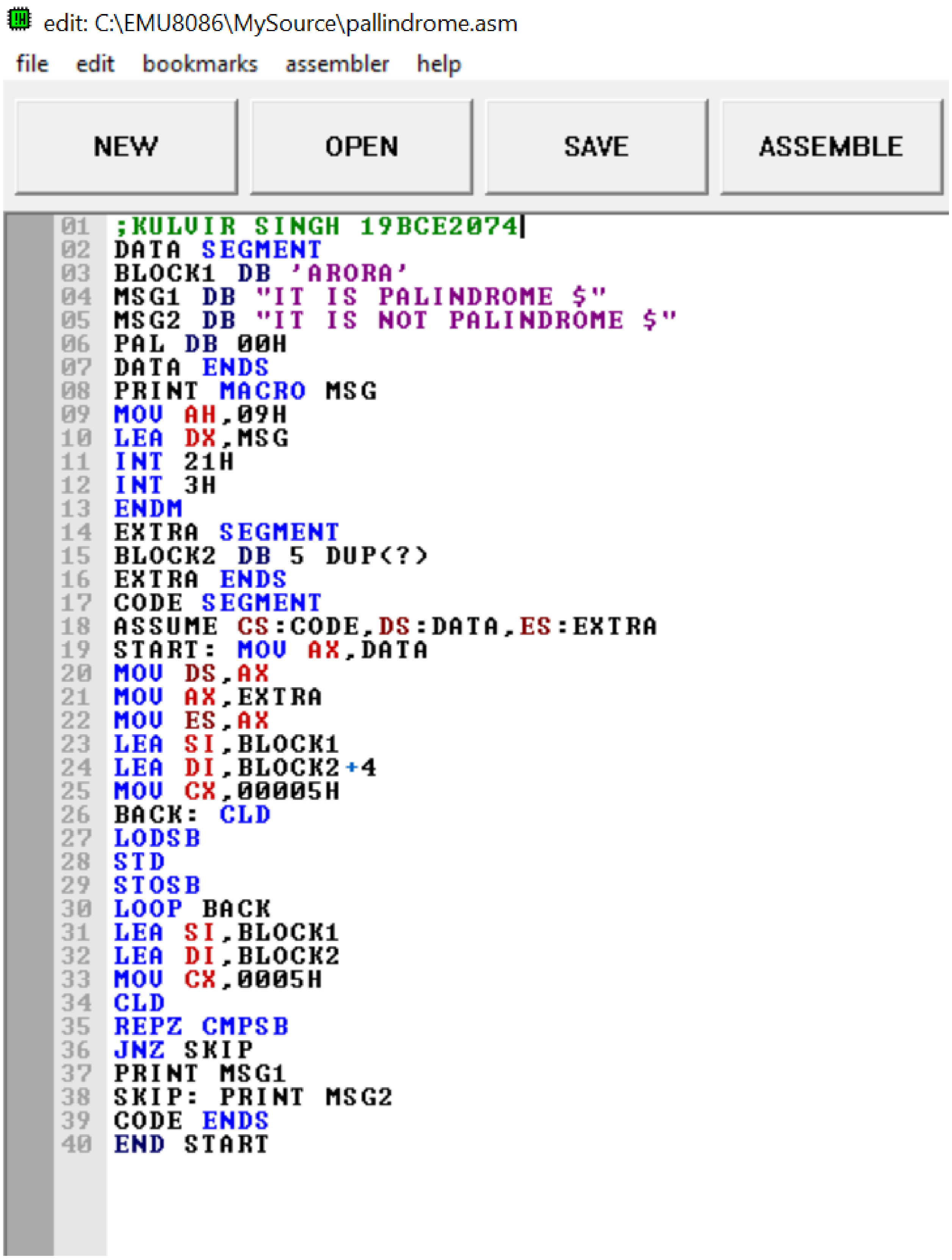
* Pallindrome String or not program.

```
DATA  SEGMENT
      BLOCK1  DB  'ARORA'                    ; input string
      MSG1  DB  'IT IS PALINDROME $'          ; result msg
      MSG2  DB  'IT IS NOT PALINDROME $'
      PAL  DB  00H
DATA  ENDS

PRINT  MACRO  MSG                             ; user defined function
MOV  AH, 09H                                  ; to print the given
LEA  DX, MSG                                  ; variable contents
INT  21H                                      ; as a function parameter
INT  3H
ENDM

EXTRA  SEGMENT
BLOCK2  DB  05  DUP <?>
EXTRA  ENDS
```

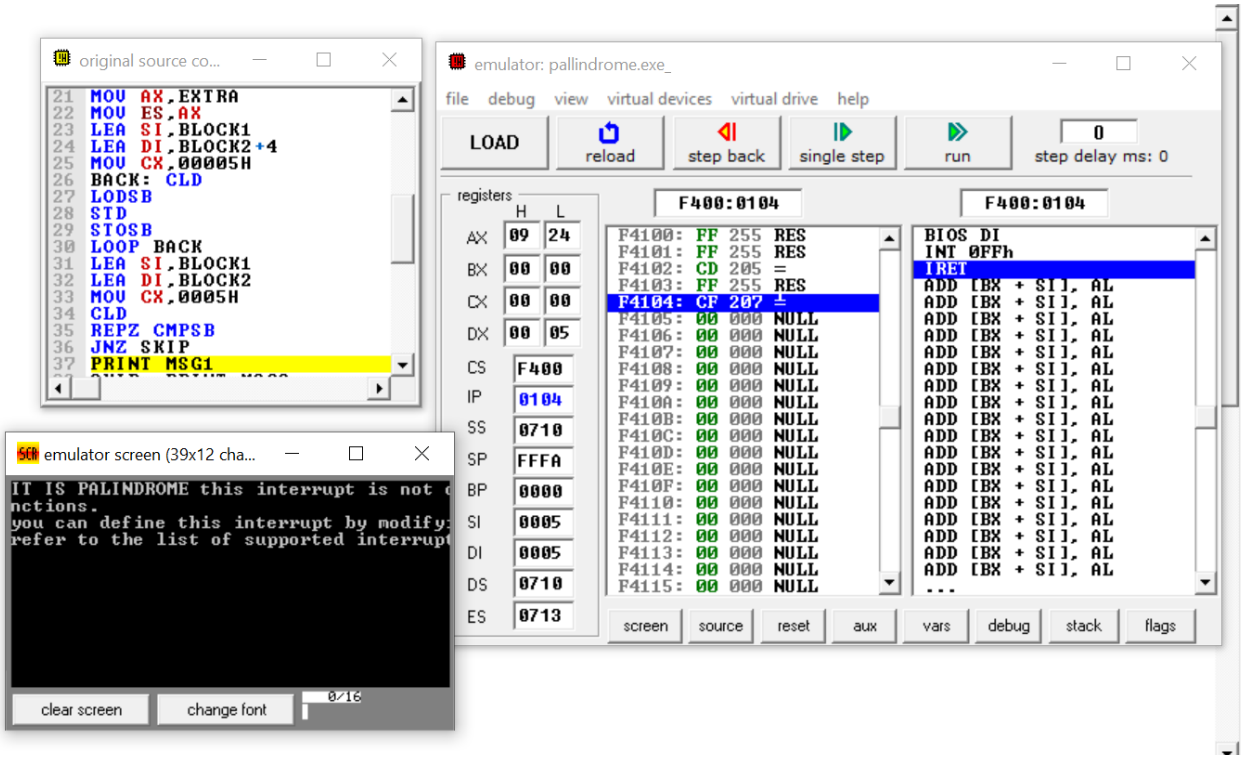


```
CODE   SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:EXTRA
    START : MOV   AX, DATA
     MOV    DS, AX
     MOV    AX, EXTRA
      MOV    ES, AX
      LEA     SI, BLOCK1        ; store the location of block1
      LEA     DI, BLOCK2 + 4    ; store the location of block2
      MOV     CX, 0005H         ; along with length of str.
       BACK :  CLD              ; ~~[illegible]~~ Clear direction flag
       LODSB                   ; Load the string
        STD                    ; set direction flag.
        STOSB                  ; copy from AL to Extra segment
         LOOP BACK             ; loop to back (start of back)
       LEA    SI, BLOCK1    ; store location of block1
       LEA    DI, BLOCK2    ; store location of block2
       MOV    CX, 0005H     ; setup counter
       CLD                  ; clear direction flag
       REPZ  CMPSB          ; compare blocks
       JNZ  SKIP            ; skip if z=1 (Not palli.)
       PRINT MSG1           ; display pallindrom
       SKIP : PRINT MSG2    ; display not pallindrom
    CODE  ENDS
END  START
```

Screenshots :


edit: C:\EMU8086\MySource\pallindrome.asm
file edit bookmarks assembler help
NEW
OPEN
SAVE
ASSEMBLE

```
;KULVIR SINGH 19BCE2074
DATA SEGMENT
BLOCK1 DB 'ARORA'
MSG1 DB "IT IS PALINDROME $"
MSG2 DB "IT IS NOT PALINDROME $"
PAL DB 00H
DATA ENDS
PRINT MACRO MSG
MOV AH,09H
LEA DX,MSG
INT 21H
INT 3H
ENDM
EXTRA SEGMENT
BLOCK2 DB 5 DUP(?)
EXTRA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA,ES:EXTRA
START: MOV AX,DATA
MOV DS,AX
MOV AX,EXTRA
MOV ES,AX
LEA SI,BLOCK1
LEA DI,BLOCK2+4
MOV CX,00005H
BACK: CLD
LODSB
STD
STOSB
LOOP BACK
LEA SI,BLOCK1
LEA DI,BLOCK2
MOV CX,0005H
CLD
REPZ CMPSB
JNZ SKIP
PRINT MSG1
SKIP: PRINT MSG2
CODE ENDS
END START
```



```
MOV AX,EXTRA
MOV ES,AX
LEA SI,BLOCK1
LEA DI,BLOCK2+4
MOV CX,00005H
BACK: CLD
LODSB
STD
STOSB
LOOP BACK
LEA SI,BLOCK1
LEA DI,BLOCK2
MOV CX,0005H
CLD
REPZ CMPSB
JNZ SKIP
PRINT MSG1
```

```
IT IS PALINDROME this interrupt is not
nctions.
you can define this interrupt by modify
refer to the list of supported interrupt
```

```
      H   L
AX   09  24
BX   00  00
CX   00  00
DX   00  05
CS   F400
IP   0104
SS   0710
SP   FFFA
BP   0000
SI   0005
DI   0005
DS   0710
ES   0713
```

```
F400:0104

F4100:  FF 255 RES
F4101:  FF 255 RES
F4102:  CD 205 =
F4103:  FF 255 RES
F4104:  CF 207 ±
F4105:  00 000 NULL
F4106:  00 000 NULL
F4107:  00 000 NULL
F4108:  00 000 NULL
F4109:  00 000 NULL
F410A:  00 000 NULL
F410B:  00 000 NULL
F410C:  00 000 NULL
F410D:  00 000 NULL
F410E:  00 000 NULL
F410F:  00 000 NULL
F4110:  00 000 NULL
F4111:  00 000 NULL
F4112:  00 000 NULL
F4113:  00 000 NULL
F4114:  00 000 NULL
F4115:  00 000 NULL
```

```
F400:0104

BIOS DI
INT 0FFh
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

original source co...
emulator: pallindrome.exe_
file debug view virtual devices virtual drive help
LOAD reload step back single step run step delay ms: 0
registers
emulator screen (39x12 cha...
clear screen change font 0/16
screen source reset aux vars debug stack flags

Inference :

The program can successfully reverse a string as seen in the output terminal

4)Search for a character in a string

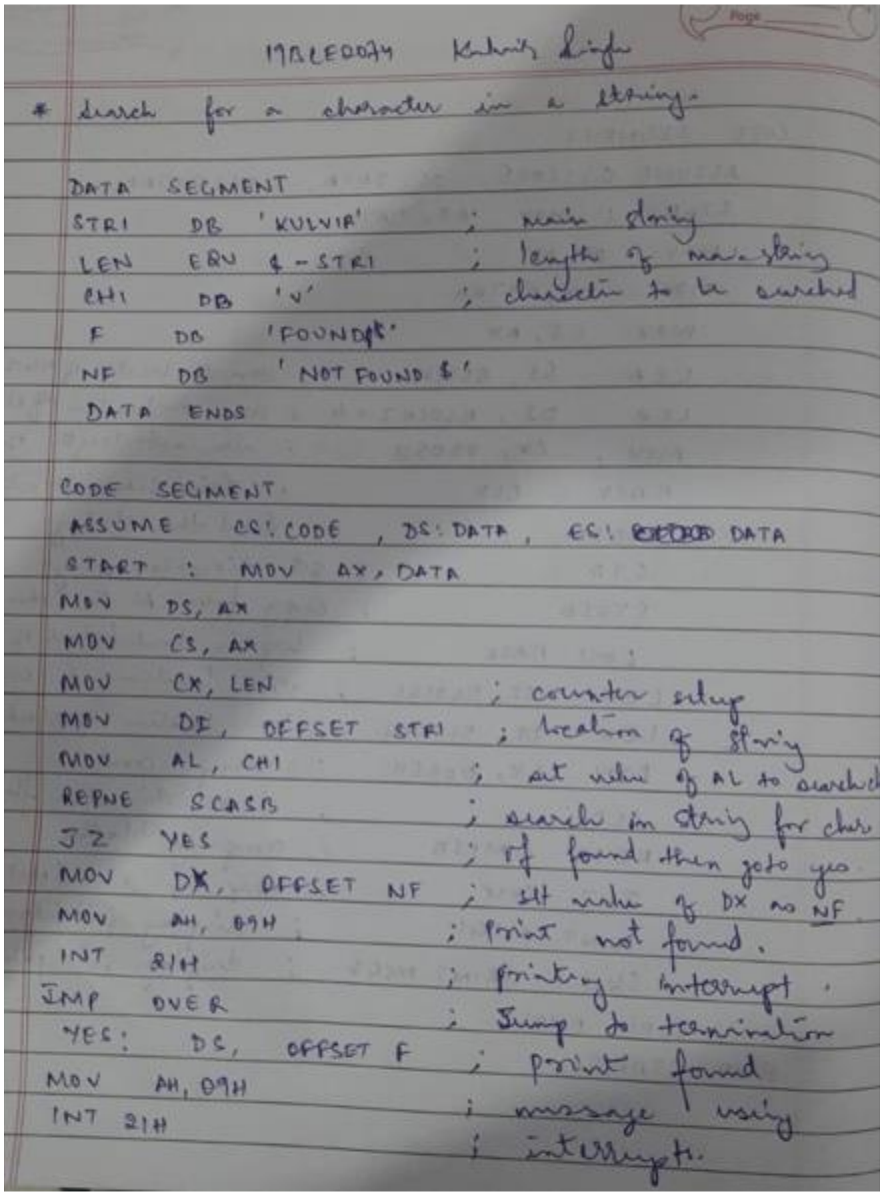
Aim :

Write a program in 8086 Assembly Language to check if a character is present in a given string or not.

Requirements :

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

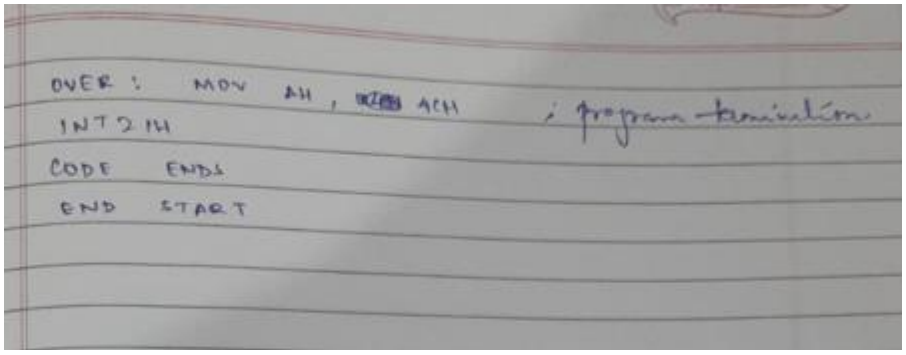
Handwritten Program :



19BLE0074 [illegible]

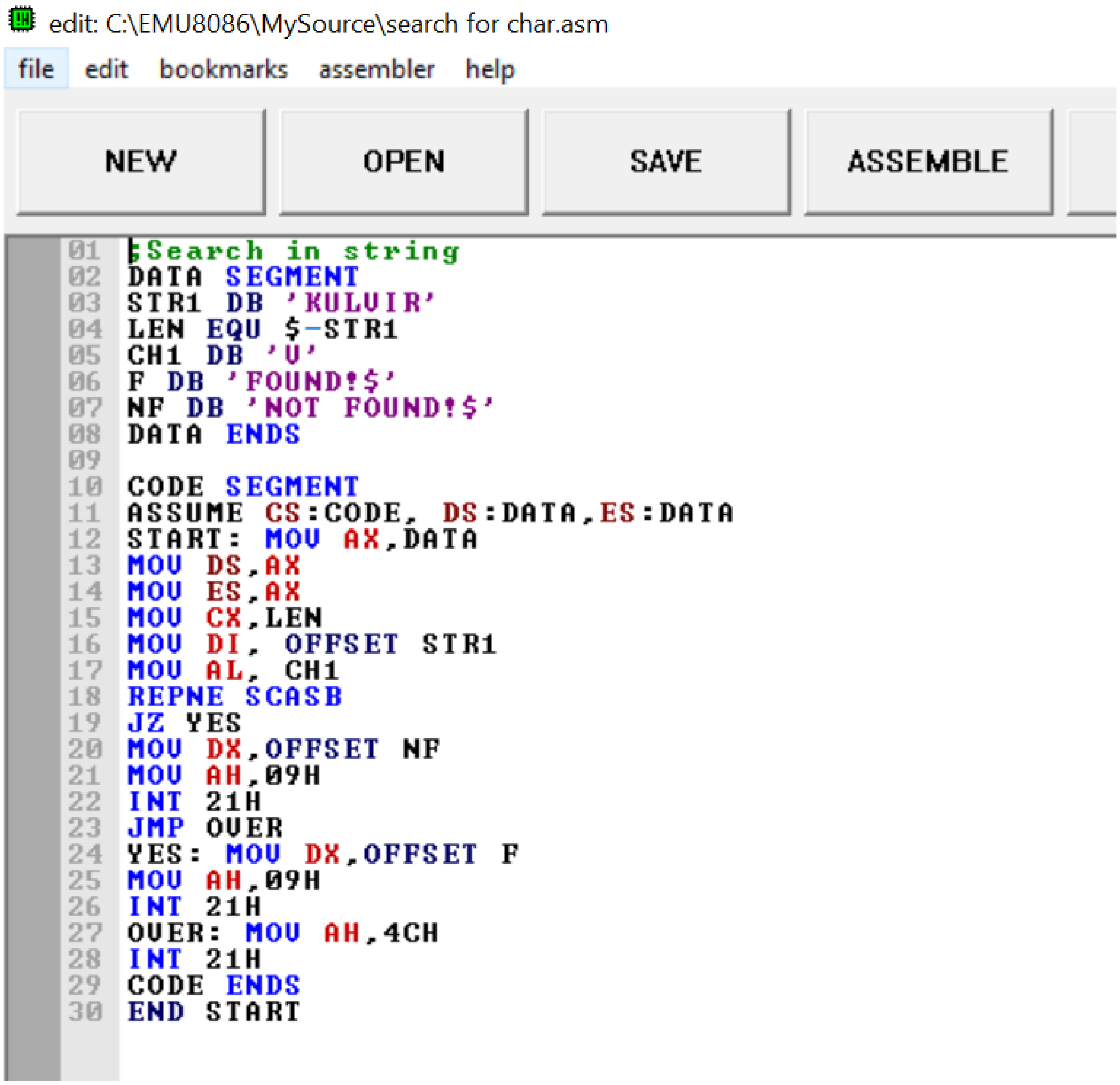
* search for a character in a string.

```
DATA SEGMENT
STR1   DB   'KULVIR'          ; main string
LEN    EQU  $ - STR1          ; length of main string
CH1    DB   'V'               ; character to be searched
F      DB   'FOUND$'
NF     DB   'NOT FOUND $'
DATA ENDS

CODE SEGMENT
ASSUME  CS: CODE , DS: DATA , ES: [illegible] DATA
START : MOV AX, DATA
MOV   DS, AX
MOV   CS, AX
MOV   CX, LEN                  ; counter setup
MOV   DI, OFFSET STR1          ; location of string
MOV   AL, CH1                  ; set value of AL to searched
REPNE SCASB                    ; search in string for char
JZ    YES                      ; if found then goto yes
MOV   DX, OFFSET NF            ; set value of DX as NF
MOV   AH, 09H                  ; Print not found.
INT   21H                      ; printing interrupt.
JMP   OVER                     ; Jump to termination
 YES:   DS, OFFSET F           ; print found
MOV   AH, 09H                  ; message using
INT 21H                        ; interrupts.
```

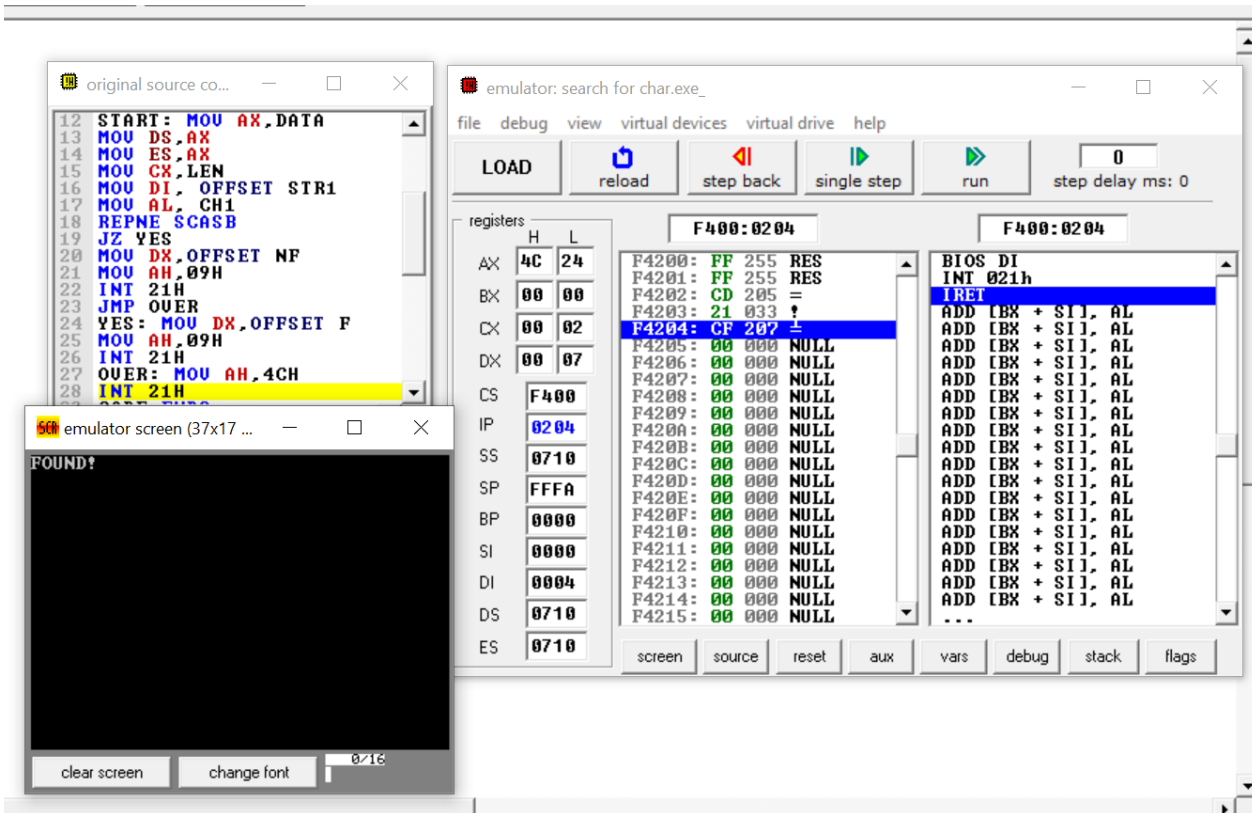


```
OVER :  MOV  AH , 4CH        ; program termination
INT 21H
CODE  ENDS
END  START
```

Screenshots :



```
;Search in string
DATA SEGMENT
STR1 DB 'KULUIR'
LEN EQU $-STR1
CH1 DB 'U'
F DB 'FOUND!$'
NF DB 'NOT FOUND!$'
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA,ES:DATA
START: MOV AX,DATA
MOV DS,AX
MOV ES,AX
MOV CX,LEN
MOV DI, OFFSET STR1
MOV AL, CH1
REPNE SCASB
JZ YES
MOV DX,OFFSET NF
MOV AH,09H
INT 21H
JMP OVER
YES: MOV DX,OFFSET F
MOV AH,09H
INT 21H
OVER: MOV AH,4CH
INT 21H
CODE ENDS
END START
```



```
START: MOV AX,DATA
MOV DS,AX
MOV ES,AX
MOV CX,LEN
MOV DI, OFFSET STR1
MOV AL, CH1
REPNE SCASB
JZ YES
MOV DX,OFFSET NF
MOV AH,09H
INT 21H
JMP OVER
YES: MOV DX,OFFSET F
MOV AH,09H
INT 21H
OVER: MOV AH,4CH
INT 21H
```

```
FOUND!
```

```
registers
     H   L
AX  4C  24
BX  00  00
CX  00  02
DX  00  07
CS  F400
IP  0204
SS  0710
SP  FFFA
BP  0000
SI  0000
DI  0004
DS  0710
ES  0710
```

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

Inference :

The program can successfully find out the character ‘v’ in the given string ‘kulvir’ as seen in the output terminal