Name:- Sravan Chittupalli

VJTI ID: 181060018

EXPERIMENT - 3

AIM

1. Design an 8086-assembly language program to identify the number of occurrences of a shorter string in a longer string. Both the strings are stored in the DS and the variable RESULT will hold the number of occurrences.

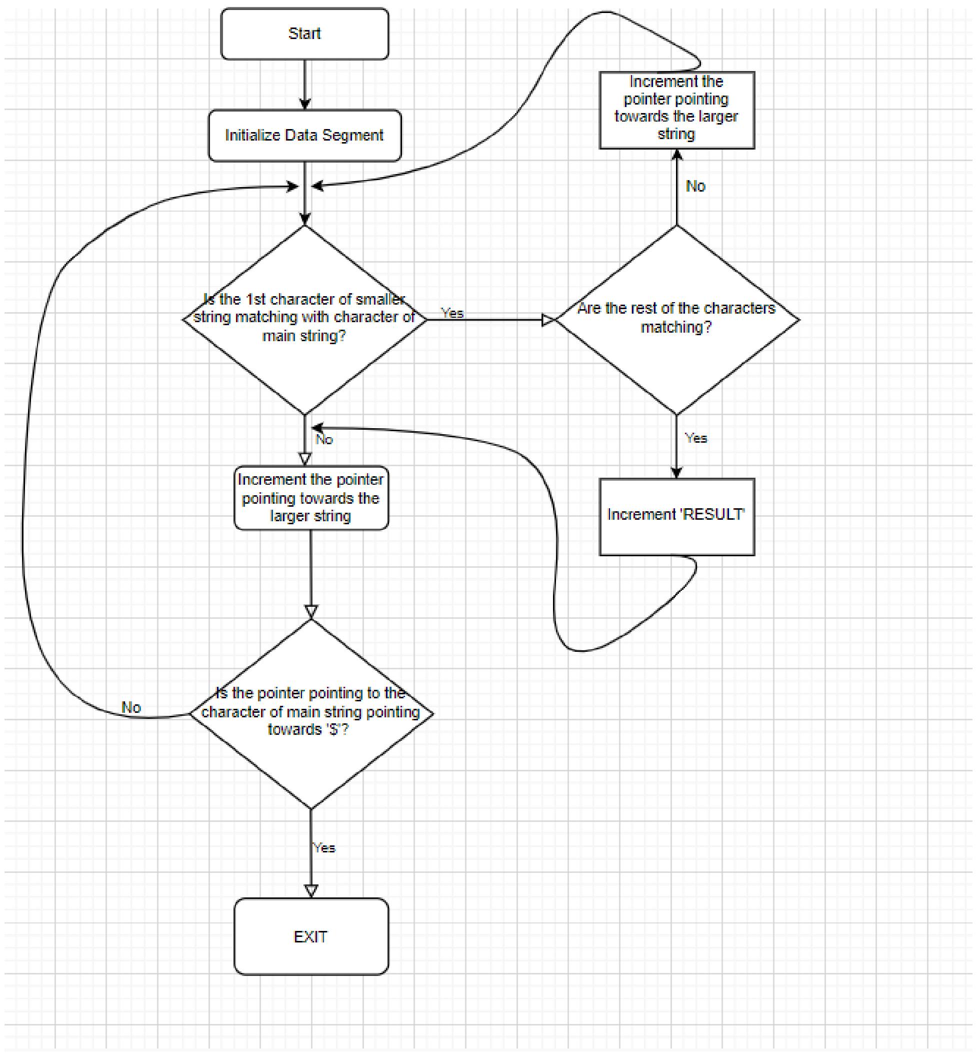
SOFTWARE

- EMU8086 emulator

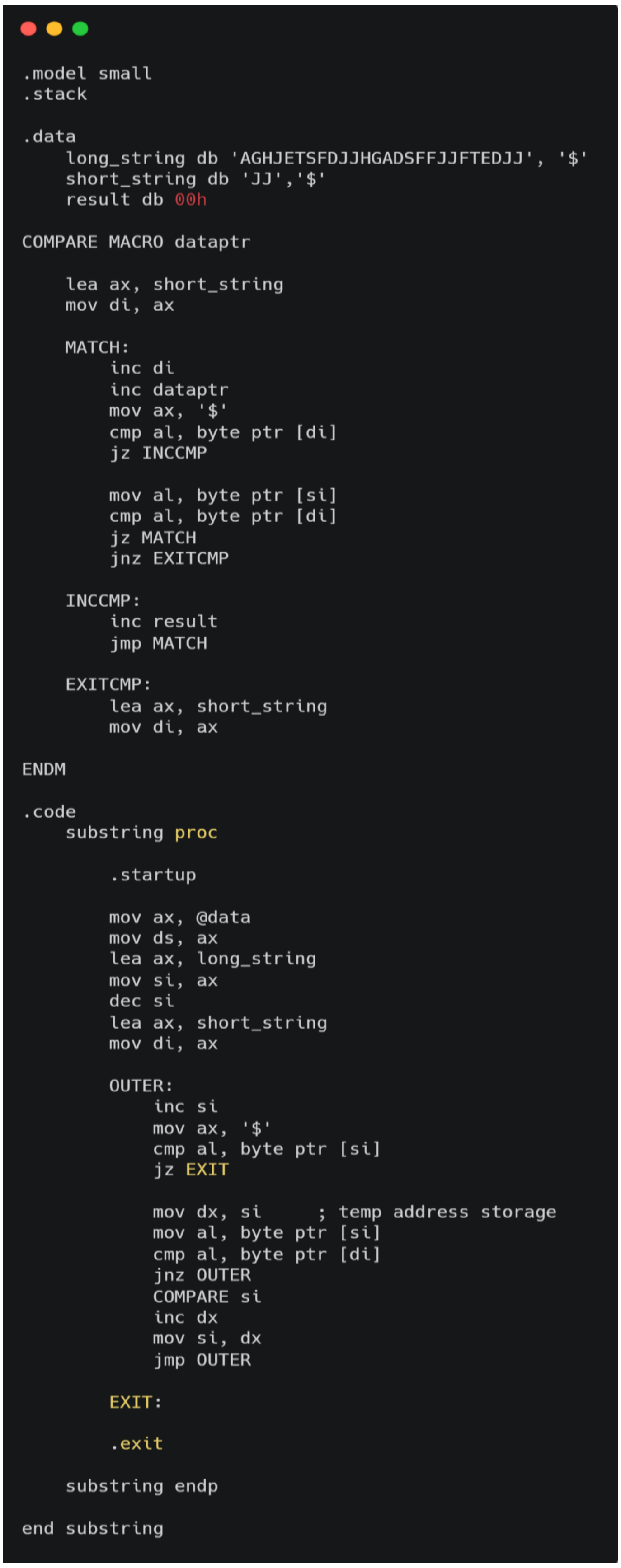
OVERVIEW

Here we first compare the 1st character of smaller string with every character of the larger string. If me get a match then the rest of the smaller string is compared.

FLOWCHART

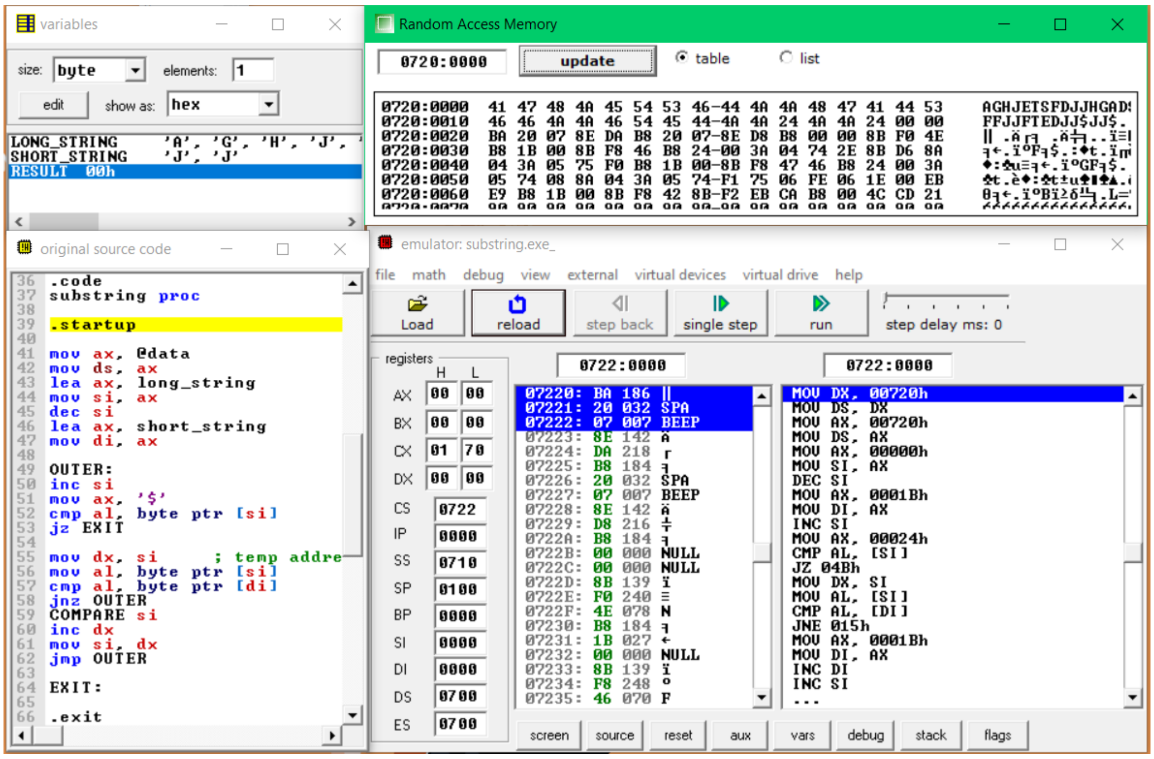

Start
Initialize Data Segment
Is the 1st character of smaller string matching with character of main string?
Yes
Are the rest of the characters matching?
No
Increment the pointer pointing towards the larger string
Yes
Increment 'RESULT'
No
Increment the pointer pointing towards the larger string
Is the pointer pointing to the character of main string pointing towards '$'?
No
Yes
EXIT

- **CODE: -**



```
.model small
.stack

.data
    long_string db 'AGHJETSFDJJHGADSFFJJFTEDJJ', '$'
    short_string db 'JJ','$'
    result db 00h

COMPARE MACRO dataptr

    lea ax, short_string
    mov di, ax

    MATCH:
        inc di
        inc dataptr
        mov ax, '$'
        cmp al, byte ptr [di]
        jz INCCMP

        mov al, byte ptr [si]
        cmp al, byte ptr [di]
        jz MATCH
        jnz EXITCMP

    INCCMP:
        inc result
        jmp MATCH

    EXITCMP:
        lea ax, short_string
        mov di, ax

ENDM

.code
    substring proc

        .startup

        mov ax, @data
        mov ds, ax
        lea ax, long_string
        mov si, ax
        dec si
        lea ax, short_string
        mov di, ax

        OUTER:
            inc si
            mov ax, '$'
            cmp al, byte ptr [si]
            jz EXIT

            mov dx, si      ; temp address storage
            mov al, byte ptr [si]
            cmp al, byte ptr [di]
            jnz OUTER
            COMPARE si
            inc dx
            mov si, dx
            jmp OUTER

        EXIT:

        .exit

    substring endp

end substring
```

- **INPUT**


variables — size: byte, elements: 1, show as: hex

```
LONG_STRING   'A', 'G', 'H', 'J', '
SHORT_STRING  'J', 'J'
RESULT  00h
```

Random Access Memory — 0720:0000 — update — (•) table ( ) list

```
0720:0000  41 47 48 4A 45 54 53 46-44 4A 4A 48 47 41 44 53   AGHJETSFDJJHGAD$
0720:0010  46 46 4A 4A 46 54 45 44-4A 4A 24 4A 4A 24 00 00   FFJJFTEDJJ$JJ$.
0720:0020  BA 20 07 8E DA B8 20 07-8E D8 B8 00 00 8B F0 4E   ‖ .Äг┐ .Ä╪┐..ï≡N
0720:0030  B8 1B 00 8B F8 46 B8 24-00 3A 04 74 2E 8B D6 8A   ┐←.ï°F┐$.:♦t.ïπ
0720:0040  04 3A 05 75 F0 B8 1B 00-8B F8 47 46 B8 24 00 3A   ♦:♣u≡┐←.ï°GF┐$.
0720:0050  05 74 08 8A 04 3A 05 74-F1 75 06 FE 06 1E 00 EB   ♣t.è♦:♣t±u♠■♠▲.
0720:0060  E9 B8 1B 00 8B F8 42 8B-F2 EB CA B8 00 4C CD 21   Θ┐←.ï°Bï≥δ╩┐.L=!
```

original source code

```
.code
substring proc

.startup

mov ax, @data
mov ds, ax
lea ax, long_string
mov si, ax
dec si
lea ax, short_string
mov di, ax

OUTER:
inc si
mov ax, '$'
cmp al, byte ptr [si]
jz EXIT

mov dx, si       ; temp addre
mov al, byte ptr [si]
cmp al, byte ptr [di]
jnz OUTER
COMPARE si
inc dx
mov si, dx
jmp OUTER

EXIT:

.exit
```

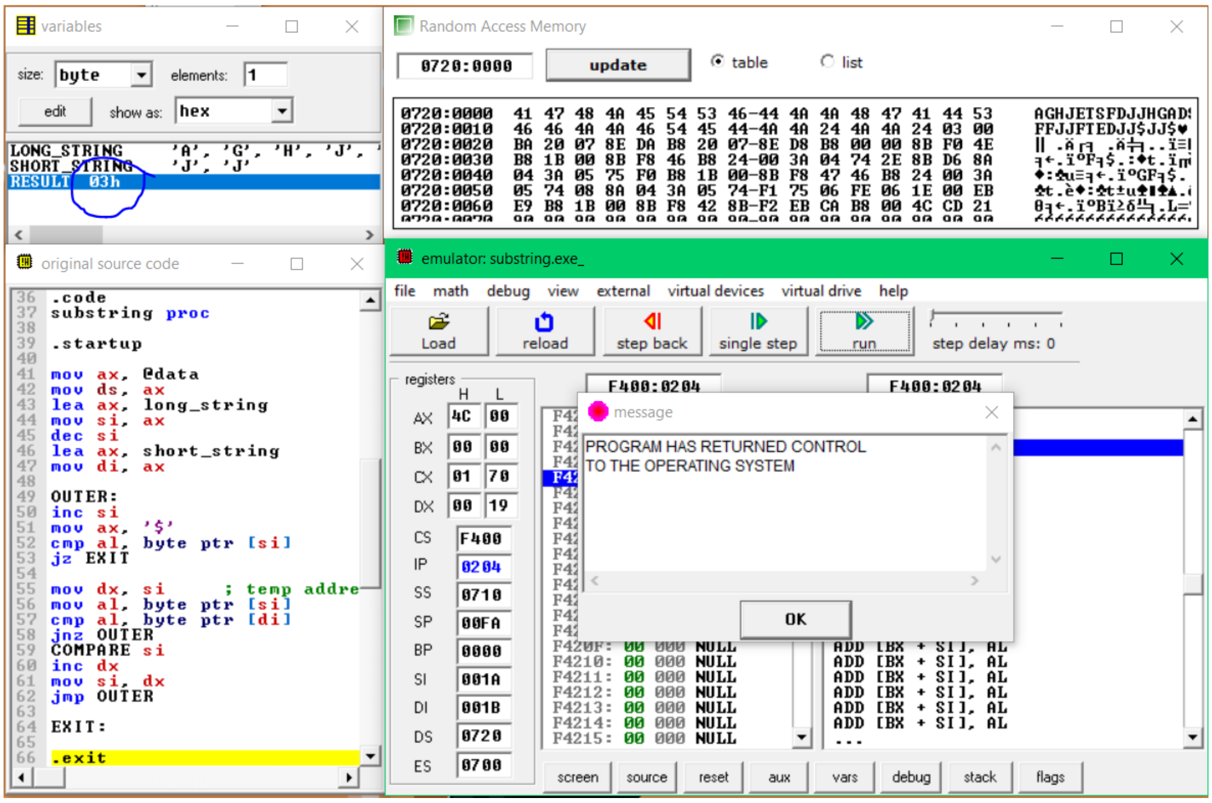
emulator: substring.exe_

registers

	H	L
AX	00	00
BX	00	00
CX	01	70
DX	00	00
CS	0722	
IP	0000	
SS	0710	
SP	0100	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
0722:0000                   0722:0000
07220: BA 186 ‖             MOV DX, 00720h
07221: 20 032 SPA           MOV DS, DX
07222: 07 007 BEEP          MOV AX, 00720h
07223: 8E 142 Ä             MOV DS, AX
07224: DA 218 ┌             MOV AX, 00000h
07225: B8 184 ┐             MOV SI, AX
07226: 20 032 SPA           DEC SI
07227: 07 007 BEEP          MOV AX, 0001Bh
07228: 8E 142 Ä             MOV DI, AX
07229: D8 216 ╪             INC SI
0722A: B8 184 ┐             MOV AX, 00024h
0722B: 00 000 NULL          CMP AL, [SI]
0722C: 00 000 NULL          JZ 04Bh
0722D: 8B 139 ï             MOV DX, SI
0722E: F0 240 ≡             MOV AL, [SI]
0722F: 4E 078 N             CMP AL, [DI]
07230: B8 184 ┐             JNE 015h
07231: 1B 027 ←             MOV AX, 0001Bh
07232: 00 000 NULL          MOV DI, AX
07233: 8B 139 ï             INC DI
07234: F8 248 °             INC SI
07235: 46 070 F             ...
```

- **OUTPUT**


variables — size: byte, elements: 1, show as: hex

```
LONG_STRING   'A', 'G', 'H', 'J', '
SHORT_STRING  'J', 'J'
RESULT  03h
```

Random Access Memory — 0720:0000 — update — (•) table ( ) list

```
0720:0000  41 47 48 4A 45 54 53 46-44 4A 4A 48 47 41 44 53   AGHJETSFDJJHGAD$
0720:0010  46 46 4A 4A 46 54 45 44-4A 4A 24 4A 4A 24 03 00   FFJJFTEDJJ$JJ$♥
0720:0020  BA 20 07 8E DA B8 20 07-8E D8 B8 00 00 8B F0 4E   ‖ .Äг┐ .Ä╪┐..ï≡N
0720:0030  B8 1B 00 8B F8 46 B8 24-00 3A 04 74 2E 8B D6 8A   ┐←.ï°F┐$.:♦t.ïπ
0720:0040  04 3A 05 75 F0 B8 1B 00-8B F8 47 46 B8 24 00 3A   ♦:♣u≡┐←.ï°GF┐$.
0720:0050  05 74 08 8A 04 3A 05 74-F1 75 06 FE 06 1E 00 EB   ♣t.è♦:♣t±u♠■♠▲.
0720:0060  E9 B8 1B 00 8B F8 42 8B-F2 EB CA B8 00 4C CD 21   Θ┐←.ï°Bï≥δ╩┐.L=!
```

original source code

```
.code
substring proc

.startup

mov ax, @data
mov ds, ax
lea ax, long_string
mov si, ax
dec si
lea ax, short_string
mov di, ax

OUTER:
inc si
mov ax, '$'
cmp al, byte ptr [si]
jz EXIT

mov dx, si       ; temp addre
mov al, byte ptr [si]
cmp al, byte ptr [di]
jnz OUTER
COMPARE si
inc dx
mov si, dx
jmp OUTER

EXIT:

.exit
```

emulator: substring.exe_

registers

	H	L
AX	4C	00
BX	00	00
CX	01	70
DX	00	19
CS	F400	
IP	0204	
SS	0710	
SP	00FA	
BP	0000	
SI	001A	
DI	001B	
DS	0720	
ES	0700	

message

```
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
```

OK

```
F400:0204                   F400:0204
F420F: 00 000 NULL          ADD [BX + SI], AL
F4210: 00 000 NULL          ADD [BX + SI], AL
F4211: 00 000 NULL          ADD [BX + SI], AL
F4212: 00 000 NULL          ADD [BX + SI], AL
F4213: 00 000 NULL          ADD [BX + SI], AL
F4214: 00 000 NULL          ADD [BX + SI], AL
F4215: 00 000 NULL          ...
```

CONCLUSION

In this program we have learnt how to work with strings in assembly language.