



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

CSE2006

Microprocessor and Interfacing

Lab Assignment 3

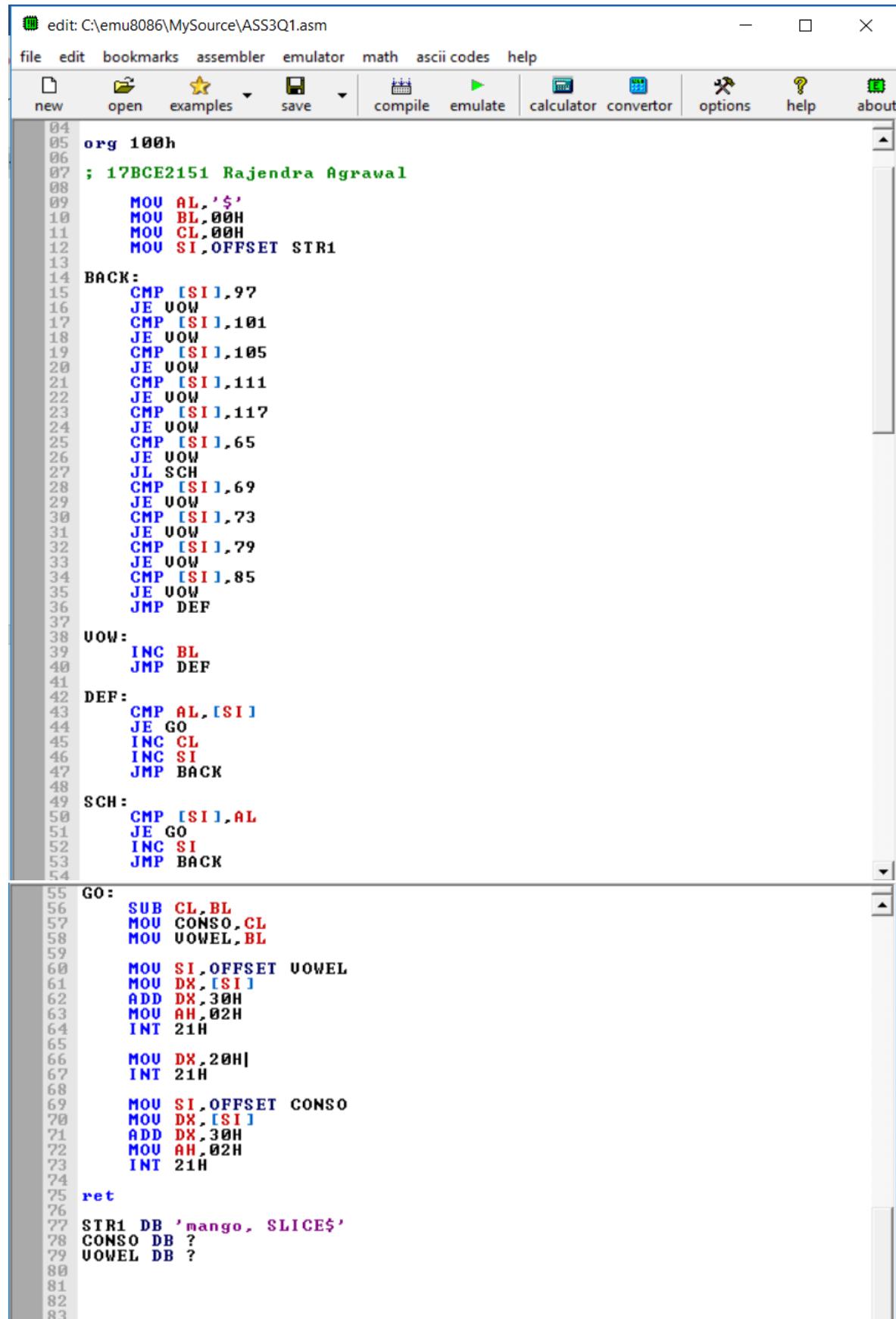
Slot: L39+L40

Faculty: Dr. Kumaravelu R

Name: Rajendra Agrawal
Reg. No: 17BCE2151

Question 1. Write an 8086 Assembly Language Program to count
a) vowels b) consonants in a given string.

Code:



The screenshot shows the Emu8086 assembly editor interface with the file C:\emu8086\MySource\ASS3Q1.asm open. The menu bar includes file, edit, bookmarks, assembler, emulator, math, ascii codes, and help. The toolbar includes new, open, examples, save, compile, emulate, calculator, converter, options, help, and about. The assembly code is as follows:

```
04 org 100h
05 ; 17BCE2151 Rajendra Agrawal
06
07 MOU AL,'$'
08 MOU BL,00H
09 MOU CL,00H
10 MOU SI,OFFSET STR1
11
12 BACK:
13     CMP [SI],97
14     JE UOW
15     CMP [SI],101
16     JE UOW
17     CMP [SI],105
18     JE UOW
19     CMP [SI],111
20     JE UOW
21     CMP [SI],117
22     JE UOW
23     CMP [SI],65
24     JE UOW
25     JL SCH
26     CMP [SI],69
27     JE UOW
28     CMP [SI],73
29     JE UOW
30     CMP [SI],79
31     JE UOW
32     CMP [SI],85
33     JE UOW
34     JMP DEF
35
36 DEF:
37     CMP AL,[SI]
38     JE GO
39     INC CL
40     INC SI
41     JMP BACK
42
43 SCH:
44     CMP [SI],AL
45     JE GO
46     INC SI
47     JMP BACK
48
49 GO:
50     SUB CL,BL
51     MOU CONSO,CL
52     MOU UOWEL,BL
53
54     MOU SI,OFFSET UOWEL
55     MOU DX,[SI]
56     ADD DX,30H
57     MOU AH,02H
58     INT 21H
59
60     MOU DX,20H
61     INT 21H
62
63     MOU SI,OFFSET CONSO
64     MOU DX,[SI]
65     ADD DX,30H
66     MOU AH,02H
67     INT 21H
68
69     ret
70
71 STR1 DB 'mango, SLICE$'
72 CONSO DB ?
73 UOWEL DB ?
74
75
76
77
78
79
80
81
82
83
```

Here we compare the character with ascii codes for all vowels, in small case as well as upper case. We also check if character is special character. So we count number of vowels in BL and total number of letters in CL. Later, we find the number of vowels BL and number of consonants as difference of CL and BL.

Step by Step Execution:

1. Initialize all the registers. AL with ascii of \$. BL and CL to be 0.

	H	L
AX	00	24
BX	00	00
CX	00	00
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	017B	
DI	0000	
DS	0700	
ES	0700	

```

09 MOU AL, '$'
10 MOU BL, 00H
11 MOU CL, 00H
12 MOU SI, OFFSET STR1
13
14 BACK:
15 CMP [SI], 97
16 JE UOW
17 CMP [SI], 101
18 JE UOW
19 CMP [SI], 105
20 JE UOW
21 CMP [SI], 111
22 JE UOW
23 CMP [SI], 117
24 JE UOW
25 CMP [SI], 65
26 JE UOW
27 IT EQU

```

2. “mango, SLICE” since letter ‘m’ is under scan BL will be 0 and CL will be incremented to 1.

	H	L
AX	00	24
BX	00	00
CX	00	01
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	017C	
DI	0000	
DS	0700	
ES	0700	

```

15 CMP [SI], 97
16 JE UOW
17 CMP [SI], 101
18 JE UOW
19 CMP [SI], 105
20 JE UOW
21 CMP [SI], 111
22 JE UOW
23 CMP [SI], 117
24 JE UOW
25 CMP [SI], 65
26 JE UOW
27 JL SCH
28 CMP [SI], 69
29 JE UOW
30 CMP [SI], 73
31 JE UOW
32 CMP [SI], 79
33 IT EOU

```

3. “mango, SLICE” since vowel ‘a’ is under scan BL will be incremented to 1 and CL will also be incremented to 2.

The screenshot shows a debugger interface with two panes. The left pane displays assembly code and memory dump windows, while the right pane shows the original source code. The assembly code window highlights the instruction at address 0700:0109, which is `CMP b,[SI], 061h`. The registers window shows various CPU registers with their values. The source code window on the right shows the C code for the string "mango, SLICE". The character 'a' is highlighted in blue, indicating it is the current character being processed by the scan loop.

```

    emulator: ASS3Q1.com
    file math debug view external virtual devices virtual drive help
    Load reload step back single step run step delay ms: 0
    registers - H L
    AX 00 24
    BX 00 01
    CX 00 02
    DX 00 00
    CS 0700
    IP 0109
    SS 0700
    SP FFFE
    BP 0000
    SI 017D
    DI 0000
    DS 0700
    ES 0700
    0700:0109 CMP b,[SI], 061h
    0700:010A JZ 013Fh
    0700:010B CMP b,[SI], 065h
    0700:010C JZ 013Fh
    0700:010D CMP b,[SI], 069h
    0700:010E JZ 013Fh
    0700:010F CMP b,[SI], 06Fh
    0700:0110 JZ 013Fh
    0700:0111 CMP b,[SI], 075h
    0700:0112 JZ 013Fh
    0700:0113 CMP b,[SI], 041h
    0700:0114 JZ 013Fh
    0700:0115 JL 014Ch
    0700:0116 CMP b,[SI], 045h
    0700:0117 JZ 013Fh
    0700:0118 CMP b,[SI], 049h
    0700:0119 JZ 013Fh
    0700:011A CMP b,[SI], 04Fh
    0700:011B JZ 013Fh
    0700:011C CMP b,[SI], 055h
    0700:011D JZ 013Fh
    0700:011E ...
    original source code...
    15 CMP [SI],97
    16 JE UOW
    17 CMP [SI],101
    18 JE UOW
    19 CMP [SI],105
    20 JE UOW
    21 CMP [SI],111
    22 JE UOW
    23 CMP [SI],117
    24 JE UOW
    25 CMP [SI],65
    26 JE UOW
    27 JL SCH
    28 CMP [SI],69
    29 JE UOW
    30 CMP [SI],73
    31 JE UOW
    32 CMP [SI],79
    33 JE UOW
  
```

4. “mango, SLICE” since letter ‘n’ is under scan BL will be 1 and CL will be incremented to 3.

This screenshot is identical to the one above, showing the debugger interface with assembly code, registers, and source code. The assembly code window highlights the instruction at address 0700:0109, which is `CMP b,[SI], 061h`. The registers window shows the same register values. The source code window on the right shows the C code for the string "mango, SLICE". The character 'n' is highlighted in blue, indicating it is the current character being processed by the scan loop.

```

    emulator: ASS3Q1.com
    file math debug view external virtual devices virtual drive help
    Load reload step back single step run step delay ms: 0
    registers - H L
    AX 00 24
    BX 00 01
    CX 00 03
    DX 00 00
    CS 0700
    IP 0109
    SS 0700
    SP FFFE
    BP 0000
    SI 017E
    DI 0000
    DS 0700
    ES 0700
    0700:0109 CMP b,[SI], 061h
    0700:010A JZ 013Fh
    0700:010B CMP b,[SI], 065h
    0700:010C JZ 013Fh
    0700:010D CMP b,[SI], 069h
    0700:010E JZ 013Fh
    0700:010F CMP b,[SI], 06Fh
    0700:0110 JZ 013Fh
    0700:0111 CMP b,[SI], 075h
    0700:0112 JZ 013Fh
    0700:0113 CMP b,[SI], 041h
    0700:0114 JZ 013Fh
    0700:0115 JL 014Ch
    0700:0116 CMP b,[SI], 045h
    0700:0117 JZ 013Fh
    0700:0118 CMP b,[SI], 049h
    0700:0119 JZ 013Fh
    0700:011A CMP b,[SI], 04Fh
    0700:011B JZ 013Fh
    0700:011C CMP b,[SI], 055h
    0700:011D JZ 013Fh
    0700:011E ...
    original source code...
    15 CMP [SI],97
    16 JE UOW
    17 CMP [SI],101
    18 JE UOW
    19 CMP [SI],105
    20 JE UOW
    21 CMP [SI],111
    22 JE UOW
    23 CMP [SI],117
    24 JE UOW
    25 CMP [SI],65
    26 JE UOW
    27 JL SCH
    28 CMP [SI],69
    29 JE UOW
    30 CMP [SI],73
    31 JE UOW
    32 CMP [SI],79
    33 JE UOW
  
```

5. “mango, SLICE” since letter ‘g’ is under scan BL will be 1 and CL will be incremented to 4.

The screenshot shows a debugger interface with two main panes. The left pane displays assembly code and registers for the word "mango". The right pane shows the original source code. The assembly code pane highlights the instruction at address 0700:0109, which is `CMP b,[SI], 061h`. The registers pane shows the following values:

	H	L
AX	00	24
BX	00	01
CX	00	04
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	017F	
DI	0000	
DS	0700	
ES	0700	

The source code pane shows the following assembly code:

```

15 CMP [SI],97
16 JE UOW
17 CMP [SI],101
18 JE UOW
19 CMP [SI],105
20 JE UOW
21 CMP [SI],111
22 JE UOW
23 CMP [SI],117
24 JE UOW
25 CMP [SI],65
26 JE UOW
27 JL SCH
28 CMP [SI],69
29 JE UOW
30 CMP [SI],73
31 JE UOW
32 CMP [SI],79
33 JE UOW

```

6. “mang**o**, SLICE” since vowel ‘o’ is under scan BL will be incremented to 2 and CL will also be incremented to 5.

The screenshot shows a debugger interface with two main panes. The left pane displays assembly code and registers for the word "mang'o". The right pane shows the original source code. The assembly code pane highlights the instruction at address 0700:0109, which is `CMP b,[SI], 061h`. The registers pane shows the following values:

	H	L
AX	00	24
BX	00	02
CX	00	05
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	0180	
DI	0000	
DS	0700	
ES	0700	

The source code pane shows the following assembly code:

```

15 CMP [SI],97
16 JE UOW
17 CMP [SI],101
18 JE UOW
19 CMP [SI],105
20 JE UOW
21 CMP [SI],111
22 JE UOW
23 CMP [SI],117
24 JE UOW
25 CMP [SI],65
26 JE UOW
27 JL SCH
28 CMP [SI],69
29 JE UOW
30 CMP [SI],73
31 JE UOW
32 CMP [SI],79
33 JE UOW

```

7. “mango, SLICE” since special character ‘,’ is under scan BL and CL both will be same.

The screenshot shows a debugger interface with two windows. The left window displays assembly code and registers for address 0700:0109. The right window shows the original source code. The assembly code includes instructions like CMP [SI], 97, JE UOW, and CMP [SI], 101. The source code shows the string "mango, SLICE". The registers window shows various CPU registers with their values.

```

    emulator: ASS3Q1.com
    file math debug view external virtual devices virtual drive help
    Load reload step back single step run step delay ms: 0
    registers H L
    AX 00 24
    BX 00 02
    CX 00 05
    DX 00 00
    CS 0700
    IP 0109
    SS 0700
    SP FFFE
    BP 0000
    SI 0181
    DI 0000
    DS 0700
    ES 0700
    0700:0109
    0700:0109
    15 CMP [SI], 97
    16 JE UOW
    17 CMP [SI], 101
    18 JE UOW
    19 CMP [SI], 105
    20 JE UOW
    21 CMP [SI], 111
    22 JE UOW
    23 CMP [SI], 117
    24 JE UOW
    25 CMP [SI], 65
    26 JE UOW
    27 JL SCH
    28 CMP [SI], 69
    29 JE UOW
    30 CMP [SI], 73
    31 JE UOW
    32 CMP [SI], 79
    33 JE UOW
    07109: 80 128 C
    0710A: 3C 060 <
    0710B: 61 097 a
    0710C: 74 116 t
    0710D: 31 049 i
    0710E: 80 128 C
    0710F: 3C 060 <
    07110: 65 101 e
    07111: 74 116 t
    07112: 2C 044 ,
    07113: 80 128 C
    07114: 3C 060 <
    07115: 69 105 i
    07116: 74 116 t
    07117: 27 039 ,
    07118: 80 128 C
    07119: 3C 060 <
    0711A: 6F 111 o
    0711B: 74 116 t
    0711C: 22 034 "
    0711D: 80 128 C
    0711E: 3C 060 <
    ...
  
```

8. “mango, SLICE” since space character ‘ ’ is under scan BL and CL both will be same.

The screenshot shows a debugger interface with two windows. The left window displays assembly code and registers for address 0700:0109. The right window shows the original source code. The assembly code includes instructions like CMP [SI], 97, JE UOW, and CMP [SI], 101. The source code shows the string "mango, SLICE". The registers window shows various CPU registers with their values.

```

    emulator: ASS3Q1.com
    file math debug view external virtual devices virtual drive help
    Load reload step back single step run step delay ms: 0
    registers H L
    AX 00 24
    BX 00 02
    CX 00 05
    DX 00 00
    CS 0700
    IP 0109
    SS 0700
    SP FFFE
    BP 0000
    SI 0182
    DI 0000
    DS 0700
    ES 0700
    0700:0109
    0700:0109
    15 CMP [SI], 97
    16 JE UOW
    17 CMP [SI], 101
    18 JE UOW
    19 CMP [SI], 105
    20 JE UOW
    21 CMP [SI], 111
    22 JE UOW
    23 CMP [SI], 117
    24 JE UOW
    25 CMP [SI], 65
    26 JE UOW
    27 JL SCH
    28 CMP [SI], 69
    29 JE UOW
    30 CMP [SI], 73
    31 JE UOW
    32 CMP [SI], 79
    33 JE UOW
    07109: 80 128 C
    0710A: 3C 060 <
    0710B: 61 097 a
    0710C: 74 116 t
    0710D: 31 049 i
    0710E: 80 128 C
    0710F: 3C 060 <
    07110: 65 101 e
    07111: 74 116 t
    07112: 2C 044 ,
    07113: 80 128 C
    07114: 3C 060 <
    07115: 69 105 i
    07116: 74 116 t
    07117: 27 039 ,
    07118: 80 128 C
    07119: 3C 060 <
    0711A: 6F 111 o
    0711B: 74 116 t
    0711C: 22 034 "
    0711D: 80 128 C
    0711E: 3C 060 <
    ...
  
```

9. “mango, SLICE” since letter ‘S’ is under scan BL will be 2 and CL will be incremented to 6.

```

    emulator: ASS3Q1.com
    file math debug view external virtual devices virtual drive help
    Load reload step back single step run step delay ms: 0
    registers H L
    AX 00 24
    BX 00 02
    CX 00 06
    DX 00 00
    CS 0700
    IP 0109
    SS 0700
    SP FFFE
    BP 0000
    SI 0183
    DI 0000
    DS 0700
    ES 0700
    0700:0109
    0700:0109
    15 CMP [SI],97
    16 JE UOW
    17 CMP [SI],101
    18 JE UOW
    19 CMP [SI],105
    20 JE UOW
    21 CMP [SI],111
    22 JE UOW
    23 CMP [SI],117
    24 JE UOW
    25 CMP [SI],65
    26 JE UOW
    27 JL SCH
    28 CMP [SI],69
    29 JE UOW
    30 CMP [SI],73
    31 JE UOW
    32 CMP [SI],79
    33 JE UOW
    07109: 80 128 C
    0710A: 3C 060 <
    0710B: 61 097 a
    0710C: 74 116 t
    0710D: 31 049 i
    0710E: 80 128 C
    0710F: 3C 060 <
    07110: 65 101 e
    07111: 74 116 t
    07112: 2C 044 ,
    07113: 80 128 C
    07114: 3C 060 <
    07115: 69 105 i
    07116: 74 116 t
    07117: 27 039 ,
    07118: 80 128 C
    07119: 3C 060 <
    0711A: 6F 111 o
    0711B: 74 116 t
    0711C: 22 034 "
    0711D: 80 128 C
    0711E: 3C 060 <
    ...
    screen source reset aux vars debug stack flags
  
```

10.“mango, SLICE” since letter ‘L’ is under scan BL will be 2 and CL will be incremented to 7.

```

    emulator: ASS3Q1.com
    file math debug view external virtual devices virtual drive help
    Load reload step back single step run step delay ms: 0
    registers H L
    AX 00 24
    BX 00 02
    CX 00 07
    DX 00 00
    CS 0700
    IP 0109
    SS 0700
    SP FFFE
    BP 0000
    SI 0184
    DI 0000
    DS 0700
    ES 0700
    0700:0109
    0700:0109
    15 CMP [SI],97
    16 JE UOW
    17 CMP [SI],101
    18 JE UOW
    19 CMP [SI],105
    20 JE UOW
    21 CMP [SI],111
    22 JE UOW
    23 CMP [SI],117
    24 JE UOW
    25 CMP [SI],65
    26 JE UOW
    27 JL SCH
    28 CMP [SI],69
    29 JE UOW
    30 CMP [SI],73
    31 JE UOW
    32 CMP [SI],79
    33 JE UOW
    07109: 80 128 C
    0710A: 3C 060 <
    0710B: 61 097 a
    0710C: 74 116 t
    0710D: 31 049 i
    0710E: 80 128 C
    0710F: 3C 060 <
    07110: 65 101 e
    07111: 74 116 t
    07112: 2C 044 ,
    07113: 80 128 C
    07114: 3C 060 <
    07115: 69 105 i
    07116: 74 116 t
    07117: 27 039 ,
    07118: 80 128 C
    07119: 3C 060 <
    0711A: 6F 111 o
    0711B: 74 116 t
    0711C: 22 034 "
    0711D: 80 128 C
    0711E: 3C 060 <
    ...
    screen source reset aux vars debug stack flags
  
```

11.“mango, SLICE” since vowel ‘I’ is under scan BL will be incremented to 3 and CL will also be incremented to 8.

Registers	H	L
AX	00	24
BX	00	03
CX	00	08
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	0185	
DI	0000	
DS	0700	
ES	0700	

```

0700:0109 CMP b.[SI1], 061h
0700:010A JZ 013Fh
0700:010B CMP b.[SI1], 065h
0700:010C JZ 013Fh
0700:010D CMP b.[SI1], 069h
0700:010E JZ 013Fh
0700:010F CMP b.[SI1], 06Fh
0700:0110 JZ 013Fh
0700:0111 CMP b.[SI1], 075h
0700:0112 JZ 013Fh
0700:0113 CMP b.[SI1], 041h
0700:0114 JZ 013Fh
0700:0115 JL 014Ch
0700:0116 CMP b.[SI1], 045h
0700:0117 JZ 013Fh
0700:0118 CMP b.[SI1], 049h
0700:0119 JZ 013Fh
0700:011A CMP b.[SI1], 04Fh
0700:011B JZ 013Fh
0700:011C CMP b.[SI1], 055h
0700:011D JZ 013Fh
0700:011E ...

```

12.“mango, SLICE” since letter ‘C’ is under scan BL will be 3 and CL will be incremented to 9.

Registers	H	L
AX	00	24
BX	00	03
CX	00	09
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	0186	
DI	0000	
DS	0700	
ES	0700	

```

0700:0109 CMP b.[SI1], 061h
0700:010A JZ 013Fh
0700:010B CMP b.[SI1], 065h
0700:010C JZ 013Fh
0700:010D CMP b.[SI1], 069h
0700:010E JZ 013Fh
0700:010F CMP b.[SI1], 06Fh
0700:0110 CMP b.[SI1], 085h
0700:0111 JZ 013Fh
0700:0112 CMP b.[SI1], 075h
0700:0113 JZ 013Fh
0700:0114 CMP b.[SI1], 041h
0700:0115 JZ 013Fh
0700:0116 JL 014Ch
0700:0117 CMP b.[SI1], 045h
0700:0118 JZ 013Fh
0700:0119 CMP b.[SI1], 049h
0700:011A JZ 013Fh
0700:011B CMP b.[SI1], 04Fh
0700:011C JZ 013Fh
0700:011D CMP b.[SI1], 055h
0700:011E ...

```

13.“mango, SLICE” since vowel ‘o’ is under scan BL will be incremented to 4 and CL will also be incremented to 10.

The screenshot shows the emulator interface for ASS3Q1.com. On the left, the 'Registers' window displays memory addresses 0700:0109 and 0700:0109. The AX register has a value of 00 24. The BX register has a value of 00 04. The CX register has a value of 00 0A. The DX register has a value of 00 00. The CS register has a value of 0700. The IP register has a value of 0109. The SS register has a value of 0700. The SP register has a value of FFFE. The BP register has a value of 0000. The SI register has a value of 0187. The DI register has a value of 0000. The DS register has a value of 0700. The ES register has a value of 0700. The right side of the interface shows the assembly code:

```

15 CMP [SI], 97
16 JE UOW
17 CMP [SI], 101
18 JE UOW
19 CMP [SI], 105
20 JE UOW
21 CMP [SI], 111
22 JE UOW
23 CMP [SI], 117
24 JE UOW
25 CMP [SI], 65
26 JE UOW
27 JL SCH
28 CMP [SI], 69
29 JE UOW
30 CMP [SI], 73
31 JE UOW
32 CMP [SI], 79
33 JE UOW

```

14. Now CL is subtracted from BL and result is stored in CL. Later Both are printed on screen with Number of vowels (4) first then a space and then number of consonants (6) as 4 6.

The screenshot shows the emulator interface for ASS3Q1.com. On the left, the 'Registers' window displays memory addresses F400:0154 and F400:0154. The AX register has a value of 02 36. The BX register has a value of 00 04. The CX register has a value of 00 06. The DX register has a value of 04 36. The CS register has a value of F400. The IP register has a value of 0154. The SS register has a value of 0700. The SP register has a value of FFFA. The BP register has a value of 0000. The SI register has a value of 0188. The DI register has a value of 0000. The DS register has a value of 0700. The ES register has a value of 0700. The right side of the interface shows the assembly code:

```

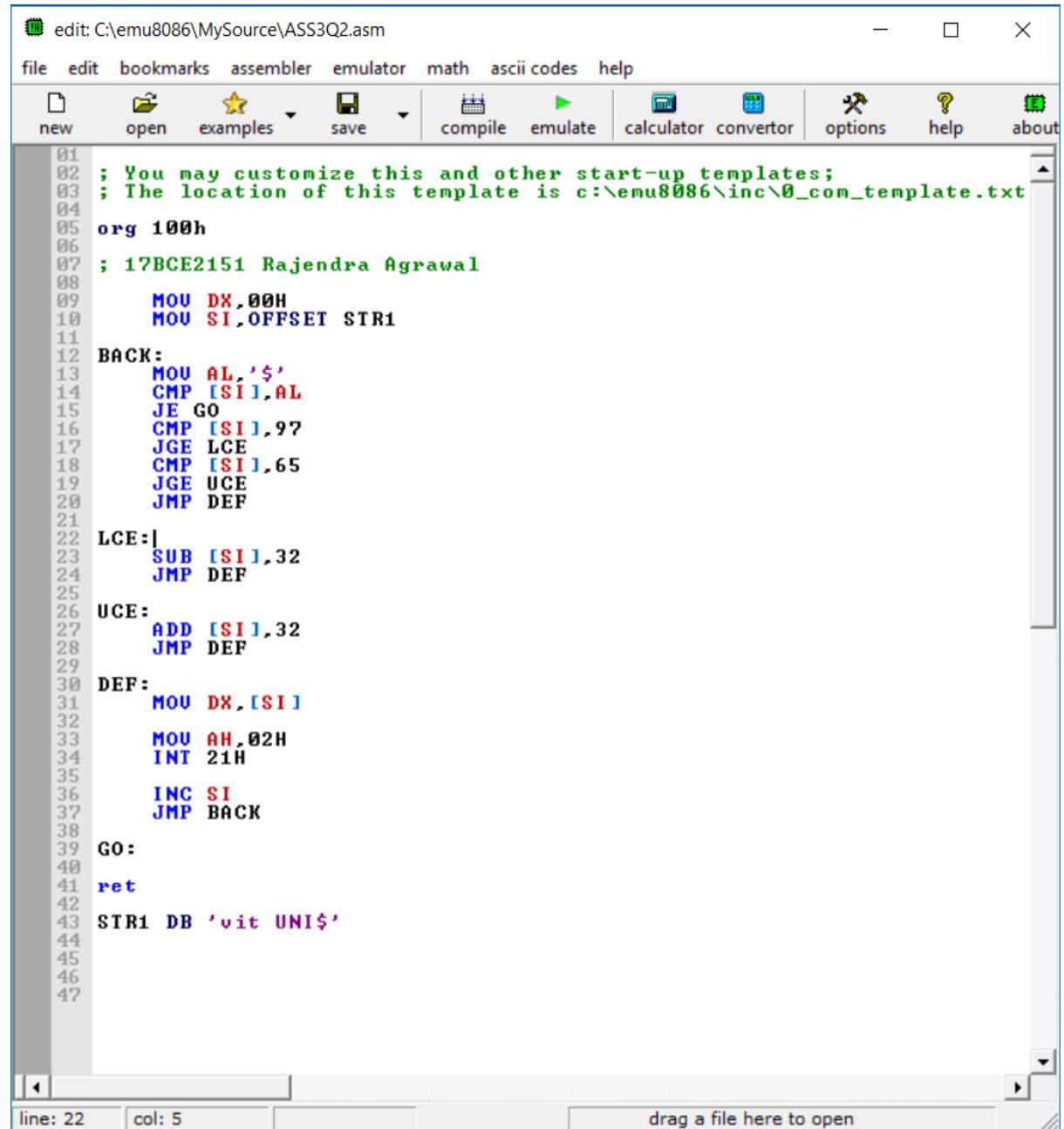
58 MOU UOWEL, BL
59
60 MOU SI, OFFSET UOWEL
61 MOU DX, [SI]
62 ADD DX, 30H
63 MOU AH, 02H
64 INT 21H
65
66 MOU DX, 20H
67 INT 21H
68
69 MOU SI, OFFSET CONSO
70 MOU DX, [SI]
71 ADD DX, 30H
72 MOU AH, 02H
73 INT 21H
74
75 ret

```

A message box at the bottom right says "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM". Below the message box is an "OK" button. At the bottom left, there is a terminal window titled "SCR emulator screen (56x11 chars)" showing the output "4 6".

Question 2. Write an 8086 Assembly Language Program to change Lower to Upper Case and vice versa in a given string?

Code:



The screenshot shows the emu8086 assembly editor interface. The title bar reads "edit: C:\emu8086\MySource\ASS3Q2.asm". The menu bar includes file, edit, bookmarks, assembler, emulator, math, ascii codes, help. The toolbar includes new, open, examples, save, compile, emulate, calculator, converter, options, help, about. The code window displays the following assembly code:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ; 17BCE2151 Rajendra Agrawal
07
08 MOU DX,00H
09 MOU SI,OFFSET STR1
10
11 BACK:
12     MOU AL,'$'
13     CMP [SI],AL
14     JE GO
15     CMP [SI],97
16     JGE LCE
17     CMP [SI],65
18     JGE UCE
19     JMP DEF
20
21 LCE:
22     SUB [SI],32
23     JMP DEF
24
25 UCE:
26     ADD [SI],32
27     JMP DEF
28
29 DEF:
30     MOU DX,[SI]
31
32     MOU AH,02H
33     INT 21H
34
35     INC SI
36     JMP BACK
37
38 GO:
39
40 ret
41
42
43 STR1 DB 'vit UNI$'
44
45
46
47
```

The status bar at the bottom shows "line: 22 col: 5" and "drag a file here to open".

Here we analyse the ASCII value of letter under scan and then branch into lower case (LCE) and decrease ASCII by 32 or into upper case (UCE) and increase ASCII by 32 or into default case (includes special character also). In DEF, we also print the value of [SI] or changed character.

Step By Step Execution:

1. ‘vit UNI’ since letter ‘v’ under scan is lower case letter, it’s ASCII value will be decreased by 32 and upper case ‘V’ will be printed.

The screenshot shows a debugger interface with three main windows:

- Registers Window:** Shows CPU register values. AX: 02 56, BX: 00 00, CX: 00 34, DX: 69 56, CS: 0700, IP: 0186, SS: 0700, SP: FFFE, BP: 0000, SI: 012D, DI: 0000, DS: 0700, ES: 0700.
- Memory Dump Window:** Displays memory starting at address 0700:0106. The first byte is 00 (ASCII 'V').
- Assembly Code Window:** Shows the assembly code being executed. The code includes instructions like MOU AL, 024h, CMP [SI], AL, JZ 012Bh, and various loops and jumps. The original source code is also visible on the right side of the window.

The assembly code is as follows:

```
13 MOU AL, '$'
14 CMP [SI], AL
15 JE GO
16 CMP [SI], 97
17 JGE LCE
18 CMP [SI], 65
19 JGE UCE
20 JMP DEF
21
22 LCE:
23 SUB [SI], 32
24 JMP DEF
25
26 UCE:
27 ADD [SI], 32
28 JMP DEF
29
30 DEF:
31 MOU DX, [SI]
32
33 MOU AH, 02H
34 INT 21H
35
36 INC SI
37 JMP BACK
38
39 GO:
40
41 ret
42
43 STR1 DB 'vit UNI$'
44
45
46
47
48
49
```

2. ‘vit UNI’ since letter ‘i’ under scan is lower case letter, it’s ASCII value will be decreased by 32 and upper case ‘I’ will be printed.

The screenshot shows a debugger interface with several windows:

- Registers Window:** Shows CPU register values. AX: 02 49, BX: 00 00, CX: 00 34, DX: 74 49, CS: 0700, IP: 0106, SS: 0700, SP: FFFE, BP: 0000, SI: 012E, DI: 0000, DS: 0700, ES: 0700.
- Memory Dump Window:** Displays memory starting at address 0700:0106. The first byte is 00 (ASCII ' '), followed by 176 (ASCII 'v'), 036 (ASCII 'i'), 056 (ASCII 'N'), 060 (ASCII 'U'), 044 (ASCII 'N'), 032 (ASCII 'I'), and 235 (ASCII ' '). Below the dump are assembly instructions and their addresses.
- Assembly Window:** Shows the assembly code being executed. It includes labels like GO, LCE, UCE, and DEF, and various instructions such as MOU, CMP, JZ, JNL, ADD, SUB, INC, INT, and RET.
- Output Window:** Displays the character 'V' printed to the screen.

```

original source co...
13 MOU AL, '$'
14 CMP [SI], AL
15 JE GO
16 CMP [SI], 97
17 JGE LCE
18 CMP [SI], 65
19 JGE UCE
20 JMP DEF
21
22 LCE:
23 SUB [SI], 32
24 JMP DEF
25
26 UCE:
27 ADD [SI], 32
28 JMP DEF
29
30 DEF:
31 MOU DX, [SI]
32
33 MOU AH, 02H
34 INT 21H
35
36 INC SI
37 JMP BACK
38
39 GO:
40
41 ret
42
43 STR1 DB 'vit UNI$'
44
45
46
47
48
49

```

3. ‘vit UNI’ since letter ‘t’ under scan is lower case letter, it’s ASCII value will be decreased by 32 and upper case ‘T’ will be printed.

The screenshot shows a debugger interface with several windows:

- Registers Window:** Shows CPU registers (AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, ES) with their values and sizes (H/L).
- Stack Window:** Displays memory starting at address 0700:0106. It shows assembly instructions and their corresponding opcodes. The stack grows downwards, with the current instruction being MOU AL, [SI].
- Code Window:** Displays the original source code in assembly language. The code includes labels like GO, LCE, UCE, DEF, and RET, along with various memory operations and jumps.
- Terminal Window:** Shows the terminal output where the string "VIT" is displayed.
- Status Bar:** Shows the screen size (55x11 chars) and the current page (0/16).

```

original source code
13 MOU AL, '$'
14 CMP [SI], AL
15 JE GO
16 CMP [SI], 97
17 JGE LCE
18 CMP [SI], 65
19 JGE UCE
20 JMP DEF
21
22 LCE:
23 SUB [SI], 32
24 JMP DEF
25
26 UCE:
27 ADD [SI], 32
28 JMP DEF
29
30 DEF:
31 MOU DX, [SI]
32
33 MOU AH, 02H
34 INT 21H
35
36 INC SI
37 JMP BACK
38
39 GO:
40
41 ret
42
43 STR1 DB 'vit UNI$'
44
45
46
47
48
49

emulator: ASS3Q2.com
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers H L
AX 02 54
BX 00 00
CX 00 34
DX 20 54
CS 0700
IP 0106
SS 0700
SP FFFE
BP 0000
SI 012F
DI 0000
DS 0700
ES 0700
0700:0106
07106: B0 176
07107: 24 036
07108: 38 056
07109: 04 004
0710A: 74 116
0710B: 1F 031
0710C: 80 128
0710D: 3C 060
0710E: 61 097
0710F: 7D 125
07110: 07 007
07111: 80 128
07112: 3C 060
07113: 41 065
07114: 7D 125
07115: 07 007
07116: EB 235
07117: 0A 010
07118: 80 128
07119: 2C 044
0711A: 20 032
0711B: EB 235
0700:0106
MOU AL, 024h
CMP [SI], AL
JZ 012Bh
JNL 0118h
JNL 011Dh
JMP 0122h
SUB b,[SI], 020h
JMP 0122h
ADD b,[SI], 020h
JMP 0122h
MOU DX, [SI]
MOU AH, 02h
INT 021h
INC SI
JMP 0106h
...
screen source reset aux vars debug stack flags
50 emulator screen (55x11 chars)
VIT
clear screen change font 0/16

```

4. ‘vit UNI’ since space character ‘ ‘ is under scan no change will take place and it will be printed as it is.

The screenshot shows a debugger interface with several windows:

- Registers Window:** Shows CPU register values. AX: 02 20, BX: 00 00, CX: 00 34, DX: 55 20, CS: 0700, IP: 0106, SS: 0700, SP: FFFE, BP: 0000, SI: 0130, DI: 0000, DS: 0700, ES: 0700.
- Memory Dump Window:** Displays memory starting at address 07106. It shows a sequence of bytes: B0 176, 24 036 \$, 38 056 8, 04 004 ♦, 74 116 t, 1F 031 ▼, 80 128 C, 3C 060 <, 61 097 a, 7D 125 >, 07 007 BEEP, 80 128 C, 3C 060 <, 41 065 A, 7D 125 >, 07 007 BEEP, EB 235 6, 0A 010 NEWL, 80 128 C, 2C 044, 20 032 SPA, EB 235 6. The byte 24 036 \$ is highlighted.
- Assembly Window:** Shows the assembly code with line numbers 13 to 49. The code includes instructions like MOU AL, '\$', CMP [SI], AL, JE GO, CMP [SI], 97, JGE LCE, CMP [SI], 65, JGE UCE, JMP DEF, LCE:, SUB [SI], 32, JMP DEF, UCE:, ADD [SI], 32, JMP DEF, DEF:, MOU DX, [SI], MOU AH, 02H, INT 21H, INC SI, JMP BACK, GO:, ret, STR1 DB 'vit UNI\$'.
- Output Window:** Shows the emulator screen output: VIT.

5. ‘vit UNI’ since letter ‘U’ under scan is upper case letter, it’s ASCII value will be increased by 32 and lower case ‘u’ will be printed.

The screenshot shows the ASS3Q2 debugger interface with the following components:

- Registers pane:** Displays CPU register values. AX: 02 75, BX: 00 00, CX: 00 34, DX: 4E 75, CS: 0700, IP: 0186, SS: 0700, SP: FFFE, BP: 0000, SI: 0131, DI: 0000, DS: 0700, ES: 0700.
- Memory dump panes:** Two side-by-side panes showing memory starting at address 0700:0106. The left pane shows the actual memory contents, and the right pane shows the original source code.
- Source code pane:** Shows the assembly source code with labels and comments. Labels include L, S, GO, LCE, UCE, DEF, and BACK. The code includes instructions like MOU AL, '\$', CMP [SI], AL, JZ 012Bh, etc.
- Output pane:** Displays the text "VIT u" in a monospaced font.
- Status bar:** Shows "SCR emulator screen (55x11 chars)" and "0/16".

```

original source code
13 MOU AL, '$'
14 CMP [SI], AL
15 JE GO
16 CMP [SI], 97
17 JGE LCE
18 CMP [SI], 65
19 JGE UCE
20 JMP DEF
21
22 LCE:
23 SUB [SI], 32
24 JMP DEF
25
26 UCE:
27 ADD [SI], 32
28 JMP DEF
29
30 DEF:
31 MOU DX, [SI]
32
33 MOU AH, 02H
34 INT 21H
35
36 INC SI
37 JMP BACK
38
39 GO:
40
41 ret
42
43 STR1 DB 'vit UNI$'
44
45
46
47
48
49

```

6. ‘vit UNI’ since letter ‘N’ under scan is upper case letter, it’s ASCII value will be increased by 32 and lower case ‘n’ will be printed.

The screenshot shows a debugger interface with several windows:

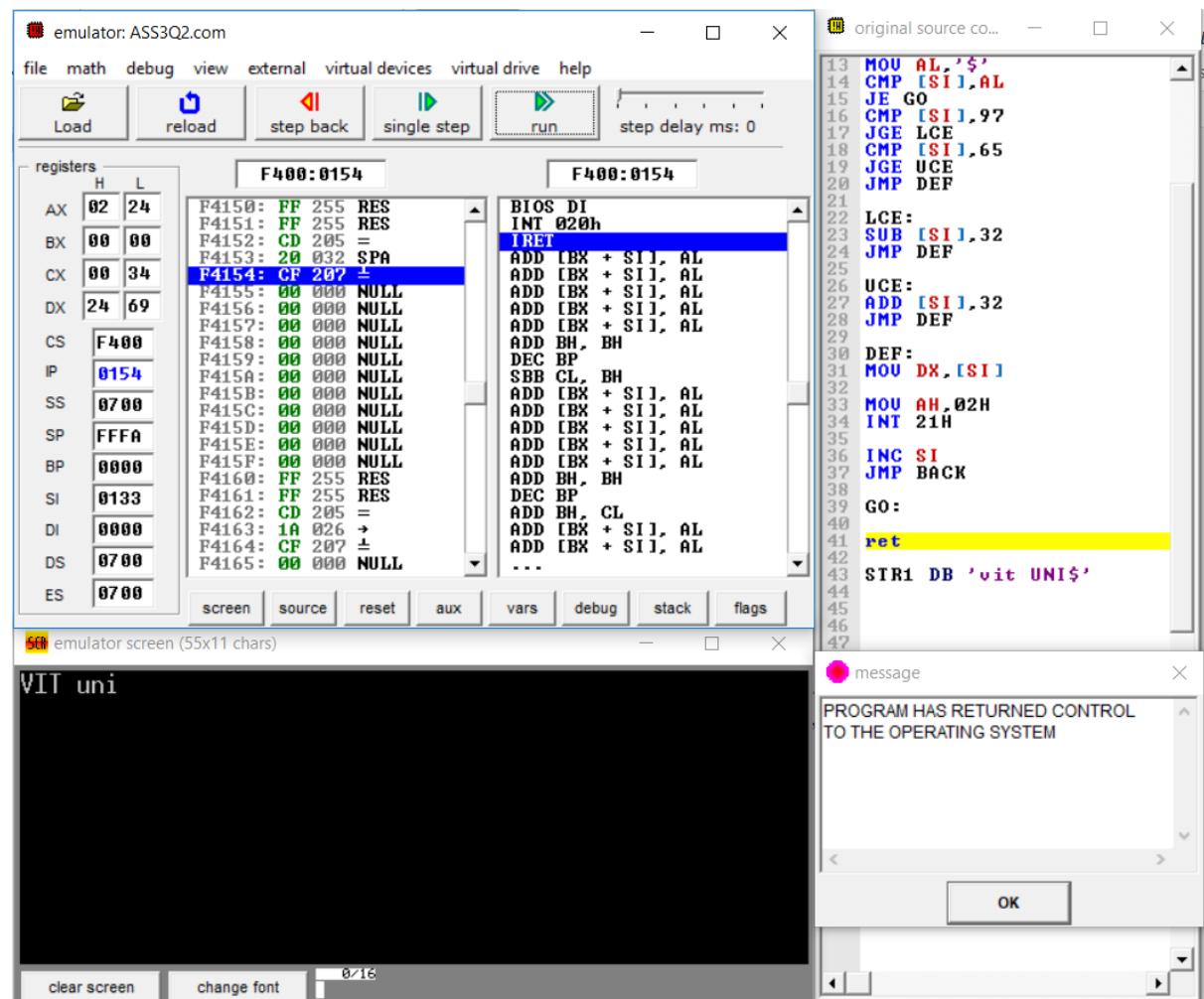
- Registers Window:** Shows CPU register values. AX: 02 6E, BX: 00 00, CX: 00 34, DX: 49 6E, CS: 0700, IP: 0106, SS: 0700, SP: FFFE, BP: 0000, SI: 0132, DI: 0000, DS: 0700, ES: 0700.
- Memory Dump Window:** Displays memory starting at address 0700:0106. It shows a sequence of instructions and data, including 'BEEP' and 'NEWL' strings.
- Assembly Window:** Shows the original source code in assembly language. The code includes labels like LCE, GO, UCE, DEF, and RET, along with various memory operations and jumps.
- Output Window:** Displays the output of the program, showing the text "VIT un".
- Status Bar:** Shows the screen size as 55x11 chars and the current memory location as 0/16.

```

original source code
13 MOU AL, '$'
14 CMP [SI], AL
15 JE GO
16 CMP [SI], 97
17 JGE LCE
18 CMP [SI], 65
19 JGE UCE
20 JMP DEF
21
22 LCE:
23 SUB [SI], 32
24 JMP DEF
25
26 UCE:
27 ADD [SI], 32
28 JMP DEF
29
30 DEF:
31 MOU DX, [SI]
32
33 MOU AH, 02H
34 INT 21H
35
36 INC SI
37 JMP BACK
38
39 GO:
40
41 ret
42
43 STR1 DB 'vit UNI$'
44
45
46
47
48
49

```

7. ‘vit UNI’ since letter ‘I’ under scan is upper case letter, it’s ASCII value will be increased by 32 and lower case ‘i’ will be printed.



Question 3: Write an assembly program to count the number of words in a given string. String may have comma, space, colon as delimiter.

Code:

The screenshot shows the Emu8086 assembly editor window. The title bar reads "edit: C:\emu8086\MySource\ASS3Q3.asm". The menu bar includes file, edit, bookmarks, assembler, emulator, math, ascii codes, and help. Below the menu is a toolbar with icons for new, open, examples, save, compile, emulate, calculator, convertor, options, help, and about. The main code area contains the following assembly code:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ; 17BCE2151 Rajendra Agrawal
07
08 MOU AL,'$'
09 MOU CX,00H
10 MOU BX,00H
11 MOU DX,00H
12 MOU SI,OFFSET STR1
13
14 BACK:
15     CMP [SI],AL
16     JE GO
17     CMP [SI],65
18     JGE LET
19     JMP SCH
20
21 LET:
22     MOU BX,01H
23     JMP DEF
24
25 SCH:
26     ADD CX,BX
27     MOU BX,00H
28     JMP DEF
29
30 DEF:
31     INC SI
32     JMP BACK
33
34 GO:
35     ADD CX,BX
36     MOU DX,CX
37     ADD DX,30H
38     MOU AH,02H
39     INT 21H
40
41
42 ret
43
44 STR1 DB 'UIT IS A,GOOD; PLACES'
45
46
47
48
49
50
51
```

The status bar at the bottom shows "line: 45 col: 30" and "drag a file here to open".

Here we use two registers BX and CX. CX counts the number of words. If we encounter a character, we change BX to 1 and if we encounter a delimiter we add BX to CX and change BX back to 0. At last we get total number of words in CX and print it. In our algorithm we take care of two delimiters occurring together and also of missing delimiter at end of sentence.

Step by Step Execution:

- Initially BX and CX both are zero.

```

01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06
07 ; 17BCE2151 Rajendra Agr
08
09 MOU AL,'$'
10 MOU CX,00H
11 MOU BX,00H
12 MOU DX,00H
13 MOU SI,OFFSET STR1
14
15 BACK:
16 CMP [SI],AL
17 JE GO
18 CMP [SI],65
19 JGE LET
20 JMP SCH
21
22 LET:
23 MOU BX,01H
24 JMP DEF
25
26 SCH:
27 ADD CX,BX
28 MOU BX,00H
29 JMP DEF
30
31 DEF:
32 INC SI
33 JMP BACK
34

```

- “VIT IS A,GOOD; PLACE” during scan of “VIT” value of BX will be 1 and CX is 0.

```

03 ; The location of this t
04
05 org 100h
06
07 ; 17BCE2151 Rajendra Agr
08
09 MOU AL,'$'
10 MOU CX,00H
11 MOU BX,00H
12 MOU DX,00H
13 MOU SI,OFFSET STR1
14
15 BACK:
16 CMP [SI],AL
17 JE GO
18 CMP [SI],65
19 JGE LET
20 JMP SCH
21
22 LET:
23 MOU BX,01H
24 JMP DEF
25
26 SCH:
27 ADD CX,BX
28 MOU BX,00H
29 JMP DEF
30
31 DEF:
32 INC SI
33 JMP BACK
34

```

3. “VIT IS A,GOOD; PLACE” as we encounter a space, BX will be added to CX and so CX is 1 and BX will be made to 0.

```

; The location of this t
org 100h
; 17BCE2151 Rajendra Agr
MOU AL,'$'
MOU CX,00H
MOU BX,00H
MOU DX,00H
MOU SI,OFFSET STR1
BACK:
CMP [SI],AL
JE GO
CMP [SI],65
JGE LET
JMP SCH
LET:
MOU BX,01H
JMP DEF
SCH:
ADD CX,BX
MOU BX,00H
JMP DEF
DEF:
INC SI
JMP BACK

```

4. “VIT **IS** A,GOOD; PLACE” during scan of “IS” value of BX will be 1 and CX is 1.

```

; The location of this t
org 100h
; 17BCE2151 Rajendra Agr
MOU AL,'$'
MOU CX,00H
MOU BX,00H
MOU DX,00H
MOU SI,OFFSET STR1
BACK:
CMP [SI],AL
JE GO
CMP [SI],65
JGE LET
JMP SCH
LET:
MOU BX,01H
JMP DEF
SCH:
ADD CX,BX
MOU BX,00H
JMP DEF
DEF:
INC SI
JMP BACK

```

5. “VIT IS A,GOOD; PLACE” as we encounter a space, BX will be added to CX and so CX is 2 and BX will be made to 0.

```

03 ; The location of this t
04 org 100h
05
06 ; 17BCE2151 Rajendra Agr
07
08 MOU AL,'$'
09 MOU CX,00H
10 MOU BX,00H
11 MOU DX,00H
12 MOU SI,OFFSET STR1
13
14 BACK:
15 CMP [SI],AL
16 JE GO
17 CMP [SI],65
18 JGE LET
19 JMP SCH
20
21 LET:
22 MOU BX,01H
23 JMP DEF
24
25 SCH:
26 ADD CX,BX
27 MOU BX,00H
28 JMP DEF
29
30 DEF:
31 INC SI
32 JMP BACK
33
34

```

6. “VIT IS A,GOOD; PLACE” during scan of “A” value of BX will be 1 and CX is 2.

```

03 ; The location of this t
04 org 100h
05
06 ; 17BCE2151 Rajendra Agr
07
08 MOU AL,'$'
09 MOU CX,00H
10 MOU BX,00H
11 MOU DX,00H
12 MOU SI,OFFSET STR1
13
14 BACK:
15 CMP [SI],AL
16 JE GO
17 CMP [SI],65
18 JGE LET
19 JMP SCH
20
21 LET:
22 MOU BX,01H
23 JMP DEF
24
25 SCH:
26 ADD CX,BX
27 MOU BX,00H
28 JMP DEF
29
30 DEF:
31 INC SI
32 JMP BACK
33
34

```

7. “VIT IS A,GOOD; PLACE” as we encounter a comma, BX will be added to CX and so CX is 3 and BX will be made to 0.

```

; The location of this t
org 100h
; 17BCE2151 Rajendra Agr
MOU AL,'$'
MOU CX,00H
MOU BX,00H
MOU DX,00H
MOU SI,OFFSET STR1
BACK:
CMP [SI],AL
JE GO
CMP [SI],65
JGE LET
JMP SCH
LET:
MOU BX,01H
JMP DEF
SCH:
ADD CX,BX
MOU BX,0000H
JMP DEF
DEF:
INC SI
JMP BACK

```

8. “VIT IS A,GOOD; PLACE” during scan of “GOOD” value of BX will be 1 and CX is 3.

```

; The location of this t
org 100h
; 17BCE2151 Rajendra Agr
MOU AL,'$'
MOU CX,00H
MOU BX,00H
MOU DX,00H
MOU SI,OFFSET STR1
BACK:
CMP [SI],AL
JE GO
CMP [SI],65
JGE LET
JMP SCH
LET:
MOU BX,01H
JMP DEF
SCH:
ADD CX,BX
MOU BX,0000H
JMP DEF
DEF:
INC SI
JMP BACK

```

9. "VIT IS A,GOOD; PLACE" as we encounter a semicolon, BX will be added to CX and so CX is 4 and BX will be made to 0.

```

03 ; The location of this t
04
05 org 100h
06
07 ; 17BCE2151 Rajendra Agr
08 MOU AL,'$'
09 MOU CX,00H
10 MOU BX,00H
11 MOU DX,00H
12 MOU SI,OFFSET STR1
13
14 BACK:
15 CMP [SI],AL
16 JE GO
17 CMP [SI],65
18 JGE LET
19 JMP SCH
20
21 LET:
22 MOU BX,01H
23 JMP DEF
24
25 SCH:
26 ADD CX,BX
27 MOU BX,00H
28 JMP DEF
29
30 DEF:
31 INC SI
32 JMP BACK
33
34

```

10. "VIT IS A,GOOD; PLACE" as we encounter a space, BX will be added to CX and so CX is 4(since BX was set to 0) and BX will be made to 0

```

03 ; The location of this t
04
05 org 100h
06
07 ; 17BCE2151 Rajendra Agr
08 MOU AL,'$'
09 MOU CX,00H
10 MOU BX,00H
11 MOU DX,00H
12 MOU SI,OFFSET STR1
13
14 BACK:
15 CMP [SI],AL
16 JE GO
17 CMP [SI],65
18 JGE LET
19 JMP SCH
20
21 LET:
22 MOU BX,01H
23 JMP DEF
24
25 SCH:
26 ADD CX,BX
27 MOU BX,00H
28 JMP DEF
29
30 DEF:
31 INC SI
32 JMP BACK
33
34

```

11.“VIT IS A,GOOD; PLACE” during scan of “PLACE” value of BX will be 1 and CX is 4.

The screenshot shows the ASS3Q3.com debugger interface. The assembly code window displays the following code:

```

03 ; The location of this t
04
05 org 100h
06
07 ; 17BCE2151 Rajendra Agr
08
09 MOU AL,'$'
10 MOU CX,00H
11 MOU BX,00H
12 MOU DX,00H
13 MOU SI,OFFSET STR1
14
15 BACK:
16 CMP [SI],AL
17 JE GO
18 CMP [SI],65
19 JGE LET
20 JMP SCH
21
22 LET:
23 MOU BX,01H
24 JMP DEF
25
26 SCH:
27 ADD CX,BX
28 MOU BX,00H
29 JMP DEF
30
31 DEF:
32 INC SI
33 JMP BACK

```

The registers window shows the following values:

	H	L
AX	00	24
BX	00	01
CX	00	04
DX	00	00
CS	0700	
IP	010E	
SS	0700	
SP	FFFE	
BP	0000	
SI	0147	
DI	0000	
DS	0700	
ES	0700	

The memory dump window shows the string "VIT IS A,GOOD; PLACE".

12.“VIT IS A,GOOD; PLACE” as the string is completed, value of BX is added to CX and CX is totalled to 5 which is printed on screen.

The screenshot shows the ASS3Q3.com debugger interface. The assembly code window displays the following code:

```

13 MOU SI,OFFSET STR1
14
15 BACK:
16 CMP [SI],AL
17 JE GO
18 CMP [SI],65
19 JGE LET
20 JMP SCH
21
22 LET:
23 MOU BX,01H
24 JMP DEF
25
26 SCH:
27 ADD CX,BX
28 MOU BX,00H
29 JMP DEF
30
31 DEF:
32 INC SI
33 JMP BACK
34
35 GO:
36 ADD CX,BX
37 MOU DX,CX
38 ADD DX,30H
39 MOU AH,02H
40 INT 21H
41
42
43 ret

```

The registers window shows the following values:

	H	L
AX	02	35
BX	00	01
CX	00	05
DX	00	35
CS	0700	
IP	0002	
SS	0700	
SP	0000	
BP	0000	
SI	0148	
DI	0000	
DS	0700	
ES	0700	

The memory dump window shows the string "VIT IS A,GOOD; PLACE".

The emulator screen window shows the character '5'.

The message window shows the text "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM".