

1) Write an assembly language program that stores a string in a variable. Now, first display the whole string and then display the first small letter and last small letter in the string. If no small letters are entered, then display "No small letters".

Sample Input / Output: Input in a String: input\_string DB 'WE aRE DUET STuDeNTs', 0Dh, 0Ah, '\$'

Output: a

S

The screenshot shows an x86 assembly language emulator. The main window displays the following assembly code:

```

01 ORG 100H
02 .DATA
03 s DB "WE aRE DUET STuDeNTs$", 0
04 fst DB ' '
05 snd DB ' '
06 len DW 20
07 MSG_F DB "First= $"
08 MSG_SND DB " Second= $"
09 .CODE
10 MAIN PROC
11 MOV AX, @DATA
12 MOV DS, AX
13
14 MOV SI, OFFSET s
15 MOV AL, ' '
16 MOV fst, AL
17 MOV snd, AL
18 MOV CX, len
19
20 FIND_LOWER:
21 LODSB
22 CMP AL, 'a'
23 JL NOT_LOWER
24 CMP AL, 'z'
25 JG NOT_LOWER
26
27 MOV DL, fst
28 CMP DL, ' '
29 JNE CHECK_SECOND
30 MOV fst, AL
31 JMP NEXT_CHAR
32
33 CHECK_SECOND:
34 MOV snd, AL
35 ; St
36
37 NOT_LOWER:
38 NEXT_CHAR:
39 LOOP FIND_LOWER
40
41 MOV AH, 09H
42 LEA DX, MSG_F
43 INT 21H
44
45 MOV DL, fst
46 MOV AH, 09H
47 INT 21H
48
49 MOV AH, 09H
50 LEA DX, MSG_SND
51 INT 21H
52
53 MOV DL, snd
54 MOV AH, 09H
55 INT 21H
56
57 MOV AH, 4CH
58 INT 21H

```

The output window shows the string "First: a Second: \$". A message box indicates "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM".

2) Write an assembly language code to derive the final value of the number sequence  $1 + 2 + 3 + 4 + \dots + N$ . (use ARRAY and Loop). Take the input value of N (in between 2 to 9) as a single ASCII character and then adjust it to actual decimal value in your program. Finally, store and show the output in a variable named RESULT.

Sample Input / Output:

Input: The value of N in between 2 ~ 9 : 9

The result is: 285

The screenshot shows an x86 assembly emulator with the following components:

- Assembly Code (Left Panel):**

```

01 ORG 100H
02 .DATA
03 arr DW 10 DUP(0)
04 n DW 0
05 result DW 0
06
07 MSG_INPUT DB 'Enter n (2~9): $'
08 MSG_RESULT DB 13, 10, 'Result: $'
09
10 .CODE
11 MOV AX, @DATA
12 MOV DS, AX
13
14 ; Input n
15 LEA DX, MSG_INPUT
16 MOV AH, 9
17 INT 21h
18
19 MOV AH, 1
20 INT 21h
21 SUB AL, '0'
22 XOR AH, AH
23 MOV n, AX
24
25 MOV CX, n
26 MOV BX, 0
27 MOV AX, 1
28
29 INIT_ARRAY:
30 MOV arr[BX], AX
31 INC BX
32 INC AX
33 LOOP INIT_ARRAY
34
35 MOV CX, n
36 MOV BX, 0
37 MOV result, 0
38
39 SUM_SQUARES:
40 XOR AX, AX
41 MOV AX, arr[BX]
42 AND AX, 0FH
43 MOV DX, AX
44 IMUL DX
45
46 ADD result, AX
47 INC BX
48 LOOP SUM_SQUARES
49
50 LEA DX, MSG_RESULT
51 MOV AH, 9
52 INT 21h
53
54 MOV AX, result
55
56 MOV CX, 4Ch
57 INT 21h
58
59 END MAIN

```
- Emulator Screen (Top Left):** Displays the input prompt "Enter n (2~9): 8" and the output "Result: 0/16".
- Registers (Bottom Left):** Shows the state of registers. AX contains 0204, BX contains 0008, CX contains 0000, DX contains 012A, IP contains F400, SS contains 0204, SP contains 0700, BP contains FFF8, SI contains 0000, DI contains 0000, DS contains 0700, ES contains 0700.
- Message Box (Center):** Displays "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM".
- Variables (Top Right):** Shows the state of variables. N is 8, RESULT is 285, MSG\_INPUT is 45h, and MSG\_RESULT is 00h.

3) Write an assembly code to sort the following data in ascending order using any sorting algorithm.

Sample Input: Sample 2 6 1 9 4    Output: The sorted list is: 1 2 4 6 9

The screenshot displays an x86-64 assembly emulator with the following components:

- Assembly Code Window:** Shows the source code for a bubble sort program. The code includes data definitions for an array, a prompt, and a code section with labels for the sorting algorithm.
- Registers Window:** Displays the current state of the CPU registers. The instruction pointer (RIP) is at 00000000, and the instruction is at 00000000.
- Memory Window:** Shows the memory layout, including the array data and the prompt string.
- I/O Window:** Displays the output of the program, showing the sorted list: 1 2 4 6 9.
- Emulator Controls:** Includes buttons for file operations, debugging, and running the program.

The assembly code is as follows:

```
01 ORG 100H
02 .DATA
03     ARRAY DB 2, 6, 1, 9, 4
04     LENGTH DW 5
05     MSG_PROMPT DB "The sorted list is: "
06
07 .CODE
08 MAIN PROC
09     MOV AX, @DATA
10     MOV DS, AX
11
12     LEA DX, MSG_PROMPT
13     MOV AH, 09H
14     INT 21H
15
16     XOR SI, SI
17
18 OUTER_LOOP:
19     CMP SI, LENGTH
20     JGE END_SORT
21
22     MOV DI, SI
23     INC DI
24 INNER_LOOP:
25     CMP DI, LENGTH
26     JGE NEXT_OUTER
27
28     MOV AL, ARRAY[SI]
29     MOV BL, ARRAY[DI]
30     CMP AL, BL
31     JLE NO_SWAP
32
33     XCHG AL, BL
34     MOV ARRAY[SI], AL
35     MOV ARRAY[DI], BL
36
37     NO_SWAP:
38     INC DI
39     JMP INNER_LOOP
40
41 NEXT_OUTER:
42     INC SI
43     JMP OUTER_LOOP
44 END_SORT:
45     MOV CX, LENGTH
46     MOV SI, OFFSET ARRAY
47
48 PRINT_LOOP:
49     MOV DL, [SI]
50     ADD DL, '0'
51     MOV AH, 02H
52     INT 21H
53     MOV DL, ' '
54     MOV AH, 02H
55     INT 21H
56     INC SI
57     LOOP PRINT_LOOP
58     MOV AH, 4CH
59     INT 21H
```

**Q1:** The user will input 3 numbers (single digit) and the program will output the second highest number. The sample input and output is given below:

Sample Input	Sample Output
9, 5, 8	8
9, 9, 6	6
9, 9, 9	All the numbers are equal
9, 6, 6	6

```

001 ORG 100H
002
003 .DATA
004 EQUAL DB "All the numbers are equal$
005 NEWLINE DB 0AH, 0DH, "$
006 A DB ?
007 B DB ?
008 C DB ?
009 FST DB 0
010 SND DB 0
011
012 .CODE
013
014 MAIN PROC
015     ; DATA LOAD
016     MOV AX, @DATA
017     MOV DS, AX
018
019     ; THREE INPUT A, B, C VARIABLE
020     MOV AH, 01H
021     INT 21H
022     MOV A, AL
023
024     INT 21H
025     MOV B, AL
026
027     INT 21H
028     MOV C, AL
029
030 ; NEW LINE
031 MOV AH, 09H
032 LEA DX, NEWLINE
033 INT 21H
034
035 MOV BL, A
036 MOV FST, BL ; FST
037
038 ; CHECK EQUAL OR NOT
039
040 MOV BL, B
041 CMP BL, A ; A
042 JE A_B_EQUAL ; YE
043 JMP FIND_SND ; 3
044
045 A_B_EQUAL:
046 MOV BL, B
047 CMP BL, C
048 JE ABC_EQUAL ; A == B
049 JMP FIND_SND ;
050
051 ABC_EQUAL:
052 MOV AH, 09H
053 LEA DX, EQUAL ; PRINT
054 INT 21H
055 JMP EXIT
056
057 FIND_SND:
058 MOV BL, B
059 CMP BL, FST
060 JLE CHK_FSTE_C
061
062 MOV BL, FST ;
063 MOV SND, BL ; F
064 MOV BL, B
065 MOV FST, BL
066
067
068 CHK_FSTE_C:
069 MOV BL, C
070 CMP BL, FST
071 JLE DOUBLE_CHK
072 MOV BL, FST ; S
073 MOV SND, BL ; F
074 MOV BL, B
075 MOV FST, BL
076
077 DOUBLE_CHK:
078 MOV BL, A
079 CMP BL, FST
080 JGE CHK_SND_B
081 MOV BL, A
082 CMP BL, SND
083 JLE CHK_SND_B
084 MOV BL, A
085 MOV SND, BL
086
087 CHK_SND_B:
088 MOV BL, B
089 CMP BL, FST
090 JGE CHK_SND_C
091 MOV BL, B
092
093 CHK_SND_C:
094 MOV BL, C
095 CMP BL, FST
096 JGE OUTPUT
097 MOV BL, C
098 CMP BL, SND
099 JLE OUTPUT
100 MOV BL, C
101 MOV SND, BL
102
103 ; OUTPUT SHOW
104 OUTPUT:
105 MOV AH, 02H
106 MOV DL, SND
107 INT 21H
108 JMP EXIT
109
110 EXIT:
111 MOV AH, 4CH
112 INT 21H
113
114 ENDP MAIN
115 END MAIN
116
117
118
119
120 RET
121

```

**Q2.** Write an assembly language code to derive the final value of the number sequence:  $-1+2-3+4-5+\dots+N$ . (Using Loop). Take the input value of N (in between 2 to 9) as a single ASCII character and then adjust it to the actual decimal value in your program. Finally, store the output in a variable named RESULT. You do not need to display the output in the console

```

01 ORG 100H
02
03 .DATA
04 NEWLINE DB 0AH, 0DH, "$" ; Newline string
05 RESULT DB 0
06 SIGN DB -1 ; Alternating sign
07 N DB ? ; Input number
08 CUNT DB 1 ; Counter for loop
09
10 .CODE
11 MAIN PROC
12 ; Load data
13 MOV AX, @DATA
14 MOV DS, AX
15
16 MOV AH, 01H ; Input N via keyboard
17 INT 21H
18 SUB AL, 48 ; Convert ASCII to decimal
19 MOV N, AL ; Store in N
20
21 ; new line
22 MOV AH, 09H
23 LEA DX, NEWLINE
24 INT 21H
25
26 XOR CX, CX ; Clear CX
27 MOV CL, N ; Loop count
28
29 CALCULATION:
30 MOV BL, CUNT
31 INC CUNT
32
33 MOV AL, SIGN
34 IMUL BL
35
36 ADD RESULT, AL
37 NEG SIGN
38
39 LOOP CALCULATION
40
41 ; NO NEED TO PRINT THE RESULT
42
43 MOV AH, 4CH ; Exit
44 INT 21H
45 END MAIN

```

**Q3:** First of all we will input a mark of the subject from the candidate and according to the following condition we will calculate the grade.

Sample Input: 80 Output: Grade: A

```

01 ORG 100H
02
03 .DATA
04 GRADE_F DB 0AH, 0DH, 'Grade: F$'
05 GRADE_D DB 0AH, 0DH, 'Grade: D$'
06 GRADE_C DB 0AH, 0DH, 'Grade: C$'
07 GRADE_B DB 0AH, 0DH, 'Grade: B$'
08 GRADE_A DB 0AH, 0DH, 'Grade: A$'
09 GRADE_A_PLUS DB 0AH, 0DH, 'Grade: A+$'
10 N DB 0
11 TEMP DB 0
12
13 .CODE
14 MAIN PROC
15 ; Load data segment
16 MOV AX, @DATA
17 MOV DS, AX
18
19 INPUT_LOOP:
20 ; Read a character
21 MOV AH, 01H
22 INT 21H
23
24 ; Check for CR (Carriage Return)
25 CMP AL, 0DH
26 JE END_INPUT_LOOP
27 CMP AL, 0AH
28 JE END_INPUT_LOOP
29
30 ; Convert character to decimal
31 SUB AL, 48
32 MOV TEMP, AL
33
34 MOV BL, N
35 MOV AL, 10
36 MUL BL
37 ADD AL, TEMP
38 MOV N, AL
39
40 ; Loop back for the next character
41 JMP INPUT_LOOP
42
43 END_INPUT_LOOP:
44 MOV AL, N
45
46 CMP AL, 50 ; If mark < 50
47 JL PRINT_GRADE_F ; Jump
48
49 CMP AL, 60 ; If mark < 60
50 JL PRINT_GRADE_D ; Jump
51
52 CMP AL, 70 ; If mark < 70
53 JL PRINT_GRADE_C ; Jump
54
55 CMP AL, 80 ; If mark < 80
56 JL PRINT_GRADE_B ; Jump
57
58 CMP AL, 90 ; If mark < 90
59 JL PRINT_GRADE_A ; Jump
60
61 JMP PRINT_GRADE_A_PLUS ; If mark >= 90
62
63 PRINT_GRADE_F:
64 LEA DX, GRADE_F
65 JMP DISPLAY_GRADE
66
67 PRINT_GRADE_D:
68 LEA DX, GRADE_D
69 JMP DISPLAY_GRADE
70
71 PRINT_GRADE_C:
72 LEA DX, GRADE_C
73 JMP DISPLAY_GRADE
74
75 PRINT_GRADE_B:
76 LEA DX, GRADE_B
77 JMP DISPLAY_GRADE
78
79 PRINT_GRADE_A:
80 LEA DX, GRADE_A
81 JMP DISPLAY_GRADE
82
83 PRINT_GRADE_A_PLUS:
84 LEA DX, GRADE_A_PLUS
85
86 DISPLAY_GRADE:
87 ; Print grades
88 MOV AH, 09H
89 INT 21H
90
91 MOV AH, 4CH
92 INT 21H

```