Program 13 Date 10 Nov 2012

Aim:

Write assembly language macros:

- i) To read a character from the keyboard
- ii) To display a character
- iii) Use the above two functions to read a string of characters from the keyboard terminated by the carriage return and prints the string on the display in the next line.

Theory:

The program uses the 01h and 02h service routines of the 21h interrupt to accept or write a single character. A macro is created using these instructions used in series to accept a string.

```
1
    .8086
2
    .MODEL SMALL
3
    .STACK 512
4
    .DATA
5
            CRLF DB 13,10,'$'
6
            STRING DB 64 DUP(?)
7
    . CODE
8
    START:
9
            MOV AX,@DATA
            MOV DS,AX
10
11
    READCHAR MACRO
12
13
            MOV AX,0100H
14
            INT 21H
15
    ENDM
16
17
    WRITECHAR MACRO
            ;CHAR TO BE MOVED TO DL
18
19
            MOV AX,0200H
20
            INT 21H
21
    ENDM
22
23
    READSTR MACRO
24
            MOV SI,0
25
    RST:
26
            READCHAR
27
            MOV STRING[SI], AL
28
            INC SI
            CMP AL,13
29
30
            JNZ RST
31
            MOV STRING[SI], '$'
    ENDM
32
33
    WRITESTR MACRO
34
35
           MOV SI,0
   WST:
36
37
            MOV DL,STRING[SI]
```

```
38
            WRITECHAR
39
            INC SI
40
            CMP DL,13
41
            JNZ WST
42
    ENDM
43
44
            READCHAR
45
            MOV DL,AL
46
            WRITECHAR
47
            LEA DX, CRLF
48
            MOV AH,09H
49
            INT 21H
50
51
52
            READSTR
53
54
            LEA DX, CRLF
            MOV AH,09H
55
56
            INT 21H
57
58
            WRITESTR
59
60
            MOV AH,4CH
61
            INT 21H
   END START
62
```

An 8086 assembly language macro to read/write a character and strings was written, assembled, linked and debugged to obtain expected output.

Program 14 Date 10 Nov 2012

Aim:

Write assembly language to read an alphanumeric character and display its equivalent ASCII code at the centre of the screen.

Theory:

The program uses interrupt service routines to carry out tasks like moving the cursor position, clearing the screen etc. The 8086 assembly language provides an extensive list of interrupt services to perform a wide list of functions. The two interrupts used here are 10h and 21h. The former controls the output device, in this case, the monitor. The latter controls the input/output streams and buffers for the monitor and the keyboard.

```
1
     .8086
2
     .MODEL SMALL
3
    .STACK 512
4
     .DATA
5
            CHAR
                            DB ?
                            DB 4 DUP(0)
6
            INTEG
7
    .CODE
    START:
10
            MOV AX,@DATA
            MOV DS,AX
11
12
            MOV AH,01H
13
            INT 21H
14
15
16
            MOV CHAR, AL
17
            ;INVOKE VIDEO INTERRUPT
18
19
            ;CLEAR SCREEN
            MOV AL,00H
20
21
            MOV CX,00H
22
            MOV DX,1850H
23
            MOV AH,06H
24
            MOV BH,07H
25
            INT 10H
26
            ;SET CURSOR POSITION
27
28
            MOV DX,0C23H
29
            MOV AH, 02H
30
            MOV BH,00H
31
            INT 10H
32
            MOV [INTEG+3], '$'
33
            MOV AL, CHAR
34
35
            MOV CL, OAH
36
            MOV BX,02H
37
    L00P0:
```

```
38
            MOV AH,00H
39
            DIV CL
40
            ADD AH,'0'
41
            MOV INTEG[BX], AH
42
            DEC BX
43
            JNS LOOP0
44
            LEA DX, INTEG
45
            MOV AH,09H
46
            INT 21H
47
48
            MOV AH,08H
49
50
            INT 21H
51
            MOV AH, 4CH
52
53
            INT 21H
54
   END START
```

An 8086 assembly language macro to read a character and display its equivalent ASCII value in the centre of the screen was written, assembled, linked and debugged to obtain expected output.

Aim:

To write 8086/8087 program to compute the real roots of the quadratic equation.

Theory:

The quadratic equation being solved is $2x^2 + 5x + 3 = 0$. This is done by using the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where the values of x are the roots of the given equation. This gives two possible values of x. These values are stored in variables X and Y respectively. This discriminant is stored in the variable disc.

```
.8087
1
2
    .MODEL SMALL
3
    .STACK 512
    .DATA
5
                        DW 2.0
                         DW 5.0
6
            В
7
            C
                         DW 3.0
            DISC
8
                        DW ?
9
                          DW ?
            Χ
10
            Υ
                          DW ?
11
    .CODE
12
    START:
13
            MOV AX,@DATA
14
            MOV DS,AX
15
16
            FINIT
17
18
            FLD B
            FLD B
19
20
            FMUL
21
22
            FLD A
23
            FLD C
24
            FMUL
25
            FLD 4.0
26
            FMUL
27
            FSUB
28
29
            FST DISC
30
31
            FINIT
32
            FLD B
33
            FLD -1.0
34
            FMUL
35
            FLD DISC
36
            FSUB
37
            FLD A
38
            FLD 2.0
            FMUL
39
```

```
FDIV
40
41
            FST X
42
43
            FINIT
44
            FLD B
45
           FLD -1.0
46
           FMUL
47
            FLD DISC
48
            FADD
49
           FLD A
50
           FLD 2.0
51
            FMUL
52
            FDIV
53
           FST Y
54
55
           MOV AH,4CH
56
           INT 21H
57 END START
```

An 8086/8087 assembly language program to calculate the roots of a quadratic equation was written, assembled, linked and debugged to obtain expected output.

Program 15 Date 10 Nov 2012

Aim:

Write assembly language to reverse a string and check whether it is a palindrome or not.

Theory:

A palindrome is a word, phrase, number, or other sequence of units that may be read the same way in either direction. The program checks for the inbuilt strng for being a palindrome and displays the result.

```
.8086
     .MODEL SMALL
2
3
     .STACK 512
4
     .DATA
5
                                    DB "RACECAR", '$'
            STRNG
6
            STRNGLEN
                                    DW $-STRNG
                                    DB 20 DUP(' '),13,10
7
            STRNGREV
                                    DB 13,10,"PALINDROME",'$'
8
            PROMPT0
            PROMPT1
                                    DB 13,10,"NOT A PALINDROME",'$'
9
10
     .CODE
11
12
    START:
            MOV AX,@DATA
13
14
            MOV DS,AX
15
            MOV ES, AX
16
17
            MOV CX, STRNGLEN
18
            ADD CX,-2
19
20
            LEA SI, STRNG
21
            LEA DI, STRNGREV
22
            ADD SI, STRNGLEN
23
24
            ADD SI,-2
    L00P0:
25
26
            MOV AL,[SI]
27
            MOV [DI],AL
28
            DEC SI
29
            INC DI
30
            L00P L00P0
31
32
            MOV AL, [SI]
33
            MOV [DI], AL
34
            INC DI
35
            MOV DL, '$'
36
            MOV [DI], DL
37
            MOV CX, STRNGLEN
38
            LEA DX, PROMPT1
39
            MOV AH.09H
40
41
            INT 21H
42
```

```
43
            LEA DX,STRNGREV
44
            MOV AH,09H
45
            INT 21H
46
47
    PCHECK:
48
            LEA SI,STRNG
49
            LEA DI,STRNGREV
            REPE CMPSB
50
            JNE PFALSE
51
52
   PTRUE:
53
            MOV AH,09H
54
55
            LEA DX, PROMPTO
56
            INT 21H
57
            JMP TERMINATE
58
59
   PFALSE:
60
            MOV AH,09H
61
            LEA DX, PROMPT2
62
            INT 21H
63
    TERMINATE:
64
65
            MOV AX,4C00H
66
            INT 21H
67
68
    END START
```

An 8086 assembly language program to determine if a string is a palindrome or not was written, assembled, linked and debugged to obtain expected output.

Program 16 Date 10 Nov 2012

Aim:

To write an assembly language program to read two strings, store them in locations STR1 and STR2 and check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.

```
. 8086
 1
 2
      .MODEL SMALL
 3
      .STACK 512
 4
      .DATA
 5
 6
               STRNG1
                                  DB 64 DUP('$')
 7
               STRNG2
                                  DB 64 DUP('$')
               STRNG2 DB 64 DDP(*S
STRNG1_LEN DW $-STRNG1
STRNG2_LEN DW $-STRNG2
8
9
               CRLF DB 13,10,'$'
PROMPT1 DB 10,13,"STRING 1: ",'$'
PROMPT2 DB 10,13,"STRING 2: ",'$'
PROMPT3 DB 10,13,"STRINGS ARE EQU
PROMPT4 DB 10,13,"STRINGS ARE NOT
10
                                  DB 13,10,'$
11
12
13
                                  DB 10,13, "STRINGS ARE EQUAL", '$'
                                  DB 10,13, "STRINGS ARE NOT EQUAL", '$'
14
15
     .CODE
16
17
               EXTRN READSINT: NEAR, WRITESINT: NEAR
18
     START:
19
               MOV AX,@DATA
20
               MOV DS,AX
21
22
     READSTRNG MACRO STRNG
23
               MOV AH, OAH
24
               LEA DX, STRNG
25
               INT 21H
26
     ENDM
27
28
     WRITESTRNG MACRO STRNG
29
               MOV AH,09H
               LEA DX, STRNG
30
               INT 21H
31
32
     ENDM
33
               WRITESTRNG PROMPT1
34
               READSTRNG STRNG1
35
36
               WRITESTRNG PROMPT2
37
               READSTRNG STRNG2
38
39
40
               LEA DX, CRLF
41
               MOV AH,09H
42
               INT 21H
43
44
               MOV AX, STRNG1 LEN
45
               CALL WRITESINT
46
               LEA DX, CRLF
47
               MOV AH,09H
48
```

```
INT 21H
49
50
51
            MOV AX,STRNG2_LEN
52
            CALL WRITESINT
53
54
            LEA SI,STRNG1
            LEA DI,STRNG2
55
56
57
            MOV CL,STRNG1+1
58
            MOV CH,00H
59
            REPE CMPSB
60
61
            JNE NOTEQUAL
62
            WRITESTRNG PROMPT3
63
64
            JMP TERMINATE
65
66
    NOTEQUAL:
67
            WRITESTRNG PROMPT4
68
69
    TERMINATE:
70
            MOV AX,4C00H
71
            INT 21H
72
   END START
```

An 8086 assembly language program to determine if two read strings were equal or not was written, assembled, linked and debugged to obtain expected output.

Program 17 Date 10 Nov 2012

Aim:

To write an assembly language program to read a name from the keyboard and display it at a specified location on the screen in front of the message 'WHAT IS YOUR NAME?'. The entire screen has to be cleared before display.

Source code:

48

INC SI

```
.8086
1
2
     .MODEL SMALL
3
     .STACK 512
4
     .DATA
5
            PROMPT1 DB "WHAT IS YOUR NAME?", '$'
            PROMPT2 DB "YOU ARE ",'$'
6
7
            NAMEO DB 99 DUP(0)
8
9
     .CODE
10
    START:
11
            MOV AX,@DATA
            MOV DS,AX
12
13
14
    POS PROC
15
            MOV AH,02H
            MOV BH,00H
16
17
            INT 10H
18
            RET
    POS ENDP
19
20
21
            MOV AH,06H
22
            MOV AL,00H
23
            MOV BH,07H
24
            MOV CX,00H
25
            MOV DX,1850H
26
            INT 10H
27
28
            MOV DX,0C23H
            CALL POS
29
30
            LEA DX, PROMPT1
31
32
            MOV AH,09H
            INT 21H
33
34
            LEA SI, NAMEO
35
36
            LEA DI, NAMEO
37
    L00P1:
38
            MOV AH,01H
39
            INT 21H
40
41
            CMP AL,08H
42
            JE LOOP2
43
44
            CMP AL, 0DH
45
            JE TAG1
46
47
            MOV [SI],AL
```

```
49
            JMP LOOP1
50
51
   L00P2:
52
            MOV DL,''
            MOV AH,02H
53
54
            INT 21H
55
            MOV DL,08H
56
57
            INT 21H
58
            CMP SI,DI
59
            JE LOOP1
60
61
62
            DEC SI
            JMP LOOP1
63
    TAG1:
64
65
            MOV AL, '$'
66
            MOV [SI],AL
67
68
            MOV DX,0D23H
69
            CALL POS
70
71
            LEA DX,PROMPT2
72
            MOV AH,09H
73
            INT 21H
74
75
            MOV AH,4CH
            INT 21H
76
77
   END START
```

An 8086 assembly language program to read a name and write it according to the given requirements was written, assembled, linked and debugged to obtain expected output.

Program 18 Date 10 Nov 2012

Aim:

To write an assembly language program to compute ⁿC_r using a recursive procedure. Assume that 'n' and 'r' are non-negative integers

Theory:

In mathematics a combination is a way of selecting several things out of a larger group, where (unlike permutations) order does not matter. It is defined as

$$c_r^n = \frac{n!}{(n-r)! \, r!}$$

The program defines the factorial as a procedure and calculates it recursively.

```
.8086
1
2
     .MODEL SMALL
3
     .DATA
                             DB 05H
 4
5
             R
                             DB 02H
 6
             NCR
                     DW ?
8
     .CODE
9
             EXTRN READSINT: NEAR, WRITESINT: NEAR
10
     START:
11
             MOV AX,@DATA
12
             MOV DS,AX
13
14
     NCRP PROC
15
             CMP AX, BX
             JE TAG1
16
17
             CMP BX,00H
18
19
             JE TAG1
20
             CMP BX,01H
21
22
             JE TAG2
23
24
             DEC AX
25
             CMP AX, BX
26
             JE TAG3
27
             PUSH AX
28
29
             PUSH BX
             CALL NCRP
30
31
             POP BX
32
33
             POP AX
34
             DEC BX
35
             PUSH AX
36
             PUSH BX
37
             CALL NCRP
38
```

```
POP BX
39
40
            POP AX
41
            RET
42
    TAG1:
43
            INC NCR
44
            RET
45
    TAG2:
46
            ADD NCR, AX
47
            RET
48
    TAG3:
            ADD NCR, AX
49
50
            INC NCR
51
            RET
52
    NCRP ENDP
53
54
            MOV AX,00H
55
            MOV AL,N
56
            MOV BL,R
57
            MOV NCR,00H
58
            CALL NCRP
59
            MOV AH,4CH
60
61
            INT 21H
62
   END START
```

An 8086 assembly language program to compute ${}^{n}C_{r}$ using a recursive procedure was written, assembled, linked and debugged to obtain expected output.

Program 19 Date 10 Nov 2012

Aim:

To write an assembly language program to read the current time from the system and display it in the standard format on the screen.

Theory:

The program uses 2ch service of 21h interrupt to obtain the system time. The interrupt stores hours in CH, minutes in CL, seconds in DH and 1/100 seconds in DL. The values are converted into character strings before displayed on the screen.

```
1
    . 8086
     .MODEL SMALL
    .STACK 512
     . DATA
 5
    . CODE
 6
     START:
7
             MOV AX,@DATA
8
             MOV DS,AX
9
10
             MOV AH, 2CH
             INT 21H
11
12
             MOV BL, OAH
13
            MOV AL, CH
14
             CALL ATIME
15
16
             MOV AL, CL
17
18
             CALL ATIME
19
20
             MOV AL, DH
21
             CALL MTIME
22
23
             MOV AH, 4CH
24
             INT 21H
25
26
    ATIME PROC
27
             CALL MTIME
28
29
             MOV DL, ':'
             MOV AH,02H
30
             INT 21H
31
32
33
             RET
     ATIME ENDP
34
35
36
    MTIME PROC
37
             MOV AH,00H
38
             DIV BL
39
            MOV DL, '0'
40
```

```
41
            XCHG AL, AH
42
            ADD DL,AH
43
44
            MOV AH,02H
45
            PUSH AX
46
            INT 21H
47
48
            POP AX
49
            MOV DL,AL
50
            ADD DL,'0'
            INT 21H
51
52
53
            RET
54
   MTIME ENDP
55
    END START
```

An 8086 assembly language program to read the current time from the system and display it in the standard format on the screen was written, assembled, linked and debugged to obtain expected output.

Program 20 Date 10 Nov 2012

Aim:

To write an assembly language program to simulate a decimal up-counter to display 00-99.

Source code:

```
1
    . .8086
2
     .MODEL SMALL
3
    .STACK 512
4
    .DATA
5
            COUNTER
                           DB 64H
    .CODE
6
7
    START:
8
            MOV AX,@DATA
9
            MOV DS,AX
10
            MOV CL, COUNTER
11
            MOV AL,00H
12
13
    L00P0:
14
15
            MOV AL,64H
16
            SUB AL, CL
17
            MOV BL, OAH
18
            MOV AH,00H
19
            DIV BL
            XCHG AL, AH
20
            MOV DL,AH
21
22
            ADD DL, '0'
23
            MOV AH,02H
24
            PUSH AX
25
            INT 21H
26
            POP AX
27
28
            MOV DL,AL
            ADD DL,'0'
29
30
            INT 21H
31
            MOV DL, ODH
32
33
            INT 21H
34
            PUSH CX
35
            MOV BX,01AAH
36
    L00P1:
37
            L00P L00P1
38
            DEC BX
39
            JNZ LOOP1
40
            POP CX
41
            L00P L00P0
42
43
            MOV AH,4CH
44
            INT 21H
45
    END START
```

Conclusion:

An 8086 assembly language program to simulate a decimal up-counter was written, assembled, linked and debugged to obtain expected output.