AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Computer Science and Engineering

Course Code: CSE 2214

Term Assignment

Course Teacher Name: Md. Siam Ansary

Date of Submission: 17.09.2025

Submitted by,

Name: Danial Hossain Dani

Student ID: 20230104058

Lab Section: B1

Answer To The Question 01

(a) i.

For the following instructions determines the new values of CF, SF, AF, PF and OF with a short explanation. Suppose that the flags are initially 0. "**NEG AX**" where AX contains 8000h

Answer:

CF (Carry Flag): 1

For Negative Operation, If the borrow occurs, the carry flag will be 1, and if the borrow doesn't occur, the carry flag will remain 0. Here Neg AX can be written like 0 – AX = –AX. In this type of scenario where we subtract a large value from a small value, the borrow is occurred, resulting the Carry Flag 1.

SF(Sign Flag): 1

The Sign Flag (SF) is 1 when the most significant bit (MSB) is 1, and 0 when the MSB is 0. Here, converting 8000h into 16-bit binary gives 1000 0000 0000 0000. Since the MSB is 1, the Sign Flag is set to 1.

OF(Overflow Flag): 1

The Overflow Flag is set to 1 when the result of an operation is too large to fit in the destination operand's signed range. For 16-bit signed numbers, the valid range is:

$$-2^{15} \text{ to } 2^{15}-1 \rightarrow -32768 \text{ to } +32767$$

Here, AX = 8000h which represents −32768 (since the MSB is 1).

The operation is: NEG 8000h = 0−(−32768) = +32768

But +32768 cannot be represented in 16-bit signed range, so overflow occurs

AF(Auxiliary Flag): 0

The Auxiliary Flag (AF) is set to 1 if there is a borrow (in subtraction) or carry (in addition) between the low nibble (bits 0–3) and the high nibble (bits 4–7) of the lower byte. In the operation NEG AX with AX = 8000h, the lower byte is 0h, so no borrow occurs between
bit 3 and bit 4. Therefore, AF = 0

PF(Parity Flag): 1

The Parity Flag (PF) is set to 1 if the low byte of the result contains an even number of 1s, and 0 if it contains an odd number of 1s. In NEG AX with AX = 8000h, the low byte is 0h, which has 0 ones. Since 0 is an even number, PF = 1.

(a) ii.

For the following instructions determine the new values of CF, SF, AF, PF and OF with
a short explanation. Suppose that the flags are initially 0. "**INC AX"** where AX contains FFFFh

Answer:

CF (Carry Flag): 0

The Carry Flag (CF) is not affected by the INC instruction. Even though incrementing FFFFh wraps around to 0000h, CF stays unchanged.

SF (Sign Flag): 0

The Sign Flag (SF) is 0 when the most significant bit (MSB) is 0, and 1 when the MSB is 1

For INC FFFFh:

$$\begin{array}{r} 1111\ 1111\ 1111\ 1111\ (\text{FFFFh}) \\ +\ 0000\ 0000\ 0000\ 0001 \qquad\qquad \\ \hline 0000\ 0000\ 0000\ 0000\ (\text{0000h}) \end{array}$$

The MSB of the result is 0, so the Sign Flag = 0.

OF (Overflow Flag): 0

The Overflow Flag (OF) is set when a signed result cannot be represented within the available number of bits. For 16-bit signed numbers, the range is −32768 to +32767 (in decimal). The result of INC FFFFh is 0000h (0 in decimal), which falls within this range, so OF = 0.

AF (Auxiliary Flag): 1

The Auxiliary Flag is set if there is a carry out from bit 3 to bit 4 (low nibble to high nibble) in the lower byte during addition. In INC FFFFh, the lower byte is FFh (1111 1111), and adding 1 produces 00h, causing a carry from bit 3 to bit 4. Therefore, AF = 1.

PF (Parity Flag): 1

The Parity Flag (PF) is set to 1 if the lower byte of the result contains an even number of 1s. After INC FFFFh, the result is 0000h, whose lower byte has 0 ones, which is even. Therefore,
PF = 1.

(b) i.

A memory location has a physical address 8DCE2h. Write down the logical address if it has segment 7EDAh.

Answer:

Given physical address is 8DCE2h and segment 7EDAh.

We know:

Physical Address = Segment × 16 + Offset

Converting segment into binary : 0111 1110 1101 1010

Shift left 4 bits (×16):

Segment x16 = 0111 1110 1101 1010 0000

= 7EDA0h

Offset = Physical Address – Segment x 16

= 8DCE2h − 7EDA0h

= 0EF42h

logical address = Segment : Offset

so , Here the logical address is 7EDA:0EF42h

(b) ii.

A memory location has a physical address ABCDEh. Write down the logical address if it has offset BF2Eh.

Answer:

Given physical address is ABCDEh and offset BF2Eh.

We know:

Physical Address = Segment × 16 + Offset

Segment x 16 = Physical Address – Offset

= ABCDEh -BF2Eh

= 9FDB0h

Converting Segment x16 into binary : 1001 1111 1101 1011 0000

Shit right 4 bits(/16):

Segment = 1001 1111 1101 1011

= 9FDBh

logical address = Segment: Offset

so, Here the logical address is 9FDB:BF2Eh

(c) i.

Show how the decimal integer -267 would be represented in 16 bits. Express the answer in hexadecimal.

Answer:

To represent the decimal number −267 in 16 bits, we use two's complement.

267_{10} = 0000 0001 0000 1011_2

0000 0001 0000 1011

1111 1110 1111 0100 (1's complement)

+1

1111 1110 1111 0101 (2's complement)

Converting this binary number to hexadecimal: FEF5h

(c) ii.

Show how the decimal integer -263 would be represented in 16 bits. Express it in hexadecimal.

Answer:

To represent the decimal number −263 in 16 bits, we use two's complement.

263_{10} = 0000 0001 0000 0111_2

0000 0001 0000 0111

1111 1110 1111 1000 (1's complement)

+1

1111 1110 1111 1001 (2's complement)

Converting this binary number to hexadecimal: FEF9h

Question 02

(a) i.

Write an assembly code to perform the following operations:

- Declare A as constant and set its value as 7.
- Declare B and C as regular variables and initialize it with 3 and 2 respectively.
- Perform the following equation and store the value in Z. Display it with an appropriate message.

$$Z = A + 2B - 1 - 2C$$

Answer:

```
 .model small
.stack 100h

.DATA
A EQU 7
B DB 3
C DB 2
Z DB ?
MSG DB 'Z = $'

.CODE

MAIN PROC
   MOV AX,@DATA
   MOV DS,AX

   MOV AL,B
   MOV BL,2
   MUL BL
   MOV Z,AL

   DEC Z

   MOV AL,C
   MOV BL,2
   MUL BL
   SUB Z,AL

   MOV AL,A
   ADD Z,AL
   ADD Z,48

   MOV AH,9
   LEA DX, MSG
   INT 21h
   MOV AH,2
   MOV DL,Z
   INT 21H

   MOV AH,4Ch
   INT 21h
   MAIN ENDP

END MAIN
```

(a) ii.

Write an assembly code to perform the following operations:

- Declare P and Q as regular variables and initialize them with 6 and 4,respectively.
- Declare R as a constant and set its value to 3.
- Perform the following equation and store the value in Y. Display it with an appropriate message.

$$\mathbf{Y = P + 1 + Q - 2R + 1}$$

Answer:

```
.model small
.stack 100h

.DATA
R EQU 3
P DB 6
Q DB 4
Z DB ?
MSG DB 'Y = $'

.CODE
MAIN PROC
MOV AX,@DATA
MOV DS,AX

MOV AL,P
MOV Z,AL
MOV AL,Q
ADD Z,AL

INC Z
INC Z

MOV AL,R
MOV BL,2
MUL BL
SUB Z,AL
ADD Z,48

MOV AH,9
LEA DX, MSG
INT 21h

MOV AH,2
MOV DL,Z

INT 21H
MOV AH,4Ch
INT 21h
MAIN ENDP

END MAIN
```

(b) Write an assembly language program that takes an integer N as input (1 ≤ N ≤ 14) and prints the Fibonacci series up to the Nth term.

Answer:

```
.MODEL SMALL
.STACK 100H

.DATA
R     DB 'ENTER RANGE: $'
FAB   DB 13,10,13,10,'FIBONACCI
SERIES $'
N     DB ?
TEMP  DW ?
DIVISOR DW 10

.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX

    MOV DX,OFFSET R
    MOV AH,9
    INT 21H

    XOR BX,BX
    MOV BL,10

    MOV AH,1
    INT 21H
    CMP AL,13
    JE NEXT
    SUB AL,30H
    MOV CL,AL

    MOV AH,1
    INT 21H
    CMP AL,13
    JE NEXT
    SUB AL,30H
    MOV CH,AL

    MOV AL,CL
    MUL BL
    ADD AL,CH
    MOV N,AL
    JMP INPUT_DONE

NEXT:
    MOV N,CL

INPUT_DONE:

    LEA DX,FAB
    MOV AH,9
    INT 21H

    MOV CX,0
    MOV CL,N
    CMP CL,0
    JE EXIT_PROGRAM

    MOV BL,0
    MOV BH,1

    PUSH BX
    PUSH CX
    MOV AL,BL
    MOV AH,0
    CALL DISPLAY_NUMBER
    POP CX
    POP BX
    DEC CX
    JZ EXIT_PROGRAM

    PUSH BX
    PUSH CX
    MOV AL,BH
    MOV AH,0
    CALL DISPLAY_NUMBER
    POP CX
    POP BX
    DEC CX
    JZ EXIT_PROGRAM

FIB_LOOP:

    MOV AL,BL
    ADD AL,BH
    MOV BL,BH
    MOV BH,AL

PUSH BX
    PUSH CX
    MOV AL,BH
    MOV AH,0
    CALL DISPLAY_NUMBER
    POP CX
    POP BX

    LOOP FIB_LOOP

EXIT_PROGRAM:
    ; exit
    MOV AH,4CH
    INT 21H
MAIN ENDP


DISPLAY_NUMBER PROC
    MOV CX,0
    MOV BX,DIVISOR

DIGIT_LOOP:
    XOR DX,DX
    DIV BX
    PUSH DX
    INC CX
    CMP AX,0
    JNE DIGIT_LOOP

PRINT_DIGITS:
    POP DX
    ADD DL,30H
    MOV AH,2
    INT 21H
    LOOP PRINT_DIGITS


    MOV DL,' '
    MOV AH,2
    INT 21H

    RET
DISPLAY_NUMBER ENDP

END MAIN
```

(c) Write an assembly language program to find the maximum and minimum elements of a byte array containing 10 unsigned 8-bit numbers. Also, print the range (Maximum Value - Minimum Value) of the array. You do not need to take user input for the array.

Answer:

```
.MODEL SMALL
.STACK 100H

.DATA
INPUT DB 2,2,3,4,5,6,7,8,9,4
MSG1 DB 10,13,'LARGEST VALUE: $'
MSG2 DB 10,13,'SMALLEST VALUE: $'
MSG3 DB 10,13,'RANGE: $'
LARGER DB ?
SMALLER DB ?

.CODE

MAIN PROC
    MOV AX,@DATA
    MOV DS,AX

    LEA SI,INPUT
    MOV CX,0
    MOV BL,0

LARGEST:
    MOV AL,[SI]
    CMP AL,BL
    JLE NOT_LARGER
    MOV BL,AL

NOT_LARGER:
    INC SI
    INC CX
    CMP CX,10
    JL LARGEST

PRINT1:
    MOV LARGER,BL
    MOV AH,9
    LEA DX,MSG1
    INT 21H
    MOV AH,2
    MOV DL,BL
    ADD DL, '0'
    INT 21H

    LEA SI,INPUT
    MOV CX,0
    MOV BL,100



SMALLEST:
    MOV AL,[SI]
    CMP AL,BL
    JGE NOT_SMALLER
    MOV BL,AL

NOT_SMALLER:
    INC SI
    INC CX
    CMP CX,10
    JL SMALLEST

PRINT2:
    MOV SMALLER,BL
    MOV AH,9
    LEA DX,MSG2
    INT 21H
    MOV AH,2
    MOV DL,BL
    ADD DL, '0'
    INT 21H


    MOV AH,9
    LEA DX,MSG3
    INT 21H

    MOV BL, LARGER
    SUB BL, SMALLER
    MOV AH,2
    MOV DL,BL
    ADD DL, '0'
    INT 21H

EXIT:
    MOV AH,4Ch
    INT 21h
MAIN ENDP
END MAIN
```