Name :- M. Daniyal. Reg no :- L1F20BSCS0036

Cource = COAL Section :- C13

Assignment 1

Q1

Values of DS = 0xA000

```
Mov Ax, [0xBCD8]
Mov Bx, [0xBCDA]
Add Ax, Bx
Mov [0xBCD4], Ax
```

Ax = | 00 | 1D |

Bx = | 00 | 61 |

Add ax, bx

ax = | 00 | 7E |

mov [0xBED4], ax

This value will be overwritten and the value of ax will be moved in it i.e;

0x7E -> 0xABCD4

0x23 -> 0xABCCB
0xA0 -> 0xABCCC
0xB1 -> 0xABCCD
0xC2 -> 0xABCCE
0x00 -> 0xABCCF
0x15 -> 0xABCD0
0x32 -> 0xABCD1
0x25 -> 0xABCD2
0x11 -> 0xABCD3
0xFE -> 0xABCD4
0x11 -> 0xABCD5
0x95 -> 0xABCD6
0xD5 -> 0xABCD7
0x1D -> 0xABCD8
0x2E -> 0xABCD9
0x61 -> 0xABCDA
0x58 -> 0xABCDB
0xAA -> 0xABCDC
0xAA -> 0xABCDD

Q2

Sum of first and last element of array.

```
[org 0x100]

xor ax, ax
xor bx, bx
xor cx, cx
xor dx, dx

mov ax, [arr]
mov bx, [arr+5]
add ax, bx

mov ax, 0x4c00
int 21h

arr: dw 5, 10, 15, 20, 25
```

Q3

Add elements of two arrays and store carry Flag in Index Carry

```
[org 0x100]

 xor  ax,ax
 xor  bx, bx
 xor  cx, cx
 xor  dx, dx

 mov  ax, [arr]
 mov  bx , [arr1]
add  ax , bx
mov  [arr2],ax

mov  ax, [arr+2]
mov  bx, [arr1+2]
add  ax, bx
mov  [arr2+2],ax

mov  ax , [arr+2]
mov  bx , [arr1+2]
add  ax, bx
mov [arr2+2],ax

mov  ax, [arr+3]
mov  bx, [arr1+3]
 add  ax, bx
 mov [arr2+3],ax

mov ax, [arr+4]
mov bx, [arr1+4]
 add  ax,bx
 mov [arr2+4],ax

mov ax, [arr+5]
mov bx, [arr1+5]
add ax, bx
 mov [arr2+5], ax

mov cx , [arr+6]
mov dx , [arr1+6]
add cx, dx
mov [arr2+6],cx

mov cx , [arr+7]
mov dx, [arr+7]
add cx,dx
mov [arr2+7],cx

mov  cx, [arr+8]
mov dx, [arr+8]      ; Here it overflow
add cx, dx
 mov [arr2+8],cx
mov [Index Carry], cx

 mov ax, 0x4c00
 int  21h

arr: dw  1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr1: dw  1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr2: dw  0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Index Carry: dw 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

Question 4

Change the code to Assembly language.

```
[org 0x100]

xor ax, ax
xor bx, bx
xor cx, cx
xor dx, dx

 mov ax, 0
 mov bx, 10h

CMP bx, cx
 jne l2
 JMP end

l2:
 add ax, 2
CMP bx, cx
 JZ l3
 JMP end

l3:
 mov dx, 2
  Jmp l1

l4:
 mov dx, 3
  Jmp exit

exit
 mov ax, 0x4c00
 int 21h.
```