Computer Organization And Assembly Language:

Assignment-I:

Student Name: [redacted]

Section: BCS-3B

Roll No.: [redacted]

Submission Date: 16-9-2022

Q.1. $A_x = 0x334A$, $B_x = 0x45F1$
$C_x = 0x8934$

a) add ax, bx

```
  0011  0011  0100  1010
  0100  0101  1111  0001
  ----------------------
  0111  1001  0011  1011
```

CF = 0, OF = 0, SF = 0, ZF = 0

b) add cx, bx

```
  1000  1001  0011  0100
  0100  0101  1111  0001
  ----------------------
  1100  1111  0010  0101
```

CF = 0 OF = 0 SF = 1, ZF = 0

c) sub bx, 6

6 = 0000 0000 0000 0110

-6 = 1111 1111 1111 1010

```
  1 1 1   1 1 1   1 1 1
  0100    0101    1111    0001
  1111    1111    1111    1010
-------------------------------
1 0100    0101    1110    1011
```

CF = 0, OF = 0, SF = 0, ZF = 0

Q.2: Storing words in Big Endian Format:

a) 0xb900

In Lower Memory Address: b9

In Higher Memory Address: 00

b) 0x4567

In Lower Memory Address: 45

In Higher Memory Address: 67

c) 0xAA99

In Lower Memory Address: AA

In Higher Memory Address: 99

Q.3: Calculate Physical addresses.

a) FFFF:4312

0x FFFF0

0x 04312

Physical Address: 0x104302 ⇒ After Wrap Around Address = 0x04302

b) 1DEF:0001

0x 1DEF0

0x 00001

Physical Address: 0x1DEF1

c) 14FF:1111

0x 14FF0

0x 01111

0x 16101

Physical Address: 0x16101

Q.NO.4:

```
1  [org 0x0100]
2
3  mov ax, 45
4  mov bx, -7
5  mov cx, 0
6  mov word[result], 0
7
8  cmp ax, 0                    ; if(ax < 0)
9  jns skip2                    ; {
10 neg ax                       ;   ax = -ax
11 inc cx                       ;   cx++
12 skip2:                       ; }
13
14 cmp bx, 0                    ; if(bx < 0)
15 jns skip3                    ; {
16 neg bx                       ;   bx = -bx
17 inc cx                       ;   cx++
18 skip3:                       ; }
19
20 cmp ax, bx                   ; if(ax < bx)
21 jg skip1                     ; {
22 mov dx, ax                   ;   temp = ax
23 mov ax, bx                   ;   ax = bx
```

```
24 mov bx, dx                   ;    bx = temp
25 skip1:                       ; }
26
27 loopStart:                   ; {
28 cmp bx, 0                    ;    if(bx == 0)
29 je loopEnd                   ;        break;
30
31 add word[result], ax         ;    result = result + ax
32
33 sub bx, 1                    ;    bx--
34
35 jmp loopStart
36 loopEnd:                     ; }
37
38 cmp cx, 1                    ; if(cx == 1)
39 jne skip4                    ; {
40 neg word[result]             ;    result = -result
41 skip4:                       ; }
42
43 mov ax, 0x4C00
44 int 0x21
```

Q.5: Calculate Effective Address.

Bx = 0x0034, Si = 0x0110, Di = 0x1101

Bp = 0x0220, Sp = 0x FFFF

a) bp - di

Invalid (Subtraction is not allowed)

b) bp + si

0x 0220

0x 0110

Effective Address : 0x 0330

c) bx - 0x12

0x0034

- 0x0012

Effective Address 0x0022

d) bx + bp

Invalid because two base reg-isters cannot be added

e) bx + ip

Invalid Because ip register cannot be accessed.

f) bx + di

0x0034

0x1101

0x1135

Effective Address : 0x1135

Q.6 CS: 0x4582 SS: 0x4582

Bx : 0x22AA IP: 0x0580 DI: 0x4247

BP: 0x4700 SI: 0x FEEF

a) [CS: bx + Si]

bx : 0x 22AA

Si : 0x FEEF

0x12199

Offset 0x 2199 (segment wraparound)

CS: 0x45820
bx+si 0x02199
0x479B9

Physical Address: 0x479B9
Wrap Around: Segment Wrap Around

b) [bp+di+10]

bp 0x4700
di 0x4247
10 0x000A
Offset: 0x8951

SS: 0x45820
bp+di+10 0x08951
0x4E171

Physical Address: 0x4E171
Wrap around: No Wrap around

Q.7: a) mov ip, bx

ip register cannot be modified.

mov bp, bx
mov ax, bx

b) mov byte bx, [ip]

ip register cannot accessed and byte size cannot be mentioned with bx register

Alternate Solutions:

```
mov bl, [bp]
mov bx, [si]
```

c) mov si, al

si is a 16bits register while al is a 8 bits register creating size mismatch error.

Alternate Solutions:

```
mov si, bx
mov si, ax
```

d) mov ax, [bx + bp + 100]

Invalid because two base registers cannot be added in effective address.

Alternate Solutions:

```
mov ax, [bx+si+100]
mov ax, [bp+di+100]
```

Q.8) Flags After running the instructions.

Zero Flag = 0 Carry Flag = 0 Parity Flag = 0

Overflow Flag = 0 Sign Flag = 1

Q.9:

There is not any apparent logical error in code of question no. 8 except that the result stored in num+4 address is overwritten in case the sum is greater than 4 so there should be an extra register or memory used to store the sum of first four numbers of num1.

Q.NO.10:

```
1  [org 0x0100]
2
3  ;write your code here to swap the contents of even index with the odd ones
4  ;that is, swap 1 &amp;amp; 2, 3 &amp;amp; 4
5  start:
6  mov si, 0              ; si = 0
7  mov di, 1              ; di = 1
8
9  swapStart:             ; {
10 cmp si, [siz]          ;   if (si >= siz)
11 jge swapEnd            ;       break
12
13 mov al, [num1 + si] ;   al = num1[si]
14 mov bl, [num1 + di] ;   bl = num1[di]
15
16 mov [num1 + si], bl ;   num1[si] = bl
17 mov [num1 + di], al ;   num1[di] = al
18
19 add si, 2              ;   si = si + 2
20 add di, 2              ;   di = di + 2
21
22 jmp swapStart
23 swapEnd:               ; }
24
25 end:
26 mov ax, 0x4c00         ; terminate program
27 int 0x21
28
29 num1: db 1, 2, 3, 4
30 siz: dw 4
```

Q.NO.11

```
1  [org 0x0100]
2
3  ;write your code here to find max in an array
4  ;and store it min
5  start:
6  mov si, 2                    ; si = 2
7  mov ax, [array1]             ; max = array1[0]
8  mov cx, 1                    ; cx = 1
9
10 loopStart:                   ; {
11 cmp cx, [siz]                ;   if (cx >= siz)
12 jge loopEnd                  ;       break
13
14 cmp [array1 + si], ax        ; if (array1[si] > max)
15 jle skip1                    ; {
16 mov ax, [array1 + si]        ;   max = array1[si]
17 skip1:                       ; }
18
19 add si, 2                    ;   si = si + 2
20 add cx, 1                    ;   cx = cx + 1
21
22 jmp loopStart
23 loopEnd:                     ; }
24
25 mov [min], ax
26
27 end:
28 mov ax, 0x4c00               ; terminate program
29 int 0x21
30
31 array1: dw 1, 3, -8, 7, 5
32 siz: dw 5
33 min: dw 0
```