Case Western Reserve University

Department of Computer and Data Sciences

EECS 349&444: Computer Security

Assignment Date:	10/15/2019
Sumission Date:	10/18/2019@9:59am
First Name:	Sabrina
Last Name:	Tay
Google Drive Link:	
Abstract of the feedback:	

* This is the part of Q2 and Q3 for HW2 which contains 20 points (10/pts per question). You are encouraged to finish independently. Any submitted work that it copied from any source or too similar to be an independent write-up will not be given credit. Please submit your solutions along with your detailed analysis for each line of code (e-copy) on Canvas by 9:59am on 10/18/2019.

HW-2: Assembly Code Analysis

Q2: Functionality?

```
DATA    SEGMENT
X   DB   25          var x = 25 1 byte
Y   DB   32          var y = 32 1 byte
Z   DW   ?           z uninitialized 2byte
DATA    ENDS

CODE    SEGMENT
    ASSUME   CS:CODE,DS:DATA
START:   MOV   AX,DATA
    MOV   DS,AX
    MOV   AL,X
    MUL   AL
    MOV   BX,0
    MOV   BL,Y
    ADD   BL,BL
    ADC   BH,0
    ADD   BL,Y
    ADC   BH,0
    SUB   AX,BX
    SHR   AX,1
    MOV   Z,AX
    MOV   AH,4CH
    INT   21H
CODE    ENDS
    END   START
```

AX = DATA
DS = AX
AL = X

AX = AX * AL
BX = 0
BL = Y

BL = BL + BL
BH = BH + 0 + CF
BL = BL + Y

BH = BH + 0 + CF
AX = AX - BX
Right shift bits in AX by 1
Z = AX
AH = 4CH
Interrupt

Q3: Functionality?

```
.data
array dd 34,12,3,18
szMsg db "%d",0ah,0

.code
start:
    mov ecx,3          ecx = 3
L2:
    push ecx           ecx on stack
    xor esi,esi        esi set to 0
    mov ecx,3          ecx = 3

L0:
    mov ebx,array[esi]          ebx = array[esi]
    cmp ebx,array[esi 4]        cmp ebx and array[esi + 1]
    jb L1                       skip L1 if ebx is larger
    xchg ebx,array[esi 4]       swaps ebx and array[esi + 1]
    mov array[esi],ebx          array[esi] = ebx
    add esi,4                   esi++
L1:
    loop L0                     go to L0
    pop ecx                     pop ecx from the stack
    loop L2                     go to L2
    xor esi,esi                 esi = 0
    mov ecx,4                   ecx = 1
L3:                             // if current is bigger than next, swap the 2 and add
    push ecx
    invoke printf,offset szMsg,array[esi]
    add esi,4
    pop ecx
    loop L3
    ret
end start
```

Q2)

Start x = 25
       y = 32
       z = ?

AX : x = 25 * 25 (after MUL AL)
   = 625

BL : y + y + y = 32 + 32 + 32 = 96
   (after ADC BH, 0)

AX : 625 − 96 = 529 (after SUB AX, BX)

AX = 529 right shift by 1 bit = 264

z = 264 (after MOV z, AX)

* ~~z is set to the x² − 3y~~

z is set to the right shift by 1 bit of $(x^2 - 3y)$

Q3)

L0 → compares current w/ next
if current is ~~smaller~~ larger than skip L1
swap the 2 ~~so~~ so smaller is at front
keep going up array till the sorted ~~list~~ part

L1 → go to L0
take ecx from the stack
then call L2
esi set to 0 reset esi
ecx = 4 reset ecx.

swaps the order of #s in the stack.

L2 → pushes the ecx onto the stack
resets esi to 0
~~MOV~~ ecx is set to 3.

L3 → pushes ecx onto stack
prints array element at that index
esi++;
pops top ecx
goes until no more elements in stack

* insertion sort from least to greatest.