Homework 2

Yue Shu (yxs626)
EECS444
Prof. Ye
October 17, 2019

Problem 2

Let's first take a look at the functionality of each line of the code segment as below:

```
MOV AL, X ; store the value of X in AL
MUL AL    ; multiply AL by AX and store that value in AX
MOV BX, 0 ; store 0 in BX
MOV BL, Y ; store the value of Y in BL
ADD BL, BL; add the value of BL to BL
ADD BH, 0 ; add 0 to BH
ADD BL, Y ; add the value of Y to BL
ADC BH, 0 ; add 0 and the value of carry in bit to BH
SUB AX, BX; substract the value of BX from AX, where BX = BH:BL
SHR AX, 1 ; shift all digits in AX to the right by one bit, which is essentially dividing AX by 2
MOV Z, AX ; store the value of AX in Z
```

Therefore from the above line-by-line code explanation we may conclude that the final functionality of the program is to execute $Z = (X^2 - 3Y)/2$, where $X = 25$, $Y = 32$, and therefore $Z = (625 - 96)/2 = 264$ since Z is an integer.

Problem 3

From problem 3, again, let's start with looking at the functionality of each line of the code segment below:

```
START:
    MOV ecx, 3                          ; store 3 in ecx
L2:
    PUSH ecx                            ; push ecx to the stack
    XOR esi, esi                        ; XOR esi with esi and store the result in esi, which is 0
    MOVE ecx, 3                         ; store 3 in ecx
L0:
    MOV ebx, array[esi]                 ; store the value of array[esi] aka array[0] = 34 in ebx
    CMP ebx, array[esi 4]               ; compare array[4] = 12 with ebx = 34,
                                        ; and set CF according to the result
                                        ; since ebx < array[esi 4], CF = 0
    JB L1                               ; jump to L1 if CF == 1
    XCHG ebx, array[esi 4]              ; exchange the value stored in ebx and array[4]
    MOV array[esi], ebx                 ; store the value of ebx in array[esi]
    ADD esi, 4                          ; add 4 to esi
L1:
    LOOP L0                             ; run the loop L0
    POP ecx                             ; pop ecx out of the stack
    LOOP L2                             ; run the loop L2
    XOR esi, esi                        ; XOR esi with esi and store the result in esi, which is 0
    MOV ecx, 4                          ; store 4 in ecx
L3:
    PUSH ecx                            ; push ecx to the stack
    INVOKE printf, offset szMSG, array[esi]; format print the array as Strings
    ADD esi, 4                          ; add 4 to eci
    POP ecx                             ; pop ecx from the stack
    LOOP L3                             ; run the loop L3
    RET
END START
```

Therefore, according to the line-by-line code explanation above, we can basically conclude that the general structure of the program is a for loop with a conditional statement nested in another for loop. For the conditional statement, the program will exchange two neighbouring elements if the leading element is larger than the following element.

Since it might seem a little too abstract reading the explanation above, let's rewrite the program in Java as below for better understanding the program:

```
int[] array = {34, 12, 3, 18};

for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        if (array[j + 1] > array[j])
            swap(array[j], array[j + 1]);
    }
}

System.out.println(array);
```

And if we run the code above, we may conclude that the program is actually a bubble sort, which still needs some optimization.