

# Comp 3350: Computer Organization & Assembly Language

## HW # 7: Theme: Conditionals, Booleans, Loops

(All main questions carry equal weight. Credit awarded to only those answers for which work has been shown.)

1. Draft a program that scans an array to determine the first odd integer in an array. If a value is found, the program should print "odd integer found" its value and index. If no odd integer is found, the program should print "no odd integer found." **Submit the asm/list file and screenshots that shows the output of your code for the following example arrays:**

	files	10 points
a. Array has all odd integers		5 points
b. Array has all even integers		5 points
c. Several arrays with a mix of odd and even integers positioned at different indices		5 points

```
; HW7 - 1
;
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:dword

INCLUDE Irvine32.inc

.data
arr          DWORD 1, 3, 5, 7, 9, 11
oddint       BYTE "odd integer found, ", 0
oddvalue     BYTE "value: ", 0
oddindex     BYTE ", index: ", 0
nooddint     BYTE "no odd integer found", 0

.code
main proc

    mov eax, OFFSET arr
    mov ecx, LENGTHOF arr
L1: mov ebx, [eax]
    and ebx, 01H
    jnz L2
    add eax, 4
    loop L1
    jmp L3
L2: mov edx, OFFSET oddint
    call WriteString
    mov edx, OFFSET oddvalue
    call WriteString
    mov eax, [eax]
    call WriteInt
    mov edx, OFFSET oddindex
    call WriteString
    mov eax, 6
    sub eax, ecx
    call WriteInt
    jmp L4
L3: mov edx, OFFSET nooddint
    call WriteString

L4: invoke ExitProcess,0
main endp
end main
```

```
.data
arr          DWORD 2, 4, 6, 8, 10, 12
oddint       BYTE "odd integer found, ", 0
oddvalue     BYTE "value: ", 0
oddindex     BYTE ", index: ", 0
nooddint     BYTE "no odd integer found", 0

.data
arr          DWORD 2, 4, 6, 8, 10, 11
oddint       BYTE "odd integer found, ", 0
oddvalue     BYTE "value: ", 0
oddindex     BYTE ", index: ", 0
nooddint     BYTE "no odd integer found", 0
```

2. Write a program which encodes any string using the XOR instruction. Test it using your <first name last name> in the data segment to produce cipher text and then decode using the program to get plain text. Use the last two digits of your student id as the key. Print plane text from the data segment, print the cipher text, and then print the plain text upon execution. **Submit the asm/list file and screenshots that shows the output of your code.** **files 3 points \*2 screenshot 13.5(encrypt 7 points, decrypt 6.5 points)**

What are the strengths and weaknesses of this encryption method (25% of points, Typewritten answer required)? 6.5 points

```
.data
plain BYTE "Ziyan Tian", 0
key BYTE 24
encrypt BYTE "plane text -> cipher text: ", 0
arrow BYTE " -> ", 0
decrypt BYTE "cipher text -> plane text: ", 0
cipher BYTE ?
.code
main proc

    mov eax, OFFSET plain
    mov ebx, OFFSET cipher
    mov ecx, LENGTHOF plain
    mov edx, 0
L1: mov dl, BYTE PTR [eax]
    xor dl, key
    mov [ebx], dl
    add eax, 1
    add ebx, 1
    loop L1

    mov edx, OFFSET encrypt
    call WriteString
    mov edx, OFFSET plain
    call WriteString
    mov edx, OFFSET arrow
    call WriteString
    mov edx, OFFSET cipher
    call WriteString

    mov eax, OFFSET plain
    mov ebx, OFFSET cipher
    mov ecx, LENGTHOF cipher
    mov edx, 0
L2: mov dl, BYTE PTR [ebx]
    xor dl, key
    mov [eax], dl
    add eax, 1
    add ebx, 1
    loop L2

    call crlf
    mov edx, OFFSET decrypt
    call WriteString
    mov edx, OFFSET cipher
    call WriteString
    mov edx, OFFSET arrow
    call WriteString
    mov edx, OFFSET plain
    call WriteString

    invoke ExitProcess, 0
main endp
end main
```

```
Select Microsoft Visual Studio Debug Console
plane text -> cipher text: Ziyan Tian -> BqayvBqayv
cipher text -> plane text: BqayvBqayv -> Ziyan Tian
C:\Users\Ziyan Tian\Desktop\AssemblyLanguageProgramming\
> 0:
Press any key to close this window . . .
```

3. Write a program that gets its input from two sensors. If the values of the sensors differ by no more than  $\pm 3$ , print “Agree”, otherwise, print “Disagree.” You can assume that the values are integers. Additionally, if the values Agree and they are each more than 50, print “Take Action”. **Submit asm/list file and show screenshots of robust testing for various inputs, including boundary conditions, in the closed interval (-90 ... 90).**

files 10 points, screenshot 15 points

```
.data
promptSensor1 BYTE "please enter the vlaue of senser1: ", 0
promptSensor2 BYTE "please enter the vlaue of senser2: ", 0
agree          BYTE "Agree", 0
disagree       BYTE "Disagree", 0
action         BYTE "Take Action", 0
sensor1        DWORD ?
sensor2        DWORD ?
.code
main proc
L1: mov edx, OFFSET promptSensor1
    call WriteString
    call ReadInt
    mov sensor1, eax
    cmp sensor1, 90
    jg L1
    cmp sensor1, -90
    jl L1

L2: mov edx, OFFSET promptSensor2
    call WriteString
    call ReadInt
    mov sensor2, eax
    cmp sensor2, 90
    jg L2
    cmp sensor2, -90
    jl L2

    mov ebx, sensor1
    cmp ebx, sensor2
    jl L3
    mov ebx, sensor1
    sub ebx, sensor2
    cmp ebx, 3
    jg L4

L3: mov ebx, sensor2
    sub ebx, sensor1
    cmp ebx, 3
    jg L4
    mov edx, OFFSET agree
    call WriteString
    call crlf
    jmp L5
L4: mov edx, OFFSET disagree
    call WriteString
    call crlf
L5: cmp sensor1, 50
    jl L6
    cmp sensor2, 50
    jl L6
    mov edx, OFFSET action
    call WriteString
L6: invoke ExitProcess,0
main endp
end main
```

```
Microsoft Visual Studio Debug Console
please enter the vlaue of senser1: 99
please enter the vlaue of senser1: -99
please enter the vlaue of senser1: 56
please enter the vlaue of senser2: 99
please enter the vlaue of senser2: -99
please enter the vlaue of senser2: 57
Agree
Take Action
C:\Users\Ziyan Tian\Desktop\AssemblyLanguageProgram\
Press any key to close this window . . .
```

4. Draw the stack (word/pdf) before every instruction that is marked red is executed to show your understanding of the call and return functions. Use N/A to represent unpredictable values.

```

Main Proc
4040018    mov ecx, 0000000Ch
404001C    mov ebx, 0000000Bh
4040020    call FMul
4040026    mov eax, ebx
...
...
Main EndP

```

```

FMul PROC
4041040    Push ecx
4041044    Push ebx
4041048    mov eax, edx
...
...
404A060    Pop ebx
404A062    Pop ecx
404A064    ret
FMul EndP

```



3 points \* 8