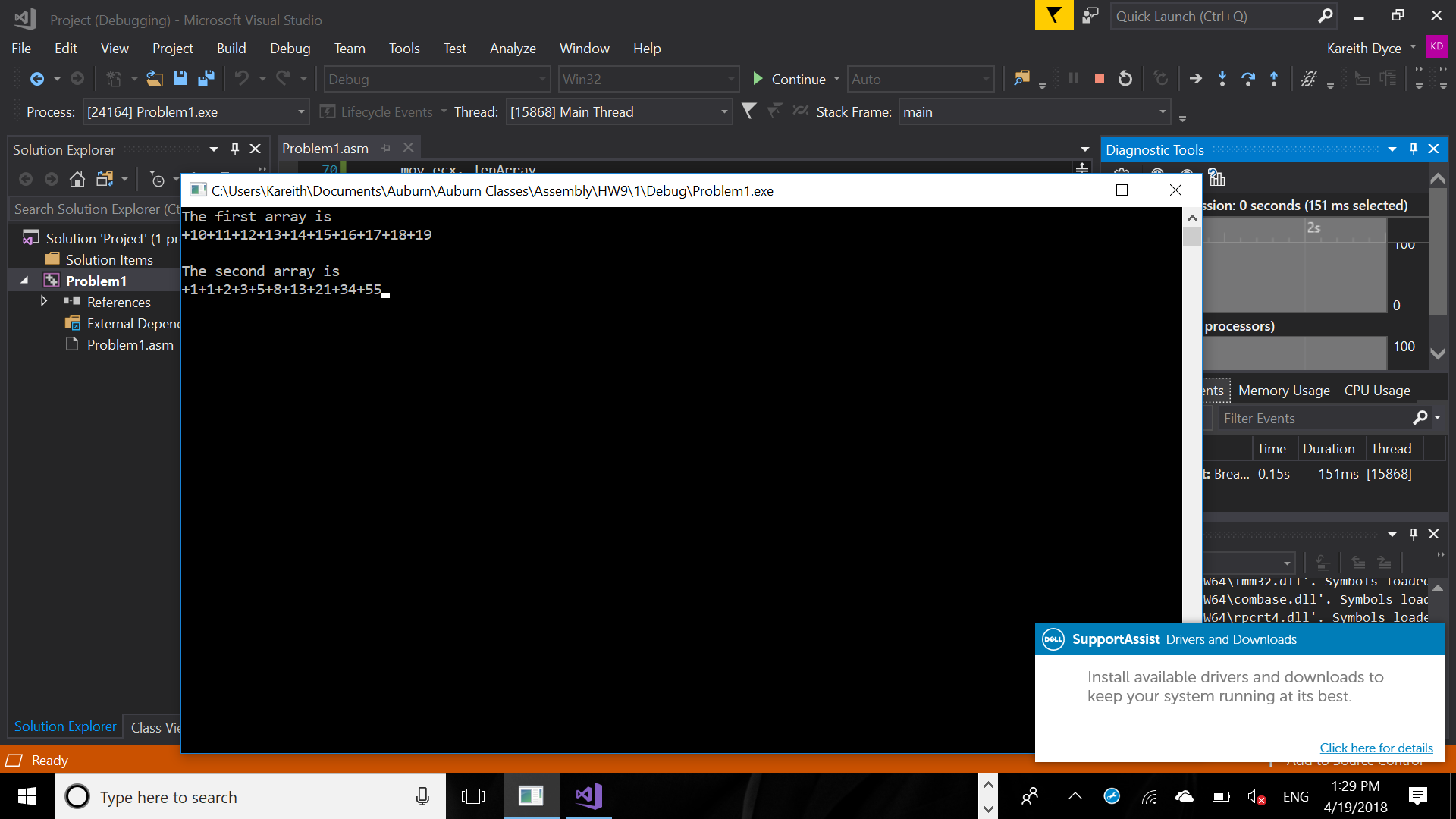
# Comp 3350: Computer Organization & Assembly Language

# HW # 9: Theme: Advanced Procedures, Stack Parameters, Locals and BCD

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

1. Write a procedure named *ArraySeries* that fills an array of ten (10) numbers with the Fibonacci series. The procedure receives two arguments: the first is the offset of an array, and the second is an integer that specifies the array length. The first argument is passed by value and the second is passed by reference. In the main program, you should set the parameters of the array and print the array values before and after call to the procedure.



Please embed your code into your homework solution along with a screen shot of the run of the program.

Include Irvine32.inc

.386

.stack 4096

ExitProcess proto,dwExitCode:dword

.data

array DWORD 10,11,12,13,14,15,16,17,18,19

arraySize DWORD SIZE array

temp DWORD ?

lenArray DWORD LENGTHOF array

word1 BYTE "The first array is ", 0

word2 BYTE "The second array is ", 0

.code

ArraySeries PROC

push ebp

mov ebp,esp

mov ecx, [ebp+8]

mov esi, [ebp+12]

mov eax, 1

mov ebx, 0

mov temp, 0

mov [esi], eax

mov edx, 4

L1:

mov temp, eax

add eax, ebx

mov [esi+edx], eax

;call writeInt

add edx, 4

mov ebx, temp

loop L1

pop ebp

ret

ArraySeries endp

main PROC

mov edx, offset word1

call writestring

call crlf

mov ecx, lenArray

mov edx, 0

L1:

mov eax, array[edx]

Call WriteInt

add edx, 4

loop L1

mov eax, offset array

mov ebx, lenArray

call crlf

push eax

push ebx

call ArraySeries

pop ebx

pop eax

call crlf

mov edx, offset word2

call writestring

mov ecx, lenArray

mov edx, 0

mov ecx, lenArray

mov edx, 0

mov eax, 0

call crlf

L2:

mov eax, array[edx]

Call WriteInt

add edx, 4

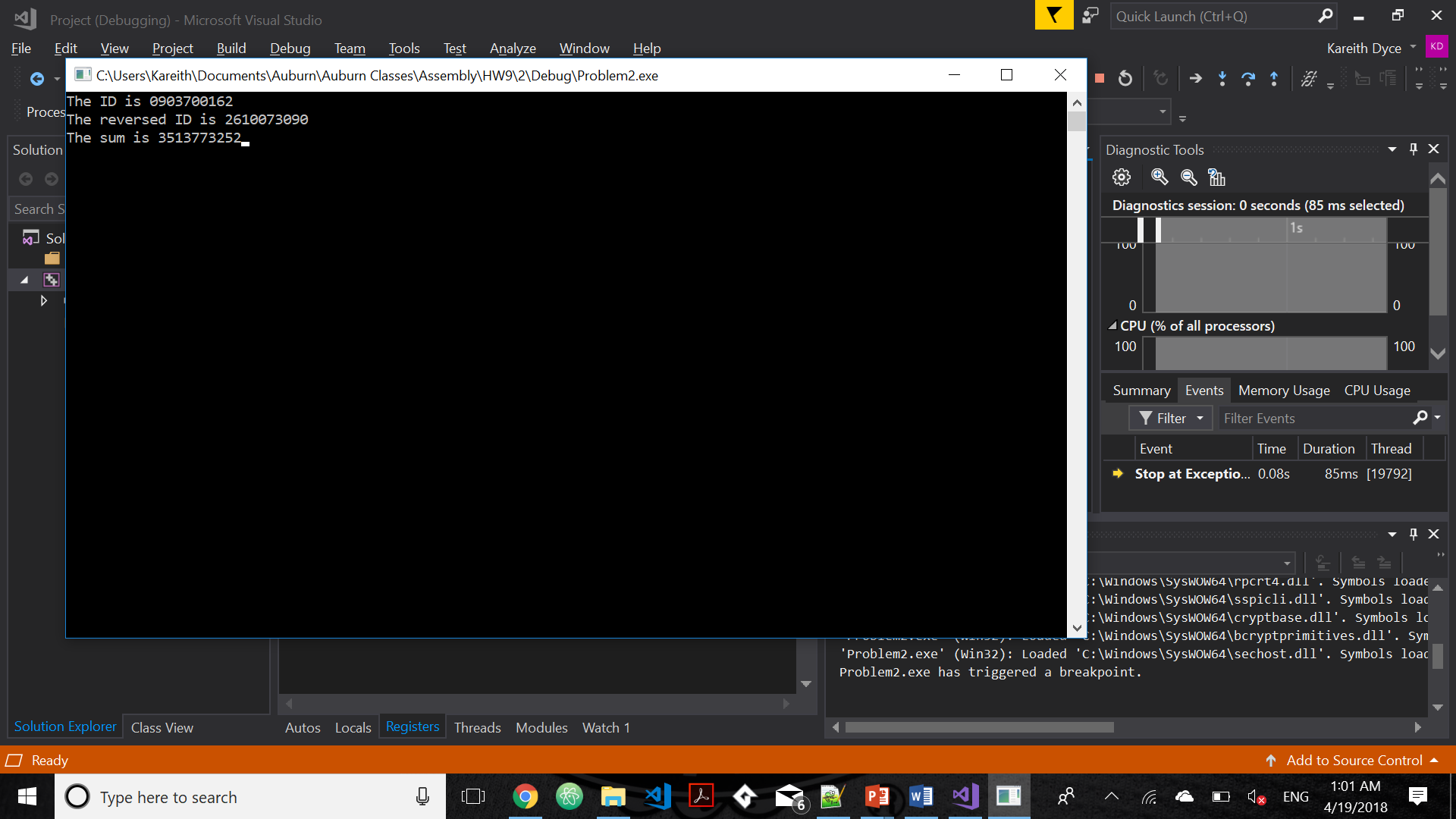
loop L2

invoke ExitProcess,0

main endp

end main

1. Draft a program that adds two BCD numbers (10-digits each). The first BCD number is stored in an array named *myAuburnID*, and the second in an array named *myAurbunIdRev*. The first number is your actual Auburn ID (with a prefix single zero digit and the remaining digits as the 9-digits of your *Auburn ID*); the second is the value of *MyAuburnId* written backwards. Your program should do the following:
2. Use shifts/rotates using *myAuburnID* to fill the array *myAuburnIdRev*
3. Display contents of the memory locations in question
4. Add *myAuburnID and myAurbunIdRev* using BCD arithmetic
5. Store the sum in a variable named *Result*, and
6. Display contents of memory post execution.



Include Irvine32.inc

.386

.stack 4096

ExitProcess proto,dwExitCode:dword

.data

ID BYTE "The ID is ",0

revId BYTE "The reversed ID is ",0

sum BYTE "The sum is ",0

myAuburnID BYTE 09h,03h,70h,01h,62h

myAuburnIDRev BYTE 5 DUP(?)

result BYTE 5 DUP(?)

len DWORD LENGTHOF myAuburnID

.code

main proc

mov edx, offset ID

call writestring

mov eax, 0

mov ecx, len

mov edx, 0

mov ebx, 1

L1:

mov al, myAuburnId[edx]

call writehexB

inc edx

loop L1

call crlf

mov edx, offset revID

call writestring

mov ecx, len

mov edx, 0

L2:

mov al, myAuburnId[ecx-1]

rol al, 4

mov myAuburnIdRev[edx], al

inc edx

loop L2

mov ecx, len

mov edx, 0

L3:

mov al, myAuburnIdRev[edx]

call writehexB

inc edx

loop L3

mov ecx, len

mov edx, 0

L4:

mov al, myAuburnIdRev[ecx-1]

add al, dl

mov dl, 0

add al, myAuburnId[ecx-1]

daa

mov result[ecx-1], al

jnc skip

mov dl, 1

skip:

loop L4

call crlf

mov edx, offset sum

call writestring

mov ecx, len

mov edx, 0

L5:

mov al, result[edx]

call writehexB

inc edx

loop L5

main ENDP

END main

Please embed your code into your homework solution along with a screen shot post execution.

1. Consider an isosceles triangle A with base 12 and height 20. Consider another triangle B formed using vertices which are the center of the sides of triangle A. Consider another triangle C whose vertices are similarly formed from B. Repeat this process ad infinitum. Express the sum of the areas of all such triangles using a series and its closed form sum. Compute the areas (a) by using only the first two terms of the series and (b) by using the closed form of the series sum. Write a program to find the sums and use shifts to compute. What is the difference in the two computed sums?

Please embed your code into your homework solution along with a screen shot post execution.

Include Irvine32.inc

.386

.stack 4096

ExitProcess proto,dwExitCode:dword

.data

method1 BYTE "The sum using method 1 is ",0

method2 BYTE "The sum using method 2 is ",0

diff BYTE "The difference between the two methods is ",0

base WORD 12

height WORD 20

sum1 WORD ?

sum2 WORD ?

.code

main proc

movzx ebx, height

movzx ecx, base

mov eax, ecx

mul bx

shr ax, 1

add sum1, ax

shr ax, 2

add ax, sum1

mov sum1, ax

mov edx, offset method1

call writestring

call writeint

call crlf

movzx ebx, height

movzx ecx, base

L1:

mov eax, ecx

mul bx

shr ax, 1

jz next

add sum2, ax

shr bx, 1

shr cx, 1

jmp L1

next:

movzx eax, sum2

mov edx, offset method2

call writestring

call writeint

call crlf

mov ax, sum2

sub ax, sum1

mov edx, offset diff

call writestring

call writeint

main ENDP

END main

