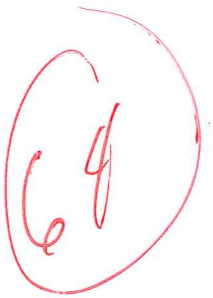
CSCI 2160
Exam 2
Fall 2011

Name: Buchanan Brandon William
print: Last First Middle



1. (5 pts) Suppose that you have the function whose prototype is defined as

```
int func(short sX, int iY, int iV)
```

Assume that all results will fit into the defined storage. (Be careful!).

```
sA    word  ?
bB    byte  ?
sC    word  ?
sD    word  ?
```

Write the assembly code call sequence which would accomplish the following tasks. (I deliberately did NOT put any casting information)

```
sD =  func(sA,sC,bB);
```

```
push dword ptr bB
push dword ptr sC
push sA
call func
add esp,10
mov sD,ax
```

can't do dword because you don't know the other 3 bytes

-3

2. (2 pts) Suppose that you have written a program **exam2.asm** that makes reference to the function **inorder** contained in the file **btree.obj**

a. What statement(s) must be supplied in the exam2.asm file that will allow your program to assemble properly?

extern inorder:Near32 -1

b. Where must the statement(s) be placed in the exam2.asm file?

beginning of file, near .Stack & .Model ✓

3. (2 pts) Suppose that there are a set of macros in a file named **macros.asm** and you wish to use them in a program **exam2.asm** that you have written.

a. What statement(s) would you need to put in exam2.asm to do this?

include macros.asm ✓

b. Where would these statement(s) be placed?

beginning of file, near .Stack & .Model ✓

4

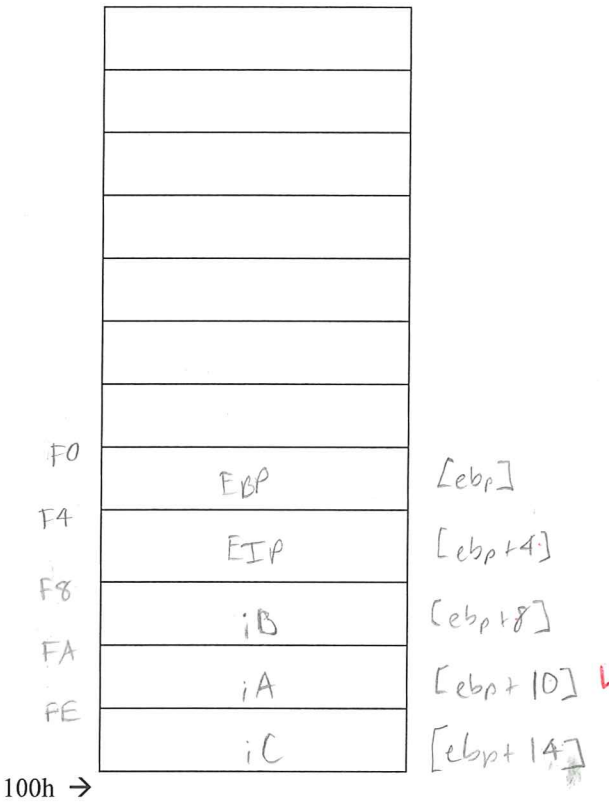
4. (8) Given the below function prototype generate the call sequence. Then, **label both sides** of the stack indicating how the stack pointer changes on the left and how the variables will be referenced as an offset from the base pointer on the right. Assume that you are to illustrate the stack after the mov EBP, ESP statement.

prototype: int func3(short sX, int iY, short sT)

```
iA      dword   ?
iC      dword   ?
iB      dword   ?
iZ      dword   ?

iZ = func3 ( iB, iA, iC);
```

```
push word ptr iC
push iA
push word ptr iB
call fun3
add esp, 8
mov iZ, eax
```



5. (3) What will be the contents (in HEX) of AX and DX after the last of the following operations? The instructions are executed in sequence – want the final results. Be sure to write EVERY hex digit in the register.

```
MOV   DX,55h
MOV   AX,32CEh
MOV   CX,10h
IMUL  CX
```

AFTER:

AX: 32CE0h

CX: 0010h

DX: 0003h

-0

6. (10 pts) Write the function **int FindMessage(node*, int)** where **node*** is the address of a node. The structure of the node

```
node   struct
errorkey    dword    ?          ; the error code to be looked for
strData     byte 100 dup (?)              ; the string to be displayed. It is null-terminated
next        dword      ?                  ;the address of the next node in sequence
node        ends
```

The method will be passed the address of the starting Node to look for the appropriate error message. For example, suppose that the string to be displayed is associated with error code 4562. The address the first error message (what is in startNode) and the errorCode is given to the method. Since errorCode 4562 IS the linked nodes, the address 4ED8 is returned. If the errorCode had not been in the linked list, then -1 would be returned. **DO NOT DISPLAY THE STRING WITHIN THE METHOD**

errorKey	strData	next
1123	Invalid social security number was entered	80E2

402C

1245	Missing the number of dependents	4ED8

80E2

4562	Address of recipient is invalid	F42C

4ED8

8712	Matching employee records do not exist	620C

F42C

9216	Employee has no security clearance	-1

620C

startNode	errorCode
402C	4562

```
findMessage  PROC
        push ebp
        mov ebp, esp
        push edx
        push esi
        mov esi, [ebp+8]        ; address of first node
        mov eax, -1             ; holds found node's address
errorCode equ [ebp+12]
stLoop:
        mov edx, (node ptr [esi]).errorKey
        cmp edx, errorCode
        je found
        mov edx, (node ptr [esi]).next
        cmp edx, -1
        je done
        mov esi, edx
        jmp stLoop
found:
        mov eax, esi
done:
        pop esi
        pop edx
        pop ebp
        ret
findMessage  ENDP
```

you don't "know" name

-1

-1

-2

7. (4) Generate a function call from the calling function that would determine whether the value of errorCode is in the list.Write the code that if it is in the list, display the message, if it is not in the list, display a message that it is NOT in the list. You may assume that you can use the macro **puts** that will display a null-terminated string on the screen. You may also assume that the address of the first node is in **startNode**

```
strNot byte "Not in the list.",0
mov esi, startNode
push esi
call findMessage
add esp, 4
cmp eax, -1
je NotInList
jmp InList

NotInList:
puts strNot
jmp done

InList:
puts (nodePtr [eax]).strData

done:
```

8. (6 pts) Given the contents of the registers and data segment: EAX: FF0C3E02
EBX: 00000E45 ESP: 0402AE2A (**NOTE: YOU MUST USE ESP FOR INIT value of stack ptr for each of a, b, c**)

0412 01B0	iA	dword	37
0412 01B4	iB	dword	25
0412 01B8	iC	dword	iA (reference to the memory location iB)

The partial contents of the stack are given below. I have only listed the last 5 hex digits of the even-numbered addresses because of space. Assume the values to the left of the variables above represent their true locations in memory.Assume, as well, that the hex value directly below a cell is that cell's address.

2A	E2	40	52	6A	5D	F2	02	41	FF	2C	B2	22	53	7E	42	87	A5	29	33	AA	E4
2AE20		2AE22		2AE24		2AE26		2AE28		2AE2A		2AE2C		2AE2E		2AE30		2AE32		2AE34	

Give the results **after** execution of the below statements. Assume in **EACH** case that the registers are as originally defined above.

a) (2 pts) POP EAX EAX: 52 40 E2 2A ESP: 2A E2 4

b) (2 pts) PUSH iC ESP: 2A E1 C

c) (2 pts)Show below what bytes were affected in b) by crossing out and writing above the addresses the new values:

00 01 12 04

~~2A~~	~~E2~~	~~40~~	~~52~~	6A	5D	F2	02	41	FF	2C	B2	22	53	7E	42	87	A5	29	33	AA	E4
2AE20		2AE22		2AE24		2AE26		2AE28		2AE2A		2AE2C		2AE2E		2AE30		2AE32		2AE34	

9. (10 pts) In the below problems, write the appropriate assembly commands for each calculation. Assume the prefix indicates the size of the variable. I (int), s (short), b (byte), q (quad word).

a) (4) iA = qD/sB

```
mov ebx, dword ptr sB
mov edx, dword ptr qD
mov eax, dword ptr qD+4
idiv ebx
mov iA, eax
```

No

OK

movsx ebx, sB — 1

b) (4) qF = sC * sE

```
mov eax, sC
imul sE
cdq
mov qF, edx
mov qF+4, eax
```

answer in dx:ax pair

convert eax to edx:eax pair

-2

c) (2 pts) sVal = iVal + bVal;

```
mov ax, word ptr iVal
add ax, word ptr bVal
mov sVal, ax
```

-2

10. (4 pts)Circle the correct response either T or F below:

a) T (F) When a macro is invoked, the CALL and RET instructions are automatically inserted into the assembled program.

b) (T) F Macro expansion is handled by the assembler's preprocessor..

c) T (F) As long as it is in the code segment, a macro definition may appear either before or after statements that invoke the macro.

d) (T) F Replacing a long procedure with a macro containing the procedure's code will typically increase the compiled code size of a program if the macro is invoked multiple times.

-5

11. (2 pts) For each of the following describe every hex digit in the resulting register

```
mov     bx,0C240h
movzx eax,bx            EAX: 0000C240h
movsx eax,bx            EAX: FFFFC240h
```

12. (3 pts) Given the below data segment:

```
.data

bVal   byte  ?
sVal   word  ?
iVal   dword ?
sArray       word  1,2,3,4,5
iArray1      dword 10,20,30,40,50
iArray2      dword 500 dup (?)
```

Suppose that you have written: `mov ebx,offset sArray`

Write one correct assembly statement referencing ebx that will replace the value 30 with 34 in the above data area.

```
mov [ebx+18], 34
```

dword ptr -2

13. (7) Write a recursive method called **calcSum** that will return the sum of the numbers from 1 to N where N is an int that is passed to the method.

```
calcSum PROC
    push ebp
    mov ebp, esp
    push edx
    XOR edx, eax          -1
    mov edx, [ebp+8]
    cmp edx, 0
    jle done
    dec edx
    push edx
    call calcSum
    add esp, 4            -1
    add eax, edx [ebp+8] ; because you have already decremented edx
done:
    pop edx
    pop ebp
    ret

calcSum ENDP
```

-4

14. (2) Write a single instruction that converts an Uppercase character in AL to lowercase but does not modify AL if it already contains a lowercase letter.

and al, 11011111b

OR al, 00100000b

Makes all upper lower case

15. (2) Write instructions that set the Zero Flag if the 32-bit value in EAX is even and clear the Zero flag if EAX is odd. You may NOT change the value in EAX.

test al, 00000001b

16.(3 pts) Given the below data segment:

```
        .data
bVal        byte  ?
sVal        word  ?
iVal        dword ?
sArray      word  1,2,3,4,5
iArray1     dword 10,20,30,40,50
iArray2     dword 500 dup (?)
```

After each of the below instructions is executed describe what is in the EAX register. Be sure to write EVERY hex digit.

```
mov   eax,  TYPE sArray        EAX: ____00000002h____
mov   eax,  LENGTHOF iArray1   EAX: ____00000005h____
mov   eax,  SIZEOF iArray2     EAX: ____0000 07CE____
```

17.(3) Suppose that the address of the **je done** instruction is at 004004C2, at that the label **done** is at address 0040042E. Given that the op code of **je** is 74 what is the rest of the instruction?

74 (94)

18, (4) Given the data segment defined below, write code that could be generated using the below .IF directive.

```
      .data
bVal        byte  ?
wVal        word  ?
dVal        dword ?
wArray      dword 1,2,3,4,5

      .code

mov   eax, wArray
.IF eax <= dVal
      mov   eax, 5
.ELSE
      mov   eax, 10
.ENDIF
```

```
cmp eax, dVal
jle makeS
jmp make10

makeS:
    mov eax, 5
    jmp done
make10:
    mov eax, 10

done:
```

jbe -1

19. (12 pts each) Given the contents of the below registers (in hex). Give what will be in the register as a result of the instructions:

CX: 69 4A sal cx,1 CX: D294 CF 0 OF 0

0110 1001 0100 1010

1101 0010 1001 0100

D 2 9 4

ZF: 0 SF: 1

AX: 69 4A sar ax,1 AX: 34A5 CF 0 OF 0

0011 0100 1010 0101

3 4 A 5

ZF: 0 SF: 0

-2

BX: 69 4A shl ax,2 BX: A528 CF 1 OF 0

1010 0101 0010 1000

A 5 2 8

undefined

ZF: 0 SF: 1

AX: 69 4A sar ax,cl AX: 06 94 CF 0 OF 0

CL: 04

06 94

undefined

ZF: 0 SF: 0

-3

20. (2) Write a sequence of shift instructions that cause AX to be sign-extended into EAX. In other words, the sign bit of AX is copied into the upper 16-bits of EAX. (Do NOT use the cwd instruction).

```
push ax
shl ax
pop ax
cmp byte ptr eax+1, 1
je signed
jmp done
signed:
    mov byte ptr eax, 0FFh
    mov byte ptr eax+1, 0FFh
done:
```

21. (3) Demonstrate using assembly code when you get a different answer using shifting instructions to divide than when you use the IDIV instruction.

```
mov eax, 9
sar eax, 1
```

```
mov eax, 9
div 2
```

22. (4 pts) Given the below function write the assembly code for the part of the method below that is given to you. Give the statements at the beginning and at the end of the procedure.

```
public int  function (int A, int B, int C)
{
       int  X, Y;
       short  U,V;


}
```

```
function PROC
    push ebp
    mov ebp, esp

    X equ dword ptr -1
    Y equ dword ptr -1
    U equ word ptr -1
    V equ word ptr -1

    pop ebp
    ret
function ENDP
```