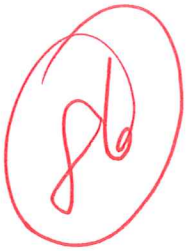
CSCI 2160

Exam 1

Fall 2011



Name: Buchanan Brandon W

Please print: Last First MI

Complete the below table:

1. (3pts)

Character	Hollerith Code (punches)	AScii Code in **hex**
H	12-8 ✓	48h ✓
T	0-3 ✓	54h ✓
6	6 ✓	36h ✓

A=12-1 A=41h
J=11-1 J=4Ah
S=0-2 P=50h

For Questions 2,3,4 use the portion of the data segment given in #2

2. (6pts -each answer worth 1 pt) Given the below portion of a data segment of a program, determine the values for the location counter and write the **actual** bytes in memory.

```
02DC  sX     word   13,-1          0D 00 FF FF ✓
02E0  dY     dword  2 dup(37)      25 00 00 00 25 00 00 00 ✓
02E8  strPr  byte   "CDE12",34     43 44 45 31 32 22 ✓
02EE ✓
```

3. (3 pts) Assuming that the following instructions have occurred, give the value that will be in the AX register after the instruction is executed. Assume for all questions that the given two instructions had just been executed . Be sure to reference #2 where necessary and note the question is asking what is in **AX** register.

```
    MOV  EBX, offset dY
a)  MOV  AX,[EBX]          00 25 ✓
b)  MOV  AX,[EBX+4]        00 25 ✓
c)  MOV  AX, sX[1]         FF 00 ✓
```

10

4. (5 pts) Given that ints are 4 bytes, shorts are 2 bytes, write the assembly code generated by the below statements (i.e. what does the data segment look like?)

```
byte  bV;                    bV byte ?
short sT;                    sT word ?
int   iV;                    iV dword ?
int[] iZ = new int[4];       iZ dword 4 dup(?)
short[]  sVal = {-1,15,-2);  sVal word -1, 15, -2
```

5. (1) Given the above definitions, write the code generated in assembly by the Java statement:

```
sT = iV + sVal[2];
```

```
mov ebx, iV
mov ax, sVal+2
cwde
add eax, ebx
mov sT, ax
```

6. (4 pts) Given that the prefix indicates the type of the variable, write the code generated by each of the below Java expressions:

a) `sX += bVal;`

```
; sX = sX + bVal
mov bx, sX      ; bx = sX
mov al, bVal    ; al = bVal
cbw             ; ax
add ax, bx      ; ax = ax + bx
mov sX, ax      ; sX = ax
```

b) `iV = sX * bVal;`

```
mov al, bVal    ; al = bVal
cbw             ; ax = bVal
imul sX         ; eax = sX * ax
mov iV, ax      ; iV = eax
```

c) `sX = bVal - iNum;`

```
mov al, bVal
cbw
cdwe
sub eax, iNum
mov sX, ax
```

d) `iX = sX * sY;`

```
mov ax, sX
imul sY
mov iX, eax
```

7. (2) What are the 32-bit index registers? esi, edi

8. (1 pt)What is the hexadecimal representation of each of the following binary numbers:

a) 1100 1111 0101 0111 CF57

9. (1 pt) What is the 8-bit binary (2's complement) representation of each of the following signed decimal integers?

a) -5 FFFB

FB -1/2

Use the below assembly listing of **exam1.lst** to answer the next set of questions.

```
                        C
                          .stack 50h
                          ExitProcess     PROTO NEAR32 stdcall, dExitCode:DWORD
                          extrn ascint:near32, intasc:near32
00000000                      .data
00000000 42                   charInput   byte    'B'
00000001 53 74 72 69 6E       strOutput   byte    "String to print",0
     67 20 74 6F 20
     70 72 69 6E 74
     00
00000011 0A 0D 00             crlf        byte    10,13,0
                 ;strInput    byte    20 dup(?)
00000014 0A 0D 45 6E 74       strPrompt   byte    10,13,"Enter a string (then ENTER): ",0
     65 72 20 61 20
     73 74 72 69 6E
     67 20 28 74 68
     65 6E 20 45 4E
     54 45 52 29 3A
     20 00
00000034 01F403BA             iPosNum     dword   32768954
00000038 FB7008DE             iNegNum     dword   -76543778
0000003C  00000014 [          strNum      word    20 dup(?)
        0000
       ]
00000064 2D 31 32 00          strInput    byte    "-12",0
00000068 0A 0D 4F 76 65       strOverflow byte    10,13,"Overflow occurred",0
     72 66 6C 6F 77
     20 6F 63 63 75
     72 72 65 64 00
0000007C 0A 0D 49 6E 76       strInvalid  byte    10,13,"Invalid numeric string",0
     61 6C 69 64 20
     6E 75 6D 65 72
     69 63 20 73 74
     72 69 6E 67 00
00000095 FE                   bVal        byte    -2
00000096 7F 61                sVal        byte    127,97
00000098 FF                   sVal2       byte    255
00000000                          .code
00000000                 _start:
00000000  0F BE 05                movsx   eax, bVal
```

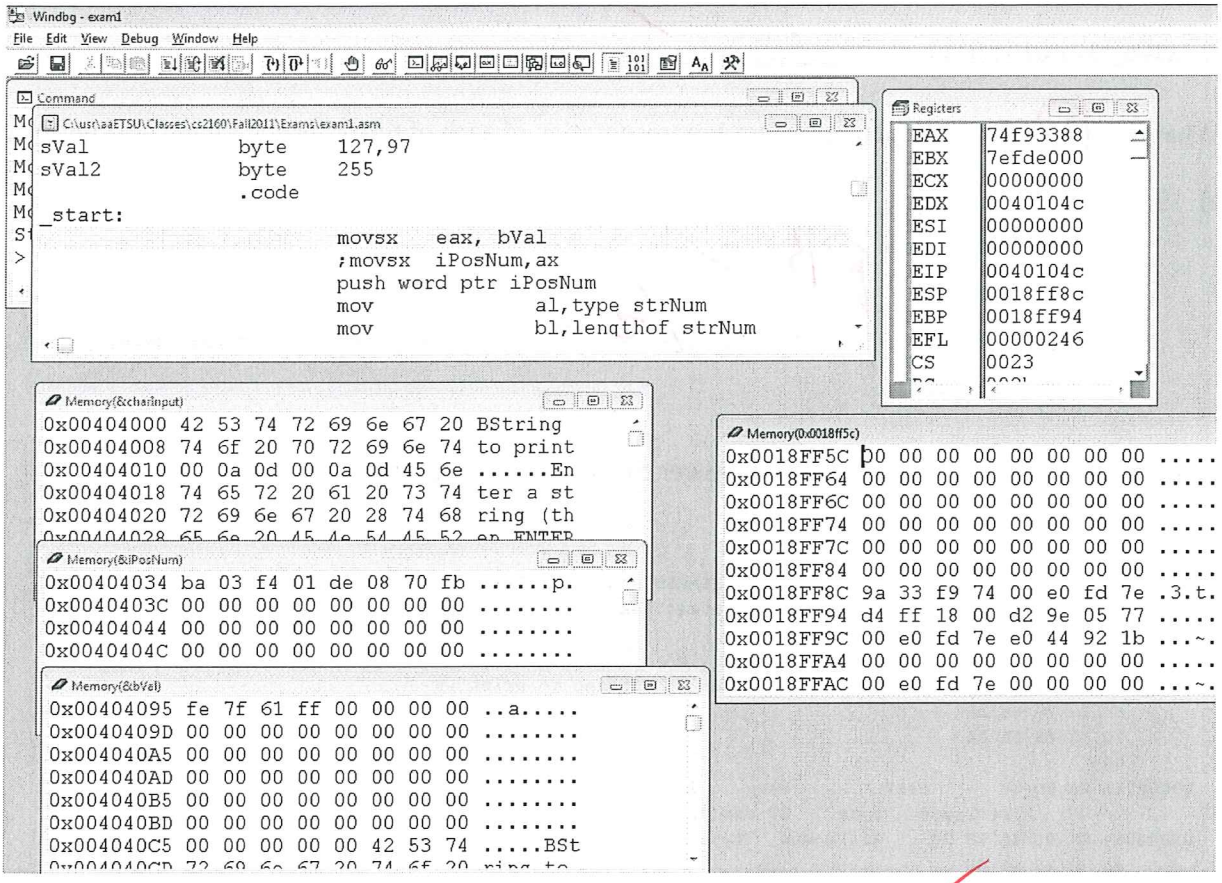
ormal text file 14826 chars 15588

10. (2 pts)What is the size of the

a) stack in hex ? 50h b) data segment in hex? 99h

-1/2

11. Given the below screenshot of windbg running against exam1.exe answer the set of questions below:



a) (1 pt) What is the starting address of the code segment? 0040104C

b) (1 pt) What is the address of the first byte beyond the TOP of the stack? 0018FF8C

c) (1 pt) What is the address of the very FIRST byte at the BOTTOM of the stack? 0018FF3C

d) (1 pt) What is the address of the beginning of the ~~code~~ data segment? 00404000

e) (1 pt) Given the state of the registers above, Show memory look like after the below instruction is executed: **push ebx** (I took the liberty of "drawing" the individual bytes of memory. You need to label the first byte of memory that is affected by the instruction, and what the contents of the following bytes would be.

address of the low-order byte: 0018FF88 →

00	E0	FD	7E	9A	33	F9

f) (3 pts) What would the contents of the receiving operand in each of the following. You may assume that **before** each instruction below that the below assembly instruction was executed.

mov ebx, offset iPosNum

mov ax, [ebx] ax: 03 ba

-0

mov eax, [ebx] eax: 01 f4 03 ba

mov eax, word ptr [ebx +1] eax: f4 03

... word ptr into eax... is that valid?

12.(2) Given the below portion of a code segment, what is the size of the code segment? 1E3h

```
C:\Users\aaETSU\Classes\cs2165\Fall2011\Exams\exam1.lst - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
exam1.lst

00000169  B8 FFFFFFFF                    mov     eax,-1
0000016E  B8 0000000A                    mov     eax,10
00000173  68 00000014 R                  push    offset strPrompt
00000178  E8 00000000 E                  call    PutString
0000017D  83 C4 08                add    esp,8
00000180  50                      push   eax     ;get a string of max 10 characters
00000181  68 00000064 R                  push    offset strInput     ;address of string to insert characters into
00000186  E8 00000000 E                  call    GetString           ;get the string
0000018B  83 C4 08                add    esp,8               ;restore the stack pointer

0000018E  68 00000011 R                  push    offset crlf         ;go to the beginning of a newline
00000193  E8 00000000 E                  call    PutString
00000198  83 C4 04                add    esp,4               ;restore the stack pointer

0000019B  68 00000064 R                  push    offset strInput     ;display the string back
000001A0  E8 00000000 E                  call    PutString
000001A5  83 C4 04                add    esp,4
000001A8  EB 32                   jmp    done

000001AA  E8 00000000 E                  call    GetCharecho
000001AF  A2 00000000 R                  mov     charInput,al
000001B4  A0 00000000 R                  mov     al,charInput
000001B9  50                      push   eax
000001BA  E8 00000000 E                  call    PutChar
000001BF  83 C4 04                add    esp,4
000001C2  68 00000011 R                  push    offset crlf
000001C7  E8 00000000 E                  call    PutString
000001CC  83 C4 04                add    esp,4
000001CF  68 00000001 R                  push    offset strOutput
000001D4  E8 00000000 E                  call    PutString
000001D9  83 C4 04                add    esp,4

000001DC          done:

                        INVOKE  ExitProcess,0
000001E3                    PUBLIC  _start

                        end
Microsoft (R) Macro Assembler Version 6.11          10/05/11 17:40:26
exam1.asm                           Symbols 2 - 1
Normal text file                    14612 chars  15360 bytes  375 lines   Ln:1  Col:1  Sel:0 (0 bytes) in 0 ranges   Dos\Windows  ANSI  INS
```

13. (2 pts) What is the decimal representation of each of the following **unsigned** binary integers?

a) 11111000 /5

248

b) 11110000 /4

240

14. (2 pts) What is the decimal representation of each of the following **signed** binary integers?

a) 11111000 = F8

−8

b) 11110000 = F0

−16

15. (5 pts) Assume that **al** and **bl** contain each of the binary values below. Determine what is in the destination register after the assembly instruction has been executed: Determine, too, what the flag settings are.

al: 11010101 **bl:** 01101011
D 5 6 B

D5
+6B
140

add al, bl al: 40 ✓ ok

CF: 1 SF: 0 OF: 1 ZF: 0
0 -1

Each of the following 17-28 is worth 1 point

-1

16. What are the valid suffix characters used in integer constants? d, o, b, h, (q, y)

17. (T) F A5h is a valid hexadecimal constant.

18. T (F) The multiplication operator (*) has a higher precedence than the division operator (/) in integer expressions.

19. T (F) String constants must be enclosed in double quotes.

20. (T) F An identifier cannot begin with a numeric digit.

21. (T) F Assembly language identifiers are by default case insensitive.

22. (T) F Assembler directives execute at runtime.

23. (T) F Assembler directives can be written in any combination of uppercase and lowercase letters.

24. (T) F A code label is followed by a colon ; but a data label does not have a colon.

25. Write a constant expression that divides 10 by 3 and returns the integer remainder.

x = 10 % 3 ✓

26. Why would it NOT be a good idea to use numeric addresses when writing instructions that access variables?

When the program is loaded, offsets are often used. Using a numeric address may mean that the program is looking at a different part of memory than it should be. ✓

27. What is the purpose of the END directive?

To declare the end of the assembly code procedure. -1

28. (2pts) If the source code is successfully assembled, the assembler produces two files: one automatically and one optionally? Give a brief description of each.

The object file contains the assembly program in machine code and is automatic.
The list file contains data and code segment information and is optional.

29.(1 pt)Which operating system component reads and executes programs?

the loader

30. (2 pts) Which operating system component is the second in the cycle (if assembling is first)? linker What file does it produce? the executable program

31.(4 pts) Create an uninitialized data declaration for

I make this mistake all the time, so I am not going to penalize you for it!

an 8-bit integer bX byte ?

a 16-bit integer wX word ?

a 32-bit integer dX dword ?

a 64-bit integer qX qword ?

32. (1 pt) Declare a 16-bit integer variable named **sArray** that references the array 20,20,42,46

sArray word 20,20,42,46

33. (1 pt) Declare a string variable containing the name of your favorite color. Initialize it as a null-terminated string

strColor byte "the name of your favorite color",0 lol

34. (1 pt) Declare an uninitialized array of 50 doublewords named **dArray**

dArray dword 50 dup(?)

-0

35. (1 pt) Declare a string variable containing the word **TEST** repeated 500 times

strTest byte 500 dup("TEST") ✓

36. (1 pt) Show what the individual bytes in memory (lowest to highest) for the following doubleword variable

```
dVal1 DWORD 87654321h
```

21 43 65 87 ✓

37. (1 pt) Declare a symbolic constant using the equal-sign directive that contains the ASCII code (08h) for the Backspace key.

BACKSPACE = 08h ✓

38. (1 pt) Declare a symbolic constant named **secondsInDay** using the equal-sign directive and assign it to an arithmetic expression that calculates the number of seconds in a 24-hour day

secondsInDay = 60 x 60 x 24 ✓

39. (1 pt) Write a statement that causes the assembler to calculate the number of bytes in the following array, and assign the value to a symbolic constant named **ArraySize.**

```
myArray word 20 dup(?)
```

ArraySize = sizeof myArray ✓

good!

40. (1 pt) Write the assembly instruction which would set eax to the number of elements in myArray (You may NOT write

```
mov eax, 40)
```

mov eax, lengthof myArray ✓

41. (2 pts) Assuming that a 32-bit register is numbered from left-to-right as 0, 1,2, 3, ...,31 Give the positions within the EFLAGS register for each of the below flags:

CF: 0 SF: 7 ZF: 6 OF: 11 ✓

yes! c1

0 carry
2 parity
6 zero
7 sign
11 overflow

0

42 (4 pts). Assume that the registers and variables below contain the given values.

```
iVal    dword   17,22,45
sNum    word    -1,4,-5
bVal    byte    23,37,9
```

EAX: 1234C6E8h EDX: F2C564A7h

movsx	eax, sNum	eax: FF FF FF FF
movzx	eax,sNum	eax: 00 00 FF FF
mov	eax, iVal +2	eax: 2D 00 00 00 -1
mov	al, bVal+2	ax: C6 09

43. (9 pts) Use the following variable definitions for the questions below:

```
        .data
bVar1   byte    -4,-2,3,1
sVar2   word    1000h, 2000h, 3000h, 4000h
sVar3   word    -16,-42
dVar4   dword   1,2,3,4,5
```

For each of the following statements, state valid if the instruction is **valid**, and **invalid**. If the response is INVALID, give the reason why.

a)	mov	ax, bVar1	invalid - ax is 2 bytes, bVar1 is 1 byte
b)	mov	ax, sVar2	valid
c)	mov	eax, sVar3	invalid - eax is 4 bytes, sVar3 is 2 bytes
d)	mov	sVar2, sVar3	invalid - cannot do memory to memory operations
e)	mov	eax, dVar4	valid
f)	mov	word ptr eax, sVar2	invalid - cannot use ptr on registers
g)	movzx	eax, bVar1	valid
h)	movsx	dVar4, ax	valid -1
i)	mov	eax, dword ptr bVar1	valid

-2

44. (3 pts) Assume that you have the message **strMsg** that you want displayed 100 times. You can use the macro **puts msg** that will display the string **msg** on the screen. Write the code that would do this. (You may assume, as well, that CRLF is a null-terminated characters string for advancing to a newline)

```
mov ecx, 100
msgLoop:
    puts strMsg
    puts CRLF
    loop msgLoop
```

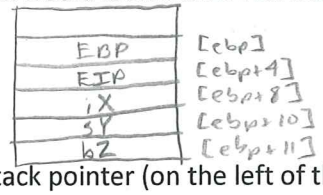
45. (5)Suppose that you have the function whose UML description is below:

```
funct(int iX, short sY, byte bZ):short
```

Assume that the prefix indicates the type of variable that the program is "trying" to pass to the method (I deliberately did not put any casting information). Write the assembly code call sequence which could be generated by the compiler to accomplish the following java statement

Draw a picture of the stack and show what the stack would look like **AFTER executing the first 2 instructions in the method:**

```
push ebp
mov  ebp,esp
```



Draw the stack and label both the changes in the stack pointer (on the left of the stack) and label how the procedure "sees" the stack (on the right with reference to EBP).

```
dVal =  funct(sA,bB,sC);      ;assuming the stack is 100h bytes
```