CSCI 2450 Assembler Midterm

October 14th, 2010

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Instructions**: There are 6 questions on this test, 20pts each, which may provide you with 120 points out of possible 100.

1a(15pts). Identify errors within the executable code bellow, and then provide the corrected code.

.DATA

buffer BYTE “Yo mama jokes here”,0

bufSize WORD $-buffer-1

KEY = 245

.......

.......

TranslateBuffer PROC

pushads

mov ecx,bufSize

mov esi,0

L1:

xor buffer[esi],KEY

inc si

loop jmp L1

popad

TranslateBuffer ENDP

1b(5pts). In not more than two sentences explain to best of your abilities what the function above does.

2. Given the BYTE array “destination”, write an ASM procedure which looks for a character given by the variable “char” displaying the position number of that character on the screen.

destination BYTE “Here I am looking for a letter A”,0

char BYTE “A” ; *ASCII A is equivalent to 65 decimal*

3. Given the assembly code and the opcodes for Intel instructions, draw a memory image representing how **executable** code below is stored in memory using HEX values.

counter BYTE 30h

**MOV cl, counter**

**L1: MOV eax, 0DEADBEEFh**

**ADD ax, 0F00Fh**

**LOOP L1**

Opcodes:

MOV eax, imm/mem32bit => B8 imm/mem32bit

MOV cl, imm/mem8bit => B0 imm/mem8bit

ADD ax, imm/mem16bit => 05 imm/mem8bit

LOOP => E2

Remember that a location of L1 is equivalent to negative $-(amount of bytes from L1 to the loop)

4a. List all general purpose registers we covered in class, and explain what their purposes are **in one sentence**.

4b. List at least 4 non-general purpose registers. Describe their purposes **in one sentence**.

5. Convert the following arithmetic into **assembly level code procedure** which accepts Xval, YVal and Zval required registers to be and returns Rval using the same principle.

**Rval = Xval - (-Yval – Zval)**

6. What are the values of each register after each line of execution?

**.data**

**val1 BYTE 10h,20h,30h**

**arrayW WORD 1000h,2000h,3000h**

**.code**

**mov esi,OFFSET val1 ESI:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov al,[esi] AL:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**inc esi ESI:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov al,[esi] AL:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**inc esi**

**mov eax, esi EAX:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov esi, 0**

**mov ax,[arrayW + esi] AX:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov ax,arrayW[esi] AX:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov esi, 2**

**mov al, val1[esi\*TYPE val1] AL:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov bx, arrayW[esi\*TYPE arrayW] BX:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**mov dx, WORD PTR arrayW DX:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**