Dec. 18, 2010

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Major Exam II

First Semester (101)

Time: 8:15 PM-10:30 PM

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

Student ID. : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Question** | **Max Points** | **Score** |
| **Q1** | **30** |  |
| **Q2** | **36** |  |
| **Q3** | **10** |  |
| **Q4** | **24** |  |
| **Total** | **100** |  |

Dr. Aiman El-Maleh

# **[30 Points]**

# **(Q1)** Fill the blank in each of the following:

## Assume that ESP=0000012FH, AX=1234H and BX=5678H. Assume that the address of MPROC is 00200FFA. After executing the instruction sequnece{PUSH AX, PUSH BX, CALL MPROC } the content of ESP=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

## Assume that ESP=0000012FH. After executing the instruction RET 4, the content of ESP=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

## Assuming that register AL contains an alphabatic character, to convert the content of register AL to upper case, we use the following instruction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

## The code to Jump to label L1 if regiser AL bits 0 and 2 are 1 or bits 1 and 4 are zero is:

## 

## Assuming that EAX=12345678H and ECX=9ABCDEF0H, executing the instruction SHLD EAX, ECX, 12 will set EAX=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and ECX=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

## To multiply the **signed** content of register EAX by 14.5 without using multiplications instructions, we use the following instructions:

## Assuming that all variables are 32-bit signed integers, the assembly code implementing the following equation **var3 = -4\*(var1+2)/(15-8\*var2)** is:

## Suppose that we have a 64-bit number stored in memory in the variable I defined as I Qword. The assembly code to multiply this number by 9 is:

## Given that the CPU is receiving a word in AX register from the printer. Assume that bits 5 to 10 represent a number. The assembly code to display the decimal value of this number is:

## Suppose that we would like to translate 8-bit numbers into characters according to a given tanslation table. Part of the translation table is shown below. The assembly code to translate a number in register AL according to the translation table below and store the resulting character in the same register is:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | … |
| ‘e’ | ‘A’ | ‘y’ | ‘Z’ | ‘o’ | ‘B’ | ‘1’ | ‘!’ | ‘h’ | … |

## Assuming signed operands, the assembly code to implement the high-level statement if ( ( (AL ≥ BL) && (BL ≠ CL) ) || (AL<100) ) {X = X+1;} is:

## We can define the macro SAVE\_REGS to save only the registers passed as arguments by pushing them on the stack as follows:

**[36 Points]**

# **(Q2) Answer the following questions. Show how you obtained your answer:**

## **(i)** Given the following definition in the data segment:

## **Array dword 0, 1, 2, 3, 4**

## **dword 5, 6, 7, 8, 9**

## **dword 10, 11, 12, 13, 14**

## Determine the content of Array after executing the following code**:**

MOV ECX, lengthof Array

MOV ESI, 1\*sizeof Array

MOV EDI, 2\*sizeof Array

XOR EBX, EBX

Next:

MOV EAX, Array[ESI+EBX\*4]

XCHG EAX, Array[EDI+EBX\*4]

MOV Array[ESI+EBX\*4], EAX

INC EBX

LOOP Next

## **(ii)** Determine the content of EBX after executing the following code**:**

MOV EAX, 7

CALL MyProc

MyProc PROC

CMP EAX, 2

JAE Next

MOV EBX, EAX

RET

Next:

PUSH EAX

SUB EAX, 2

CALL MyProc

POP EAX

ADD EBX, EAX

RET

MyProc ENDP

## **(iii)** Given the following definition in the data segment:

## Array Dword 4, 2, 1, 3

## Determine the content of Array after executing the following code**:**

MOV ESI, lengthof Array

DEC ESI

MOV EDI, ESI

DEC EDI

XOR ECX, ECX

FLOOP:

CMP ECX, EDI

JG EFLOOP

MOV EAX, Array[ECX\*4]

MOV EDX, ECX

MOV EBX, ECX

INC ECX

FLOOP2:

CMP ECX, ESI

JG EFLOOP2

CMP Array[ECX\*4], EAX

JGE EIF

MOV EAX, Array[ECX\*4]

MOV EDX, ECX

EIF:

INC ECX

JMP FLOOP2

EFLOOP2:

MOV ECX, EBX

CMP ECX, EDX

JE EIF2

MOV EBP, Array[ECX\*4]

MOV Array[EDX\*4], EBP

MOV Array[ECX\*4], EAX

EIF2:

INC ECX

JMP FLOOP

EFLOOP:

## **(iv)** Given that **TABLE1** and **TABLE2** are defined as:

## **TABLE1 BYTE ‘This is COE 205’**

**TABLE2 BYTE ‘This is COE 308’**

## Determine the content of **AX** after executing the following code:

MOV ECX, lengthof TABLE1

MOV EBX, -1

XOR AX, AX

AGAIN: JECXZ DONE

INC EBX

MOV DL, TABLE1[EBX]

CMP DL, TABLE2[EBX]

LOOPE AGAIN

JE DONE

INC AX

JMP AGAIN

DONE:

## **(v)** Determine what will be displayed after executing the following code**:**

PUSH 23

PUSH 5

CALL Test

Test PROC

MOV EBX, [ESP+4]

MOV EAX, [ESP+8]

XOR EDX, EDX

DIV EBX

CALL WriteDec

MOV AL, '.'

CALL WriteChar

MOV EAX, 10

MUL EDX

DIV EBX

CALL WriteDec

RET 8

Test ENDP

## **(vi)** Determine what will be displayed after executing the following code:

MOV EAX, 5

MOV EBX, 9

XOR ECX, ECX

Next:

SHR EBX, 1

JNC Skip

ADD ECX, EAX

Skip:

SHL EAX, 1

CMP EBX, 0

JNE Next

MOV EAX, ECX

## Call WriteDec

**[10 Points]**

# **(Q3)** Write a macro, **DigitSum**, to compute and display the sum of the digits of a number **n** passed as a aparmeter to the macro. The macro should preserve the content of all temporary registers used. Then, use the macro to compute the sum of the digits in the number 123. Your macro should display the result as 6 since 6=1+2+3.

**[24 Points]**

# **(Q4)**

# **(i)** Write a procedure, **MatrixMul**, to multiply two matrices of integers where Matrix1 is of size R1xC1 integers and Matrix2 is of size R2xC2 integers. If C1≠R2, then the two matrices cannot be multiplied and the procedure prints a statement indicating that the two matrices cannot be multiplied. Otherwise, the procedure computes the resultant matrix and displayes it. The addresses of the two matrices and their dimensions are assumed to be passed on the stack. The procedure should maintain the content of all registers to their state before its execution.

The pseudocode for multiplying two matrices **m1** and **m2** and storing the result in matrix **mult** is given below:

for(i=0; i<R1; i++){  
       for(j=0; j<C2; j++){  
                mult[i][j]=0;  
                for(k=0; k<C1; k++){  
                    mult[i][j] = mult[i][j]+m1[i][k]\*m2[k][j];  
                }  
       }

}

# **(ii)** Write a complete program, showing the place of procedure definition, to use the procedure **MatrixMul** to multiply the two matrices given below:

M1 dword 1, 2, 3

dword 4, 5, 6

M2 dword 1, 2

dword 3, 4

dword 5, 6

Your program should display the result of matrix multiplication as:

22 28

49 64