Jan. 7, 2010

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Major Exam II

First Semester (091)

Time: 3:30 PM-6:00 PM

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

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

|  |  |  |
| --- | --- | --- |
| **Question** | **Max Points** | **Score** |
| **Q1** | **34** |  |
| **Q2** | **42** |  |
| **Q3** | **24** |  |
| **Total** | **100** |  |

Dr. Aiman El-Maleh

# **[34 Points]**

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

## Assume that ESP=00000020H and EAX=12345678H. Assume that the address of MPROC is 0010005E. After executing the instruction sequnece{PUSH EAX, CALL MPROC } the content of ESP=ESP-8=00000018H.

##  Assume that EAX=12345678H and EBX=90ABCDEFH. After executing the following sequence of instructions, the content of EAX=90ABCDEFH and EBX=12345678H.

PUSH EAX

PUSH EBX

POP EAX

POP EBX

##  The code to Jump to label L1 if bits 1 and either bit 3 or bit 5 in AL are zero is:

Test AL, 00100010b

JZ L1

Test AL, 00001010b

JZ L1

##

## Assuming that EAX=8765432CH and ECX=FEDBA7E4H, executing the instruction SHL EAX, CL will set EAX=765432C0H and CF=0.

##

## Assuming that EAX=8765432CH, executing the instruction SAR EAX, 4 will set EAX=F8765432H and CF=1.

## Assuming that EAX=8765432CH, executing the instruction ROL EAX, 8 will set EAX=65432C87H and CF=1.

## Assuming that EAX=8765432CH and ECX=FEDBA7E4H, executing the instruction SHLD EAX, ECX, 12 will set EAX=5432CFEDH and ECX= FEDBA7E4H.

##  To multiply the content of register EAX by 35.5 without using multiplications instructions, we use the following instructions:

MOV EBX, EAX

MOV ECX, EAX

SHL EAX, 5

SHL ECX, 2

ADD EAX, ECX

SAR EBX, 1

SUB EAX, EBX

## Assuming that AX=FFF0H and BX=FFF9H, executing the instruction IDIV BL will result in AX=FE02.

## Assuming that AX= FFF0H and BX= FFF8H, executing the instruction IMUL BX will result in AX=0080H and CF=0.

## Assuming that all variables are 32-bit signed integers, the assembly code implementing the following equation **var5 = (-3\*var1\*var2)/(4\*var3 + var4)** is:

MOV EAX, var1

MOV EDX, EAX

SHL EAX, 1

ADD EAX, EDX

NEG EAX

IMUL var2

MOV ECX, var3

SHL ECX, 2

ADD ECX, var4

IDIV ECX

MOV var5, EAX

## Given that the CPU is receiving a byte in AL register from the printer. Assume that bits 3 to 6 represent a number. The assembly code to display the decimal value of this number is:

SHR AL, 3

AND AL, 0FH

MOVZX EAX, AL

Call WriteDec

## Suppose that we would like to encrpyt text according to an encryption table. Part of the encryption table is shown below. The assembly code to encrpyt a character in register AL according to the encrption table below and store the encrypted character in the same register is:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | … |
| ‘E’ | ‘Z’ | ‘I’ | ‘X’ | ‘M’ | ‘A’ | ‘C’ | ‘L’ | ‘F’ | … |

We will define the encryption table as follows:

EncTable BYTE ‘EZIXMACLF…’

The encryption code will be as follows:

SUB AL, ‘A’

MOVZX EAX, AL

MOV AL, EncTable[EAX]

##

**[42 Points]**

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

## **(i)** Given the following data declarations: **TABLE1 BYTE ‘COE 205 Exam II’ and TABLE2 BYTE ‘aeoui’**

## Determine the content of register **AH** after executing the following code:

 XOR AH, AH

 MOV ESI, offset TABLE2

 MOV ECX, 5

Top:

 PUSH ECX

 MOV ECX, lengthof TABLE1

 MOV EBX, offset TABLE1

 DEC EBX

 MOV DL, [ESI]

Next:

 JECXZ ENL

 INC EBX

 MOV AL, [EBX]

 OR AL, 20H

 CMP AL, DL

 LOOPNE Next

 JNE ENL

 INC AH

 JMP Next

ENL:

 POP ECX

 INC ESI

 LOOP Top

The code counts the number of vowel characters in TABLE1 and stores the count in register AH. Thus, AH=6.

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

GDisp MACRO PG

LOCAL E1, E2, E3, E4, E5

 MOV ESI, PG

 CMP ESI, 90

 JBE E1

 MOV AL, 'A'

 JMP E5

E1:

 CMP ESI, 80

 JBE E2

 MOV AL, 'B'

 JMP E5

E2:

 CMP ESI, 70

 JBE E3

 MOV AL, 'C'

 JMP E5

E3:

 CMP ESI, 60

 JBE E4

 MOV AL, 'D'

 JMP E5

E4:

 MOV AL, 'F'

E5:

 CALL WriteChar

 CALL Crlf

ENDM

 GDisp 83

 GDisp 68

The code will display the following the grades correspond to the passed mark and hence will display:

B

D

## **(iii)** Given the following definition in the data segnment N DWORD 8 DUP (1), determine what will be displayed after executing the following code**:**

MOV ESI, 2

 MOV ECX, 6

F1:

 MOV EAX, N[ESI\*4-4]

 ADD EAX, N[ESI\*4-8]

 MOV N[ESI\*4], EAX

 INC ESI

 LOOP F1

 XOR ESI, ESI

 MOV ECX, 8

F2:

 MOV EAX, N[ESI\*4]

 CALL WriteDec

 CALL Crlf

 INC ESI

 LOOP F2

The code will first compute the eight elements of the array according to the Fibonacci sequenceand then display them as follows:

1

1

2

3

5

8

13

21

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

 MOV ESI, 654

 MOV EBX, 9

W1: CMP ESI, 9

 JBE EndW1

 XOR EDI, EDI

W2: CMP ESI, 0

 JBE Endw2

 XOR EDX, EDX

 MOV EAX, ESI

 DIV EBX

 ADD EDI, EDX

 MOV ESI, EAX

 JMP W2

Endw2:

 MOV ESI, EDI

 JMP W1

Endw1:

 MOV EAX, ESI

 CALL WriteDec

The code will extract the remainders of dividing the number by 9 and then add the digits together. If the result is greater than 9 the process is repeated. In the first iteration, 6+0+8=14. In the second iteration 5+1=6. Thus, the result dispalyed will be 6.

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

PUSH 1

CALL HILL

HILL PROC

 MOV EAX, [ESP+4]

 CALL WriteDec

 CALL Crlf

 CMP EAX, 5

 JA Endif1

 MOV EBX, 3

 MUL EBX

 DEC EAX

 PUSH EAX

 CALL HILL

 MOV EAX, [ESP+4]

 CALL WriteDec

 CALL Crlf

Endif1:

 RET 4

HILL ENDP

The code will display the following:

1

2

5

14

5

2

1

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

 MOV ESI, 14

 MOV EDI, 21

 CALL MTest

 MOV EAX, ECX

 Call WriteDec

 MTest PROC

 CMP ESI, 0

 JNE Skip2

 MOV ECX, EDI

 JMP End1

Skip2:

 CMP EDI, 0

 JNE Skip3

 MOV ECX, ESI

 JMP End1

Skip3:

 MOV EAX, ESI

 XOR EDX, EDX

 DIV EDI

 MOV ESI, EDI

 MOV EDI, EDX

 CALL MTest

End1:

 RET

MTest ENDP

The code will display the greatest comon divisor between the two numbers in ESI and EDI and hence will display the result as 7.

## **(vii)** Given the following declaration in the data segment:

##  X DWORD 1, 5, 10, 20, 32, 50

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

 LEA EBX, X

 MOV ESI, 0

 MOV EDI, 5

 MOV EDX, 32

 CALL BSP

 CALL WriteDec

BSP PROC

 CMP ESI, EDI

 JG RET1

 MOV ECX, ESI

 ADD ECX, EDI

 SHR ECX, 1

 CMP [EBX+ECX\*4], EDX

 JNE SKIP

 MOV EAX, ECX

 RET

SKIP:

 JG SKIP2

 MOV ESI, ECX

 INC ESI

 CALL BSP

 RET

SKIP2:

 MOV EDI, ECX

 DEC EDI

 CALL BSP

 RET

RET1:

 MOV EAX, -1

 RET

BSP ENDP

The procedure implements the binary search algorithm and the code returns the index of the number 32 and hence it will display 4.

**[24 Points]**

#  **(Q3)**

# **(i)** Write a procedure, **ShellSort**, to sort an array of integers (i.e. 32-bit signed numbers) in an **ascending** order. The number of integers to be sorted and the address of the array to be sorted are assumed to be passed on the stack. The procedure should maintain the content of all registers to their state before its execution. **Do not use the USE directive, local directive, pusha and popa instructions in your solution**.

The pseudocode for the **ShellSort** procedure is given below:

 **ShellSort** (Array, Size){

hmax=Size/9;

 **for** (h= 1; h<=hmax; h=3\*h+1);

 **for** (; h>0; h=h/3){

 **for** (i=h; i<size; i++){

 v = Array[i];

 j=i;

 while(j >= h && v < Array[j-h]){

 Array[j] = Array[j-h];

 j = j-h;

 }

 Array[j] = v;

 }

}

}

# **(ii)** Write a complete program, showing the place of procedure definition, to use the procedure **ShellSort** to sort the Array given below:

Array Dword 10, 2, 0, 15, 25, 30, 7, 22, -1, -5

Note that the Content of Array after sorting will be:

Array Dword -5, -1, 0, 2, 7, 10, 15, 22, 25, 30

.686

.MODEL FLAT, STDCALL

.STACK

INCLUDE Irvine32.inc

.DATA

Array DD 10, 2, 0, 15, 25, 30, 7, 22, -1, -5

.CODE

main PROC

 PUSH offset Array

 PUSH lengthof Array

 CALL ShellSort

 exit ; exit to operating system

main ENDP

 ShellSort PROC

 PUSH EBP

 MOV EBP, ESP

 PUSH EAX ; save registers

 PUSH EBX

 PUSH ECX

 PUSH EDX

 PUSH ESI

 PUSH EDI

 MOV ECX, [EBP+8] ; size of array

 MOV EDX, [EBP+12] ; address of array

 MOV EAX, ECX

 MOV BL, 9

 DIV BL ; hmax=Size/9

 MOVZX EDI, AL

 MOV ESI, 1 ; for (h= 1; h<=hmax; h=3\*h+1);

 For1: CMP ESI, EDI

 JA Endfor1

 MOV BL, 3

 MOV EAX, ESI

 MUL BL

 INC EAX ;h=3\*h+1

 MOV ESI, EAX

 JMP For1

 Endfor1:

 For2:

 CMP ESI, 0

 JBE Endfor2

 MOV EAX, ESI ; i=h

 For3: CMP EAX, ECX

 JAE Endfor3

 MOV EBX, [EDX+EAX\*4] ; v = Array[i]

 MOV EBP, EAX ; j=i

 Whileloop:

 CMP EBP, ESI ; while(j >= h && v < Array[j-h])

 JB EndWhile

 MOV EDI, EBP

 SUB EDI, ESI ; j-h

 CMP EBX, [EDX+EDI\*4] ; v < Array[j-h]

 JGE EndWhile

 MOV EDI, [EDX+EDI\*4] ; Array[j] = Array[j-h];

 MOV [EDX+EBP\*4], EDI

 SUB EBP, ESI

 JMP Whileloop

 EndWhile:

 MOV [EDX+EBP\*4], EBX

 INC EAX

 JMP For3

 Endfor3:

 MOV BL, 3

 MOV EAX, ESI

 DIV BL

 MOVZX ESI, AL

 JMP For2

 Endfor2:

 POP EDI ; restore registers

 POP ESI

 POP EDX

 POP ECX

 POP EBX

 POP EAX

 POP EBP

 RET 8

 ShellSort ENDP

END main