May 20, 2010

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Major Exam II

First Semester (092)

Time: 1:00 PM-3:30 PM

Student Name : __KEY____

Student ID.   : ____

Question	Max Points	Score
Q1	28	
Q2	36	
Q3	12	
Q4	24	
Total	100	

Dr. Aiman El-Maleh

[28 Points]

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

(1) Assume that ESP=00000100H and the address of TEST is 0000300AH. After executing the instruction CALL TEST, the content of ESP=ESP-4=000000FCH.

(2) Assume that ESP=00000100H. After executing the instruction RET 8, the content of ESP=ESP+4+8=0000010CH.

(3) The code to Jump to label L1 if regiser AL bits 3 and 6 are 1 or bit 5 is zero is:

```
      Test AL, 100000b
      JZ L1
      Test AL, 1000000b
      JZ Skip
      Test AL, 1000b
      JNZ L1
Skip:
```

(4) Assuming that EAX=8765432CH and ECX=FEDBA7E4H, executing the instruction SHRD EAX, ECX, 16 will set EAX=A7E48765H and ECX= FEDBA7E4H.

(5) To multiply the signed content of register EAX by 33.25 without using multiplications instructions, we use the following instructions:

```
MOV EBX, EAX
SHL EAX, 5
ADD EAX, EBX
SAR EBX, 2
ADD EAX, EBX
```

(6) Assuming that all variables are 32-bit signed integers, the assembly code implementing the following equation **var3 = (-5*var1)/(8*var2 -10)** is:

```
MOV EAX, -5
IMUL var1
MOV EBX, var2
SHL EBX, 3
SUB EBX, 10
IDIV EBX
MOV var3, EAX
```

(7) 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 8 is:

```
MOV EAX, DWORD PTR I
MOV EBX, DWORD PTR I+4
SHLD, EBX, EAX, 3
SHL EAX, 3
MOV DWORD PTR I, EAX
MOV DWORD PTR I+4, EBX
```

(8) Given that MS-DOS packs the year, month, and day into 16 bits in register DX, where bits 0 to 4 store the day, bits 5 to 8 store the month and bits 9 to 15 store the year relative to 1980. Write assembly code to print the date in day, month and year. For example if DX=0010011001101010, it will print 10/3/1999:

```
MOVZX EAX, DX
AND EAX, 11111b
CALL WriteDec
MOV AL, ‘/’
CALL WriteChar
MOVZX EAX, DX
SHR EAX, 5
AND EAX, 1111b
CALL WriteDec
MOV AL, ‘/’
CALL WriteChar
MOVZX EAX, DX
SHR EAX, 9
AND EAX, 1111111b
ADD EAX, 1980
CALL WriteDec
```

[36 Points]

(Q2) Answer SIX out of 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 10,11,12,13,14
      DWORD 20,21,22,23,24
      DWORD 30,31,32,33,34
      DWORD 40,41,42,43,44
```

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

```
    mov ecx, lengthof Array
    xor esi, esi
Next:
    mov eax, lengthof Array
    mul esi
    shl eax, 2
    mov eax, Array[eax+esi*4]
    Call WriteDec
    Call CrLf
    inc esi
    loop Next
```

The program will print the diagonal of the array as follows:

```
0
11
22
33
44
```

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

```
    push 5
    push 4
    call MyProc

MyProc PROC
    push ebp
    mov ebp, esp
    sub esp, 4
    push eax
    mov DWORD PTR [ebp-4],10
    mov eax, [EBP + 8]
    sub [ebp-4], eax
    shl DWORD PTR [ebp-4], 2
    mov eax, [EBP + 12]
    add [ebp-4], eax
    shr DWORD PTR [ebp-4], 1
    mov eax, [ebp-4]
    call WriteDec
    pop eax
    mov esp, ebp
    pop ebp
    ret 8
MyProc ENDP
```

The program will display 14.
It will allocate a local variable and initialize it with 10. Then, it will copy into eax the second passed parameter 4. Then, the local variable will be 10-4=6. Then, the local variable is multiuplied by 4 i.e. its value becomes 24. The first passed parameter is moved to eax, i.e. eax=5. The content of eax is added to the local variable which becomes 29. The local variable is divided by 2 and becomes 14. The content of the local variable is then displayed.

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

Array DWord 17,-10,30,-40,4,-5,8

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

```
    xor eax, eax
    mov esi, -1
    mov ecx, lengthof Array
L1:
    inc esi
    test  Array[esi*4], 8000h
    loopz L1
    jz done
    inc eax
    cmp ecx, 0
    jnz L1
done:
    call WriteDec
```

The program will display 3 which the number of negative nyumbers in the array.

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

```
     mov ecx, 5
    mov eax, 12
Next:
    cmp   eax, 7
    ja    default
    jmp   jumptable[eax*4]

case01:
   add eax, 9
   jmp done
case23:
   add eax, 7
   jmp done
case45:
   add eax, 3
   jmp done
case67:
   add eax, 4
   jmp done
default:
   inc eax
done:
   shr eax, 1
   loop Next
   call WriteDec
exit
jumptable DWORD    case01, case01, case23, case23, case45, case45, case67,
case67
```

The program will display 5.
First, since EAX=12, the program will jump to default and EAX becomes 13. Then, EAX is divided by 2 and becomes 6. The loop then goes for the 2nd iteration as ECX is 4 and then it jumps to case67. EAX then becomes 10. It then gets divided by 2 and becomes 5. The loop is repeated as ECX becomes 3. The program then jumps to case45 and EAX becomes 8. EAX is then divided by 2 and becomes 4. The loop is continued as ECX is 2. The program then jumps to case45 and EAX becomes 7. After that it gets divided by 2 and becomes 3 and the loop is repeated as ECX is 1. The program then jumps to case23 and eax becomes 10. Then, EAX becomes 5 and it gets displayed.

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

```
        mov eax, 3
        call MyProc
        call WriteDec

MyProc Proc
        push ebx
        cmp eax, 0
        je done
        cmp eax, 1
        je done
        dec eax
        mov ebx, eax
        call MyProc
        xchg ebx, eax
        dec eax
        call MyProc
        add eax, ebx
done:
        pop ebx
        ret
MyProc Endp
```

The program wuill display 2 which is the fibonacci sequence of 3.

(vi) Given the following declaration in the data segment:

MyNumber Byte 9 dup(0)

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

```
        mov  ax, 0ABCDh
        xor  esi, esi
        mov  ecx, 8
L1:     rol  ax, 2
        mov  bx, ax
        and  bx, 3
        add  bl, '0'
        mov  MyNumber[esi], bl
        inc  esi
        loop L1

        lea edx, MyNumber
        call WriteString
```

The program will display the content of register ax in base 4 which is 22233031.

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

MyNumber Byte '1','2','3',0

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

```
            xor esi, esi
            mov   eax, 0
L1:         imul  eax, 16
            movzx edx, MyNumber[esi]
            sub   edx, '0'
            add   eax, edx
            inc   esi
            cmp   MyNumber[esi],0
            jne   L1

            dec esi
            mov  ecx, 0
            mov  ebx, 8
L2:         mov  edx, 0
            div  ebx
            add  dl, '0'
            mov MyNumber[esi], dl
            dec esi
            cmp  eax, 0
            jnz  L2

           lea edx, MyNumber
           call WriteString
```

The program will convert the hexadecimal number 123H into octal and will display the number 443.

[12 Points]

(Q3) Write a macro, **CMul**, to multiply the signed content of register EAX by a constant **n** passed as a aparmeter to the macro. The macro should be based on using shift and add instructions and should not use MUL or IMUL instructions. The macro should preserve the content of all temporary registers used.

```
CMul Macro n
Local Next, Skip
        PUSH EBX
        PUSH ECX
        MOV EBX, n
        XOR ECX, ECX
Next:
        SHR EBX, 1
        JNC Skip
        ADD ECX, EAX
Skip:
        SHL EAX, 1
        CMP EBX, 0
        JNE Next
        MOV EAX, ECX
        POP ECX
        POP EBX
ENDM
```

[24 Points]

(Q4)

(i) Write a procedure, **BubbleSort**, to sort an array of integers 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.

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

```
BubbleSort (ArrayPointer, ArraySize)
    pass = 1
    do {
        swap_occurs = 0
        for (i= 1 to ArraySize-pass)
            if (Array[i-1] > Array[i])
                swap ith and (i-1)th elements of the array
                swap_occurs = 1
            end if
        end for
        pass = pass+1
    while (swap_occurs && pass <= ArraySize -1)
end BubbleSort
```

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

Array DWord 10, 2, 0, 15, 25, 30, 7, 22

Note that the Content of Array after sorting will be:
Array DWord 0, 2, 7, 10, 15, 22, 25, 30

```
.DATA

Array DWord 10, 2, 0, 15, 25, 30, 7, 22

.code
main PROC
   MOV EAX, offset Array
   PUSH offset Array
   PUSH lengthof Array
   CALL BubbleSort

exit
main ENDP

BubbleSort PROC
       PUSHAD
       MOV EBP, ESP
       MOV ECX, [EBP+36]
       MOV EBX, [EBP+40]
```

```
        MOV EAX, 1        ; pass = 1

do_while:
        XOR EDX, EDX      ; swap_occurs = 0

        PUSH ECX
        SUB ECX, EAX
        MOV ESI, 1
for_loop:
        MOV EDI, [EBX+ESI*4-4]
        CMP EDI, [EBX+ESI*4]
        JNG NoSwap          ; if (Array[i-1] > Array[i])
        XCHG EDI,[EBX+ESI*4] ; swap ith and (i-1)th elements of the array
        MOV [EBX+ESI*4-4], EDI
        MOV EDX, 1         ; swap_occurs = 1
NoSwap:
        INC ESI
        LOOP for_loop
        POP ECX
        INC EAX            ; pass = pass+1
        CMP EDX, 0         ; while (swap_occurs && pass <= ArraySize -1)
        JE Done
        CMP EAX, ECX
        JL do_while

Done:
        POPAD
        RET 8

BubbleSort ENDP
END main
```