EE-213 COAL

Wednesday November 7, 2018

Course Instructor

Muhammad Irfan Ishaq, Sajid Iqbal, Fazeelat Mazhar

Serial No:

2nd Mid Term Exam
Total Time: 1 Hour
Total Marks: 50

____
Signature of Invigilator

____________________ Roll No	_______ Section	_____________ Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Verify at the start of the exam that you have a total of five (05) questions printed on seven (07) pages including this title page.
2. Attempt all questions on the question-book and in the given order.
3. The exam is closed books, closed notes. Please see that the area in your threshold is free of any material classified as ‘useful in the paper’ or else there may a charge of cheating.
4. Read the questions carefully for clarity of context and understanding of meaning and make assumptions wherever required, for neither the invigilator will address your queries, nor the teacher/examiner will come to the examination hall for any assistance.
5. Fit in all your answers in the provided space. You may use extra space on the back if required. If you do so, clearly mark question/part number on that page to avoid confusion.
6. **Calculators are not allowed in the exam**. Use only your own stationery.
7. Use only permanent ink-pens. Only the questions attempted with permanent ink-pens will be considered. Any part of paper done in lead pencil cannot be claimed for checking/rechecking.

	Q1	Q2	Q3	Q4	Q5	**Total**
Total Marks	**10**	**10**	**10**	**5**	**15**	**50**
Marks Obtained						

Vetted By: ________________________ **Vetter Signature:** __________________

University Answer Sheet Required: **No** [✓ X] **Yes** ☐

Question Number 1 **(5 + 5 = 10 Marks)**

a) Write an assembly language code that displays the contents of BL in hexadecimal, then reverse the data stored in BL (as shown below) and display it again in Hexadecimal.

Sample:

```
Contents of BL are  :   4B (0100 1011)
Reverse Output is   :   D2 (1101 0010)
```

Solution

```
.data
     str1 DB "Contents of BL are : ", 0
     str2 DB "Reverse Output is  : ", 0
.code
     mov edx, offset str1
     call WriteString
     mov eax, ebx
     mov ebx, 1
     call WriteHexB
     call crlf
     mov ebx, eax
     mov ecx, 8
     mov edx, 0
     L1:
          shl bl, 1  ; OR shr bl, 1
          rcr dl, 1  ; OR rcl dl, 1
     Loop L1
     mov edx, offset str2
     call WriteString
     movzx eax, dl
     mov ebx, 1
     call WriteHexB
```

b) The parity of a 32 bit data can be checked by XORing its individual bytes. Write an assembly language code to check the parity of a data stored in EAX.

Solution

```
.data
     var DD 0
.code
     mov var, eax
     xor al, DB ptr var + 1
     xor al, DB ptr var + 2
     xor al, DB ptr var + 3
```

Question Number 2 **(10 Marks)**

Write an ALP that prompts the user to enter a string terminated by Enter key. Search for vowels in the string using loop instructions. Your program should display the total number of vowels present in the string. Your program should handle both lower and upper case vowels.

Sample:

```
Enter a string terminated by Carriage return:
Today is COAL Mid 2

Total number of vowels are: 6
```

Solution

```
.data
     str1 DB 30 DUP (?)
     str2 DB "aeiouAEIOU", 0
     vowel_length = 10
     msg1 DB "Enter a string terminated by carriage return", 0
     msg2 DB "Total number of vowels are : ", 0
     var DD 0
     count DD 0
.code
     mov edx, offset str1
     mov ecx, lengthof str1
     call ReadString
     call crlf
     mov ecx, vowel_length
     mov esi, offset str1      ; user entered string
     mov edi, offset str2      ; vowels
     L1:
          mov bl, [edi]        ; fetch a character of vowels
          mov var, ecx         ; save the outer loop count
          mov ecx, eax         ; inner loop count, string length is in eax
          L2:
               mov bh, [esi]   ; fetch a character of user entered string
               cmp bl, bh      ; compare vowel and string character
               JNE again       ; if not vowel
               inc count       ; if vowel, increase count
               again:
                    inc esi    ; next character of string
          loop L2              ; search for vowel again
          mov ecx, var         ; restore the outer loop count
          inc edi              ; next vowel character
     loop L1                   ; search again
     mov edx, offset msg2
     call WriteString          ; display msg2
     mov eax, count
     call WriteDec             ; display vowel count
     call crlf
```

Question Number 3 **(10 Marks)**

Convert the following C++ code to equivalent Assembly Language instructions. (All the numbers are unsigned).

```
main()
{
     int var1, var2;
     cout<<"Enter two integers: var1 of 32 bits and var2 of 8
     bits";
     cin>>var1>>var2;
     int sum = Addtwo(var1,var2);
     cout<<"The sum is:"<<sum<<endl;
     int diff = Subtwo(var1,var2);
     cout<<"The difference is:"<<diff<<endl;
     int multiply = Multwo(var1);
     cout<<"var1 x 8 is:"<<multiply<<endl;
     int division = Divtwo(var1);
     cout<<"The quotient of var1/32 is:"<<division<<endl;
}

int Addtwo(int &var1,int &var2)
{
     var1 = var1 + var2;
     return var1;
}

int Subtwo(int var1,int var2)
{
     var1 = var1 - var2;
     return var1;
}

int Multwo(int var1)
{
     return var1*8;
}

int Divtwo(int var1)
{
     return var1/32;
}
```

Solution of Question Number 3

```
.data
     msg1 DB “Enter two
     integers: var1 of 32 bits
     and var2 of 8 bits”,0

     add_msg DB “The sum is :
     ”,0

     diff_msg DB “The
     difference is : ”,0

     mul_msg DB “var1 x 8 is :
     ”, 0

     div_msg DB “The quotient
     of var1/32 is : ”, 0

     var1 DD 0
     var2 DD 0
     temp DD 0

.code
MAIN PROC
     mov edx, msg1
     call WriteString
     call ReadDec
     mov var1, eax
     call ReadDec
     mov var2, eax
     mov esi, offset var1
     mov edi, offset var2
     mov ebx, var1
     mov ecx, var2
     call AddTwo
     mov edx, offset add_msg
     call WriteString
     mov eax, [esi]
     call WriteDec
     call crlf
     mov temp, ebx
     call SubTwo
     mov edx, offset diff_msg
     call WriteString
     mov eax, ebx
     call WriteDec
     call crlf
     mov eax, temp
     call MulTwo
     mov edx, offset mul_msg
     call WriteString
     call WriteDec
     call crlf
     mov eax, temp
     call DivTwo
     mov edx, offset div_msg
     call WriteString
     call WriteDec
     call crlf
MAIN ENDP
;----------------------------
     AddTwo PROC
          mov eax, [edi]
          add [esi], eax
          RET
     AddTwo ENDP
;----------------------------

     SubTwo PROC
          sub ebx, ecx
          RET
     SubTwo ENDP
;-----------------------------

     MulTwo PROC
          shl eax, 3
          RET
     MulTwo ENDP
;-----------------------------

     DivTwo PROC
          shr eax, 5
          RET
     DivTwo ENDP

EXIT
END MAIN
```

Question Number 4 **(05 Marks)**

The RandomRange procedure from the Irvine32 library generates a pseudorandom integer between 0 and N – 1. Your task is to create an improved version that generates an integer between M and N – 1. Let the caller pass M in EBX and N in EAX.
Where M is the lower bound and N is the upper bound.

Solution

```
ORG 100h
INCLUDE Irvine32.inc
.model small
.stack 100h
.data
     Upper_range = 300
     Lower_range = 100
     msg DB "Generated number in the given range is : ", 0
.code
     MAIN PROC
          mov eax, Upper_range
          mov ebx, Lower_range
          call BetterRandomRange
          mov edx, offset msg
          call WriteString
          call WriteInt
     MAIN ENDP
     ;-----------------------------------------------------------
     ; Definition of BetterRandomRange
     ;-----------------------------------------------------------
     BetterRandomRange PROC
          sub eax, ebx
          call RandomRange
          add eax, ebx
          RET
     BetterRandomRange ENDP
     EXIT
     END MAIN
```

Question Number 5 **(5 + 8 + 2 = 15 Marks)**

a) State **with reason**, that where the jump will be made in the following instructions? (treat every part as a different code). ***(No marks will be awarded for answers written without reasons).***

i.
```
mov ax,8109h
cmp ax,26h
```
`JG Target1;` jump not taken as JG is for signed numbers and 8109 is negative

`JA Target2 ;` jump taken as 8109h > 26h

ii.
```
mov ax,0
STC
```
`JZ Target1 ;` jump not taken as mov instruction does not affect any flag

`JC Target2 ;` jump taken as CF is set due to STC instruction

iii.
```
mov ecx,10000000h
```
`JCXZ Target1 ;` jump taken as CX = 0

`JECXZ Target2 ;` jump not taken as ECX ≠ 0

iv.
```
mov ecx,0
cmp ecx,0
```
`Jg Target1 ;` jump not taken as 0 is not greater than 0

`JNL Target2 ;` jump taken as 0 is not less than 0 is true

b) You are given with the following code. Identify the errors by underlining them and re-write the code after applying suitable corrections.

Code With Errors	Corrected Code
.data	.data
arr DD 1,2,3,4,5,6	arr DD 1,2,3,4,5,6
theSum DW ?	theSum **DD** ?
.code	.code
MAIN PROC	MAIN PROC
mov esi,OFFSET arr	mov esi,OFFSET arr
mov ecx,SIZEOF arr	mov ecx,**LENGTHOF** arr
call ArraySum Uses esi	**call ArraySum**
mov theSum,eax	mov theSum,eax
MAIN ENDP	MAIN ENDP
ArraySum PROCEDURE	ArraySum **PROC**
push esi	push esi
push ecx	push ecx
push ebp	push ebp
mov eax,0	mov eax,0
L1: add eax, esi	L1: add eax, **[esi]**
add esi,TYPE WORD	add esi,TYPE **DWORD**
loop L1	loop L1
pop esi	**pop ecx**
pop ecx	**pop esi**
ArraySum ENDP	**pop ebp**
END MAIN	**RET**
	ArraySum ENDP
	END MAIN

c) A stack in protected mode is shown in the figure below. Write instruction(s) to move parameter 2 in EBX register by using EBP.

```
Push ebp
Mov ebp, esp
Mov ebx, [ebp + 12]
```

. .
Parameter 1
Parameter 2
Parameter 3
Return to Main
. .