National University of Computer and Emerging Sciences



Assignment 04

COAL

|  |  |
| --- | --- |
| Name | Muhammad Zain |
| Roll No. | 19F-0228 |
| Course  INSTRUCTOR | Mr. Abdul Qadir Bilal |
| Semester | Fall 2020 |

Question 1

# Source Code:

*;Author :Muhammad Zain*

;Task :Task 1

;Name :xor ,2 use registers

INCLUDE Irvine32.inc

.data

msg1 db "Enter Integer ", 0

var dd 0

.code

main PROC

mov eax, white + (lightred \* 16)

call SetTextColor

mov edx,offset msg1

call writestring

call readint

mov var,eax

mov ebx,-1

add eax,ebx

call crlf

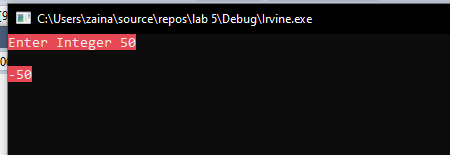
xor eax, ebx

call writeint

main ENDP

END main

# Snip:



Question 2

# Source Code:

*;Author :Muhammad Zain*

;Task :Task 2

;Name :Extended subtraction

INCLUDE Irvine32.inc

.DATA

msg\_1 BYTE "The Subtraction of Array 1 and Array 2 ",0

msg\_2 BYTE "The Subtraction of Array 3 and Array 4 ",0

Array\_1 BYTE 10h,0Ah,00h,40h,10h,0Ah,00h,40h,10h,0Ah,34h,12h,98h,74h,06h,0A4h,0B2h,0A2h,56h,78h,02h,45h,23h,00h,00h,87h,10h,80h,13h,24h,0FCh,6Ah

Array\_2 BYTE 02h,45h,23h,00h,00h,87h,10h,80h,13h,24h,10h,0Ah,00h,40h,10h,0Ah,00h,40h,10h,0Ah,02h,45h,23h,00h,00h,87h,10h,80h,13h,24h,0FCh,6Ah

Array\_3 BYTE 10h,0Ah,00h,40h,10h,0Ah,00h,40h,10h,0Ah,5Bh,0FCh,6Ah,22h,00h,20h,70h,00h,00h,40h,02h,45h,23h,00h,00h,87h,10h,80h,13h,24h,0FCh,6Ah

Array\_4 BYTE 10h,0Ah,00h,40h,10h,0Ah,00h,40h,10h,02h,45h,23h,00h,00h,87h,10h,80h,13h,24h,0Ah,10h,0Ah,00h,40h,10h,0Ah,00h,40h,10h,0Ah,0FCh,6Ah

result BYTE 33 DUP(0)

.CODE

main PROC

mov eax, white + (lightred \* 16)

call SetTextColor

mov edx,OFFSET msg\_1

call writestring

mov esi,OFFSET Array\_1

mov edi,OFFSET Array\_2

mov ebx,OFFSET result

mov ecx,LENGTHOF Array\_1

call Extended\_Sub

mov esi,OFFSET result

mov ecx,LENGTHOF result

call Display

call crlf

call crlf

mov edx,OFFSET msg\_2

call writestring

mov esi,OFFSET Array\_3

mov edi,OFFSET Array\_4

mov ebx,OFFSET result

mov ecx,LENGTHOF Array\_3

call Extended\_Sub

mov esi,OFFSET result

mov ecx,LENGTHOF result

call Display

call crlf

exit

main ENDP

;===========================================================================

; This procedure subtracts two extended integers stored as arrays of bytes.

; ESI and EDI point to the two integers

; EBX points to a variable that will hold the result,

; ECX indicates the number of bytes to be subtracted

;=============================================================================

Extended\_Sub PROC

pushad

clc

L1:

mov al,[esi]

sbb al,[edi] ; subtract the second integer

pushfd ; save the Carry flag

mov [ebx],al ; store result

inc esi

inc edi

inc ebx

popfd ; restore the Carry flag

loop L1

mov BYTE PTR [ebx],0 ; clear high byte of result

sbb BYTE PTR [ebx],0 ; subtract any leftover carry

popad

ret

Extended\_Sub ENDP

;============================================================

;display proc

;from high to low

;============================================================

Display PROC

pushad

add esi,ecx

sub esi,TYPE BYTE

mov ebx,TYPE BYTE

L1:

;loop for displaying each byte

mov al,[esi]

call WriteHexB

sub esi,TYPE BYTE

loop L1

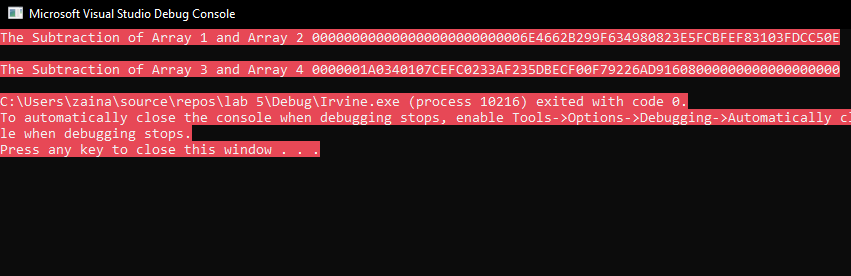
popad

ret

Display ENDP

END main

# Snip:



Question 3

# Source Code:

*;Author :Muhammad Zain*

;Task :Task 3

;Name :Encryption

INCLUDE Irvine32.inc

.DATA

key1 BYTE -2,4,1,0,-3,5,2,-4,-4,6

Entered BYTE "Entered Text : ",0

input\_1 BYTE "Its Zain Here 10",0

input\_2 BYTE "Encrypted Text : ",0

keySize = $ - key1

.CODE

main PROC

mov eax, white + (lightred \* 16)

call SetTextColor

mov edx,OFFSET Entered

call WriteString

mov edx,OFFSET input\_1

call WriteString

call Crlf

mov esi,OFFSET input\_1

mov edi,OFFSET key1

mov ecx,SIZEOF input\_1

call Encryption\_proc

mov edx,OFFSET input\_2

mov esi,OFFSET input\_1

call Display

exit

main ENDP

Display PROC

pushad

call WriteString

mov edx,esi

call WriteString

call Crlf

popad

ret

Display ENDP

;===================================================================

; Encryption by rotation

; ESI = points to the

;ECX = size of buffer

;EDI = points to the key

;====================================================================

Encryption\_proc PROC

pushad

mov edx,0

L1:

push ecx

mov cl,[edi + edx]

inc edx

cmp edx,keySize ; is EDI >= SIZEOF key ?

jb Continue ; if not, continue

sub edx,keySize ; if yes, reset EDI tofirst char of key

Continue:

or cl,cl ; set Zero and Sign flags

je Done ; if Zero flag is set, CL = 0

js IsNegative ; if Sign flag is set, CL < 0

ror BYTE PTR [esi],cl ; CL > 0 so rotate right

jmp Done

IsNegative:

neg cl ; CL < 0 so rotate left

rol BYTE PTR [esi],cl

Done:

inc esi ; point to the next byte

pop ecx

loop L1

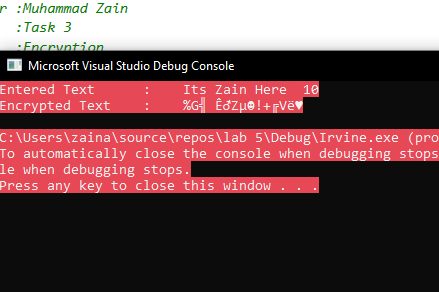
popad

ret

Encryption\_proc ENDP

# END main

# Snip:



Question 4

# Source Code:

*;Author :Muhammad Zain*

;Task :Task 4

;Name :Find GCD

INCLUDE Irvine32.inc

.data

msg1 DB "Enter 1st Number: ", 0

msg2 DB "Enter 2nd Number: ", 0

gcdmsg DB "GCD is: ", 0

GCD DWord ?

msg6 BYTE "Invalid Input :-( try again " , 0

.code

main PROC

mov eax, white + (lightred \* 16)

call SetTextColor

call DEC\_IN

cmp ebx, edx

JNZ NotEqual

mov GCD, ebx

JMP EndA

NotEqual:

cmp ebx, 0

JNZ ebxNotZero

mov GCD, edx

JMP EndA

ebxNotZero:

cmp edx, 0

JNZ edxNotZero

mov GCD, ebx

JMP EndA

edxNotZero:

call GCD\_AB

EndA:

call DEC\_OUT

main endp

DEC\_IN proc

up1:

call crlf

mov eax , 0

mov edx , offset msg1

call writestring

call readdec

cmp eax , 1

jb down1

cmp eax , 99

ja down1

mov bx , ax

up2:

call crlf

mov eax , 0

mov edx , offset msg2

call writestring

call readdec

cmp eax , 1

jb down2

cmp eax , 99

ja down2

mov dx , ax

jmp done1

down1:

mov edx , offset msg6

call writestring

jmp up1

down2:

mov edx , offset msg6

call writestring

jmp up2

done1:

ret

DEC\_IN endp

GCD\_AB PROC

mov ax, bx

mov cx, dx

mov dx, 0

div cx

add dx, 0

JNZ next

movzx edx, cx

mov GCD, edx

ret

next:

mov bx, cx

call GCD\_AB

ret

GCD\_AB endp

DEC\_OUT PROC

mov ebx, gcd

mov edx, offset gcdmsg

call WriteString

mov eax, gcd

call Writedec

call crlf

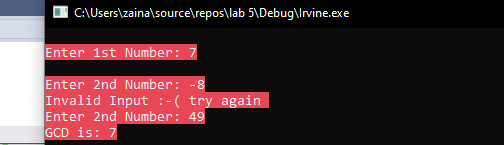
exitprogramme:

ret

DEC\_OUT endp

end main

# Snip:



Question 5

# Source Code:

The most basic design issue to be faced is the instruction format length.

This decision affects, and is affected by, memory size, memory organization, bus structure, processor, complexity, and processor speed.

This decision determines the richness and

flexibility of the machine as seen by the assembly- language programmer.

Programmers want more opcodes, more operands, more addressing modes, and greater address range, For a given instruction length, there is clearly a trade- off between the number

of opcodes and the power of the addressing capability. More opcodes obviously

mean more bits in the opcode field. For an instruction format of a given length,

this reduces the number of bits available for addressing Number of addressing modes Number operands

**PDP-8**

Despite the limitations of this design, the addressing is quite flexible.

Each memory reference consists of 7 bits plus two 1-bit modifiers. The memory is divided into fixed- length

pages of 27 = 128 words each. In addition, 8 dedicated words on page 0 are autoindex “registers.” When an indirect reference is made to one of these locations, preindexing occurs.

To enlarge the group of operations, opcode 7 defines a register reference or microinstruction.

In this format, the remaining bits are used to encode additional operations. In general, each bit defines a specific operation (e.g., clear accumulator), and these bits can be combined in a single instruction.

It instruction format is remarkably efficient. It supports indirect addressing, displacement addressing, and indexing

**PDP-10**

A sharp contrast to the instruction set of the PDP- 8 is that of the PDP- 10.

The PDP- 10 was designed to be a large- scale time- shared system, with an emphasis on

making the system easy to program, even if additional hardware expense was involved.

has a 36-bit word length and a 36-bit instruction length.

The fixed instruction format is shown in Figure 13.6. The opcode occupies 9 bits, allowing

up to 512 operations. In fact, a total of 365 different instructions are defined.

Most instructions have two addresses, one of which is one of 16 general-

Purpose registers. Thus, this operand reference occupies 4 bits.

The use of variable- Length instructions does not remove the desirability of

making all of the instruction lengths integrally related to the word length.

Because the processor does not know the length of the next instruction to be fetched, a

typical strategy is to fetch a number of bytes or words equal to at least the longest

possible instruction.

PDP-11

PDP-11 employs a set of eight 16-bit general- purpose registers. Two of these registers have additional significance: one is used as a stack pointer for special- purpose stack operations, and one is used as the program counter, which contains the address of the next instruction.

Thirteen different formats are used, encompassing zero-, one-, and two- address instruction types.

The opcode can vary from 4 to 16 bits in length. Register references are 6 bits in length.

Three bits identify the register, and the remaining 3 bits identify the addressing mode. The

**PDP- 11** is endowed with a rich set of addressing modes.