LAB 5

Questions:

Use following array declarations:

arrayB BYTE 10, 20, 30 arrayW

WORD 150, 250, 350 arrayD

DWORD 600, 1200, 1800

Now initialize three double word variables SUM1, SUM2, SUM3 and perform following operations

(expressed in pseudo-code here):

SUM1 = arrayB[0] + arrayW[0] + arrayD[0]

SUM2 = arrayB[1] + arrayW[1] + arrayD[1]

SUM3 = arrayB[2] + arrayW[2] + arrayD[2]

5. Initialize two arrays: array1

BYTE 10, 20, 30, 40

array2 BYTE 4 DUP (?)

Copy elements of array1 into array2 in reverse order using either indirect addressing or direct-offset

addressing.Use ESI and EDI Registers. (Hint: INC and DEC of OFFSET).

6. Subtract an array of 5 doublewords using indirect operands. Save the final result in a variable.

7. Use following array declarations:

arrayB BYTE 60, 70, 80

arrayW WORD 150, 250, 350

arrayD DWORD 600, 1200, 1800

For each array, add its 1st and last element using scale factors and display the result in a

separate register. (Hint: Use ESI and TYPE Operator).

Q3

Code:

INCLUDE Irvine32.inc

.data

arr BYTE 61,43,11,52,25

sortedarr BYTE 5 DUP(?)

.code

main PROC

MOV EAX , 0

MOV Al , 11

MOV sortedarr[0] , AL

call Writeint

MOV AL , 25

MOV sortedarr[1] , AL

call Writeint

MOV AL , 43

MOV sortedarr[2] , AL

call Writeint

MOV AL , 52

MOV sortedarr[3] , AL

call Writeint

MOV AL , 61

MOV sortedarr[4] , AL

call Writeint

call DumpRegs

exit

main ENDP

END main

For WORD and DWORD , the offset value is needed to be adjusted and register bit value

If array is of WORD type

.code

MOV EAX , 0

MOV AX , 11 ; AX is of 16 bits

MOV sortedarr[0] , AX

Call Writeint

MOV AX , 25

MOV sortedarr[2] , AX

Call Writeint

MOV AX , 43

MOV sortedarr[4] , AX

Call Writeint

MOV AX , 52

MOV sortedarr[6] , AX

Call Writeint

MOV AX , 61

MOV sortedarr[8] , AX

Call Writeint

If array is of DWORD type

MOV EAX , 0

MOV EAX , 11 ; EAX is of 32 bits

MOV sortedarr[0] , EAX

Call writeint

MOV EAX , 25

MOV sortedarr[4] , EAX

Call writeint

MOV EAX , 43

MOV sortedarr[8] , EAX

Call writeint

MOV EAX , 52

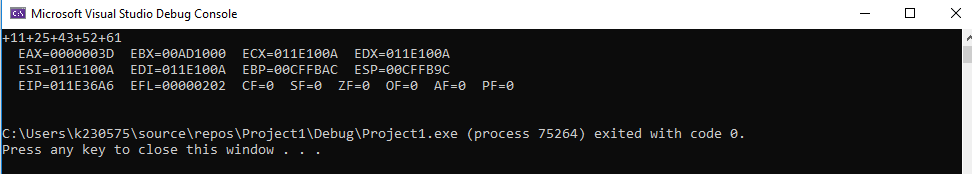
MOV sortedarr[12] , EAX

Call Writeint

MOV EAX , 61

MOV sortedarr[16] , EAX

Call Writeint



Q4

Code:

INCLUDE Irvine32.inc

.data

arrayB BYTE 10, 20, 30

arrayW WORD 150, 250, 350

arrayD DWORD 600, 1200, 1800

SUM1 DWORD ?

SUM2 DWORD ?

SUM3 DWORD ?

.code

main PROC

;SUM1 = arrayB[0] + arrayW[0] + arrayD[0]

MOV EAX , 0

MOVZX EAX , [arrayB]

MOVZX EBX , [arrayW]

ADD EAX , EBX

MOV ECX , [arrayD]

ADD EAX , ECX

call Writeint

call DumpRegs

MOV SUM1 , EAX ; SUM1 = EAX

call crlf

;SUM2 = arrayB[1] + arrayW[1] + arrayD[1]

MOV EAX , 0

MOV EBX , 0

MOV ECX , 0

MOVZX EAX , [arrayB+1]

MOVZX EBX , [arrayW+2]

ADD EAX , EBX

MOV ECX , [arrayD+4]

ADD EAX , ECX

call Writeint

call DumpRegs

MOV SUM2 , EAX ; SUM1 = EAX

call crlf

;SUM3 = arrayB[2] + arrayW[2] + arrayD[2]

MOV EAX , 0

MOV EBX , 0

MOV ECX , 0

MOVZX EAX , [arrayB+2]

MOVZX EBX , [arrayW+4]

ADD EAX , EBX

MOV ECX , [arrayD+8]

ADD EAX , ECX

call Writeint

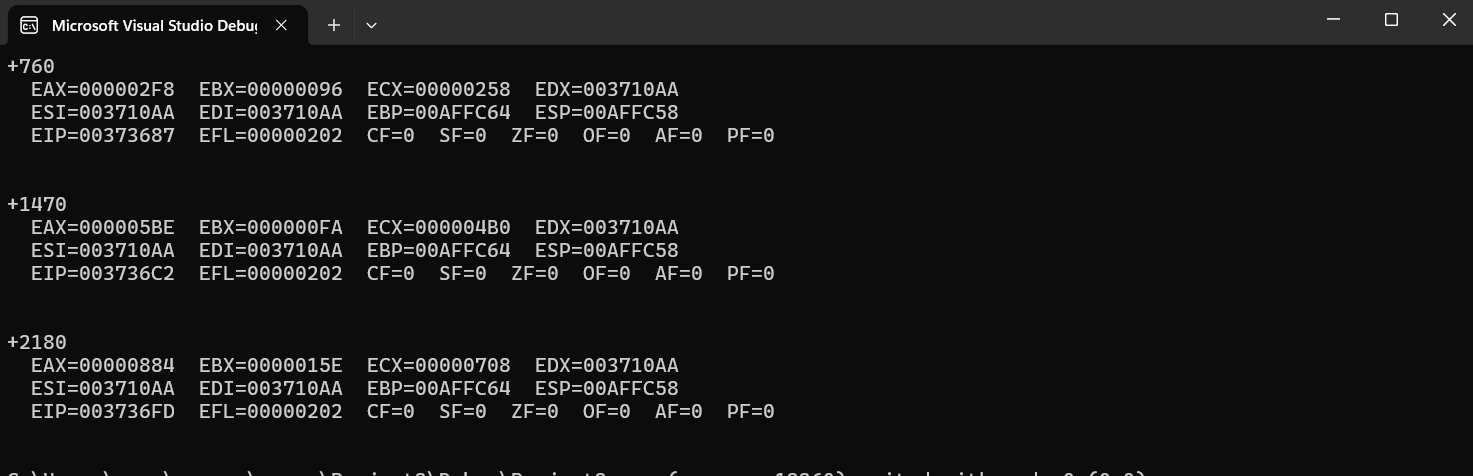
call DumpRegs

MOV SUM3 , EAX ; SUM1 = EAX

exit

main ENDP

END main



Q5

Code:

INCLUDE Irvine32.inc

.data

array1 BYTE 10, 20, 30, 40

array2 BYTE 4 DUP (?)

.code

main PROC

MOV ESI , 3

MOV EDI , 0

MOV EAX , 0

MOV AL , array2[EDI]

XCHG AL , array1[ESI]

MOVZX EAX , AL

Call Writeint

DEC ESI

INC EDI

MOV EAX , 0

MOV AL , array2[EDI]

XCHG AL , array1[ESI]

MOVZX EAX , AL

Call Writeint

DEC ESI

INC EDI

MOV EAX , 0

MOV AL , array2[EDI]

XCHG AL , array1[ESI]

MOVZX EAX , AL

Call Writeint

DEC ESI

INC EDI

MOV EAX , 0

MOV AL , array2[EDI]

XCHG AL , array1[ESI]

MOVZX EAX , AL

Call Writeint

DEC ESI

INC EDI

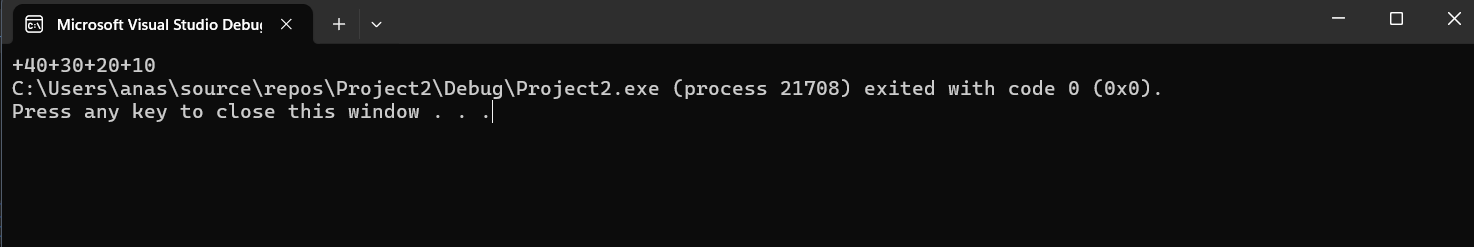
;call Writeint

;call DumpRegs

exit

main ENDP

END main



Q6

Code:

INCLUDE Irvine32.inc

.data

array1 DWORD 1000, 200, 150, 50, 20

result DWORD ? ; variable 'result' uninitialized

.code

main PROC

MOV ESI , 0

MOV EAX , 0

MOV ESI , OFFSET array1 ; Point esi to first element

MOV EAX , [ESI] ; store the first element is eax

ADD ESI , 4 ; Move to the next element

SUB EAX , [ESI]

ADD ESI , 4

SUB EAX , [ESI]

ADD ESI , 4

SUB EAX , [ESI]

ADD ESI , 4

SUB EAX , [ESI]

MOV result , EAX ; result = EAX

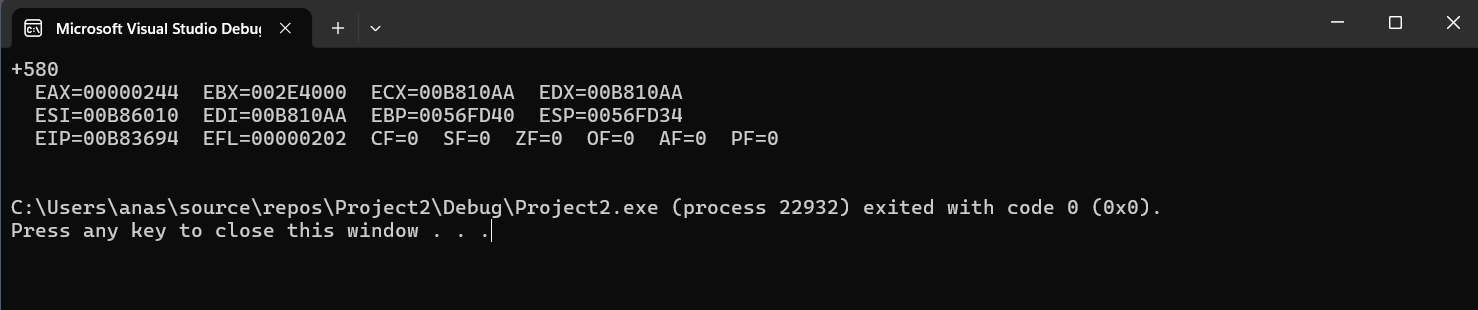
call Writeint

call DumpRegs

exit

main ENDP

END main



Q7

Code:

INCLUDE Irvine32.inc

.data

arrayB BYTE 60, 70, 80

arrayW WORD 150, 250, 350

arrayD DWORD 600, 1200, 1800

.code

main PROC

MOV ESI , 0

MOV EAX , 0

MOV AL , arrayB[ESI \* TYPE arrayB] ; arrayB[0 \* 1] = arrayB[0] = AL

ADD ESI , 2

ADD AL , arrayB[ESI \* TYPE arrayB]

call Writeint

call DumpRegs

call crlf

MOV ESI , 0

MOV EAX , 0

MOV AX , arrayW[ESI \* TYPE arrayW]

ADD ESI , 2

ADD AX , arrayW[ESI \* TYPE arrayW]

call Writeint

call DumpRegs

call crlf

MOV ESI , 0

MOV EAX , 0

MOV EAX , arrayD[ESI \* TYPE arrayD]

ADD ESI , 2

ADD EAX , arrayD[ESI \* TYPE arrayD]

call Writeint

call DumpRegs

exit

main ENDP

END main

