INCLUDE Irvine32.inc

.DATA

empID DWORD 5 DUP(?) ; Store 5 employee IDs

empRating DWORD 5 DUP(?) ; Store 5 employee ratings

sumRatings DWORD 0 ; Sum of all ratings

highestRating DWORD 0 ; Store highest rating

highestID DWORD 0 ; Store ID of highest rating employee

count DWORD 5 ; Number of employees

promptID BYTE "Enter Employee ID: ", 0

promptRating BYTE "Enter Employee Rating: ", 0

resultSum BYTE "Total sum of ratings: ", 0

resultHighest BYTE "Employee with highest rating - ID: ", 0

resultRating BYTE " Rating: ", 0

.CODE

main PROC

mov esi, 0 ; Index for array

inputLoop:

; Ask for Employee ID

mov edx, OFFSET promptID

call WriteString

call ReadInt

mov empID[esi\*4], eax ; Store ID

; Ask for Employee Rating

mov edx, OFFSET promptRating

call WriteString

call ReadInt

mov empRating[esi\*4], eax ; Store rating

; Add to sum

add sumRatings, eax

; Check for highest rating

cmp eax, highestRating

jle skipUpdate

mov highestRating, eax

mov highestID, empID[esi\*4]

skipUpdate:

inc esi

cmp esi, 5

jl inputLoop

; Print total sum of ratings

mov edx, OFFSET resultSum

call WriteString

mov eax, sumRatings

call WriteInt

call Crlf

; Print highest rated employee ID and rating

mov edx, OFFSET resultHighest

call WriteString

mov eax, highestID

call WriteInt

mov edx, OFFSET resultRating

call WriteString

mov eax, highestRating

call WriteInt

call Crlf

exit

main ENDP

END main

Right triangle without loops

INCLUDE Irvine32.inc

.DATA

row1 BYTE "\**", 0*

*row2 BYTE "\*\*****", 0***

***row3 BYTE "\*\*\****", 0

row4 BYTE "\*\*\*\*", 0

row5 BYTE "\*\*\*\*\*", 0

.CODE

main PROC ; Print first row

mov edx, OFFSET row1

call WriteString

call Crlf

; Print second row  
mov edx, OFFSET row2  
call WriteString  
call Crlf  
  
; Print third row  
mov edx, OFFSET row3  
call WriteString  
call Crlf  
  
; Print fourth row  
mov edx, OFFSET row4  
call WriteString  
call Crlf  
  
; Print fifth row  
mov edx, OFFSET row5  
call WriteString  
call Crlf  
  
exit

main ENDP

END main

Subtration of array elements with direct addressing:

INCLUDE Irvine32.inc

.DATA

arr DWORD 50, 20, 10 ; Array with 3 elements

result DWORD ? ; Store result

msg BYTE "Result of Subtraction: ", 0

.CODE

main PROC

mov eax, arr ; Load first element (arr[0])

sub eax, arr+4 ; Subtract second element (arr[1])

sub eax, arr+8 ; Subtract third element (arr[2])

mov result, eax ; Store result

; Print result

mov edx, OFFSET msg

call WriteString

mov eax, result

call WriteInt

call Crlf

exit

main ENDP

END main

Implement formula (a-b)^2:

INCLUDE Irvine32.inc

.DATA

a DWORD 10 ; Value of a

b DWORD 4 ; Value of b

result DWORD ? ; Store result

msg BYTE "(a - b)^2 = ", 0

.CODE

main PROC

; Compute (a - b)

mov eax, a ; Load a into EAX

sub eax, b ; EAX = a - b

; Square the result

imul eax, eax ; EAX = (a - b) \* (a - b)

; Store result

mov result, eax

; Print result

mov edx, OFFSET msg

call WriteString

mov eax, result

call WriteInt

call Crlf

exit

main ENDP

END main

All Instructions:

INCLUDE Irvine32.inc

.DATA

a DWORD 15 ; First number

b DWORD 5 ; Second number

sumResult DWORD ? ; Store sum

subResult DWORD ? ; Store subtraction

mulResult DWORD ? ; Store multiplication

negA DWORD ? ; Store negation of a

negB DWORD ? ; Store negation of b

xchgA DWORD ? ; Store exchanged a

xchgB DWORD ? ; Store exchanged b

signExt WORD 127 ; 8-bit number to extend

zeroExt WORD 127 ; 8-bit number to extend

msgSum BYTE "Sum: ", 0

msgSub BYTE "Difference: ", 0

msgMul BYTE "Product: ", 0

msgNegA BYTE "Negation of A: ", 0

msgNegB BYTE "Negation of B: ", 0

msgXchg BYTE "After Exchange - A: ", 0

msgSignExt BYTE "Signed Extension: ", 0

msgZeroExt BYTE "Zero Extension: ", 0

.CODE

main PROC

; Addition: a + b

mov eax, a

add eax, b

mov sumResult, eax

; Subtraction: a - b

mov eax, a

sub eax, b

mov subResult, eax

; Multiplication: a \* b

mov eax, a

imul b

mov mulResult, eax

; Negation of a and b

mov eax, a

neg eax

mov negA, eax

mov eax, b

neg eax

mov negB, eax

; Exchange values of a and b

mov eax, a

mov ebx, b

xchg eax, ebx

mov xchgA, eax

mov xchgB, ebx

; Signed Extension (extending 8-bit to 16-bit)

movsx ax, signExt

; Zero Extension (extending 8-bit to 16-bit)

movzx bx, zeroExt

; Printing Results

mov edx, OFFSET msgSum

call WriteString

mov eax, sumResult

call WriteInt

call Crlf

mov edx, OFFSET msgSub

call WriteString

mov eax, subResult

call WriteInt

call Crlf

mov edx, OFFSET msgMul

call WriteString

mov eax, mulResult

call WriteInt

call Crlf

mov edx, OFFSET msgNegA

call WriteString

mov eax, negA

call WriteInt

call Crlf

mov edx, OFFSET msgNegB

call WriteString

mov eax, negB

call WriteInt

call Crlf

mov edx, OFFSET msgXchg

call WriteString

mov eax, xchgA

call WriteInt

call Crlf

mov eax, xchgB

call WriteInt

call Crlf

mov edx, OFFSET msgSignExt

call WriteString

movsx eax, signExt

call WriteInt

call Crlf

mov edx, OFFSET msgZeroExt

call WriteString

movzx eax, zeroExt

call WriteInt

call Crlf

exit

main ENDP

END main