COAL LAB 8

Example: 01 (Stack And Nested Loops)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

msg1 BYTE "Example: Stack and Nested Loops", 0

msg2 BYTE "Final value of EBX after nested loops: ", 0

.code

main PROC

mov ebx, 0

mov ecx, 5

L1:

push ecx

mov ecx, 10

L2:

inc ebx

loop L2

pop ecx

loop L1

call Crlf

mov edx, OFFSET msg1

call WriteString

call Crlf

mov edx, OFFSET msg2

call WriteString

mov eax, ebx

call WriteDec

call Crlf

exit

main ENDP

END main



Example: 02 (How Values Are Temporarily Stored And Retrieved From The Stack During Arithmetic Operations)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

num1 WORD 15

num2 WORD 25

msg1 BYTE "Initial Numbers: ", 0

msg2 BYTE "Sum after POP (A + B): ", 0

msg3 BYTE "Final Result after PUSH & POP: ", 0

space BYTE " ", 0

.code

main PROC

call Crlf

mov edx, OFFSET msg1

call WriteString

call Crlf

movzx eax, num1

call WriteDec

mov edx, OFFSET space

call WriteString

movzx eax, num2

call WriteDec

call Crlf

mov ax, num1

push ax

mov ax, num2

push ax

pop bx

pop ax

add ax, bx

call Crlf

mov edx, OFFSET msg2

call WriteString

movzx eax, ax

call WriteDec

call Crlf

push ax

pop bx

call Crlf

mov edx, OFFSET msg3

call WriteString

movzx eax, bx

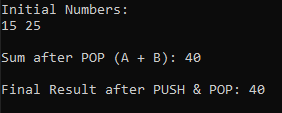
call WriteDec

call Crlf

exit

main ENDP

END main



Example: 03 (Demonstrates How Stack-Based Memory Management Supports Real- World Features Like Undo/Redo Or Version Rollback. Reinforces Understanding Of Last-In, First-Out (Lifo) Principle Crucial For Recursion, Function Calls, And Backtracking Algorithms)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

versionHistory WORD 101, 102, 103, 104, 105

rollbackHistory WORD 5 DUP(?)

msg1 BYTE "Version History (Latest Last):", 0

msg2 BYTE "Rollback Order (After Using PUSH & POP):", 0

space BYTE " ", 0

.code

main PROC

call Crlf

mov edx, OFFSET msg1

call WriteString

call Crlf

; Display original version history

mov ecx, LENGTHOF versionHistory

mov esi, OFFSET versionHistory

displayOriginal:

movzx eax, WORD PTR [esi]

call WriteDec

mov edx, OFFSET space

call WriteString

add esi, TYPE versionHistory

loop displayOriginal

; Push all version numbers to stack

mov ecx, LENGTHOF versionHistory

mov esi, OFFSET versionHistory

pushVersions:

mov ax, [esi]

push ax

add esi, TYPE versionHistory

loop pushVersions

; Pop all version numbers in reverse order

mov ecx, LENGTHOF rollbackHistory

mov edi, OFFSET rollbackHistory

popVersions:

pop ax

mov [edi], ax

add edi, TYPE rollbackHistory

loop popVersions

call Crlf

mov edx, OFFSET msg2

call WriteString

call Crlf

; Display rollback array (reversed)

mov ecx, LENGTHOF rollbackHistory

mov esi, OFFSET rollbackHistory

displayRollback:

movzx eax, WORD PTR [esi]

call WriteDec

mov edx, OFFSET space

call WriteString

add esi, TYPE versionHistory

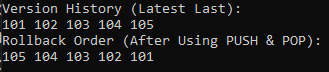
loop displayRollback

call Crlf

exit

main ENDP

END main



Example: 04 (This Program Pushes Three Integers Onto The Stack And Then Pops Them To Compute Their Product)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

mult DWORD 2

msg BYTE "Hello Sir! The product of the three integers is: ", 0

.code

main PROC

mov eax, 1

mov ecx, 3

PushLoop:

push mult

add mult, 2

loop PushLoop

mov ecx, 3

MultiplyLoop:

pop ebx

mult ebx

loop MultiplyLoop

mov edx, OFFSET msg

call WriteString

call WriteDec

call Crlf

exit

main ENDP

END main



Example: 05 (To Find The Largest Number Through A Stack)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

msg BYTE "Hello! The largest number is:", 0

.code

main PROC

push 5

push 7

push 3

push 2

mov eax, 0

mov ecx, 4

L1:

pop edx

cmp edx, eax

jle Next

mov eax, edx

Next:

loop L1

mov edx, OFFSET msg

call WriteString

call WriteDec

call Crlf

exit

main ENDP

END main



Example: 06 (Use PUSHFD and POPAD instructions in stack)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

originalFlagsMsg BYTE "Original Flags saved on stack.", 0

restoredFlagsMsg BYTE "Flags restored from stack.", 0

.code

main PROC

mov eax, 5

sub eax, 5

pushfd

mov edx, OFFSET originalFlagsMsg

call WriteString

call Crlf

mov eax, 10

add eax, 1

popfd

mov edx, OFFSET restoredFlagsMsg

call WriteString

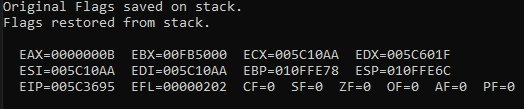
call Crlf

call DumpRegs

exit

main ENDP

END main



Example: 07 (Add Two Numbers Using Procedures)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

var1 DWORD 5

var2 DWORD 6

msg BYTE "The sum of the two numbers is: ", 0

.code

main PROC

call AddTwo

mov edx, OFFSET msg

call WriteString

call WriteDec

call Crlf

exit

main ENDP

AddTwo PROC

mov eax, var1

add eax, var2

ret

AddTwo ENDP

END main



Example: 08 (The Sum Of Integers Using Procedures)

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

SIZE = 3

.data

str1 BYTE "Enter a signed integer: ", 0

str2 BYTE "The sum of the integers is: ", 0

array DWORD SIZE DUP(?)

.code

main PROC

call Clrscr

mov esi, OFFSET array

mov ecx, SIZE

call PromptForIntegers

call ArraySum

call DisplaySum

exit

main ENDP

PromptForIntegers PROC USES esi ecx

mov edx, OFFSET str1

L1:

call WriteString

call ReadInt

call Crlf

mov [esi], eax

add esi, TYPE DWORD

loop L1

ret

PromptForIntegers ENDP

ArraySum PROC USES esi ecx

mov eax, 0

L1:

add eax, [esi]

add esi, TYPE DWORD

loop L1

ret

ArraySum ENDP

DisplaySum PROC USES edx

mov edx, OFFSET str2

call WriteString

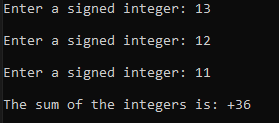
call WriteInt

call Crlf

ret

DisplaySum ENDP

END main



Example: 09

INCLUDE Irvine32.inc

;24K-0691 Abdullah Razzaq

.data

var1 DWORD 5

var2 DWORD 6

msg1 BYTE "The sum calculated in AddTwo is: ", 0

msg2 BYTE "Values printed inside AddTwo1:", 0

.code

main PROC

call AddTwo

call Crlf

exit

main ENDP

AddTwo PROC

mov eax, var1

mov ebx, var2

add eax, var2

mov edx, OFFSET msg1

call WriteString

call WriteInt

call CrlNext

call AddTwo1

ret

AddTwo ENDP

AddTwo1 PROC

mov ecx, var1

mov edx, var2

mov ebx, OFFSET msg2

call WriteString

call Crlf

mov eax, ecx

call WriteInt

call Crlf

mov eax, edx

call WriteInt

call Crlf

ret

AddTwo1 ENDP

END main

