Eric Pereira & Dmitri Piquero

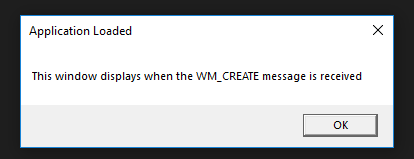
Lab 11.2

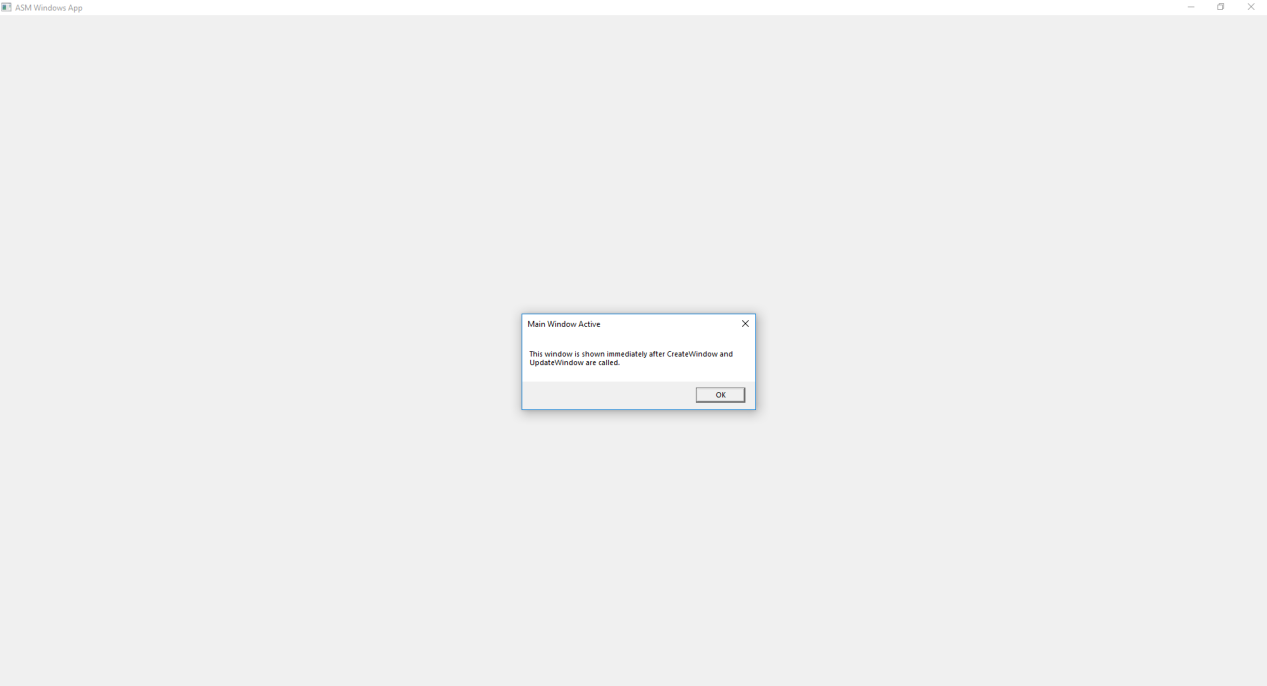
Architecture & Assembly

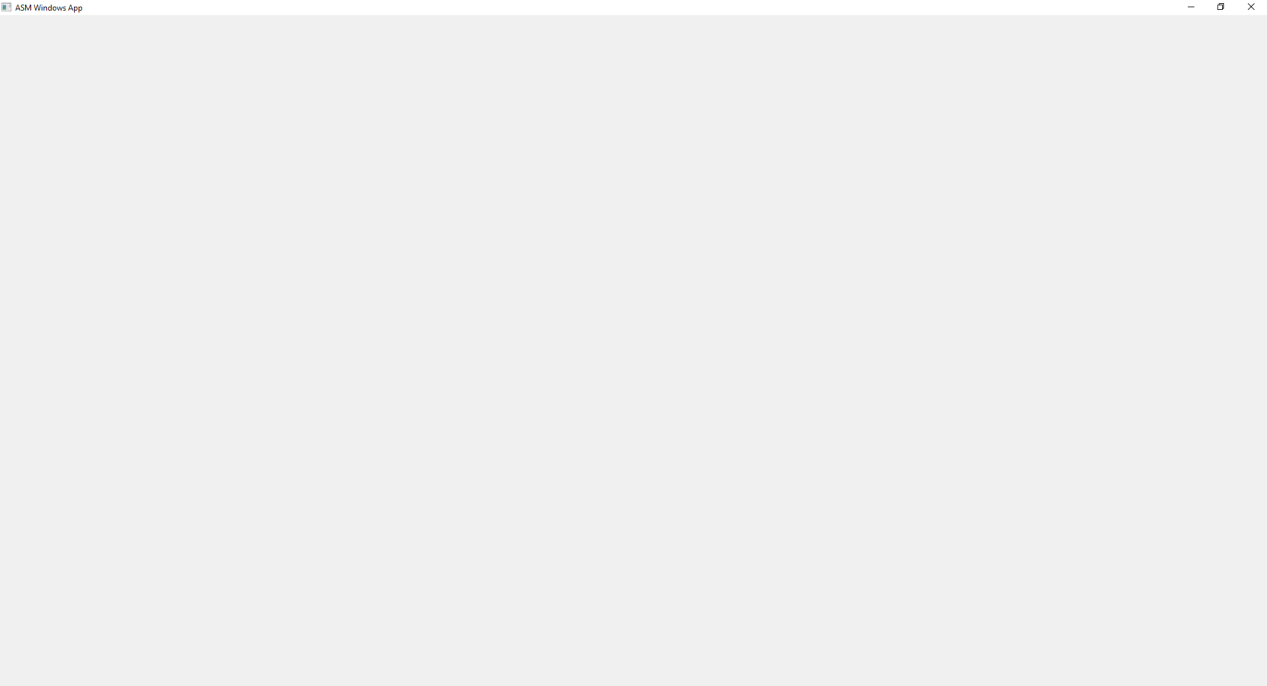
November 14, 2018

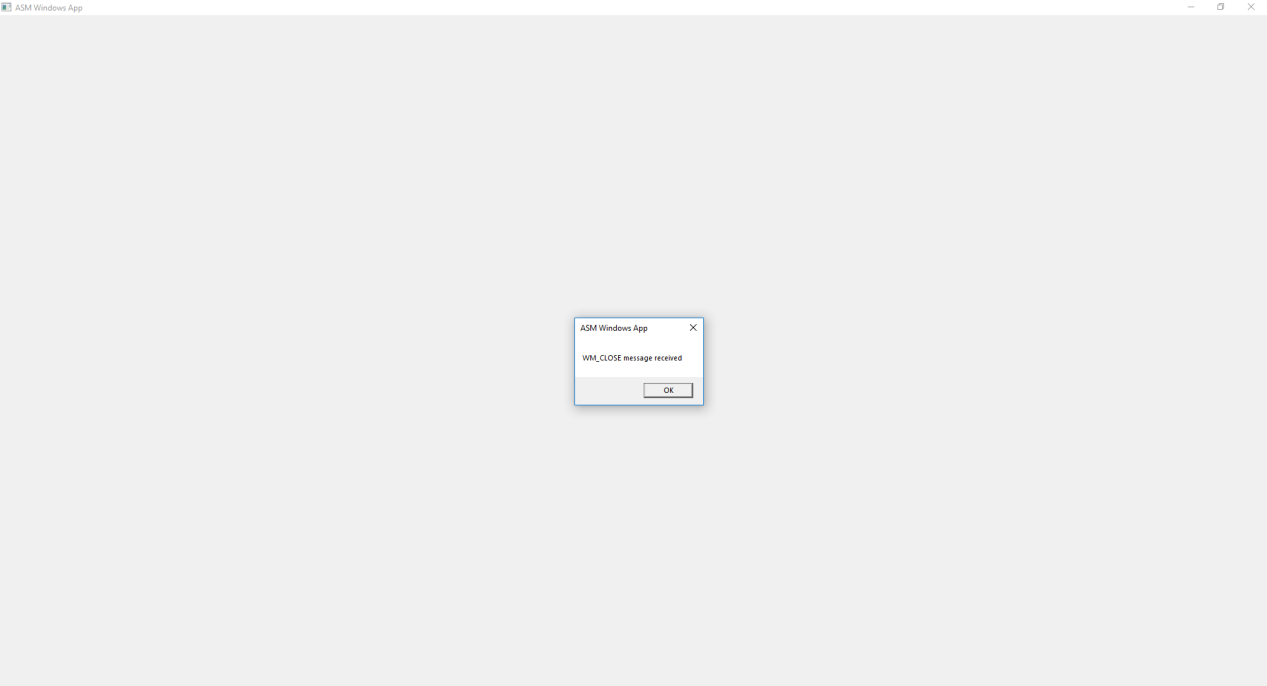
Windows Graphics

Part 1.

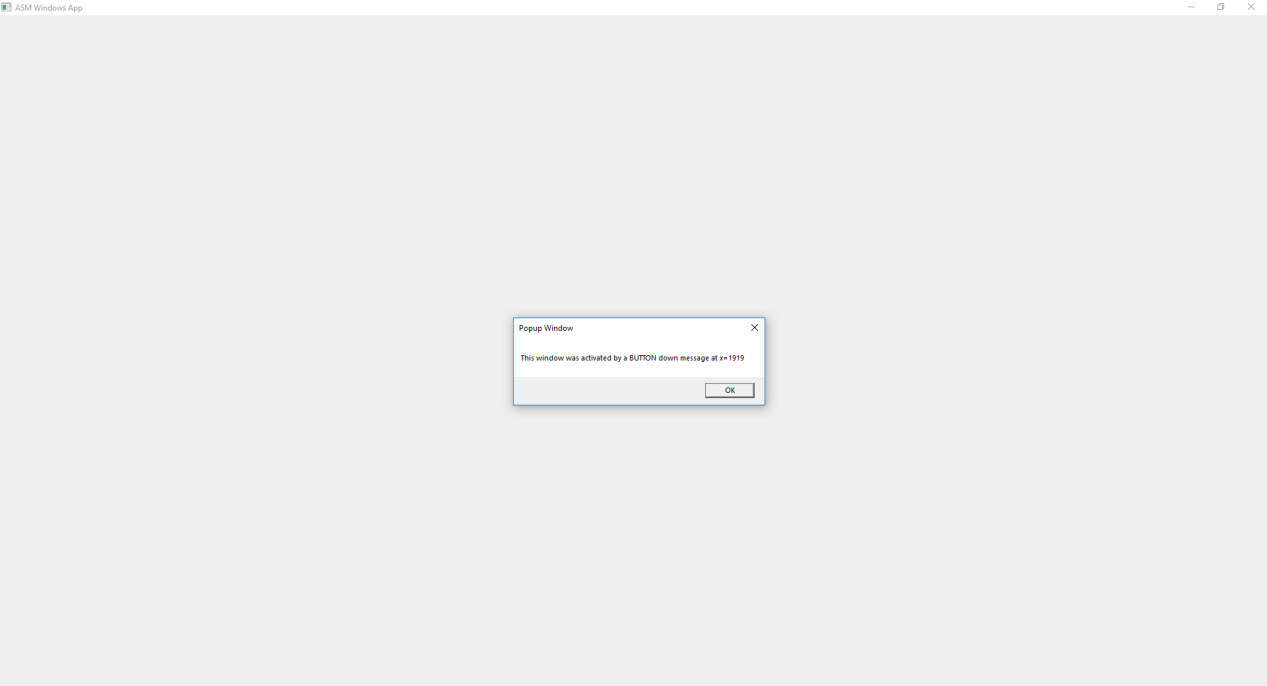








Part 2.



Source:

;-----------------------------------------------------

WinProc PROC,

hWnd:DWORD, localMsg:DWORD, wParam:DWORD, lParam:DWORD

; The application's message handler, which handles

; application-specific messages. All other messages

; are forwarded to the default Windows message

; handler.

;-----------------------------------------------------

mov DX, 0

mov BX, 10

mov AX, WORD PTR lparam

div BX

xor DX, 30h

mov popupx + 3, DL

mov DX, 0

div BX

xor DX, 30h

mov popupx + 2, DL

mov DX, 0

div BX

xor DX, 30h

mov popupx + 1, DL

mov DX, 0

div BX

xor DX, 30h

mov popupx + 0, DL

mov eax, localMsg

.IF eax == WM\_LBUTTONDOWN ; mouse button?

INVOKE MessageBox, hWnd, ADDR PopupText,

ADDR PopupTitle, MB\_OK

jmp WinProcExit

.ELSEIF eax == WM\_CREATE ; create window?

INVOKE MessageBox, hWnd, ADDR AppLoadMsgText,

ADDR AppLoadMsgTitle, MB\_OK

jmp WinProcExit

.ELSEIF eax == WM\_CLOSE ; close window?

INVOKE MessageBox, hWnd, ADDR CloseMsg,

ADDR WindowName, MB\_OK

INVOKE PostQuitMessage,0

jmp WinProcExit

.ELSE ; other message?

INVOKE DefWindowProc, hWnd, localMsg, wParam, lParam

jmp WinProcExit

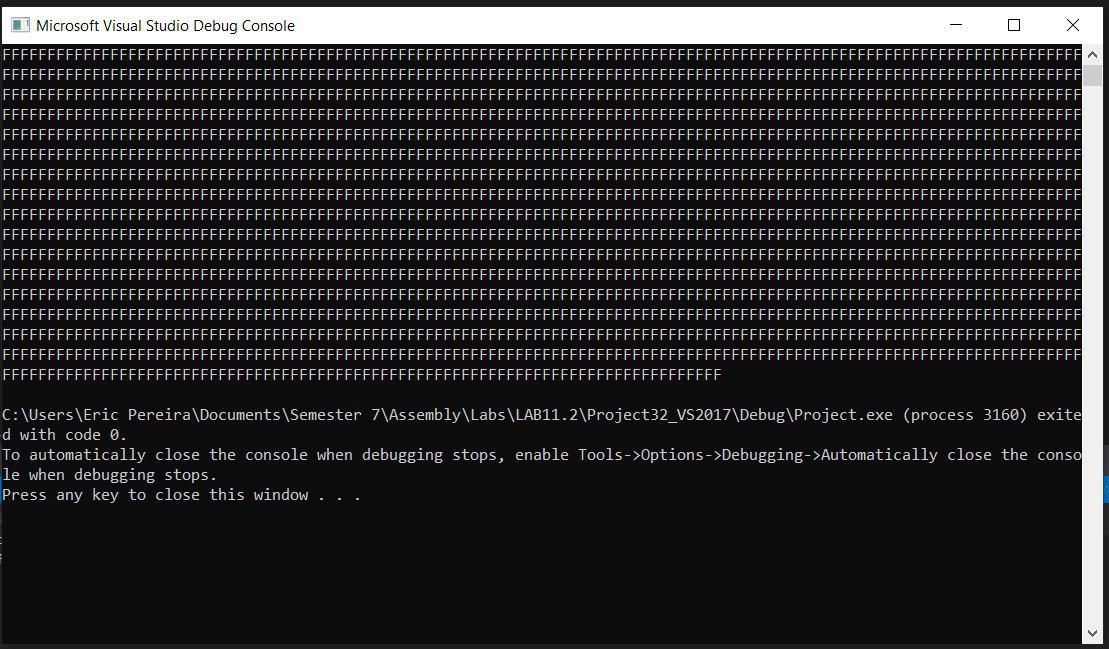
.ENDIF

WinProcExit:

ret

WinProc ENDP

Part 3.



Part 4.

; Windows Application (WinApp.asm)

; This program displays a resizable application window and

; several popup message boxes.

; Thanks to Tom Joyce for creating a prototype

; from which this program was derived.

INCLUDE Irvine32.inc

INCLUDE GraphWin.inc

;==================== DATA =======================

.data

;FOR HEAP

FILL\_VAL EQU 0FFh

hHeap DWORD ? ; handle to the process heap

AppLoadMsgTitle BYTE "Application Loaded",0

AppLoadMsgText BYTE "This window displays when the WM\_CREATE "

BYTE "message is received",0

PopupTitle Byte "Popup Window",0

PopupText byte "This window was activated by a BUTTON down message at x="

popupx BYTE 4 DUP(?)

BYTE 0

BYTE "WM\_LBUTTONDOWN message",0

PopupTextSize BYTE 4

ARRAY\_SIZE DWORD ($-PopupText)

buffer dword 55 DUP (?)

GreetTitle BYTE "Main Window Active",0

GreetText BYTE "This window is shown immediately after "

BYTE "CreateWindow and UpdateWindow are called.",0

CloseMsg BYTE "WM\_CLOSE message received",0

ErrorTitle BYTE "Error",0

WindowName BYTE "ASM Windows App",0

className BYTE "ASMWin",0

; Define the Application's Window class structure.

MainWin WNDCLASS <NULL,WinProc,NULL,NULL,NULL,NULL,NULL, \

COLOR\_WINDOW,NULL,className>

msg MSGStruct <>

winRect RECT <>

hMainWnd DWORD ?

hInstance DWORD ?

;=================== CODE =========================

.code

WinMain PROC

; Get a handle to the current process.

INVOKE GetModuleHandle, NULL

mov hInstance, eax

mov MainWin.hInstance, eax

; Load the program's icon and cursor.

INVOKE LoadIcon, NULL, IDI\_APPLICATION

mov MainWin.hIcon, eax

INVOKE LoadCursor, NULL, IDC\_ARROW

mov MainWin.hCursor, eax

; Register the window class.

INVOKE RegisterClass, ADDR MainWin

.IF eax == 0

call ErrorHandler

jmp Exit\_Program

.ENDIF

; Create the application's main window.

; Returns a handle to the main window in EAX.

INVOKE CreateWindowEx, 0, ADDR className,

ADDR WindowName,MAIN\_WINDOW\_STYLE,

CW\_USEDEFAULT,CW\_USEDEFAULT,CW\_USEDEFAULT,

CW\_USEDEFAULT,NULL,NULL,hInstance,NULL

mov hMainWnd,eax

; If CreateWindowEx failed, display a message & exit.

.IF eax == 0

call ErrorHandler

jmp Exit\_Program

.ENDIF

; Show and draw the window.

INVOKE ShowWindow, hMainWnd, SW\_SHOW

INVOKE UpdateWindow, hMainWnd

; Display a greeting message.

INVOKE MessageBox, hMainWnd, ADDR GreetText,

ADDR GreetTitle, MB\_OK

; Begin the program's message-handling loop.

Message\_Loop:

; Get next message from the queue.

INVOKE GetMessage, ADDR msg, NULL,NULL,NULL

; Quit if no more messages.

.IF eax == 0

jmp Exit\_Program

.ENDIF

; Relay the message to the program's WinProc.

INVOKE DispatchMessage, ADDR msg

jmp Message\_Loop

Exit\_Program:

INVOKE ExitProcess,0

WinMain ENDP

WinProc PROC,

hWnd:DWORD, localMsg:DWORD, wParam:DWORD, lParam:DWORD

; The application's message handler, which handles

; application-specific messages. All other messages

; are forwarded to the default Windows message

; handler.

;-----------------------------------------------------

mov DX, 0

mov BX, 10

mov AX, WORD PTR lparam

div BX

xor DX, 30h

mov popupx + 3, DL

mov DX, 0

div BX

xor DX, 30h

mov popupx + 2, DL

mov DX, 0

div BX

xor DX, 30h

mov popupx + 1, DL

mov DX, 0

div BX

xor DX, 30h

mov popupx + 0, DL

mov eax, localMsg

.IF eax == WM\_LBUTTONDOWN ; mouse button?

INVOKE GetProcessHeap ; get handle to prog's heap

.IF eax == NULL ; failed?

call WriteWindowsMsg

jmp WinProcExit

.ELSE

mov hHeap,eax ; success

.ENDIF

call allocate\_array

jnc arrayOk ; failed (CF = 1)?

call WriteWindowsMsg

call Crlf

jmp WinProcExit

arrayOk: ; ok to fill the array

call Crlf

cld

mov esi, OFFSET PopupText

mov edi, offset buffer

mov ecx, Array\_Size

rep movsb

mov eax, localMsg

INVOKE MessageBox, hWnd, ADDR PopupText,

ADDR PopupTitle, MB\_OK

; free the array

INVOKE HeapFree, hHeap, 0, buffer

jmp WinProcExit

.ELSEIF eax == WM\_CREATE ; create window?

INVOKE MessageBox, hWnd, ADDR AppLoadMsgText,

ADDR AppLoadMsgTitle, MB\_OK

jmp WinProcExit

.ELSEIF eax == WM\_CLOSE ; close window?

INVOKE MessageBox, hWnd, ADDR CloseMsg,

ADDR WindowName, MB\_OK

INVOKE PostQuitMessage,0

jmp WinProcExit

.ELSE ; other message?

INVOKE DefWindowProc, hWnd, localMsg, wParam, lParam

jmp WinProcExit

.ENDIF

WinProcExit:

ret

WinProc ENDP

;---------------------------------------------------

ErrorHandler PROC

; Display the appropriate system error message.

;---------------------------------------------------

.data

pErrorMsg DWORD ? ; ptr to error message

messageID DWORD ?

.code

INVOKE GetLastError ; Returns message ID in EAX

mov messageID,eax

; Get the corresponding message string.

INVOKE FormatMessage, FORMAT\_MESSAGE\_ALLOCATE\_BUFFER + \

FORMAT\_MESSAGE\_FROM\_SYSTEM,NULL,messageID,NULL,

ADDR pErrorMsg,NULL,NULL

; Display the error message.

INVOKE MessageBox,NULL, pErrorMsg, ADDR ErrorTitle,

MB\_ICONERROR+MB\_OK

; Free the error message string.

INVOKE LocalFree, pErrorMsg

ret

ErrorHandler ENDP

allocate\_array PROC USES eax

;

; Dynamically allocates space for the array.

; Receives: nothing

; Returns: CF = 0 if allocation succeeds.

;--------------------------------------------------------

INVOKE HeapAlloc, hHeap, HEAP\_ZERO\_MEMORY, ARRAY\_SIZE

.IF eax == NULL

stc ; return with CF = 1

.ELSE

mov buffer, eax ; save the pointer

clc ; return with CF = 0

.ENDIF

ret

allocate\_array ENDP

END WinMain