

Manual Técnico

Marvin Eduardo Catalán Véliz

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Arquitectura de Computadores y Ensambladores 1 Sección A

Ing. Otto Rene Escobar

Aux. Oscar Rolando Bernard Peralta

MANUAL TECNICO PRÁCTICA 4

Marvin Eduardo Catalán Véliz

201905554

Guatemala, 01 de abril de 2022

INDICE

REQUERIMIENTOS TÉCNICOS PARA SU EJECUCIÓN.....	3
HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO	3
DOSBOX.....	3
MASM 6.11.....	3
EDITOR DE TEXTO	4
MAQUINA UTILIZADA.....	5
SISTEMA OPERATIVO UTILIZADO.....	5
FUNCIONALIDA Y EXPLICACIÓN.....	6
INICIALIZAR EL PROYECTO.....	6
EXPLICACION main.asm y macros.asm.....	9
EXPLICACION main.asm.....	9
.data	9
.code	9
EXPLICACION macros.asm.....	10

REQUERIMIENTOS TÉCNICOS PARA SU EJECUCIÓN

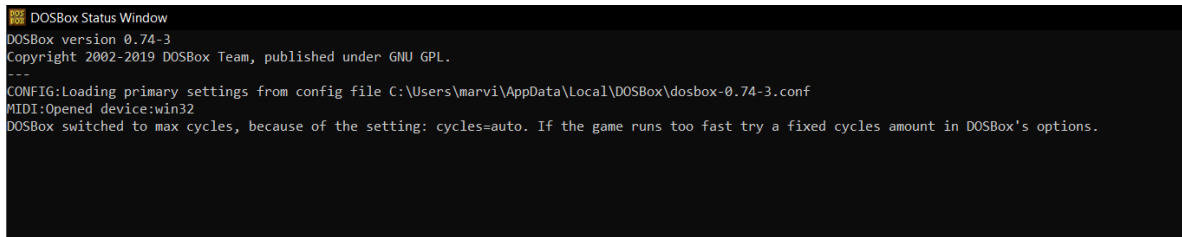
HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

DOSBOX

¿QUÉ ES?

Es un emulador que recrea un entorno similar al sistema MS-DOS con el objetivo de poder ejecutar programas y videojuegos originalmente escritos para dicho sistema en computadoras más modernas o en diferentes arquitecturas (como Power PC). También permite que estos juegos funcionen en otros sistemas operativos como GNU/Linux. Fue hecho porque Windows XP ya no se basa en MS-DOS y pasó a basarse a Windows NT.

VERSION DOSBOX UTILIZADA



```
DOSBox Status Window
DOSBox version 0.74-3
Copyright 2002-2019 DOSBox Team, published under GNU GPL.
--
CONFIG:Loading primary settings from config file C:\Users\marvi\AppData\Local\DOSBox\dosbox-0.74-3.conf
MIDI:Opened device:win32
DOSBox switched to max cycles, because of the setting: cycles=auto. If the game runs too fast try a fixed cycles amount in DOSBox's options.
```

MASM 6.11

¿QUÉ ES?

El Microsoft Macro Assembler (MASM) es un ensamblador para la familia x86 de microprocesadores. Fue producido originalmente por Microsoft para el trabajo de desarrollo en su sistema operativo MS-DOS, y fue durante cierto tiempo el ensamblador más popular disponible para ese sistema operativo. El MASM soportó una amplia variedad de facilidades para macros y programación estructurada, incluyendo construcciones de alto nivel para bucles, llamadas a procedimientos y alternación (por lo tanto, MASM es un ejemplo de un ensamblador de alto nivel). Versiones posteriores agregaron la capacidad de producir programas para los sistemas operativos Windows.

MASM es una de las pocas herramientas de desarrollo de Microsoft para las cuales no había versiones separadas de 16 bits y 32 bits.

VERSION MASM UTILIZADA



```
C:\>masm
Microsoft (R) MASM Compatibility Driver Version 6.11
Copyright (C) Microsoft Corp 1993. All rights reserved.
```

EDITOR DE TEXTO

¿QUÉ ES?

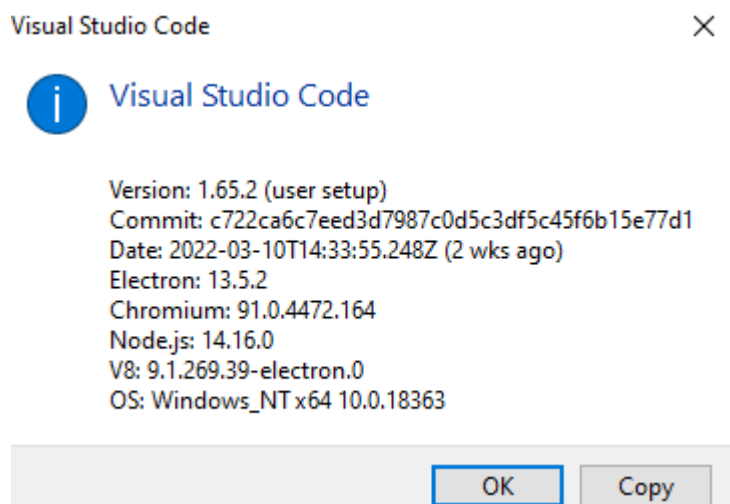
Editor de texto es un programa informático que permite crear y modificar archivos digitales compuestos únicamente por textos sin formato, conocidos comúnmente como archivos de texto o "texto plano". El programa lee el archivo e interpreta los bytes leídos según el código de caracteres que usa el editor. Es comúnmente de 7- u 8-bits en ASCII o UTF-8, rara vez EBCDIC.

EDITOR DE TEXTO UTILIZADO: VISUAL STUDIO CODE

Es un editor de código fuente que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado y refactorizar el código. Es gratuito, de código abierto y nos proporciona una utilidad para descargar y gestionar extensiones con las que podemos personalizar y potenciar esta herramienta.

Las extensiones de Visual Studio Code nos otorgan infinidad de opciones, como colorear tabulaciones, etiquetas o recomendaciones de autocompletado. También hay extensiones que nos ayudan con el lenguaje de programación que vayamos a usar, como por ejemplo para Python, C / C++, JavaScript, etc.

VERSION VSCODE



MAQUINA UTILIZADA

Especificaciones del dispositivo

Nombre del dispositivo	DESKTOP-KMQA5BN
Procesador	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
RAM instalada	12.0 GB (11.9 GB utilizable)
Id. del dispositivo	369A16F7-DB15-4299-B4CD-780C88B080B8
Id. del producto	00325-96001-12028-AAOEM
Tipo de sistema	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil	Compatibilidad con entrada táctil con 10 puntos táctiles

SISTEMA OPERATIVO UTILIZADO

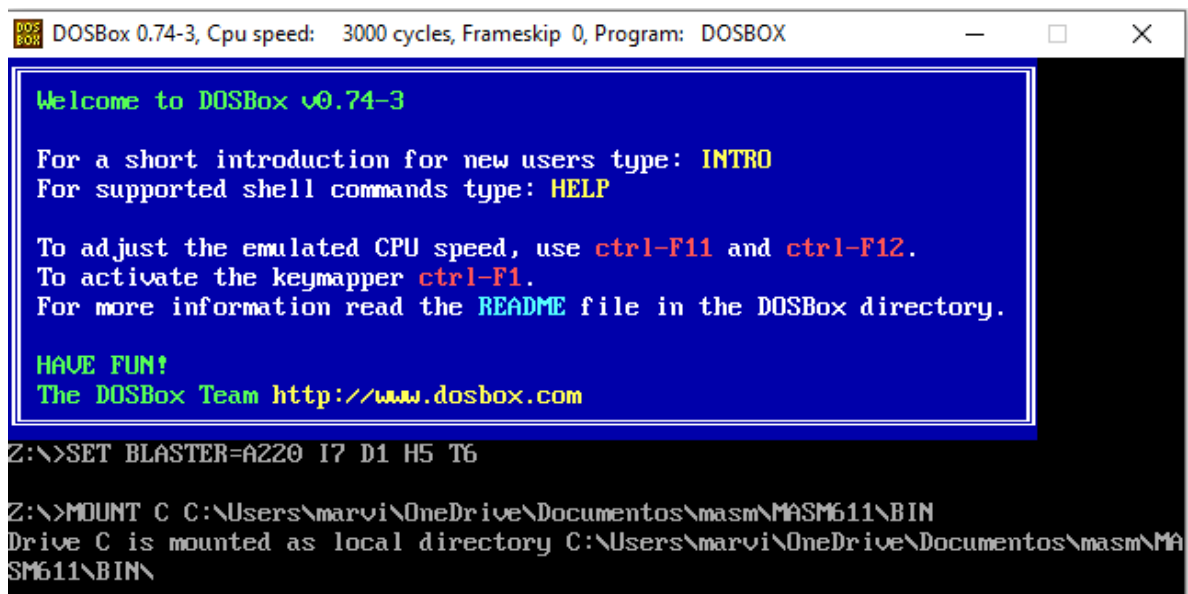
Especificaciones de Windows	
Edición	Windows 10 Home
Versión	1909
Se instaló el	15/07/2019
Compilación del SO	18363.1556

FUNCIONALIDA Y EXPLICACIÓN

El programa esta basado en lenguaje ensamblador MASM versión 6.11 que se divide en dos archivos principales que son main.asm y macros.asm posteriormente explicaremos de una mejor manera estos dos archivos, pero vamos veremos primero lo necesario para inicializar el proyecto.

INICIALIZAR EL PROYECTO

Como se mencionó en las herramientas utilizadas se creó en Dosbox por lo cual principalmente debemos abrir Dosbox y montar un disco nuevo con la dirección donde tengamos ubicada la carpeta BIN del MASM611



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C C:\Users\marvi\OneDrive\Documentos\masm\MASM611\BIN
Drive C is mounted as local directory C:\Users\marvi\OneDrive\Documentos\masm\MA
SM611\BIN\
```

La distribución de teclado utilizada en MS-DOS era un poco distinta a nuestras distribuciones actuales por lo cual posteriormente a montar el disco en la carpeta BIN del MASM cambiaremos la distribución del teclado (Este paso no es obligatorio, yo como desarrollador prefiero la distribución en inglés) con el comando keyb seguido de la distribución requerida.



```
Z:\>keyb us
Keyboard layout us loaded for codepage 437
```

En el siguiente enlace pueden encontrar la distribución que más les parezca o con la que más cómodos se sientan como desarrolladores. KEYB - DOSBoxWiki

Posteriormente a cambiar la distribución ingresaremos al disco montado, y podemos comprobar que efectivamente nos encontramos en un disco con MS-DOS.

```
Z:\>C:
C:\>masm
Microsoft (R) MASM Compatibility Driver Version 6.11
Copyright (C) Microsoft Corp 1993. All rights reserved.
```

Ahora que ya tenemos el disco montado con un ensamblador MASM podemos ejecutar el archivo main para que creamos un código objeto el cuál posteriormente será el archivo utilizado para generar un ejecutable.

Para poder compilar el código objeto lo realizamos con el comando *masm* seguido de la ruta donde se encuentra nuestro código con extensión asm.

```
C:\>masm P3EC/main.asm
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993. All rights reserved.

Invoking: ML.EXE /I. /Zm /c /Ta P3EC/main.asm

Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.

Assembling: P3EC/main.asm
```

En este momento si el código que realizamos no tiene ningún error nos ensamblara de forma correcta y nos generará en la carpeta bien un archivo de extensión .obj que posteriormente utilizaremos para generar el ejecutable.

```
MAIN      OBJ      8,215 25-03-2022  0:02
```

Cuando tengamos listo el código objeto de nuestro programa ya podemos generar nuestro ejecutable con el comando *link* seguido del nombre de el archivo .obj.

```
C:\>link main.obj

Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.

Run File [main.exe]:
List File [nul.map]:
Libraries [.lib]:
Definitions File [nul.def]:
```

Como podemos observar podemos modificar distintos parámetros del ejecutable a crear pero para este proyecto en específico podemos dejar todo default solamente presionando enter y continuando. Al momento de que se ejecute correctamente observaremos que tenemos ya un ejecutable de extensión .exe en nuestra carpeta BIN.

```
MAIN      EXE      6,702 25-03-2022  0:17
```

En este momento ya realizamos todo el proceso para generar nuestro código objeto y posteriormente generar nuestro ejecutable listo para que lo utilicé el usuario final que lo único que debe hacer es ejecutar el .exe obviamente desde el disco donde se encuentra montado el MASM611.

EXPLICACION MAIN.ASM Y MACROS.ASM

Como se mencionó anteriormente explicaríamos el por qué se utilizaban dos archivos, de igual manera puede el lector de este manual se este preguntando el por qué toda la inicialización se realizó solamente con el archivo *main.asm* esto se debe a que este es el archivo principal y que este por medio de un *include macros.asm* consume todo lo que contiene el otro archivo, esto se realizó para tener un mejor orden y que el código sea un poco más modular. Cabe resaltar que en el archivo *macros.asm* como su nombre lo hace intuir solamente contiene macros que en algún momento consume el main.

EXPLICACION MAIN.ASM

.DATA

Empezamos explicando todo lo que se encuentra dentro de este bloque de código, en este lugar declaramos todas aquellas variables que vamos a utilizar a lo largo de nuestro código, en este lugar se encuentran diferentes tipos de variables como sabemos solo podemos usar variables de tipo byte en MASM pero las podemos declarar de diferentes números de bytes dependiendo para que la utilizaremos, la gran mayoría de variables utilizadas en este proyecto son las más pequeñas que son de tipo db.

.CODE

En este bloque se encuentra nuestro procedimiento principal que obligatoriamente se tiene que ejecutar en nuestro programa y es el que se ejecuta al correr el ejecutable. Como primer punto obtenemos todas nuestra variables declaradas en data hacia el registro *ax*, posteriormente a esto inicia la lógica de nuestro proyecto que inicia con un menú de tres opciones.

```

main proc
    mov ax,@data
    mov ds,ax
    imprimirDatos:
        PrintText datosn
        PressEnter
    start:
        PrintText menu
        RecibirEntrada
        mov bl,al
        case1:
            cmp bl,"1"
            jne case2
            ;Calculadora
            PrintText modcalcu
            Calculadora
            jmp start
        case2:
            cmp bl,"2"
            jne case3
            ;Archivo
            PrintText modcarchi
            opcion1
            calcularMedia operaciones
            PrintText CommandShowMedia
            PrintText bufferMediaR
            RepoJSON IniRepo,FinRepo,TitRepo,FinTit,Datos,CierreDatos, nameFile,fecha, CierreFecha,Hora,CierreHora,Estadistica,CierreEst,Ope, CierreOpe, handleFichero
            jmp start
        case3:
            cmp bl,"3"
            jne case4
            PrintText modsalar
            mov ah,4ch
            int 21h
            jmp start
        case4:
            Cls
            jmp start
main endp

```

La macro *RecibirEntrada* tiene almacenada en *al*, la entrada que recibió del teclado del usuario que puede ser 1, 2 o 3 que entra a sus casos correspondiente dependiendo si hacen match en el *cmp*. Si en un caso no es ninguna de las tres opciones permitidas entra al *case4* que este solamente nos limpia la pantalla y nos muestra el menú de nuevo ya que se hace un salto a *start* para que se reinicie con el comando *jmp*.

Para culminar con el *main* tenemos los casos donde se ejecutan cada uno de los comandos mandando a llamar a cada uno de las macros necesarias para su funcionamiento.

EXPLICACION MACROS.ASM

Como se habló anteriormente este contiene todas las macros utilizadas por el *main*, estas macros son repetitivas con respecto a instrucciones se trata solo que acomodándolas a la lógica necesaria para poder ejecutar nuestro proyecto en curso, explicar cada una de estas sería meterse a un tema demasiado técnico y repetitivo que se puede resumir en la documentación oficial de *masm*, donde nos basamos en diferentes instrucciones y muy importantes también interrupciones para poder realizar las acciones requeridas. Se adjunta una documentación muy completa en la que estoy completamente seguro de que si no terminan de entender el por qué de algún comando o alguna interrupción les dejara todo muy claro. [emu8086 documentation \(yassinebridi.github.io\)](https://github.com/yassinebridi/emu8086_documentation)