



Universidad Nacional Autónoma de México
Facultad de Ingeniería
División de Ingeniería Eléctrica
Ingeniería en Computación
Estructura y Programación de Computadoras



Semestre: 2020-II

Grupo: 2

Manual de configuración de software en Windows

Objetivo:

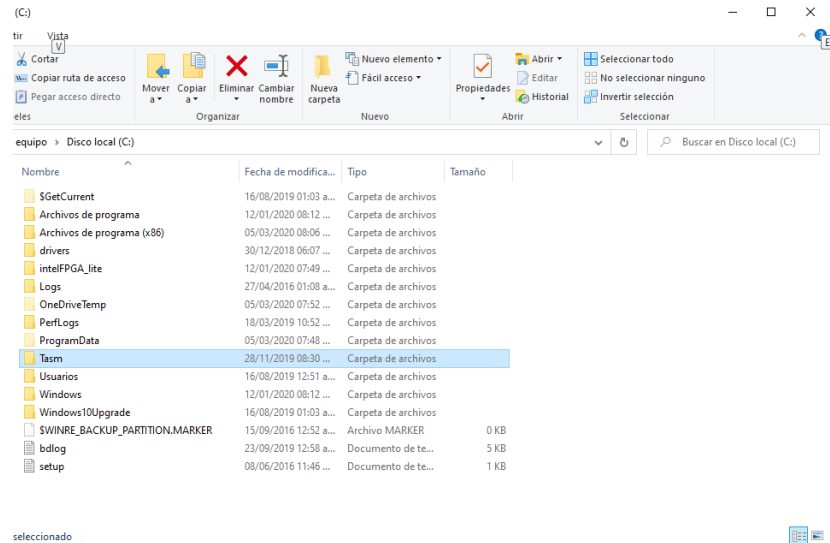
Configurar el entorno de desarrollo para el curso de Estructura y Programación de Computadoras en Windows para programación en lenguaje ensamblador de arquitectura Intel x86.

Archivos a descargar:

- DOSBox-0.74-3.exe
- Tasm.zip

Procedimiento:

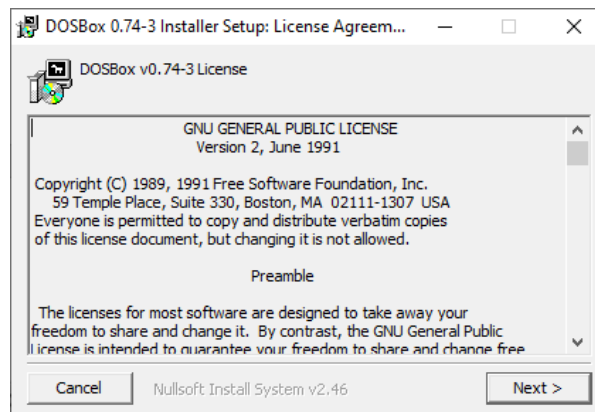
1. Descargar el archivo con extensión .exe de DOSBox para MacOS desde el sitio de DOSBox (<https://sourceforge.net/projects/dosbox/files/dosbox/0.74-3/DOSBox0.74-3-win32-installer.exe/download>) o desde la carpeta compartida en Google Drive.
2. Descargar el archivo Tasm.zip desde la carpeta compartida de Google Drive.
3. Estando en Windows, descomprimir el contenido del archivo Tasm.zip en la unidad C: (es posible que requiera permisos de administrador para tal efecto).



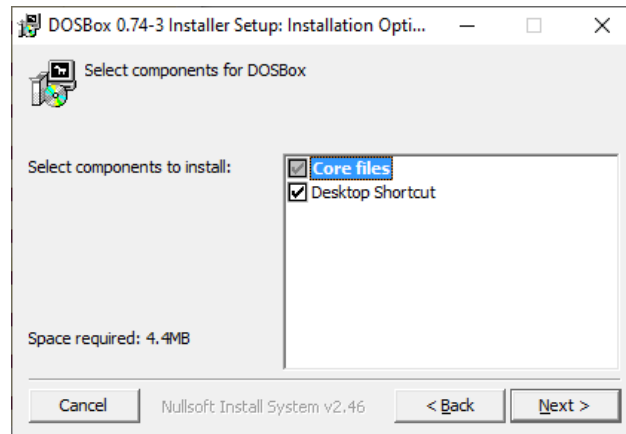
Los archivos que se encuentran en la carpeta comprimida son los que se utilizan para poder ejecutar Turbo Assembler (TASM). Principalmente utilizaremos 3: tasm.exe (Turbo Assembler), tlink.exe (Turbo Linker) y td.exe (Turbo Debugger); sin embargo, los demás archivos son útiles para las herramientas que se proporcionan por el Turbo Debugger, como las opciones de Ayuda.

4. Ejecutar el archivo DOSBox0.74-3-win32-installer.exe haciendo doble clic sobre él y una vez ejecutado, se abrirá una nueva ventana para la instalación de DOSBox.

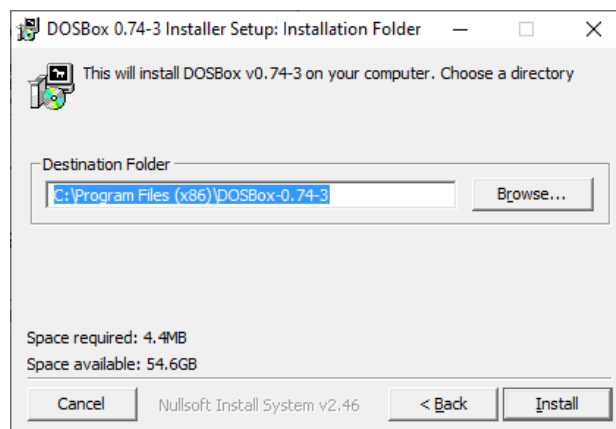
Leer los términos de la licencia GPL y clic en 'Next'.



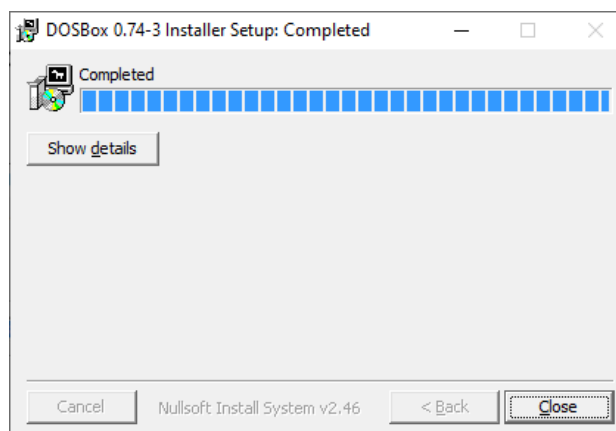
En la siguiente pantalla, mantener las opciones por defecto y dar clic en 'Next'.



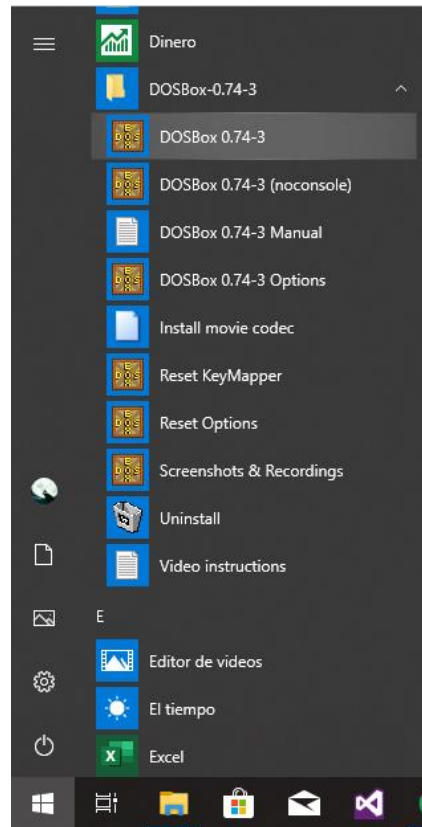
Seleccionar la ruta de instalación (o mantener la que se encuentra por default) y dar clic en 'Install'.



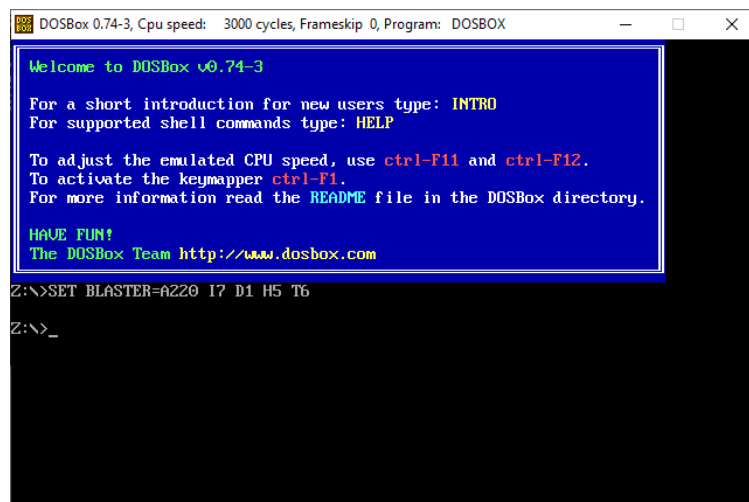
Finalizado el proceso de instalación, hacer clic en 'Close'.



5. Ir a mis programas (ícono de la ventana de Windows en la esquina inferior izquierda de la pantalla) y buscar DOSBox, abrir el programa.



6. Ahora ya podremos ver la ventana de DOSBox. Ésta luce de la siguiente manera:



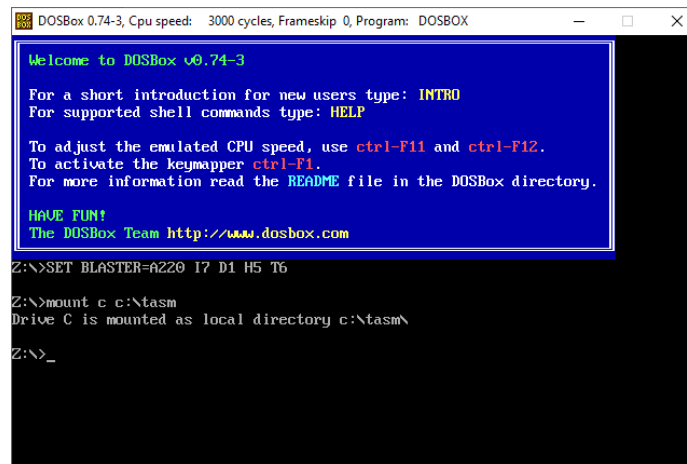
Esta terminal funciona del mismo modo que DOS. Inicialmente se ejecuta en una unidad distinta a la del disco duro. DOSBox se ejecuta en la unidad 'Z' por default. Para que se puedan ver los archivos de la computadora es necesario "montar" la unidad.

7. A continuación, sólo se montará una parte de la unidad del disco local. Se montará la carpeta de los archivos de Tasm, es decir, la carpeta que se encuentra en C:\Tasm

Para lograr esto, en la terminal de DOSBox se introduce el comando:

Z:\> mount c c:\tasm

Y dar [enter].



```
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:\tasm
Drive C is mounted as local directory c:\tasm\

Z:\>_
```

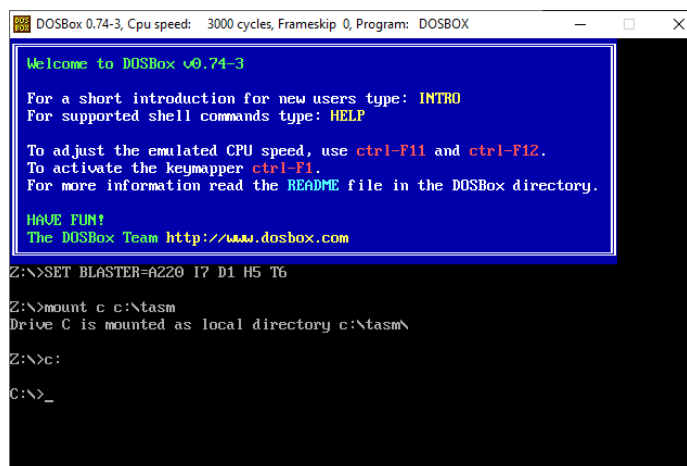
Con el comando anterior, se indica que se desea montar la carpeta C:\Tasm (segundo argumento) en una unidad virtual etiquetada con la letra C (primer argumento). De ese modo, se podrá acceder a la carpeta a través de la unidad virtual.

8. Para hacer el cambio de unidad, se introduce el siguiente comando:

Z:\> c:

Y dar enter

De esa manera, el prompt cambia a C:\> y eso indica que DOSBox ya se encuentra en carpeta C:\Tasm.



```
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:\tasm
Drive C is mounted as local directory c:\tasm\

Z:\>c:
C:\>_
```

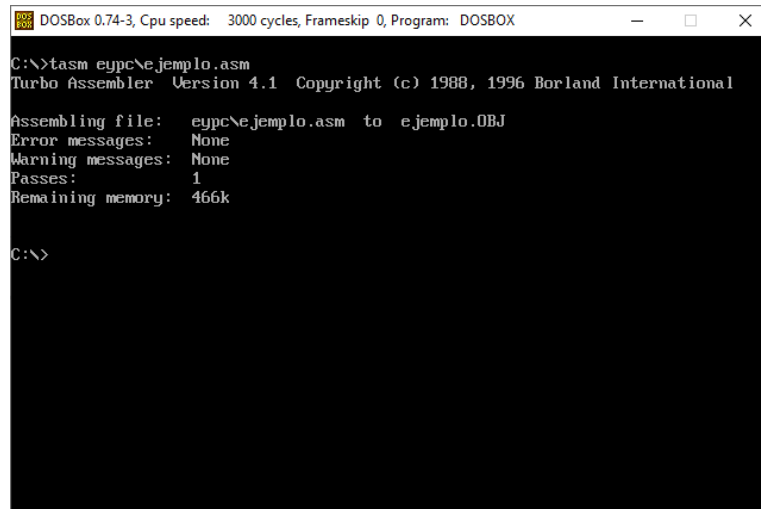
9. Para confirmar que se encuentra funcionando, dentro de la carpeta comprimida Tasm hay una carpeta más, llamada 'eypc' en donde se encuentra un código fuente de ejemplo. El procedimiento para ensamblar un código fuente en lenguaje ensamblador es el siguiente:

- a. Desde la unidad C:, introducir el siguiente comando:

```
C:\> tasm eypc\ejemplo.asm
```

Y dar enter.

Lo que sucede es que DOSBox busca el comando tasm dentro de la carpeta C:\Tasm y encuentra el archivo tasm.exe el cual será ejecutado. TASM ensambla el código fuente enviado como parámetro. En la terminal se observa el resultado del ensamblado y si no hubo errores, se genera un archivo objeto (.obj) por default en la misma ubicación que el archivo tasm.exe.

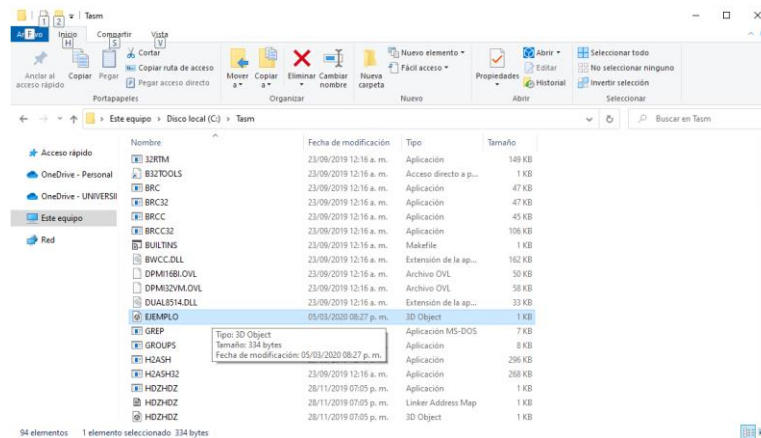


```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tasm eypc\ejemplo.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file: eypc\ejemplo.asm to ejemplo.OBJ
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 466k

C:\>
```



- b. Ahora, desde la misma ubicación, introducir:

```
C:\> tlink ejemplo.obj
```

Y dar [enter].

El comando tlink ejecuta el archivo tlink.exe, el cual es un enlazador y que va a generar un archivo ejecutable a partir del archivo objeto. Hace uso de uno o más archivo objeto de ser necesario. Al final se generará un archivo .exe y un archivo .map. Se ignorará este último.

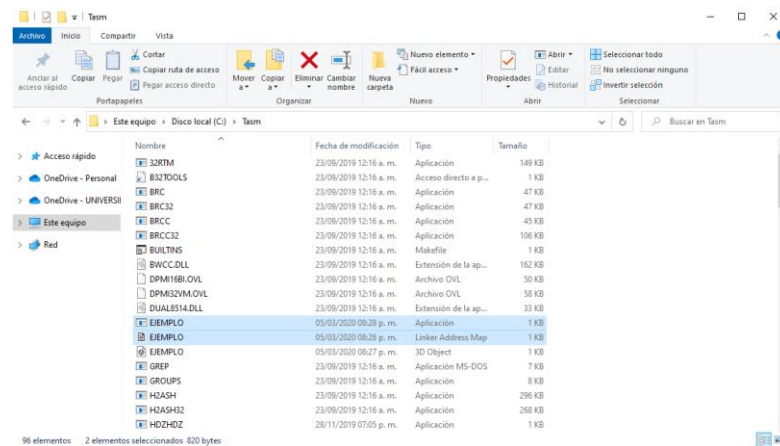
```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tasm eypc\ejemplo.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file: eypc\ejemplo.asm to ejemplo.OBJ
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 466k

C:\>tlink ejemplo.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

C:\>_
```

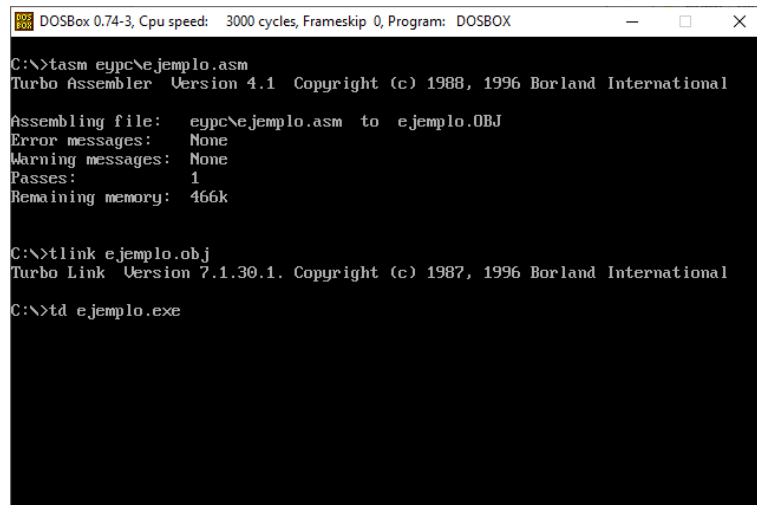


c. Para finalizar, se introducirá el siguiente comando:

C:\> td ejemplo.exe

Y dar [enter].

El comando td ejecuta el archivo td.exe, el Turbo Debugger, que se usará para hacer la depuración del programa generado. Aquí será posible revisar la ejecución el código, línea por línea, para corroborar su correcto funcionamiento.

A screenshot of a DOSBox window. The title bar reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The command prompt shows the following sequence of commands and output:

```
C:\>tasm eyy\ejemplo.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:   eyy\ejemplo.asm to ejemplo.OBJ
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 466k

C:\>tlink ejemplo.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

C:\>td ejemplo.exe
```

- d. El comando anterior abre el *Turbo Debugger* en el cual es posible depurar el código en lenguaje ensamblador. Para más detalles al respecto, ver Anexo 1 – *Turbo Debugger*.
10. Para editar el código fuente en lenguaje ensamblador se utilizará un programa editor de texto que es robusto y muy socorrido por desarrolladores en el campo laboral. Este programa se llama Sublime Text y se puede ver más información al respecto en el Anexo 2 – Editor de Texto *Sublime Text*

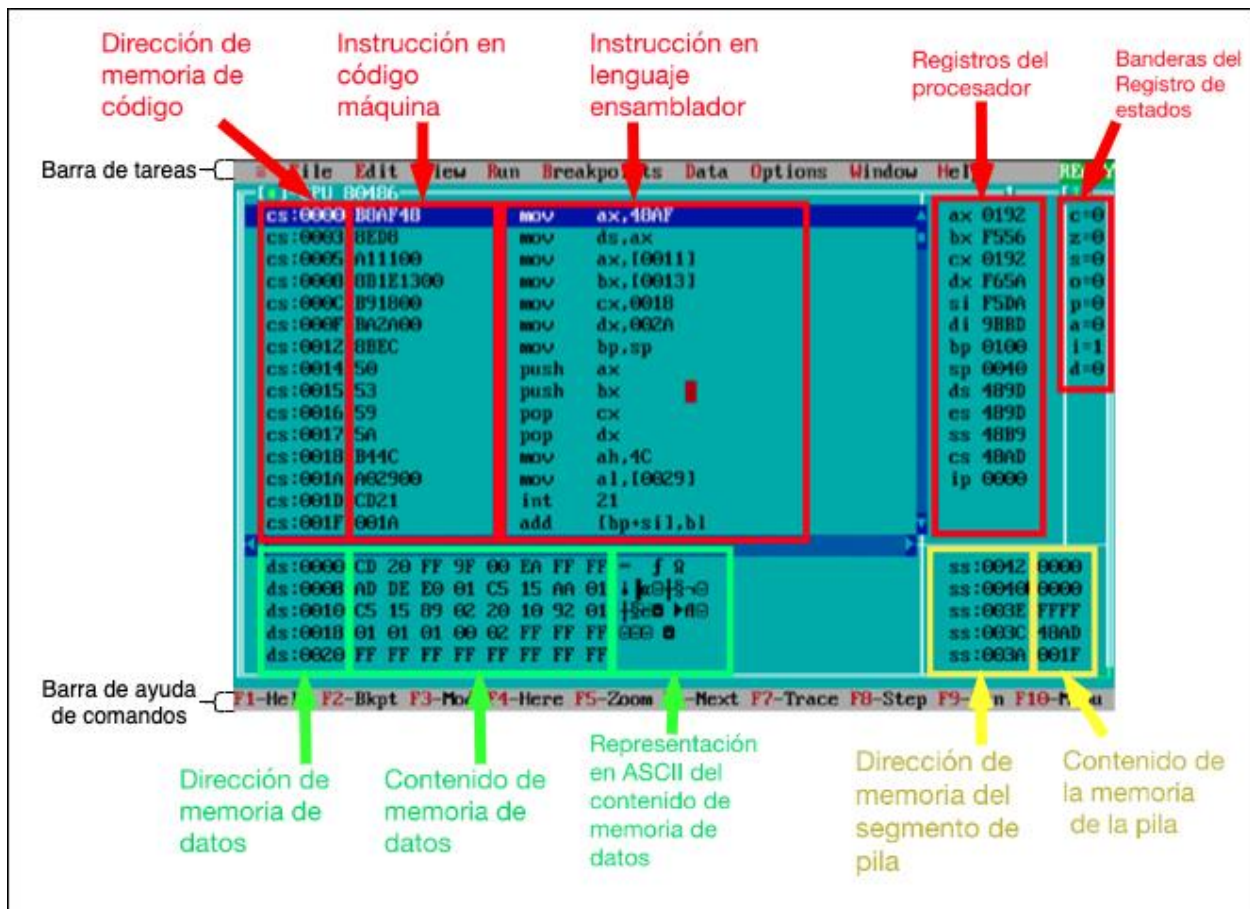
Anexo 1 – Turbo Debugger

Turbo Debugger es un programa utilizado para realizar la depuración de archivos ejecutables en DOS. Este programa es útil para depurar los programas que se generarán a partir del lenguaje ensamblador para arquitectura x86.

La ventana de *Turbo Debugger* es la siguiente:

The screenshot displays the Turbo Debugger window with the following components:

- Menu Bar:** File, Edit, View, Run, Breakpoints, Data, Options, Window, Help.
- Status Bar:** F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.
- Assembly Window:** Shows a list of instructions in the CS segment. The current instruction is at address 0000: B8AF4B, which is `mov ax, 48AF`. Other instructions include `mov ds, ax`, `mov ax, [0011]`, `mov bx, [0013]`, `mov cx, 0018`, `mov dx, 002A`, `mov bp, sp`, `push ax`, `push bx`, `pop cx`, `pop dx`, `mov ah, 4C`, `mov al, [0029]`, `int 21`, and `add [bp+si], bl`.
- Registers Window:** Displays the current values of the registers: ax=0192, bx=F556, cx=0192, dx=F65A, si=F5DA, di=98BD, bp=0100, sp=0040, ds=489D, es=489D, ss=48B9, cs=48AD, ip=0000. Flags are also shown: c=0, z=0, s=0, o=0, p=0, a=0, i=1, d=0.
- Memory Window:** Shows the contents of memory segments. The DS segment contains data at addresses 0000, 0008, 0010, 0018, and 0020. The SS segment contains data at addresses 0042, 0040, 003E, 003C, and 003A.



NOTA:

Inicialmente, el registro de segmento de datos (DS) apunta a un segmento aleatorio que no necesariamente es el segmento reservado para el programa. Cuando se hace uso del segmento de datos en un programa, siempre es necesario ejecutar 2 instrucciones al inicio para que el programa sepa cuál es el segmento correcto al que debe apuntar el registro DS. Estas dos instrucciones son:

```
mov ax, @data
```

```
mov ds, ax
```

@data es una palabra reservada por el ensamblador TASM que le indicará al programa la dirección del segmento de datos. La primera instrucción servirá para obtener esa dirección y ponerla en el registro AX, de manera que la segunda instrucción permite inicializar el registro DS con ese valor. Recordar que al registro DS no se puede asignar un valor inmediato, es por eso que se hace uso del registro AX como intermediario. Después de esas dos instrucciones.

Durante la depuración, el código se puede ejecutar línea por línea o ejecutar un bloque de código estableciendo puntos de quiebre (*breakpoints*) para detener la ejecución en donde éste se encuentre.

En la sección del segmento de datos, se puede observar el contenido de la memoria (incluyendo memoria de programa, datos y pila). Para cambiar la localidad que se visualiza, es posible dar clic derecho sobre la sección, y seleccionar la opción "Goto...". Entonces preguntará

por la dirección de la localidad, se puede introducir una dirección directamente: "48FD:0000", o se pueden utilizar los registros de segmento como apoyo: "ds:0000", luego seleccionar "Ok".

También es posible modificar el contenido de la memoria byte por byte directamente en la sección de datos. Sólo es necesario dar clic sobre el byte a modificar e introducir el valor, inmediatamente se abrirá un cuadro de texto mostrando el valor que se está introduciendo. Una vez que se finalizó la entrada de texto, hacer clic para cambiar el valor. Hay un detalle con valores a partir de A0 a FF ya que no es posible reconocer dichos valores si no se pone un '0' a la izquierda de éstos. Es decir, para introducir el valor 'B5' en una localidad, se deberá introducir el valor '0B5' en el cuadro de texto.

Anexo 2 – Editor de Texto *Sublime Text*

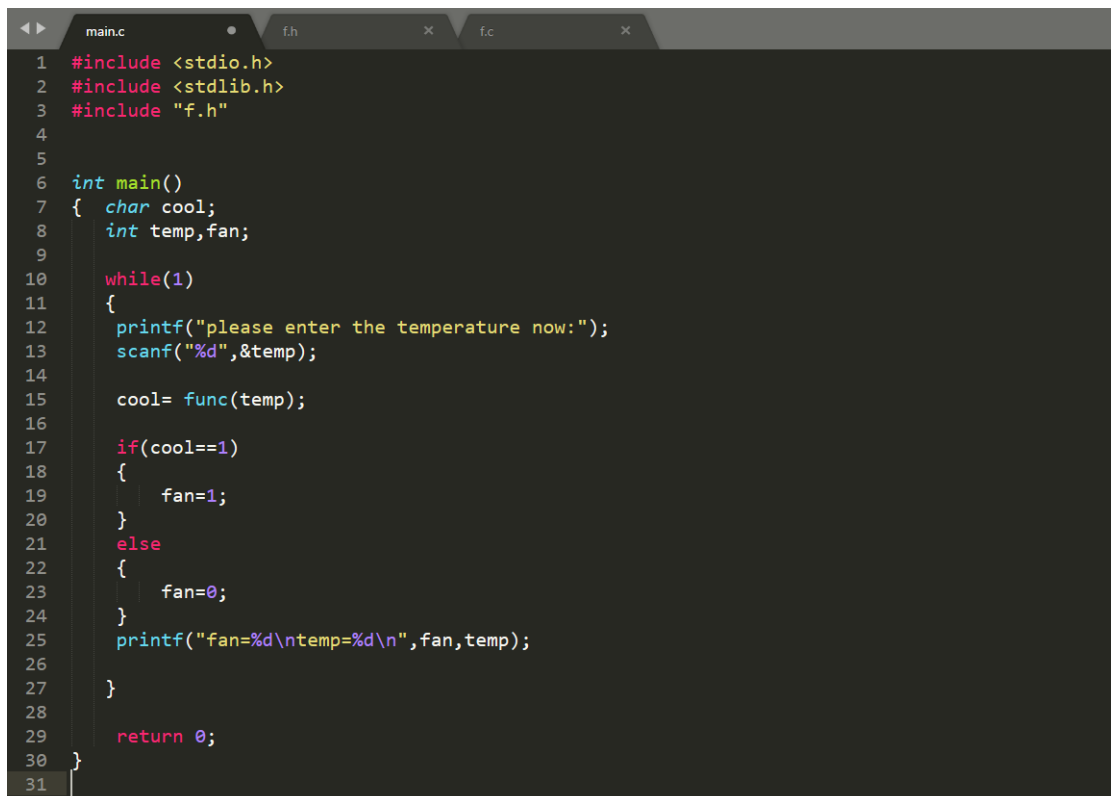
Para crear un programa en lenguaje ensamblador se puede utilizar cualquier editor de texto plano. Cada archivo en lenguaje ensamblador deberá guardarse con extensión .asm.

Los editores de texto más comunes son: Bloc de notas o Notepad++, en Windows; Gedit, Nano, o Vi, en Linux; o TextEdit en MacOS. Sin embargo, es posible que la escritura de código en alguno de éstos se vuelva complicada debido a que es difícil editar un archivo de texto plano sin formato. De esta manera es fácil cometer un error.

Existe un editor de texto útil para desarrolladores que permite reconocer diferentes extensiones de archivos para ponerles formato de acuerdo a la sintaxis del lenguaje de programación utilizado. Este editor de texto se llama *Sublime Text* y se puede encontrar tanto para Windows, como para Linux y MacOS. <https://www.sublimetext.com/>

De manera predeterminada, Sublime Text tiene herramientas que reconocen más de 30 lenguajes de programación, como C, C++, C#, Java, y Python, incluso HTML. Pero una característica importante de Sublime Text es que permite agregar paquetes (*packages*) que permiten agregar aún más lenguajes o funciones al editor de texto, haciéndolo más robusto.

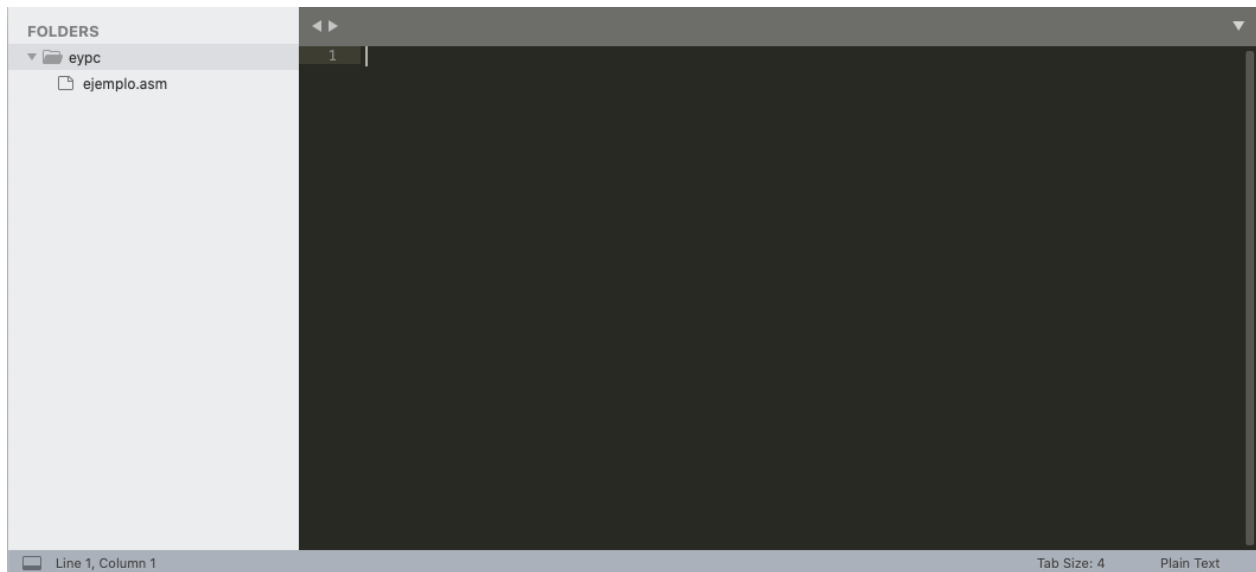
Por default, Sublime Text no contiene herramientas para reconocer la sintaxis del lenguaje ensamblador para arquitectura x86, aunque existe un *package* que lo permite. Se recomienda utilizar este editor de texto durante el curso.



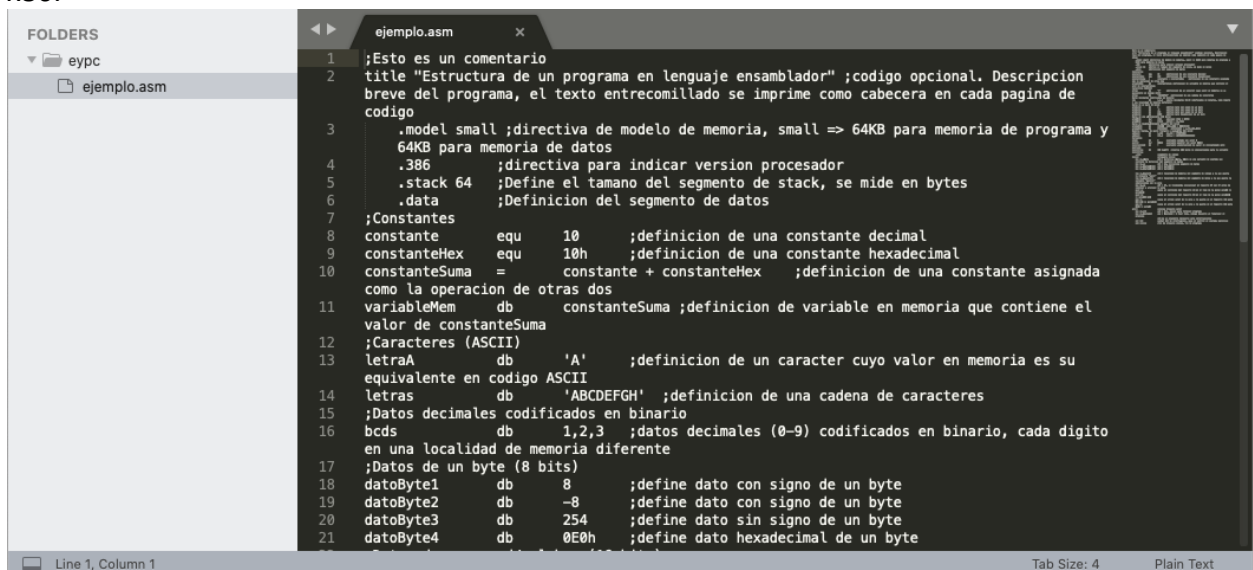
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "f.h"
4
5
6 int main()
7 { char cool;
8   int temp,fan;
9
10  while(1)
11  {
12    printf("please enter the temperature now:");
13    scanf("%d",&temp);
14
15    cool= func(temp);
16
17    if(cool==1)
18    {
19      fan=1;
20    }
21    else
22    {
23      fan=0;
24    }
25    printf("fan=%d\ntemp=%d\n",fan,temp);
26
27  }
28
29  return 0;
30 }
```

Para instalar Sublime Text en Ubuntu 18.04, seguir los pasos del siguiente tutorial:
<https://linuxconfig.org/how-to-install-sublime-text-on-ubuntu-18-04-bionic-beaver-linux>
Para otros sistemas operativos que no estén basados en Debian, consultar en internet.

Una vez instalado, abrir el editor de texto. Dentro del editor de texto, en la barra de tareas, ir a File -> Open... (Archivo -> Abrir...) y seleccionar la carpeta 'eypc' dentro de 'C:\Tasm' y esto mostrará los archivos en esa carpeta del lado izquierdo.



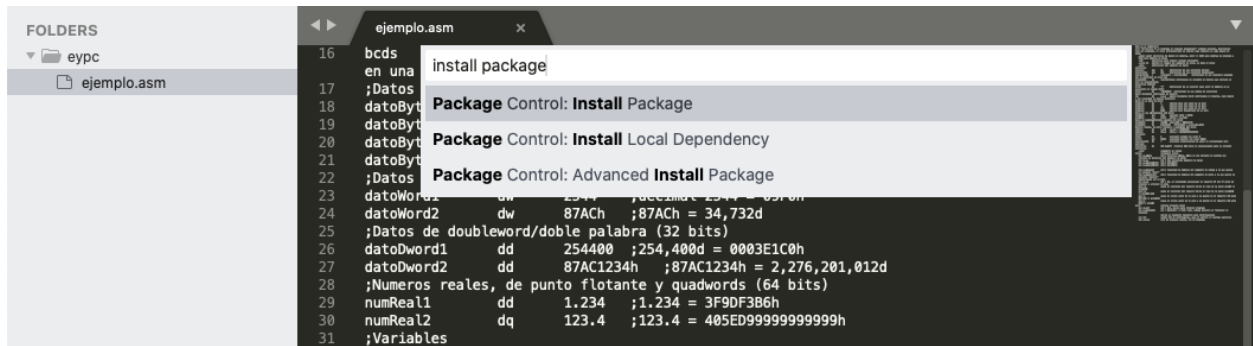
Si se abre el archivo ejemplo.asm, éste se mostrará como de texto plano, dado que aún no se instala el paquete necesario para reconocer la sintaxis del lenguaje ensamblador arquitectura x86.



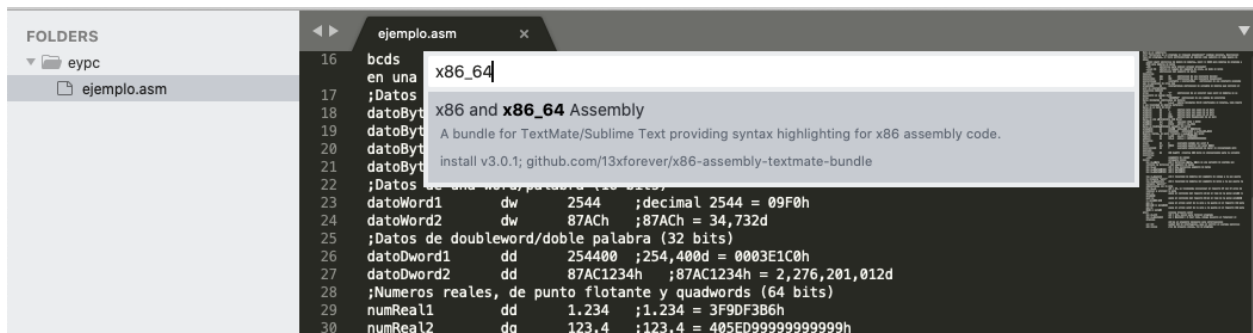
Para instalar el paquete que reconoce la sintaxis del x86, en primer lugar se debe actualizar algo llamado *Package Control*. *Package Control* es el gestor que permite instalar paquetes dentro de

Sublime Text. Para actualizarlo, ir a la opción Tools (Herramientas) en la barra de tareas y seleccionar *Install Package Control* (Instalar Package Control) del menú.

Una vez actualizado, para instalar un paquete se ir a *Tools* (Herramientas) en la barra de tareas y seleccionar *Command Palette...* (Paleta de Comandos...). Se mostrará un campo de texto y una lista de comandos, teclear “install package” (o “instalar paquete”, si está en español) y seleccionar “*Package Control: Install Package*” (“Package Control: Instalar Paquete”).

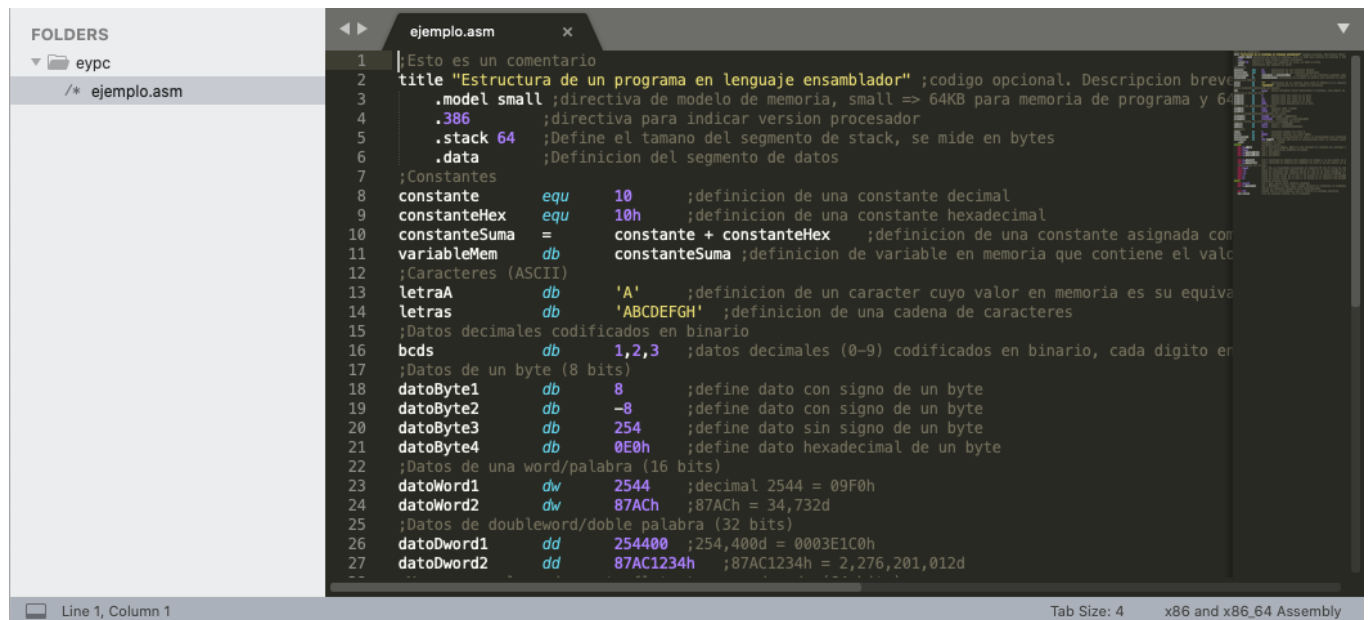


Se abre una nueva lista, teclear “x86_64” y buscar la opción “x86 and x86_64 Assembly”. Seleccionar tal paquete y en ese momento comenzará a instalarlo.



En la parte baja del editor se muestra la operación. Cuando termine la instalación del paquete, cerrar la pestaña del archivo y volver a abrirlo, entonces ahora se deberá mostrar el texto con formato y estilo que reconoce por defecto la sintaxis de la arquitectura x86.

Es posible observar en la esquina inferior derecha la sintaxis del lenguaje que se reconoció. Si se detecta que el archivo es de lenguaje ensamblador, entonces por defecto el estilo seleccionado será el del “x86 and x86_64 Assembly”



```
1 ;Esto es un comentario
2 title "Estructura de un programa en lenguaje ensamblador" ;codigo opcional. Descripcion breve
3 .model small ;directiva de modelo de memoria, small => 64KB para memoria de programa y 64KB para memoria de datos
4 .386 ;directiva para indicar version procesador
5 .stack 64 ;Define el tamaño del segmento de stack, se mide en bytes
6 .data ;Definición del segmento de datos
7 ;Constantes
8 constante equ 10 ;definición de una constante decimal
9 constanteHex equ 10h ;definición de una constante hexadecimal
10 constanteSuma = constante + constanteHex ;definición de una constante asignada con una operación
11 variableMem db constanteSuma ;definición de variable en memoria que contiene el valor de la constanteSuma
12 ;Caracteres (ASCII)
13 letraA db 'A' ;definición de un carácter cuyo valor en memoria es su equivalente en ASCII
14 letras db 'ABCDEFGH' ;definición de una cadena de caracteres
15 ;Datos decimales codificados en binario
16 bcds db 1,2,3 ;datos decimales (0-9) codificados en binario, cada dígito en un byte
17 ;Datos de un byte (8 bits)
18 datoByte1 db 8 ;define dato con signo de un byte
19 datoByte2 db -8 ;define dato con signo de un byte
20 datoByte3 db 254 ;define dato sin signo de un byte
21 datoByte4 db 0E0h ;define dato hexadecimal de un byte
22 ;Datos de una word/palabra (16 bits)
23 datoWord1 dw 2544 ;decimal 2544 = 09F0h
24 datoWord2 dw 87ACh ;87ACh = 34,732d
25 ;Datos de doubleword/doble palabra (32 bits)
26 datoDword1 dd 254400 ;254,400d = 0003E1C0h
27 datoDword2 dd 87AC1234h ;87AC1234h = 2,276,201,012d
```

Con Sublime Text se puede editar el texto y guardarlo sin ningún problema. Intente editar el texto y ensamblarlo de nueva cuenta en DOSBox para detectar los cambios en el código y ver cómo se ejecutan.