EMULADOR EMU8086

El Emulador EMU8086 es el primer programa que se utiliza en el curso de Microprocesadores que imparte la Universidad Don Bosco; se ha elegido este emulador porque posee una interfaz de usuario muy amistosa que permite familiarizarse con los fundamentos de la programación en lenguaje ensamblador de forma muy intuitiva, a parte de eso brinda una serie de recursos para ejecutar y depurar los programas. También tiene algunas desventajas como el de no soportar algunas de las interrupciones más interesantes que posee el sistema operativo y tampoco puede acceder a los puertos físicos (reales), sino que los emula usando otros programas que ya están incluidos en su respectiva carpeta.

Para iniciar se debe ejecutar el archivos EMU886.exe, que se encuentra en la carpeta del mismo nombre, en el directorio raíz; seguramente en la computadora donde corra el programa tendrá otras alternativas para activar el programa, como un acceso directo en el escritorio o en el menú de programas del sistema operativo.

Si está ejecutando la versión 4.05 del EMU8086 observará primero la pantalla de bienvenida (welcome), similar a la que se muestra en la Figura 1.

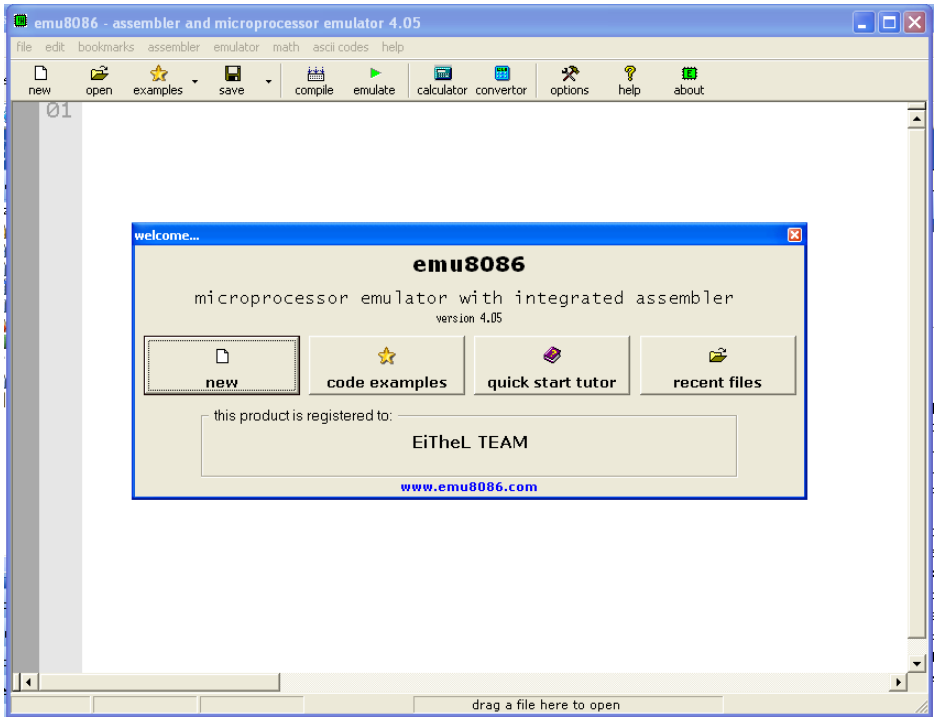

Figura 1. Pantalla de bienvenida del emulador EMU8086.

Se presentan cuatro diferentes opciones para elegir:

- **New**: Le permite escribir un nuevo código en lenguaje ensamblador (al que llamaremos "Código Fuente" y tendrá extensión .ASM)

- **Code examples**: Le permite acceder a una serie de programas ejemplos que pueden ayudarle a comprender funciones más complejas.

- **Quick star tutor**: activa un conjunto de documentos de ayuda, se recomienda revisarlos frecuentemente en caso de dudas.

- **Recent file**: Le muestra los últimos archivos que se trabajaron en la máquina.

Para continuar este primer contacto con el emulador, seleccione **New**. Observará una nueva caja de dialogo "choose code template", como se muestra en la Figura 2.

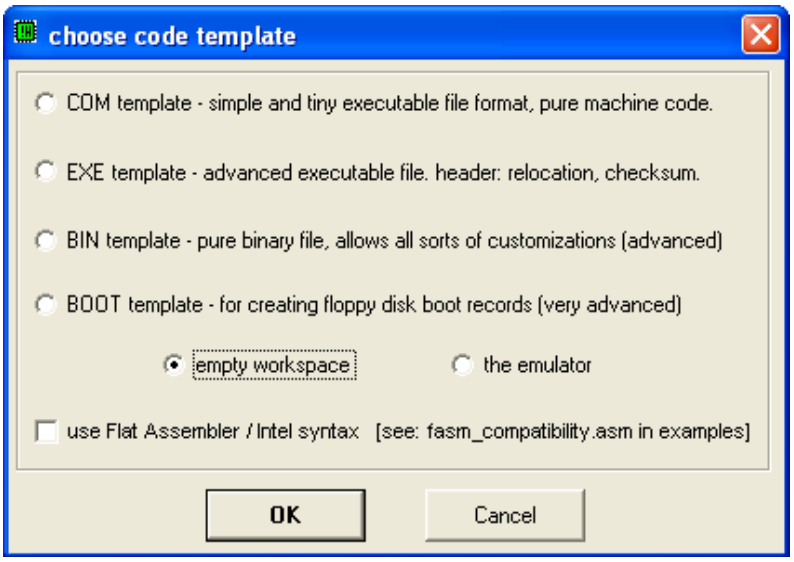

Figura 2. Caja de dialogo para seleccionar el tipo plantilla (template).

En ella se le presentan seis opciones, cuatro que le permiten usar plantillas predefinidas con algunos de los tipo de archivo que se pueden crear en lenguaje ensamblador: COM template, EXE template, BIN template y BOOT template (cada uno le permite diferentes características que se abordaran a su debido tiempo en el curso). Dos que le permiten usar un espacio vacío "empty workspace" (sin una plantilla) o activar el emulador mismo. Selecciones la opción del espacio vacío. Observará la ventana de edición o mejor dicho el Entorno de Desarrollo Integrado (Integrated Development Environme IDE), como se muestra en la Figura 3, donde escribirá sus archivos fuentes en lenguaje ensamblador, por favor lo confunda con el lenguaje de máquina.

Podrá ver una barra de menú de Windows con sus opciones file, edit, etc. pero también vera unas opciones poco usuales como assembler, emulator, etc. propias del emulador. También vera una serie de botones que le permitirán crear un nuevo archivo (new), abrir un archivo que ya existe (open), abrir un ejemplo (examples), compilar un archivo fuente (compile), emular un archivo ejecutable (emulate) y otras opciones que ira descubriendo a medida que se familiarice con el programa.

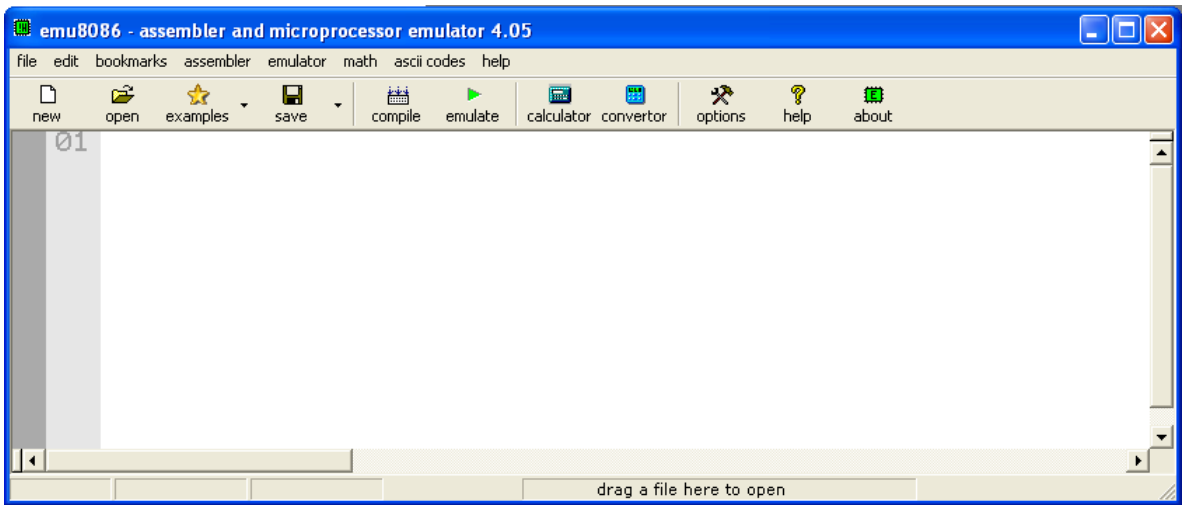

Figura 3. Ventana de edición o Entorno de Desarrollo Integrado IDE del EMU8086.

Bueno, es el momento de estudiar nuestro primer programa en lenguaje ensamblador, el cual imprime en pantalla algunas cadenas de texto. Para comprender mejor como funciona revise primero la Figura 4, donde se presenta un diagrama de flujo.

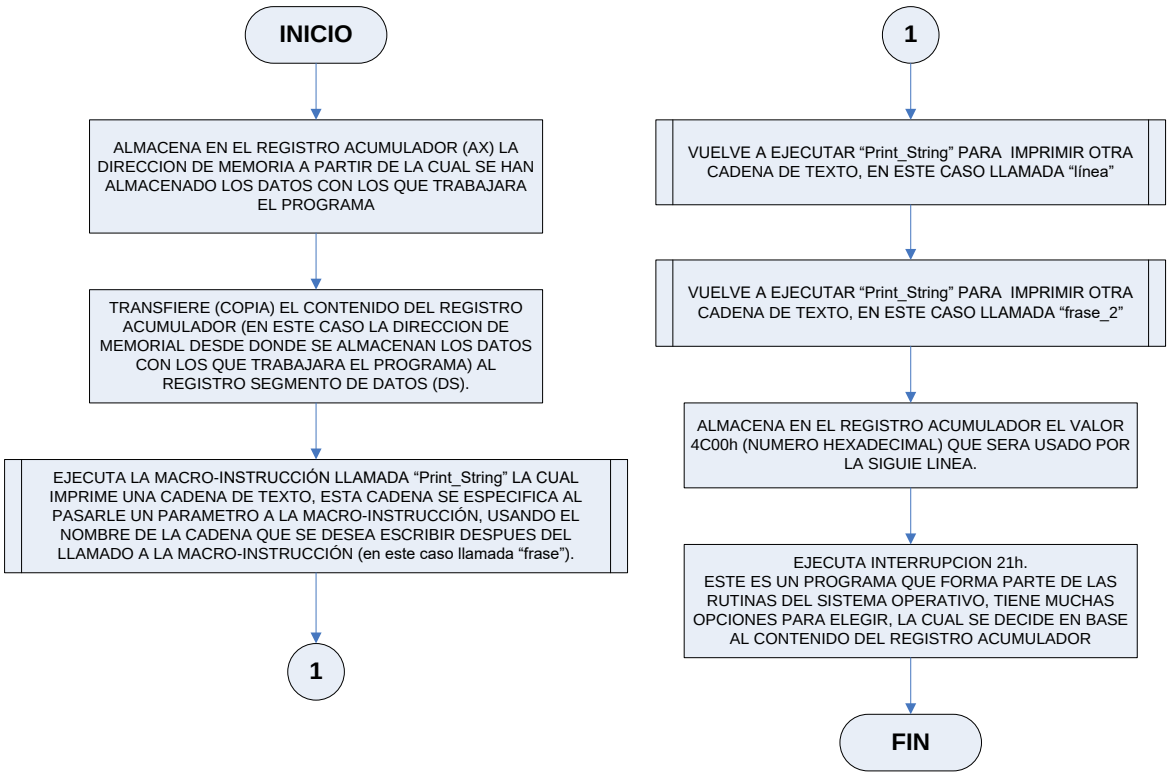

Figura 4. Diagrama de Flujo del ejemplo.

Ahora observe la Tabla 1 donde aparece el código fuente escrito en lenguaje ensamblador. Por favor note que el programa esta compuesto de diferentes bloque:

PRIMER PROGRAMA EN LENGUAJE ENSAMBLADOR

```
include IO_MAC_1.h        ;Indica al compilador que incluya la librería respectiva.
                          ;El archivo IO_MAC_1.h debe estar en la misma carpeta
                          ;que el código fuente

.model small              ;Indica al compilador el modelo de memoria que se usara
                          ;en este caso se usa un modelo pequeño
;=====================================================
; ZONA PARA DECLARAR LAS VARIABLE Y CONSTANTES DEL PROGRAMA
;SE LE CONOCE COMO SEGMENTO DE DATOS
;
; Inicia con la directiva simplificada .data
;=====================================================
.data
frase       db 'Hola!!$'          ;Declara cadena que se imprimira
frase_2     db 'Saludos$'         ;Declara cadena que se imprimira
linea    db 0Ah, 0Dh, '$'         ;Baja una linea, equivale a un ENTER
;=========================================================
; ZONA PARA ESCRIBIR LAS INSTRUCCIONES, MACRO-INSTRUCCIONES Y PROCDIMIENTOS
; QUE SE EJECUTARAN.
; SE LE CONOCE COMO SEGMENTO DE CODIGO
;
;Inicia con la directiva simplificada .code
;=========================================================
.code
INICIO:
        mov ax, @DATA     ;Bloque de instrucciones que ubican la zona de memoria donde están
                          ; almacenados los datos con los que el programa va a trabajar.
                          ; Se utiliza la instrucción MOV (mover) para trasladar al registro AX el segmento
                          ; memoria donde están almacenados los datos

        mov ds, ax        ; Transfiere (en realidad copia) el contenido del registro AX al registro DS

        Print_String frase     ;Macro-Instrucción que imprime la cadena llamada frase

        Print_String linea     ; Macro-Instrucción que imprime una cadena llamada linea
                               ; El efecto final es que el cursor baja una línea, similar a digitar un ENTER

        Print_String frase_2       ;Macro-Instrucción que imprime la cadena llamada frase_2

FIN:
        mov ax, 4C00h   ;Bloque de instrucciones que terminan el programa

        int 21h
;=========================================================
; ZONA DE MEMORIA DONDE SE ALMACENAN DATOS INTERMEDIOS QUE SE CREAN DURANTE
; LA EJECUCION DEL PROGRAMA.
; SE LE CONOCE COMO SEGMENTO DE PILA
;
; Inicia con la directiva simplificada .stack
;=========================================================
.stack

end INICIO
```

Tabla 1. Programa en lenguaje ensamblador.

a) Directivas de preproceso: Le indican al compilador que debe realizar una serie de acciones particulares en el momento de convertir el archivo fuente (ASM) en archivo objeto (OBJ).
b) Segmento de Datos: Donde se declaran las variables y constantes que el programa va a utilizar.
c) Segmento de Código: Donde especifica lo que deseamos que el programa haga. Para especificarlo se pueden usar: instrucciones (propias del microprocsador),

Macro-instrucciones (similar a los comandos de los lenguajes de alto nivel) y procedimientos (similar a las funciones definidas por el usuario de los lenguajes de alto nivel).

d) Segmento de PILA o STACK: bloque de memoria donde almacenan datos intermedios que se generan durante la ejecución de un programa. En este no se declaran variables o constantes como en el segmento de datos, sino que se administra como una memoria LIFO, el último en entrar es el primero en salir.
e) Directiva que indica el fin del archivo, cualquier instrucción posterior a esta línea será ignorada.

Digite el código en el IDE, note como se le asignan diferentes colores a las líneas, dependiendo si son instrucciones, macro-instrucciones, comentarios, cadenas de texto (formato ASCII), directivas, etc.

Al terminar de digitar el código salve el archivo usando como nombre su número de carnet (8 caracteres) y con extensión ASM (ensamblador).

Presione el botón "compile" para convertir el archivo fuente (ASM) en ejecutable (lenguaje de máquina), debe aclararse que al compilar solo se crea un archivo objeto que no es ejecutable, se necesita de otro proceso, el enlazado (LINK), para lograr que el archivo objeto se convierta en ejecutable (EXE), pero por facilidad el emulador realiza los dos procesos de forma secuencial al presionar el botón "compile".

Mientras se realiza la compilación y enlazado se abre un ventana llamada "assembler status" que le informa sobre los resultados del proceso. Si el resultado es exitoso observará un mensaje como el de la Figura 5.

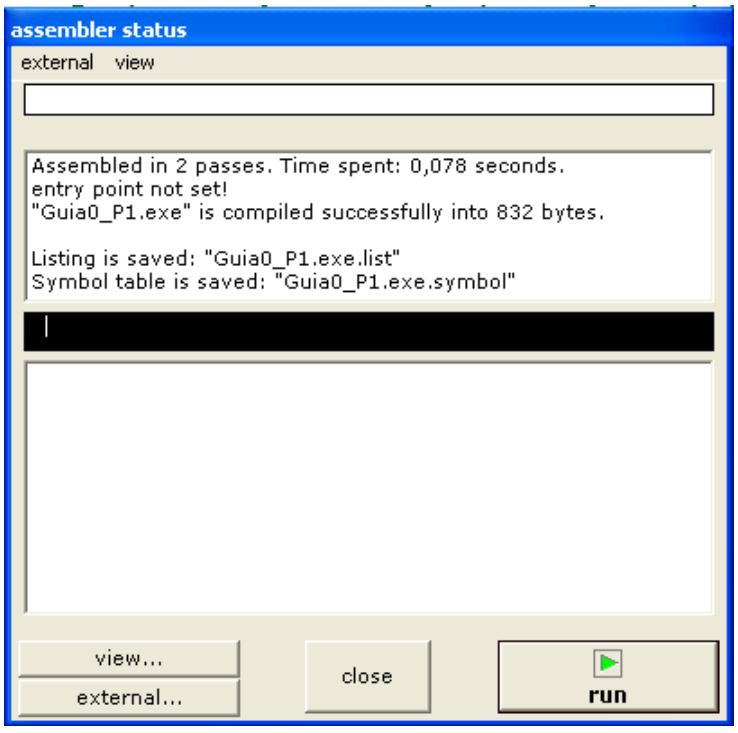

Figura 5. Estado del proceso de compilación.

Luego se le pedirá que salve el archivo EXE, por defecto se le asigna el mismo nombre del archivo fuente, pero usted puede elegir otro, siempre de 8 caracteres máximo, use el nombre por defecto, Si existe algún problema (error de sintaxis u otros) en la ventana "assembler status" se le indican las líneas donde están los errores, para hacer las correcciones cierre la ventana "assembler status" y corrija el archivo fuente, que se encuentra en el IDE e intente compilar nuevamente.

Volviendo al caso de no tener errores, la ventana "assembler status" le muestra diferentes opciones como revisar algunos de los archivos complementarios que se crearon en el proceso de compilación y enlazado (opción view), puede ejecutar el archivo usando el programa DEBUG (opción external). Esta es otra herramienta que se estudiará más adelante en el curso, puede ejecutar directamente el archive EXE (RUN) o puede cerrar la ventana (close). Seleccione esta opción.

Emule el archivo EXE presionando el botón "emulate". Notará que se abren dos ventanas: Una llamada "emulator" en la que podrá monitorear la mayoría de procesos al ejecutar el programa y otra llamada "original sourse code" que muestra el código fuente del archivo, esta ventana es diferente de la que observa en el IDE porque en ésta podrá observar cual instrucción esta a punto de ejecutarse, es ideal al corre programas pasos a paso.

Ahora observe con más detenimiento la ventana llamada "emulator" Figura 6, ésta será la que más utilice a la hora de ejecutar las prácticas de laboratorio, por lo que es importante que la conozca y maneje de forma efectiva.

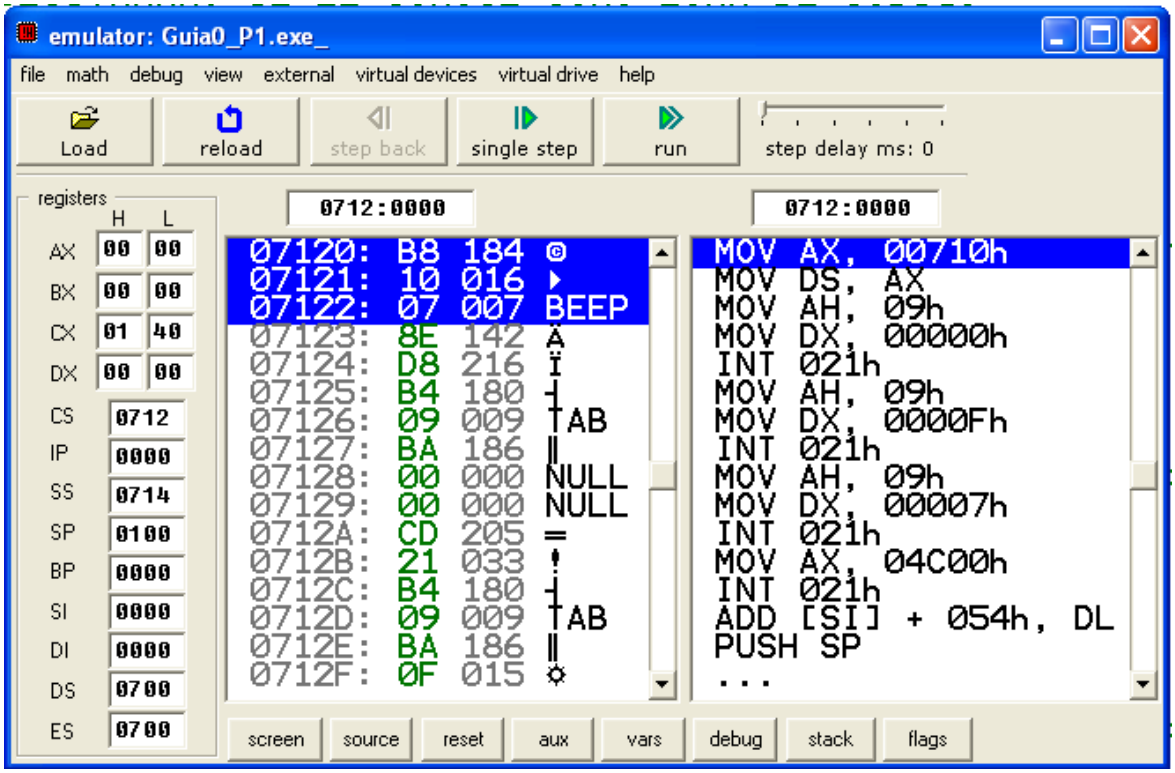

Figura 6. Ventana del Emulador.

En la parte superior tiene una barra de herramientas con las siguientes opciones:

- File, permite administrar (cargar o salvar) los archivos que va creando o ejecutando
- Math, da acceso a una calculadora y un convertidor de basas de numeración.
- Debug, provee herramientas para depurar programas.
- View, permite abrir otras ventanas que pueden ser de mucha ayuda al ejecutar o depurar programas.
- External, permite ejecutar el programa con otras herramientas diferentes del EMU8086.
- Virtual devices, activa los dispositivos virtuales con que cuenta el programa, dado que se tata de un emulador no se tiene acceso a los puertos físicos de la computadora, por lo que estos son simulados.
- Virtual drive, da opciones para administrar las unidades virtuales de almacenamiento (HDD y FDD virtuales).
- Help, activa la herramienta de ayuda.

Bajo la barra de herramientas hay cinco botones que le permiten:

- Load, carga un archivo ejecutable EXE, COM, etc. que ya existe.
- Reload, recarga (inicializa los registros) para ejecutar nuevamente un programa que acaba de ser corrido.
- Single step, ejecuta solamente una instrucción o macroinstrucción (paso a paso).
- Step back, retrocede una instrucción que ya fue ejecutada (función muy útil al depurar un programa)
- Run, ejecuta un programa en su totalidad o hasta presionar el botón "STOP". Vale la pena hacer notar que también es posible, en la opción DEBUG, insertar un "break point" cuando se está depurando programas.

La parte media esta dividida en tres partes:

- Parte izquierda denominada "registers", donde se puede monitorear el contenido de los registros del microprocesador
- Parte central, donde se puede observar el contenido de la memoria desde donde se está ejecutando el programa. Primero se notan las direcciones del bloque de memoria que se visualiza, estas direcciones de dan en un formato llamado físico o absoluto (existe otro formato para las direcciones) dado por cinco cifras hexadecimales (20 bits) lo que indica que en este bus de direcciones se puede direccional desde la dirección 00000h (dirección 0) hasta la dirección FFFFFh (dirección 148575). Luego se indica el contenido de cada palabra (cada una de 1 byte), por facilidad el contenido se presenta en hexadecimal, decimal e interpretado como un carácter ASCII.
- Parte derecha, donde puede observar el contenido de la memoria, pero esta vez no se detalla con direcciones específicas, sino que cada bloque de datos es interpretado como un conjunto de instrucciones (lo que llamaremos programa DESENSAMBLADO) que el microprocesador deberá ejecutar. Es importante mencionar que algunas instrucciones se expresan solo con un byte, pero otras necesitan varios bytes para ser expresadas.

Parte inferior, contiene una serie de botones que permiten un acceso rápido a una serie de ventanas auxiliares, algunas de las cuales se puede activar también en la barra de herramientas con la opción "view"

Regresando a la estructura del programa estudiemos el Segmento de Datos: Puede observar que se hab declarado tres cadenas de datos llamadas frase, frase_2 y línea; note que luego del nombre de la cadena de ha incluido la directiva db (definir byte) que indica al compilador que ese es el tipo de datos que contendrá la cadena. A parte de eso en las primeras dos cadenas se han usado comillas simples para definir el contenido, eso indica al compilador que el texto debe ser interpretado como caracteres ASCII. En el caso de la última cadena los datos no están entre comillas, lo que indica que serán interpretados como datos hexadecimales.

Para ejecutar el programa presione el botón RUN: note que automáticamente se activo la ventana "emulator screen" en la que se pueden observar dos textos impresos, que corresponden la frase y frase_2. Vamos a estudiar más detenidamente la ejecución del programa.

Cierre la ventana "emulator screen"

Prepare el programa para ser ejecutado nuevamente presionando el botón "reload". Observe la ventana "original sourse code", parece que es una copia fiel del código fuente que digito en el IDE (en la Figura 7 puede ver una comparación entre las dos ventanas), pero la primera línea de código está marcada de color amarillo, esto indica que es la primera que se ejecutará al correr el programa.

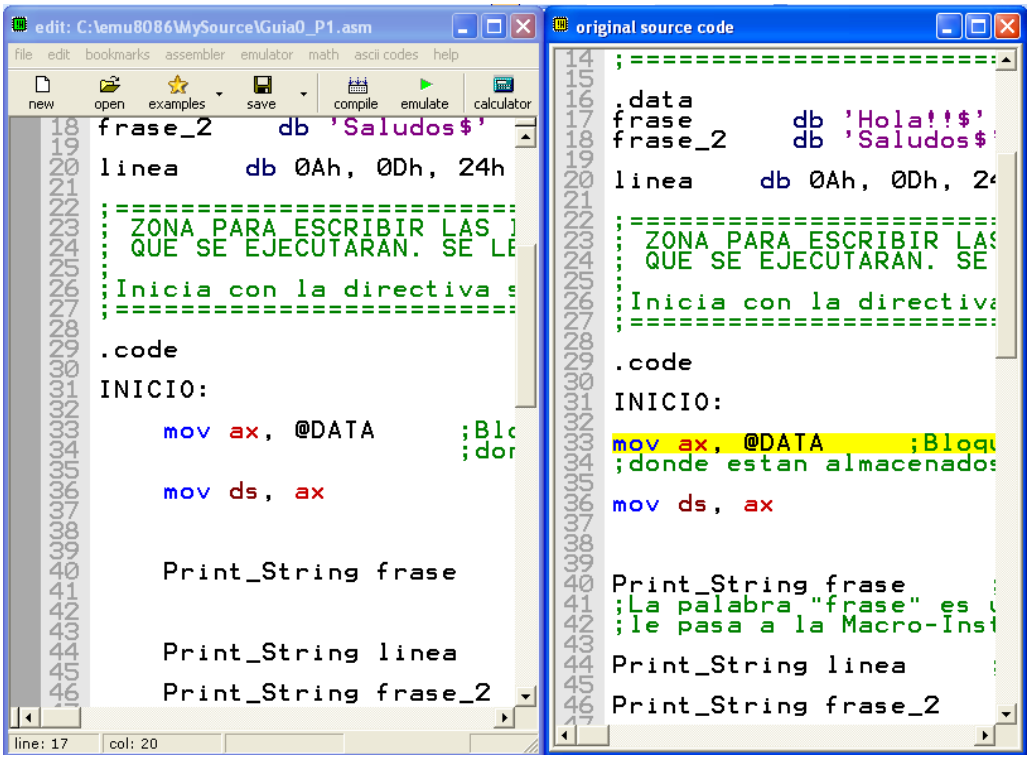

Figura 7. Comparación entre las ventanas de edición y código fuente.

Si observe la ventana "emulator" (Figura 6), se ha marcado un bloque de bytes almacenados en memoria, precisamente los de las direcciones físicas 07120h, 07121h y 07122h que contiene los datos B8h (que equivalea 184d y al carácter ASCII '©'), los otros datos son 10h y 07h.

Los tres juntos equivalen, en lenguaje de máquina, a la MOV AX, @DATA, que ya desensamblada se transforma en mov AX, 0710h, Aparece marcada porque será la primera en ejecutarse.

Un detalle importante es la sustitución de la expresión @DATA por un valor numérico, esta sustitución es producto de la compilación del código fuente que sustituye las etiquetas y nombres de variables por direcciones físicas de memoria.

Presione el botón "single step" note que se ha avanzado una sola línea, dado que se ejecuta una única instrucción, También se han dado cambios en los registros del microprocesador, pero éstos se analizarán en la primera guía de laboratorio. Ejecute la siguiente instrucción, de igual manera se avanzó solo un paso.

Ahora la línea marcada no es de color azul, sino negra, en este caso indica que se trata de una Macroinstrucción, que contiene un conjunto completo de instrucciones que se ejecutan en bloque. La lógica de esta estructura tiene varias formas de utilización, en este caso se asemejan a formas que ya aplicó programando en alto nivel, invoca a una función predefinida en una librería o biblioteca (printf o putchar) y para hacerlo, al inicio tiene que colocar una sentencia #include.

Como su nombre lo indica la macroinstrucción imprime una cadena de texto en la pantalla. Esta cadena se especifica pasándole un parámetro a la función, la palabra "frase".

Presione nuevamente "single step" observe que en la ventana "emulator" si se ha avanzado en "original sourse code" y no se avanzará hasta ejecutar todas las instrucciones que conforman la macroinstrucción.

Presione tres veces "single step" se activará de forma automática la ventana "emulator screen" e imprimiendo el texto de la cadena frase, note que el signo de dólar "$" final no se ha impreso.

Presione nuevamente "single step" y se marca la siguiente macroinstrucción, que imprimirá la cadena "línea". Esta cadena no se declarado de la misma forma que las otras, pero aun así se imprime, un detalle importante es que al final de esta cadena se encuentra el dato 24h que corresponde al código ASCII del signo de dólar.

Presione cinco veces "single step" con esto ha bajado el cursor a la siguiente línea, como si hubiera presionado ENTER en un editor de texto.

Presione cinco veces "single step" para imprimir la cadena "frase_2"

Ahora aparece marcada una línea de instrucción (recuerde el color azul). Analicemos su estructura,

La estructura sintáctica de las líneas de instrucción está formada por elementos llamados campos, dependiendo la instrucción específica se puede tener uno o varios campos.

Primer campo se le llama código de operación e indica que es lo deseamos que realice el microprocesador, básicamente nos referimos a la instrucción en misma. Este no puede faltar en una línea de instrucción

Segundo y tercer campo se les llama 1º operando y 2º operando campo. Si solo existe el 1º operando indica que la instrucción se ejecutará ese dato (el dato en cuestión puede declararse de forma explícita o estar almacenado en un registro del micro o en alguna localidad de memoria). En caso de existir los dos operandos implica que el 2º operando es la fuente donde está el dato que se procesará y el 1º operando es el destino donde se almacenará el resultado de ejecutar la instrucción

Tomando como ejemplo la instrucción que en este momento está marcada:

MOV AX, 4C00h

Se entiende que MOV es el código de operación e indica que un dato se moverá (en realidad se copiará porque la fuente conservará el dato). Por tener tres campos en esta línea de instrucción el 1º operando es el destino donde se copiará el dato (en este caso el registro Acumulador AX del microprocesador) y el 2º operando es el dato que será movido (en este caso el número hexadecimal 4C00h)
Ejecute la instrucción, note que efectivamente el dato se copió en el registro, lo que puede verificar en la ventana "emulator"

La última línea del código usa la instrucción INT (de interrupción) que es otra forma particular de invocar una función definida previamente, en este caso para terminar el programa y regresar el control del microprocesador al sistema operativo, de forma similar a la sentencia return(0) que se usa en lenguaje C.

Para visualizar de forma más clara el proceso para imprimir la cadena "línea" edite en el IDE el código fuente, eliminando de el dato 0Dh. Luego de esto realice las acciones listadas anteriormente para crear un nuevo archivo ejecutable.

Ejecute el programa paso a paso y note coma cambian los mensajes en la pantalla.

Cambie la cadena, pero ahora solo elimine el 0Ah verifique como afectan los cambios en la ejecución.

¿Cuál es la función individual de los datos 0Ah y 0Dh en el programa?
¿Cuál es la relación de estos y el código ASCII?

Para terminar este primer acercamiento con el EMU8086 es importante mostrarle una de las ventanas más útiles del emulador la ventana de "variables", es muy similar a la herramienta match del lenguaje C en la que usted podía monitorear el estado de las variables declaradas a medida que el programa se va desarrollando.

Recargue el programa y cierre la venta "emulator screen". Usando los botones de la parte inferior del "emulator" active la ventana denominada "var". Observará que están listadas las tres variables declaradas al inicio del archivo fuente como se muestra en la Figura 8a.

Note que aparece marcada la variable frase e indica que su contenido es 48 hexadecimal (indicado por la casilla llamada "show as"), pero sabemos que esta variable contiene más datos, para poder verlos todos juntos, cambien el datos en la casilla "elements" de 1 a 7, observará una serie de datos hexadecimales. Puede cambiar la representación a datos ASCII cambiando la opción en la casilla "show as" y verificar que efectivamente es el contenido de la cadena "frase", Figura 8b. Si durante

la ejecución del programa el contenido de la cadena cambia en esta ventana podrá observarlo.

Marque la segunda variable llamada "frase_2" observe la información, se le indica que el contiene el dato 53h, cambie la representación de datos a ASCII y la cantidad de elementos de 1 a 8, debería observas la misma información que ve en la Figura 8c.

Ahora marque la tercera variable "línea", el contenido es 0Ah, cambie a representación ASCII y 3 elementos, debería observas la misma información que ve en la Figura 8d.

Bueno con estas ideas básicas podrá iniciar el estudio de la programación en lenguaje ensamblador apoyándose en los recursos que le brinda el EMU8086, adelante.

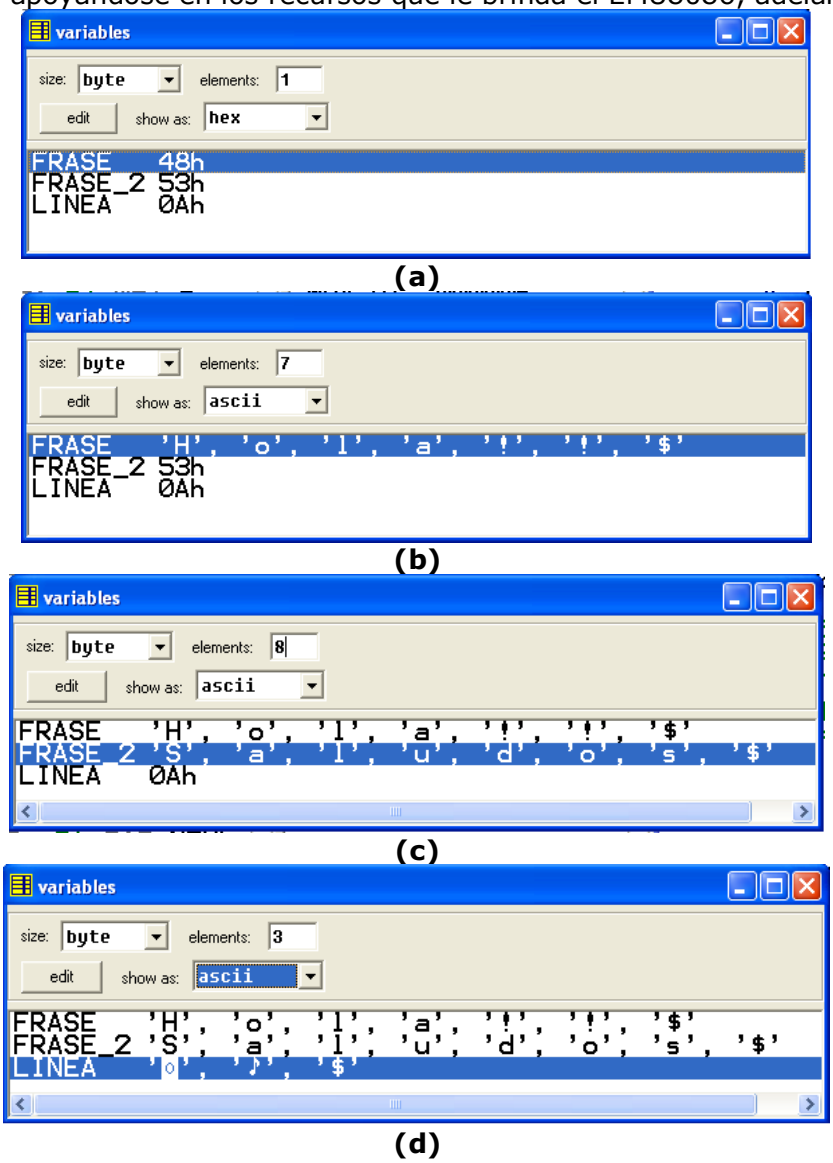

Figura 8. Ventana auxiliar para monitorear el contenido de las variables declaradas.