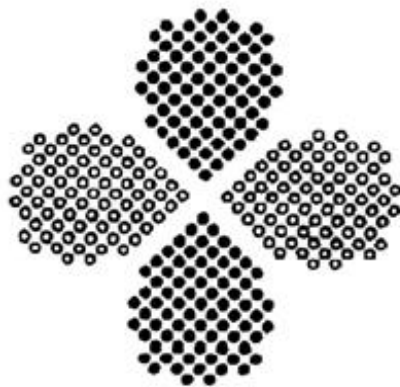


Digital II

Introducción al entorno emu8086



Departamento de Sistemas e Informática
Escuela de Electrónica
Facultad de Cs. Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Ing. Diego Alegrechi
Ing. Esteban Almirón
Octubre 2014

Tabla de contenidos

1	Introducción.....	3
2	Instalación del entorno.....	3
2.1	Emu8086	3
2.1.1	Windows XP.....	3
2.1.2	Windows Vista y 7.....	4
2.2	Dispositivos Virtuales de Digital II.....	4
3	Utilización del entorno.....	5
4	El Emulador.....	8
4.1	Mapa de memoria.....	10
4.1.1	Custom Memory Map.....	10
4.1.2	Interrupciones	11
4.1.3	Puertos de entrada/salida	12
4.2	Dispositivos Virtuales (DVIO).....	13
4.2.1	DVIO Digital II	13
4.2.2	Documentación de los Dispositivos Virtuales.....	13
4.2.3	Ejemplo – Problema Resuelto.....	14
4.2.4	Ejemplo – Dispositivos Tanque y Pulsadores	14
4.3	Documentación emu8086.....	15
4.3.1	Set de instrucciones 8086.....	16

Tabla de Figuras

Figura 1 – Instalación DVIO	4
Figura 2 – Archivo de configuración dvio.ini.....	5
Figura 3 – Ventana de inicio del emu8086.....	5
Figura 4 – Elección del tipo de template.....	6
Figura 5 – Ventana principal emu8086.....	7
Figura 6 – Editor de código fuente	7
Figura 7 – Código fuente durante la emulación.....	7
Figura 8 – Emulador.....	8
Figura 9 – Extended Viewer	9
Figura 10 – Flags y mapa de memoria.....	10
Figura 11 – Emulador, código fuente	10
Figura 12 – Custom Memory Map.....	10
Figura 13 – Interrupt Vector Table (IVT).....	11
Figura 14 – Emulador – Interrupción por Hardware	12
Figura 15 – Dispositivos Virtuales – Digital II	13
Figura 16 – Ayuda Dispositivos Virtuales	14
Figura 17 – Ejemplo de dispositivos virtuales y emulador.....	15
Figura 18 – Documentación y tutoriales emu8086	15

1 Introducción

Hasta el año 2009 en la cátedra de Digital II hemos utilizado la herramienta MASM 6.11 para la realización del segundo trabajo práctico de assembler. A partir del año 2010 hemos introducido una nueva herramienta llamada “emu8086” reemplazando al MASM.

El emu8086 es un emulador del microprocesador 8086 (Intel o AMD compatible) con assembler integrado. A diferencia del entorno de programación en assembler utilizado anteriormente en la cátedra (MASM), este entorno corre sobre Windows y cuenta con una interfaz gráfica muy amigable e intuitiva que facilita el aprendizaje el lenguaje de programación en assembler.

Dado que en un entorno emulado de microprocesador no es posible implementar una interfaz real de entrada/salida, el emu8086 permite interfacear con dispositivos virtuales y emular una comunicación con el espacio de E/S. Para esto, el emu8086 cuenta con una serie de dispositivos virtuales preexistentes en el software base, listos para ser utilizados, entre los que se encuentran una impresora, un cruce de calles con semáforos, un termómetro, un motor paso a paso, etc. No obstante, la cátedra ha desarrollado dispositivos adicionales con características particulares para la realización del segundo trabajo práctico.

Se muestra a continuación una tabla comparativa con las diferencias entre el entorno de programación en assembler utilizado anteriormente en la cátedra (MASM 6.11) y el nuevo emu8086:

Emu 8086	Microsoft Assembler (MASM 6.11)
Entorno educativo	Entorno para producción y educativo.
Basado en Windows	Basado en DOS
En forma nativa admite dispositivos virtuales.	No admite dispositivos virtuales en forma nativa
Set de instrucciones de 8086	Set de instrucciones del 8086, 80186/286/386/486
Directivas propias adicionales	Directivas comunes con TASAM (Borland Turbo Assembler)
Emula interrupciones por Hw y Sw	No permite emular interrupciones
Emula el espacio de E/S (instrucciones IN y OUT)	No permite emular el espacio de E/S
Permite emular interrupciones.	No permite emular intrrupciones.
Herramientas adicionales para el debug	

2 Instalación del entorno

En la cátedra de Digital 2 utilizaremos el emu8086 pero de manera combinada con dispositivos virtuales desarrollados por la cátedra. Por esta razón será necesario descargar de la página de la cátedra e instalar dos programas.

2.1 Emu8086

En primer lugar descargar el archivo instalador del emu8086 de la página web de la cátedra el siguiente archivo:

- http://www.dsi.fceia.unr.edu.ar/downloads/digital_II/EMU_Setup.zip

Luego ejecutar el instalador y seguir los pasos de instalación teniendo en cuenta los comentarios siguientes.

2.1.1 Windows XP

Para la instalación en Windows XP no es necesario tener cuenta consideraciones particulares, sólo basta con seguir los pasos indicados por el instalador y aceptar las configuraciones por defecto propuestas por el emu8086.

El directorio de instalación por defecto es C:\emu8086. Adicionalmente, los siguientes dos archivos se instalan en la raíz del disco C:

- C:\emu8086.hw
- C:\emu8086.io

2.1.2 Windows Vista y 7

En Windows Vista/7 no es posible, por cuestiones de seguridad, que el emu8086 se instale en la raíz del disco C. Para evitar esta situación, en el momento de la instalación del emu8086 se debe elegir otra ubicación para la instalación del programa, por ejemplo otra partición (D:) o el directorio Archivos de Programa¹.

Al haber elegido otro sitio de instalación distinto al propuesto por defecto, es necesario modificar en el archivo c:\emu8086\emu8086.ini los valores de EMUPORT=c:\emu8086.io y HW_INTERRUPT_FILE=c:\emu8086.hw para indicarle al emu8086 la nueva ubicación de estos archivos.

2.2 Dispositivos Virtuales de Digital II

Adicionalmente a los dispositivos nativos del emu8086, la cátedra de Digital II desarrolló 10 dispositivos adicionales que son los utilizados en el Trabajo Práctico N°2. Para poder utilizar estos dispositivos se debe descargar de la página de la cátedra e instalar el siguiente archivo:

- http://www.dsi.fceia.unr.edu.ar/downloads/digital_II/Digi2IO_Setup.zip

Instalar los DVIO (Dispositivos Virtuales de Entrada/Salida) en el mismo directorio en el cual se instaló el emu8086, ver por ejemplo *Figura 1*.

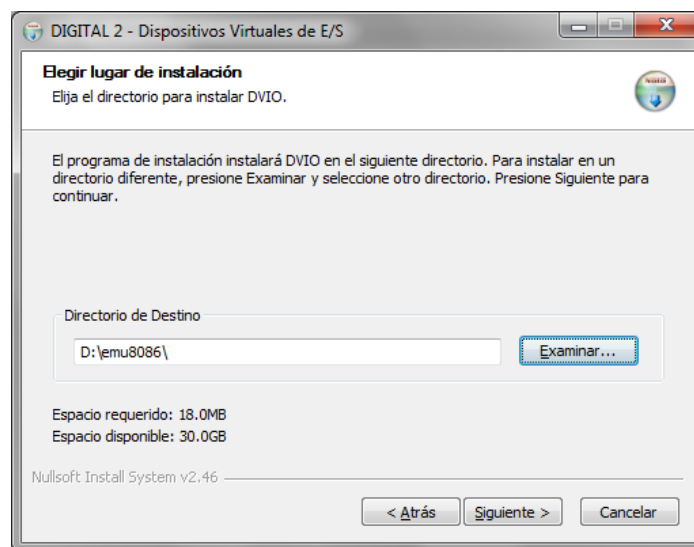


Figura 1 – Instalación DVIO

Luego de finalizado el proceso de instalación se debe editar el archivo D:\emu8086\DVIO\res\dvio.ini y modificar el path con la ubicación correcta de los archivos emu8086.io y emu8086.hw.

¹ Por defecto Windows Vista/7 no permiten modificar archivos ubicados directorio Archivo de Programas (o Program Files), por esta razón para poder editar el archivo emu8086.ini luego de la instalación se debe previamente modificar de manera temporal el nivel de seguridad y protección de usuario de Windows. Para esto ir a Control Panel → System and Security → Action Center, luego en Security debajo de User Account Control hacer click en Change Settings. Llevar la barra de desplazamiento al nivel de seguridad más bajo (Never notify) y aceptar. Editar ahora el archivo emu8086.in, luego de esto restaurar el nivel de seguridad al que tenía por defecto.

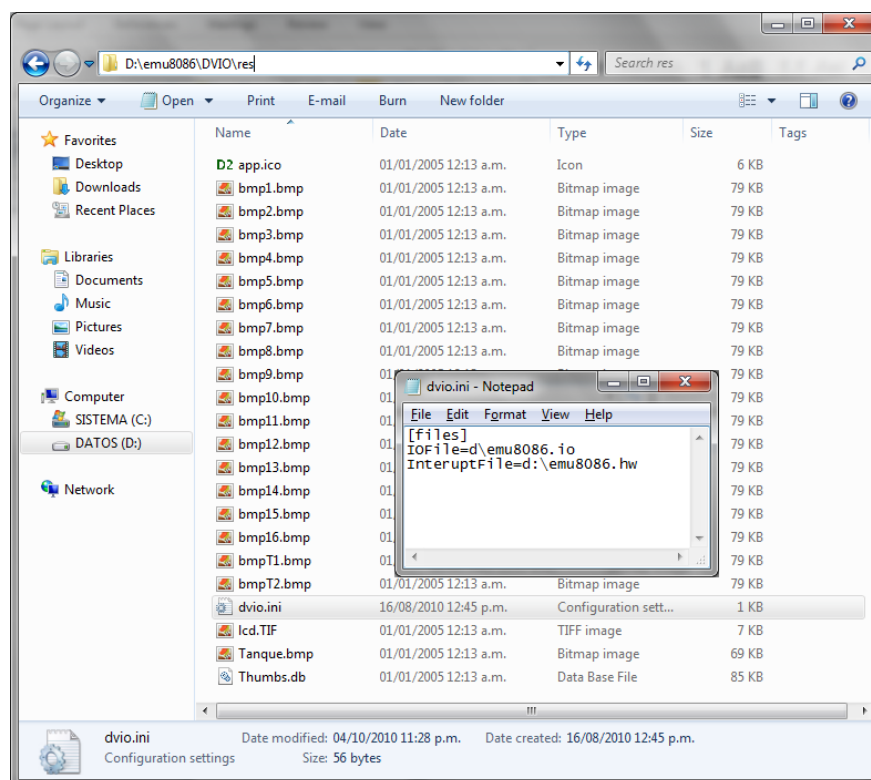


Figura 2 – Archivo de configuración dvio.ini

3 Utilización del entorno

Para iniciar el entorno se debe ejecutar el archivo emu8086.exe que se encuentra en el directorio de instalación (ej. c:\emu8086).

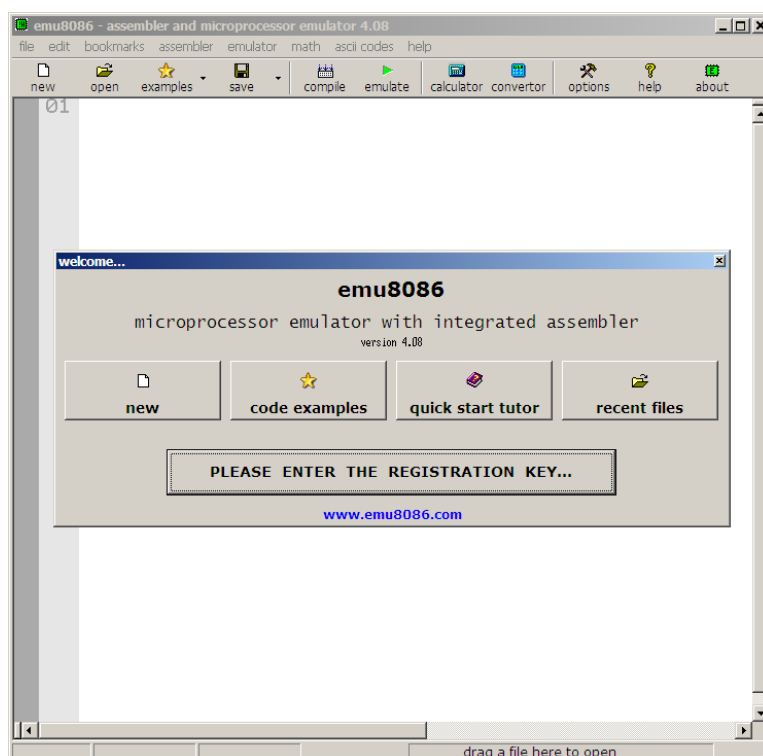


Figura 3 – Ventana de inicio del emu8086

Luego de iniciar el entorno el emu8086 ofrece diferentes opciones:

- **New:** permite escribir un nuevo código en lenguaje ensamblador (“Código Fuente” con extensión .ASM)
- **Code examples:** permite acceder a una serie de programas ejemplos muy útiles al momento de aprender a utilizar el entorno y la programación en assembler.
- **Quick start tutor:** llama al browser y permite explorar gran variedad de documentos de ayuda.
- **Recent file:** muestra los últimos archivos con los cuales se estuvo trabajando.

En el caso de hacer click en New, el entorno ofrece trabajar con diferentes plantillas o templates:

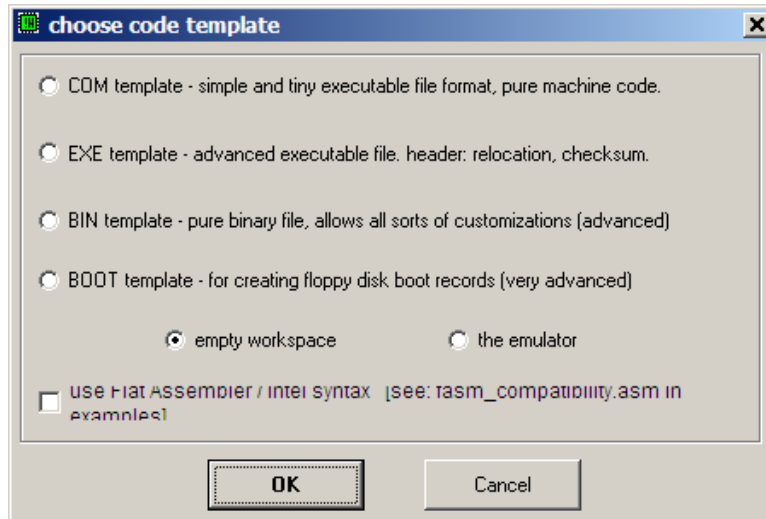


Figura 4 – Elección del tipo de template

- **COM template (directiva #make_com#):** es el formato más simple y antiguo de un archivo ejecutable, típicamente estos archivos se cargan con un offset de 100h (256 bytes). Por esta razón se debe agregar la directiva ORG 100h al comienzo del código para indicar la utilización de este tipo de archivos. Formato soportado por DOS y Windows Command Prompt.
- **EXE template (directiva #make_exe#):** este es el formato más avanzado de un archivo ejecutable. No tiene limitaciones en cuanto al tamaño del archivo y número de segmentos. Este template permite crear un programa exe simple con los segmentos de código, datos y pila predefinidos. Este tipo de archivo está soportado por Windows y Windows Command Prompt. El ensamblador elige automáticamente este tipo de archivo cuando encuentra definido un segmento de pila.
- **BIN template (directiva #make_bin#):** es un archivo ejecutable simple. Permite definir el valor de todos los registros, segmentos y el lugar de memoria donde se cargará a este programa. Cuando por ejemplo el ensamblador carga el archivo "MY.BIN" en el emulador buscará el archivo "MY.BINF" y cargará al archivo "MY.BIN" en la ubicación especificada en "MY.BINF", al igual que el valor inicial configurado para todos los registros. En el caso de que el emulador no encuentre al archivo "MY.BINF", se utilizará el valor actual de los registros al momento de la ejecución del .BIN y este código se ubicará en los valores que tengan en ese momento CS:IP.
- **BOOT template (directiva #make_boot#):** funciona igual de que un .BIN, pero utiliza valores predefinidos para ubicar el código y que coinciden con el primer track de un floppy disk (boot sector). La única diferencia con la directiva #make_bin# es que carga el código en la dirección predefinida 0000:7c00h. Este template permite emular el boot de una IBM PC desde el floppy disk.

En Digital II utilizaremos para la resolución del trabajo práctico un template modificado, pero tomaremos como base el ofrecido en la opción BIN Template. A los fines de avanzar con los primeros pasos con el emu8086, en este caso seleccionaremos la opción “empty workspace”.

Luego de esto tendremos acceso a la ventana principal del emulador que cuenta con una barra de menú de Windows (file, edit, bookmarks, assembler, etc.) y varios botones de uso frecuente como New, Open,

Save, Compile o Emulate. Esta ventana es en definitiva un editor de texto que permite crear y editar el código fuente de assembler.

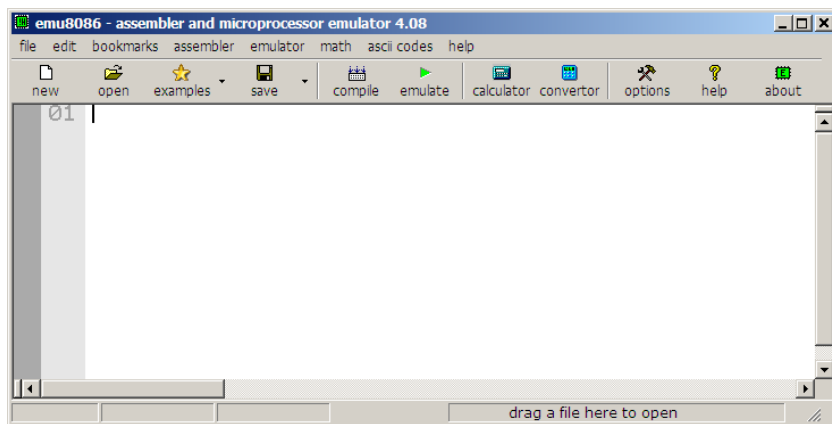


Figura 5 – Ventana principal emu8086

Para ver rápidamente las principales funciones del emu8086 procedemos a abrir uno de los programas de ejemplo, el C:\emu8086\examples\1_sample.asm o “Hello, world” si optamos por seleccionarlo desde el botón rápido “examples”.

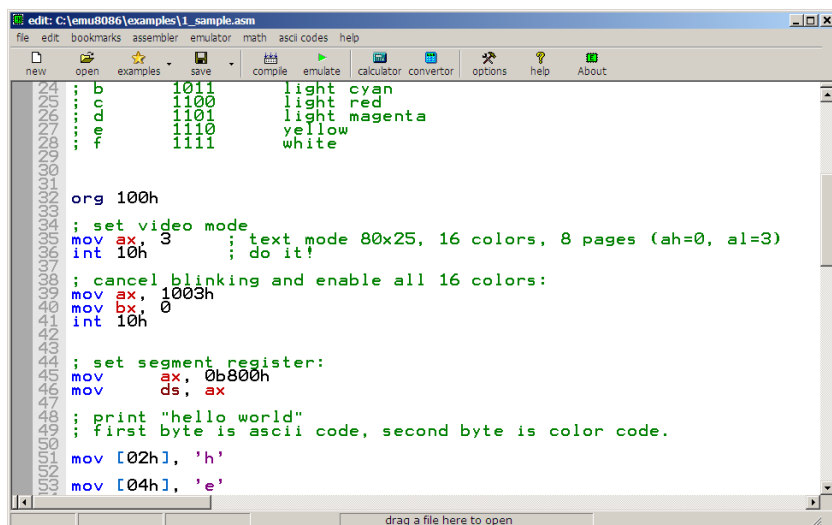


Figura 6 – Editor de código fuente

Luego, al presionar el botón “**Emulate**” se abrirán dos nuevas ventanas, una es el emulador propiamente dicho (*Figura 8 – Emulador*) y la otra muestra el código fuente durante la emulación (*Figura 7 – Código fuente durante la emulación*).

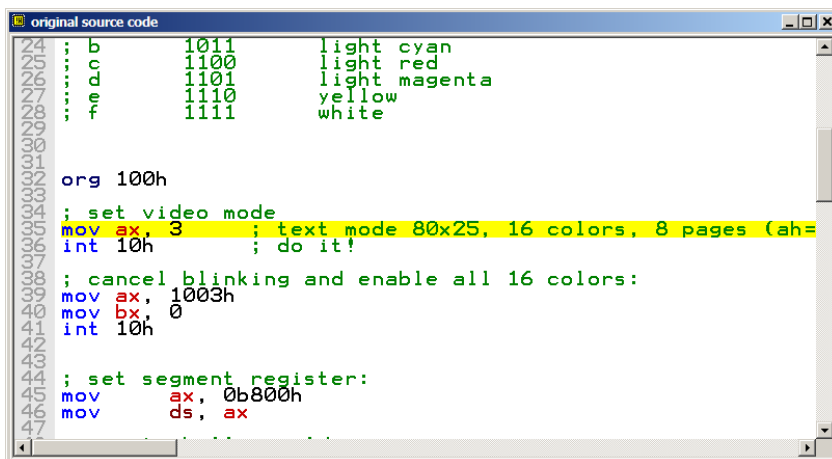



Figura 7 – Código fuente durante la emulación

4 El Emulador

Luego de cargar el código en el emulador al hacer click en el en “Emulate” , se tendrá acceso a gran variedad de funciones e información:

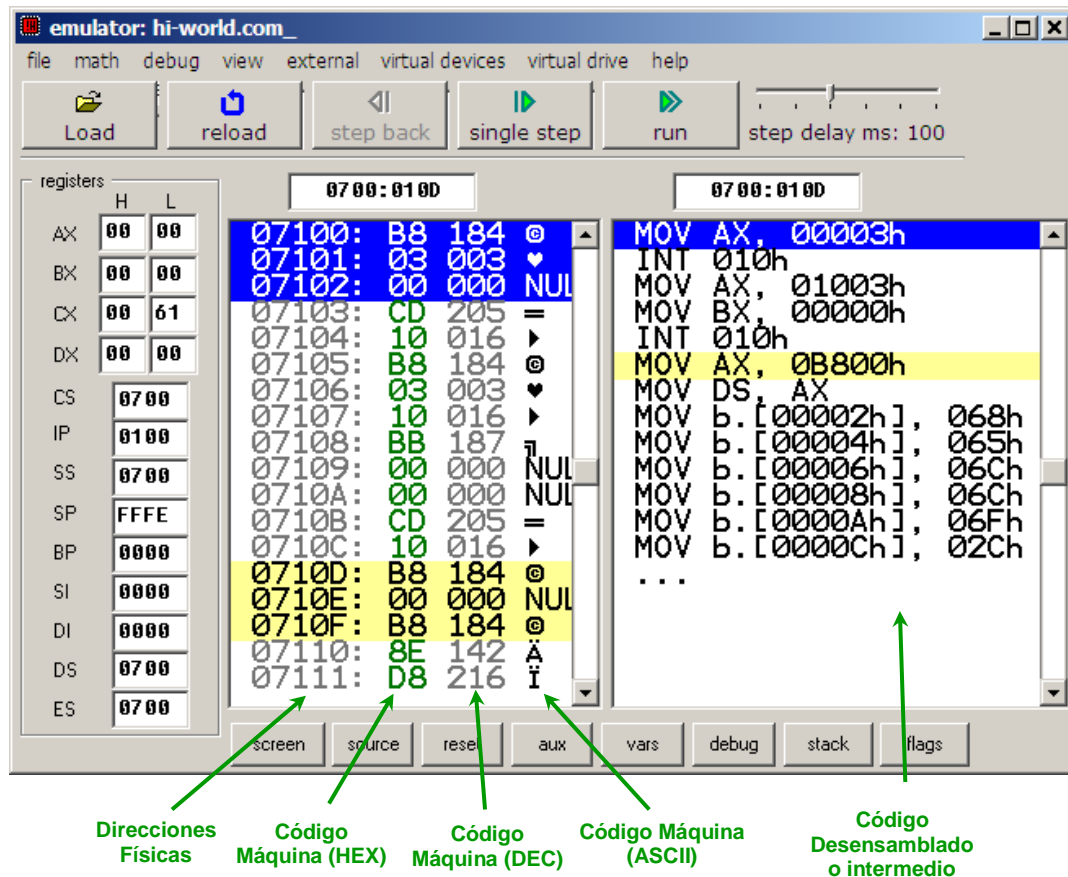


Figura 8 – Emulador

En la parte superior de la ventana se puede ver la barra de herramientas con las siguientes opciones:

- File, permite administrar (cargar o guardar) los archivos que va creando o ejecutando
- Math, da acceso a una calculadora y un convertidor en basas de numeración.
- Debug, provee herramientas para depurar programas.
- View, permite abrir otras ventanas que pueden ser de mucha ayuda al ejecutar y 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 trata 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.

Debajo de la barra de herramientas hay una serie de botones con las siguientes funciones:

- **Load:** carga un archivo ejecutable EXE, COM, etc. ya existente.
- **Reload:** reinicia el programa y comienza a ejecutar el mismo desde la primer instrucción de código, todos los registros inicializan nuevamente,
- **Single step:** permite ejecutar las instrucciones una a una deteniéndose luego de cada instrucción.

- **Step back:** retrocede a la última instrucción que ya fue ejecutada permitiendo ejecutarla nuevamente.
- **Run:** permite ejecutar todas las instrucciones una a una a la velocidad establecida por el control “step delay”. La ejecución se detiene al presionar “STOP”.

Es importante destacar que también es posible, en el menú “debug”, insertar un “break point” cuando se está depurando programa o ejecutar el programa hasta el lugar donde se encuentra el cursor (“run until”).

Debajo de la barra de botones se observan tres paneles, a la izquierda se puede ver el estado de todos los registros disponibles en el 8086. Durante la ejecución de un programa, se puede modificar el contenido de los mismos. Además, al hacer doble click sobre alguno de los registros se abre el “Extended Viewer” que permite ver el contenido del registro representando en distintas bases (binario, hexadecimal, octal, ASCII, decimal, etc.).

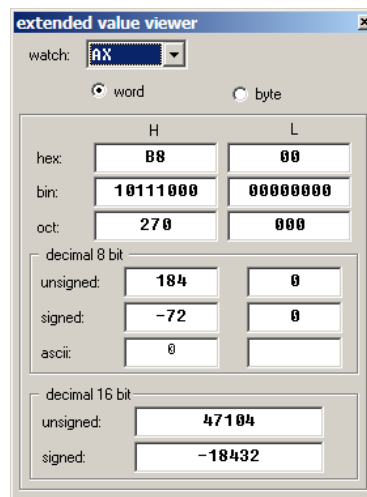


Figura 9 – Extended Viewer

Luego se pueden ver dos paneles, el de la izquierda muestra el código máquina y las direcciones físicas de la memoria de programa con la codificación hexadecimal, decimal y ASCII de cada byte de código. En el panel de la derecha se pueden ver las instrucciones de assembler pero con las direcciones y etiquetas ya resueltas por parte del ensamblador.

En ambos paneles, el código resaltado en azul muestra la próxima instrucción a ser ejecutada. Por ejemplo, en *Figura 8* veremos que la instrucción a ser ejecutada es MOV AX, 0003h, y en el panel de la izquierda que la misma insume 3 bytes de código máquina, el primero en la dirección física 07100h es el B8, que coincide con el código máquina del 8086 para la instrucción MOV AX, mientras que los dos bytes siguientes (direcciones 07101h y 07102h) representan el dato de 16 bits “0003” a ser transferido al registro AX.

A la izquierda, en la ventana del emulador, se pueden ver todos los registros del microprocesador y su valor actual. Luego de cada ejecución aquellos registros que se hayan modificado se verán resaltados en azul. Tener en cuenta que el valor de los registros se puede modificar dinámicamente durante el debug, esto es útil cuando se necesita forzar alguna situación particular durante la evolución del programa.

En la parte inferior de la ventana del emulador se pueden encontrar varios botones de utilidad, el botón Flags permite ver el estado de los Flag, el botón Aux permite, por ejemplo, ver el mapa de memoria, el botón stck muestra el estado de la pila.

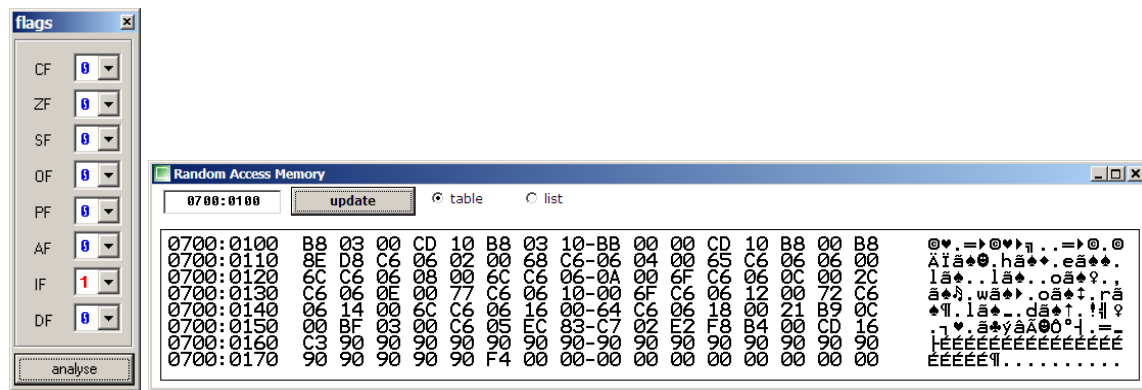


Figura 10 – Flags y mapa de memoria

Junto con la ventana principal del emulador (Figura 8 – Emulador) se abre una ventana complementaria que muestra el código fuente:

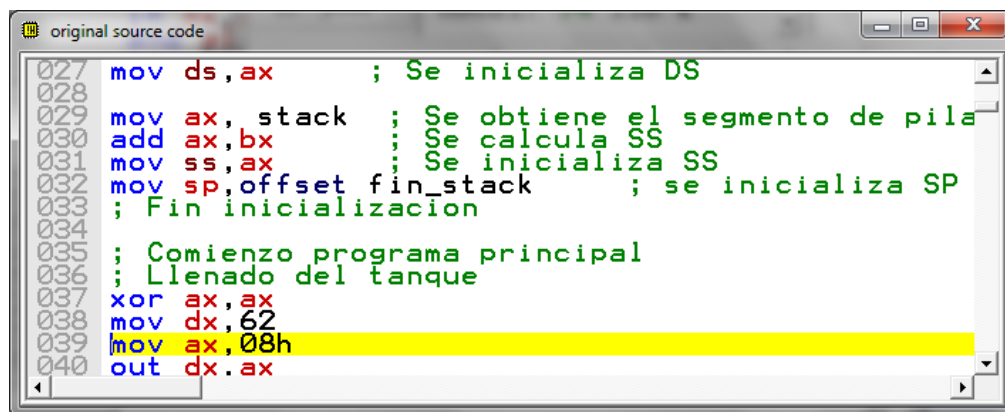


Figura 11 – Emulador, código fuente

Durante el proceso de debug (depuración) se podrá ver entonces, el código fuente, el código desensamblado (código fuente con direcciones/etiquetas resultas) y finalmente el código máquina resultante de cada instrucción.

4.1 Mapa de memoria

Como sabemos, el 8086 cuenta con un bus de direcciones de 20 bits, esto permite por lo tanto direccionar un mapa de memoria de 1 Mbyte.

El emu8086 trae por defecto un mapa de memoria predefinido equivalente al que existe en una arquitectura de PC IBM compatible, al cual el emu8086 llama “Global Memory Table”. Si bien este mapa predefinido facilita el rápido uso y aprendizaje del emu8086, en la cátedra de Digital II no utilizaremos este mapa para la realización de los trabajos prácticos ya que el alumno deberá definir su propio mapa de memoria.

4.1.1 Custom Memory Map

La posibilidad de definir mapas de memoria a medida se conoce como “Custom Memory Map”, y para esto se debe crear un archivo en el directorio raíz del emu8086 llamado **custom_memory_map.inf** (ej. c:\emu8086\custom_memory_map.inf) que especifique como estará conformado el mapa. Por ejemplo, este archivo podría estar definido de la siguiente forma:

```
NO_SYS_INFO
0000:0000 - IVT.bin
ffff:0000 - Arranque.bin
```

Figura 12 – Custom Memory Map

Esta definición del mapa de memoria se extrajo del problema resuelto de ejemplo que la cátedra entrega para utilizarlo como guía en la resolución de los problemas del Trabajo Práctico N°2. Una explicación

más detallada se podrá encontrar en este ejemplo, pero básicamente este archivo le indica al emu8086 como debe conformar el mapa de memoria. En este caso, en la primera posición del mapa se encontrará la IVT, es decir, los punteros (IP y CS) a cada una de las subrutinas de atención de interrupciones (ver sección Interrupciones). Además a partir de la dirección física FFFF0h (o lógica FFFF:0000) se encontrará el código correspondiente con Arranque.bin. Como se puede ver, se utilizan archivos .bin, que es el resultado de la compilación de archivos de código fuente.

4.1.2 Interrupciones

Una de las posibilidades que ofrece el emu8086 a diferencia del MASM 6.11 es la simulación de interrupciones, que es uno de los mecanismos de los cuales dispone el 8086 para comunicarse con dispositivos externos.

Las interrupciones alteran la ejecución del programa en respuesta a eventos externos o una condición de error. Las interrupciones permiten manejar eventos externos, provenientes de dispositivos externos al micro, también permiten resolver situaciones de error al ejecutar una determinada instrucción, por ejemplo, la presencia de una división por cero dispara automáticamente una interrupción Tipo 0. Estas últimas situaciones son ejemplos de interrupciones por hardware y por software, pero ambos tipos de interrupciones son manejadas de la misma forma por el microprocesador.

El microprocesador recibe las interrupciones por hardware a través de líneas o pines particulares propias para este fin presentes en la pastilla del microprocesador, mientras que las interrupciones por software son causadas por la ejecución de la instrucción "INT *n*", donde *n* es el tipo de interrupción a ser ejecutada. El tipo de interrupción puede ser cualquier número entre 0 y 255.

Cuando se produce una interrupción, el microprocesador detiene la ejecución del código que venía procesando y procede a ejecutar una porción de código independiente asociado con el tipo de interrupción, pero para esto, el micro debe primero conocer dentro de la memoria de programa dónde se encuentra ubicada esta porción de código. Antes de poder ejecutar la subrutina asociada a una interrupción específica debe obtener el CS e IP donde se encuentra la misma. Esta información se encuentra en lugar específico del espacio de memoria y se conoce como IVT (Interrupt Vector Table). Dado que los tipos de interrupciones van de 0 a 255, esta tabla cuenta con 256 posiciones, y en cada una de ellas se encuentran dos valores CS e IP de 16 bits cada uno.

4.1.2.1 Interrupt Vector Table

La IVT se encuentra al principio del espacio de memoria y los primeros cuatro bytes contendrán el IP y CS de la subrutina asociada a la interrupción Tipo 0, luego el offset 04h y 06h contendrán el IP y el CS asociados al código de la interrupción Tipo 1, ver *Figura 13*.

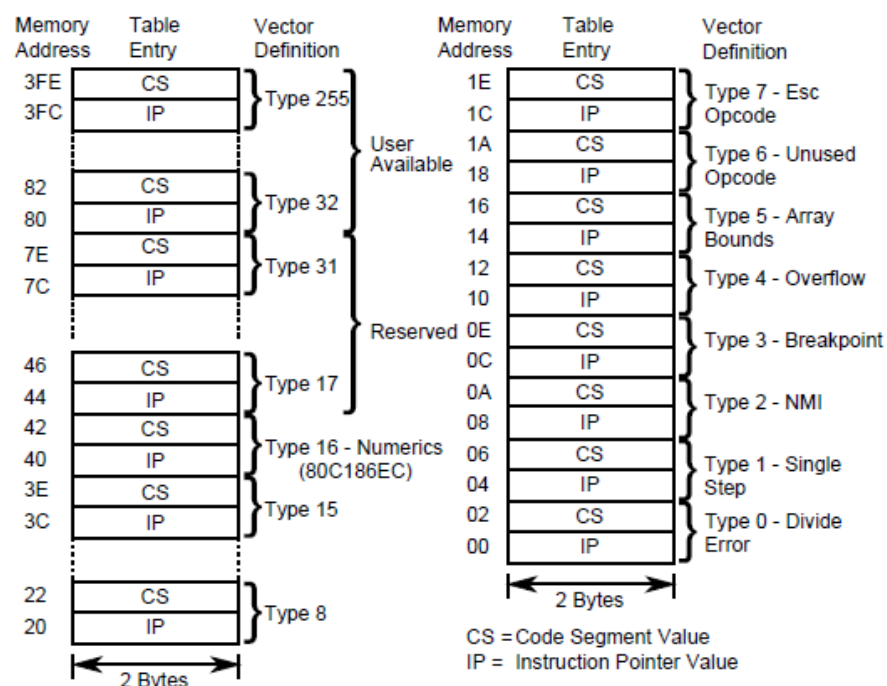


Figura 13 – Interrupt Vector Table (IVT)

Supongamos que se ejecuta la interrupción por software tipo 5 (INT 5), el microprocesador procederá de la siguiente forma para obtener el IP y CS asociado con esa interrupción, multiplica el tipo de interrupción por cuatro, $4 \times 5 = 20$ (14h), este resultado será la dirección física en la cual se encuentra el IP (00014h) y dos posiciones más arriba el CS (00016h).

El programador debe crear y definir los valores de la IVT de forma que ante la existencia de una interrupción el flujo de programa no se vea afectado.

Dado que en la realización del segundo trabajo práctico se utilizarán interrupciones será necesario que el alumno cree y defina la IVT.

4.1.2.2 Interrupción por hardware

Las interrupciones por hardware son generadas por periféricos externos, microcontroladores o por el coprocesador matemático 8087.

Las interrupciones por hardware se encuentran deshabilitadas cuando el Flag de Interrupciones (IF) se encuentra en 0. Cuando el IF está en 1, el emu8086 verifica continuamente los primeros 256 bytes del archivo "emu8086.hw", si alguno de los bytes leídos es distinto de cero, por ejemplo el byte 15, el microprocesador transfiere el control a la subrutina de atención de la interrupción tipo 15 en base a lo configurado en la IVT.

Por defecto, las interrupciones por hardware se encuentran habilitadas, pero se deshabilitan automáticamente cuando se está ejecutando una interrupción de hardware o software.

Cuando se produce una interrupción por hardware el emulador lo indica mediante una leyenda en la parte superior de la ventana del mismo.

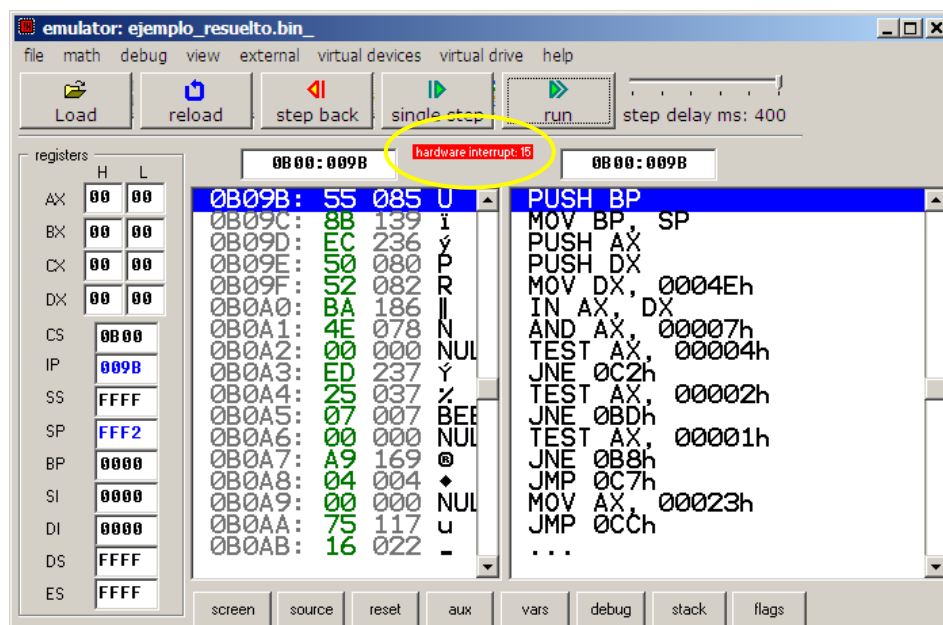


Figura 14 – Emulador – Interrupción por Hardware

En la *Figura 14* se puede ver un ejemplo de interrupción por hardware tipo 15.

En el caso de las interrupciones por software el emulador no realiza ninguna advertencia ante la aparición de la misma.

4.1.3 Puertos de entrada/salida

El emu8086 permite utilizar el espacio de E/S y emular la comunicación con los dispositivos virtuales mediante las instrucciones **in** y **out**. Para esto, el emu8086 utiliza el archivo emu8086.io como medio común para la comunicación con los dispositivos virtuales, tanto él como los dispositivos leen y escriben en este archivo, así por ejemplo el puerto 100 de E/S corresponde con el byte 100 del archivo emu8086.io.

Los puertos utilizados por los dispositivos virtuales de la cátedra se encuentran descriptos en la documentación de ayuda disponible al hacer click en el botón "Ayuda" que se ve en la *Figura 15 – Dispositivos Virtuales – Digital II*.

4.2 Dispositivos Virtuales (DVIO)

El emu8086 cuenta con 7 dispositivos virtuales accesibles desde el menú de herramientas “virtual devices” que se encuentran mapeados en el espacio de entrada/salida y pueden utilizar en conjunto con el emulador del 8086.

Estos dispositivos permiten simular la comunicación entre el micro 8086 y dispositivos externos a través de las instrucciones IN y OUT y de interrupciones por hardware. La comunicación entre los dispositivos y el emu8086 se realiza a través de los archivos emu8086.io y emu8086.hw.

4.2.1 DVIO Digital II

Adicionalmente a los dispositivos nativos del emu8086, la cátedra de Digital II desarrolló 10 dispositivos adicionales que son los utilizados en el Trabajo Práctico N°2. Estos nuevos dispositivos se encuentran disponibles luego de su instalación según se indica en el punto 2.2 Dispositivos Virtuales de Digital II. Luego de iniciar los dispositivos virtuales se muestra la siguiente ventana:



Figura 15 – Dispositivos Virtuales – Digital II

Al hacer click en cada uno de los botones se puede iniciar el dispositivo y este estará en condiciones de interactuar con el emu8086.

En la ventana de la *Figura 15* se puede ver una barra de desplazamiento “Speed” que permite modificar la velocidad temporal de comportamiento de aquellos dispositivos donde la variable tiempo es importante, por ejemplo en los Timers permite modificar la velocidad de cuenta o en el tanque la velocidad de llenado o vaciado del mismo.

Cuando alguno de los dispositivos genera un pedido de interrupción hacia el emu8086 se puede ver en la parte inferior de la ventana la leyenda “Atención. Hay interrupciones pendientes” y el tipo de interrupción “INT:15”, esto debe coincidir con lo mostrado en el emulador (ver *Figura 14*)

4.2.2 Documentación de los Dispositivos Virtuales

Al instalar los dispositivos virtuales (ver sección 2.2) se crea una carpeta que contiene un archivo (...\\emu8086\\DVIO\\docs\\help.pdf) de ayuda con la descripción y forma de utilización de cada uno de los dispositivos virtuales. Adicionalmente, esta misma información se encuentra disponible en formato http al hacer click en el botón “Ayuda” que se observa en la ventana de la *Figura 15*. Tener en cuenta que esta documentación será muy útil al momento de resolver el problema del trabajo práctico N°2.

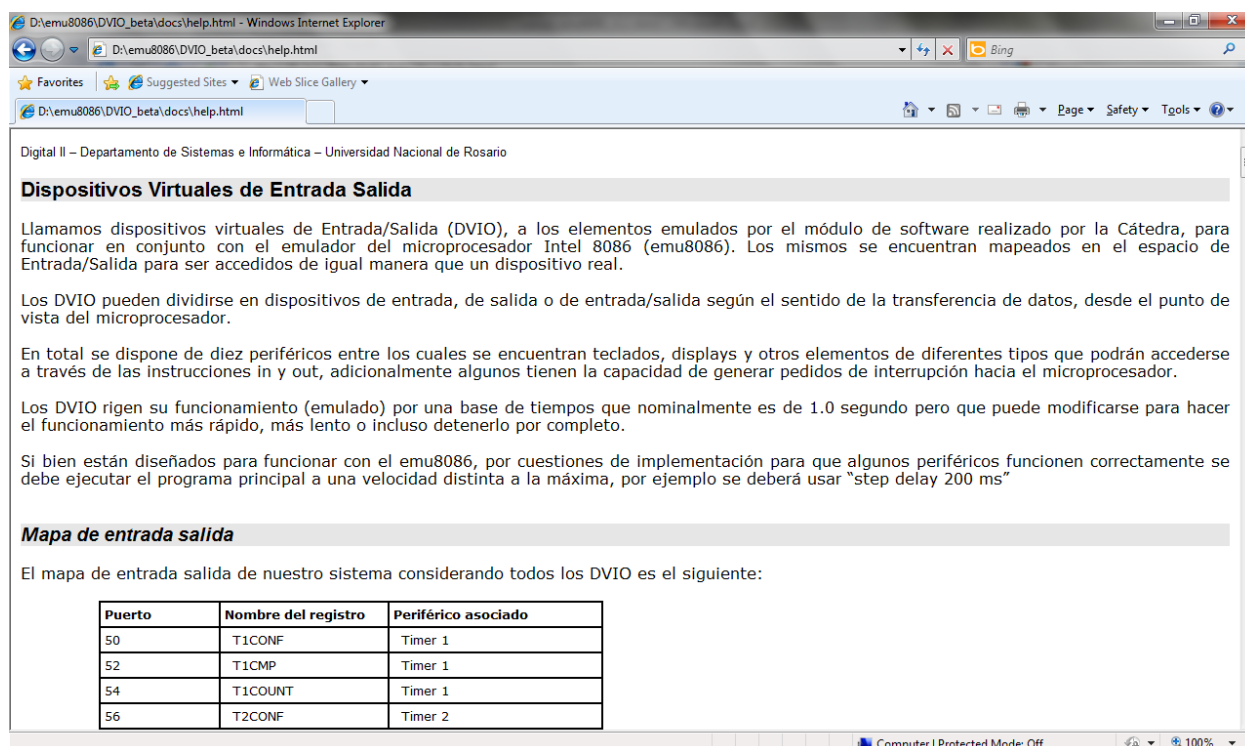


Figura 16 – Ayuda Dispositivos Virtuales

4.2.3 Ejemplo – Problema Resuelto

Al igual que la documentación sobre el uso de los dispositivos virtuales, luego de la instalación de los mismos, se encontrará a disposición un problema de ejemplo resuelto y con explicaciones paso a paso. Este material podrá resultar de mucha utilidad al momento de entender cómo se utilizan los dispositivos virtuales y para utilizarlo como guía al momento de la resolución del problema del TP2. Los archivos y documentación se podrán encontrar en los siguientes directorios:

- ...\\emu8086\\DVIO\\docs\\EjemploResuelto.pdf
- ...\\emu8086\\DVIO\\ejemplo\\ejemplo_resuelto\\

4.2.4 Ejemplo – Dispositivos Tanque y Pulsadores

A continuación se muestra un ejemplo típico de los componentes utilizados en la resolución de los problemas del trabajo práctico de assembler. En la captura de pantalla de la *Figura 17* se puede ver la ventana principal del emu8086 donde se puede crear y editar el código fuente de assembler, se ve también la ventana del emulador y código fuente, la ventana de selección de los dispositivos virtuales y los dispositivos virtuales propiamente dichos (tanque y pulsadores en este caso).

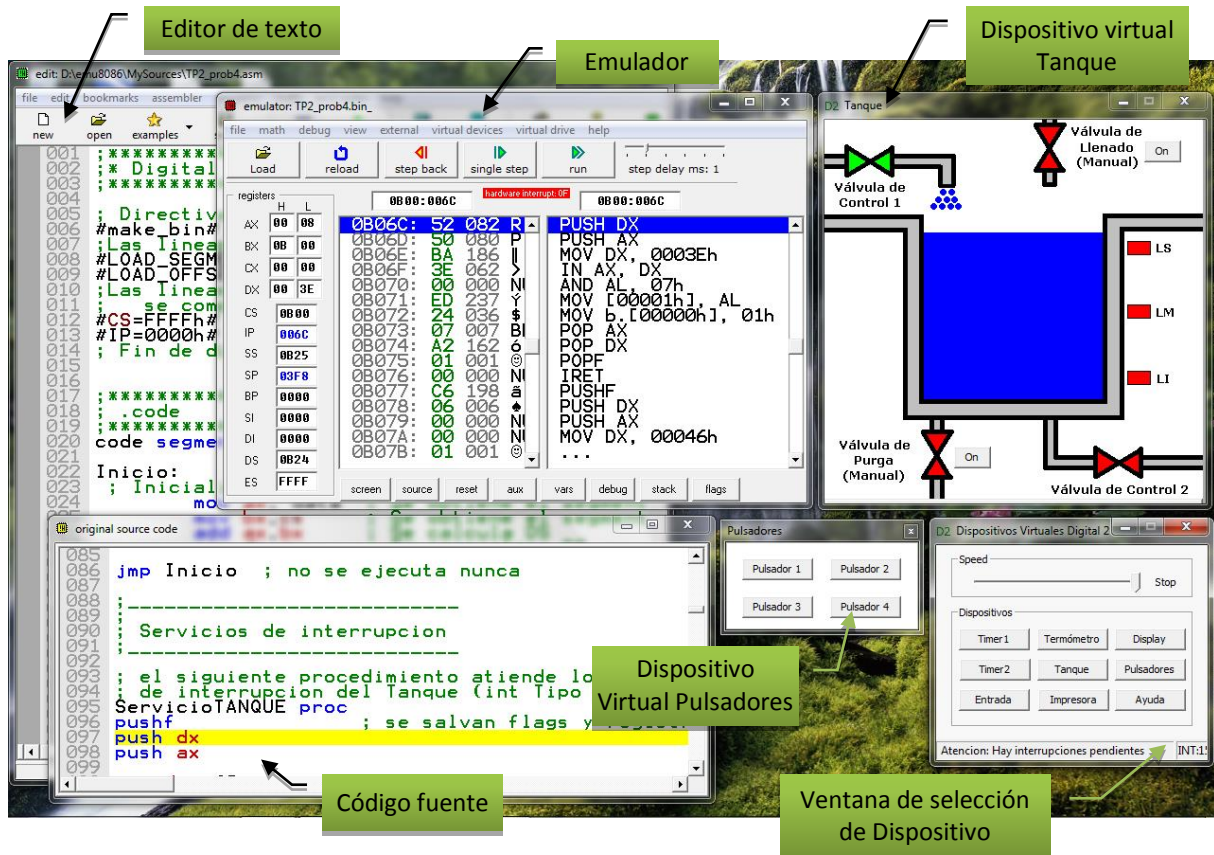


Figura 17 – Ejemplo de dispositivos virtuales y emulador

4.3 Documentación emu8086

El emu8086 cuenta con gran cantidad de documentación que se puede acceder desde “help” en el menú de herramientas en la ventana principal del entorno (Figura 5 – Ventana principal emu8086). Al hacer click en help → documentation and tutorial se abre el navegador y muestra la siguiente pantalla:

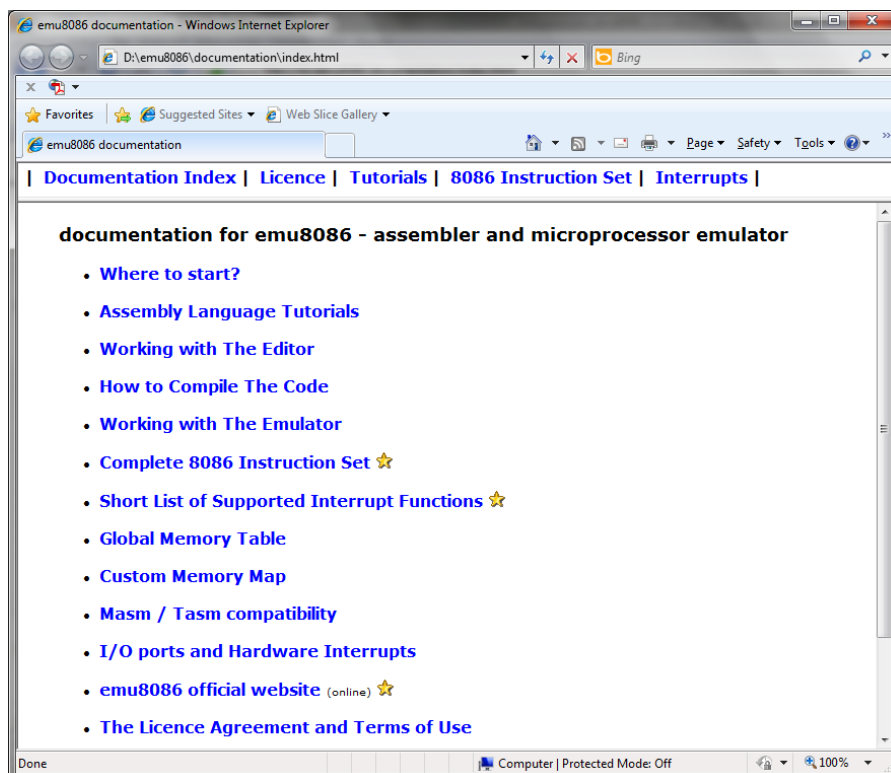


Figura 18 – Documentación y tutoriales emu8086

4.3.1 *Set de instrucciones 8086*

Como parte de la documentación del entorno se encuentra el set de instrucciones del 8086, como se puede ver en la *Figura 18*, en la parte superior de la página el link “8086 Instruction Set” permite ver la totalidad de instrucciones soportadas por el entorno y la descripción y ejemplo de uso de la mismas.

Adicionalmente en el apunte de “Microprocesadores y Microcontroladores” o la “Guía de Referencia Rápida del 80186” se encuentra el set de instrucciones completo y la descripción y función de cada instrucción.