UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo
UNIDAD IZTAPALAPA División de Ciencias Básicas e Ingeniería

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

AREA: INGENIERÍA BIOMEDICA

TARJETA DE SIMULACION DE MEMORIA DE SOLO LECTURA

MIGUEL A. CASTRO GARCIA
SILVERIO E. LEON JULIO
AGUSTIN SUAREZ FERNANDEZ
MIGUEL A. BAUTISTA LEON

TARJETA DE SIMULACION DE MEMORIA DE SOLO LECTURA

Miguel A. Castro García, Silverio E. León Julio, Agustín Suárez Fernández, Miguel A. Bautista León
Area de Ingeniería Biomédica
Departamento de Ingeniería Eléctrica
UAM-Iztapalapa, México, D.F.

INDICE **página**

INTRODUCCION.

Los Sistemas Digitales basados en un microprocesador (μp) o un microcontrolador (μc) se utilizan en aplicaciones dedicadas y la programación que permite su funcionamiento, normalmente, está en firme en una o varias memorias de sólo lectura (ROM). Si la aplicación no va a tener ningún cambio en la programación las memorias ROM son del tipo de producción que están programadas, mediante una mascarilla, desde fábrica. Ya que la cantidad mínima de fabricación de este tipo de memorias es de 1 000 es indispensable que la programación y el funcionamiento de la aplicación hayan sido totalmente revisados y depurados. Uno de los instrumentos idóneos para este tipo de aplicaciones es el analizador lógico que permite depurar la circuitería y la programación. Sin embargo, aun con el empleo de un analizador lógico el mayor tiempo de desarrollo de un Sistema Digital está en la programación. Puesto que ésta se desarrolla en forma gradual se requiere, antes de tener un programa útil, de un proceso de depuración que es largo y tedioso. Ya que consiste en desarrollar el programa, grabarlo en una memoria ROM, probar, detectar fallas, borrar la memoria, grabarla nuevamente y volver a probar. Este proceso se repite tantas veces como sea necesario hasta tener una versión funcional del programa.

El objetivo de los cursos de Sistemas Digitales I, II y III, trimestres 9, 10 y 11 respectivamente, que se imparten en la UAM-Iztapalapa es aprender los principios básicos que permiten el desarrollo de un Sistema Digital. En estos cursos los alumnos desarrollan y aplican un sistema mínimo basado en un microprocesador o en un microcontrolador(11) que requiere que la programación se desarrolle en lenguaje ensamblador y se grabe en una memoria ROM. Así, al igual que en cualquier aplicación de un microprocesador o microcontrolador, la mayor parte del tiempo de desarrollo está en la programación. Por lo tanto, es muy frecuente que los alumnos de éstos cursos adquieran un programador y un borrador de memorias EPROM a fin de tener una mayor facilidad para el desarrollo de sus programas. Esta adquisición, a un costo promedio de $ 2 000 M.N. sólo da una mayor disponibilidad para el proceso de programación-prueba-borrado-programación, pero no lo sustituye. Varios fabricantes de analizadores lógicos tienen como un accesorio un simulador de ROM que cuenta con una interfase de usuario que permite la depuración de la programación en una forma sencilla y grabar la memoria sólo cuando ya se tiene la versión final.

La utilización de un analizador lógico con un simulador de ROM es ideal para el desarrollo de la programación de Sistemas Digitales. Sin embargo, por su muy alto costo, los analizadores lógicos son de una disponibilidad muy limitada y están muy lejos, a diferencia de las microcomputadoras, de ser instrumentos personales. Esta situación, muy común en el desarrollo de Sistemas Digitales, crea la necesidad de tener un simulador de memoria ROM de bajo costo que sea accesible a la mayoría de los usuarios y en particular a los estudiantes de los cursos de Sistemas Digitales de la UAM-Iztapalapa.

OBJETIVOS

Teniendo en cuenta las características de las aplicaciones en las que se utilizará el simulador, y considerando que está disponible una computadora personal, el diseño tuvo como objetivos:

1) Facilidad de aplicación para los microprocesadores/microcontroladores más populares y que se utilizan en los cursos de Sistemas Digitales: Z-80, 80188, 8031, 68000, 68HC11.

2) Tener la programación necesaria para una interfase amable con el usuario y de preferencia correr bajo un sistema operativo. Esta interfase, debe facilitar la evaluación del sistema digital bajo prueba, permitiendo modificaciones a la programación desde la computadora anfitrión, a fin de grabar la memoria del sistema hasta tener la programación totalmente depurada.

3) Desarrollo en base a componentes con la mayor disponibilidad posible en el mercado nacional a fin de ser de una reproducción sencilla y tener el menor costo posible. Es decir, se debe alcanzar la etapa de desarrollo tecnológico.

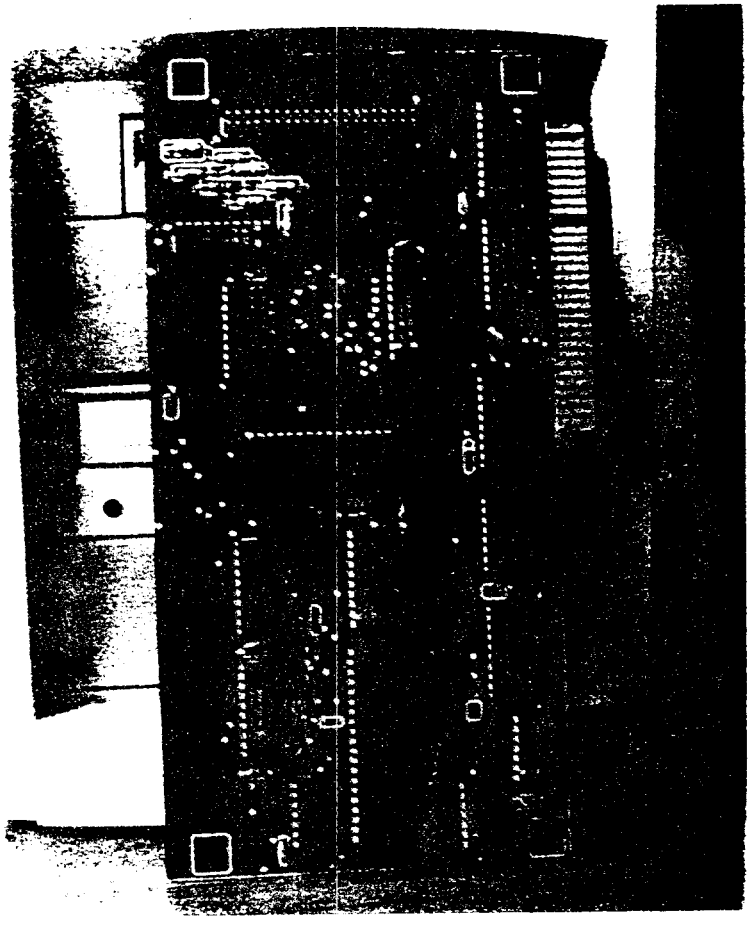


Fig. 1. Simulador SIROM-UAMI.

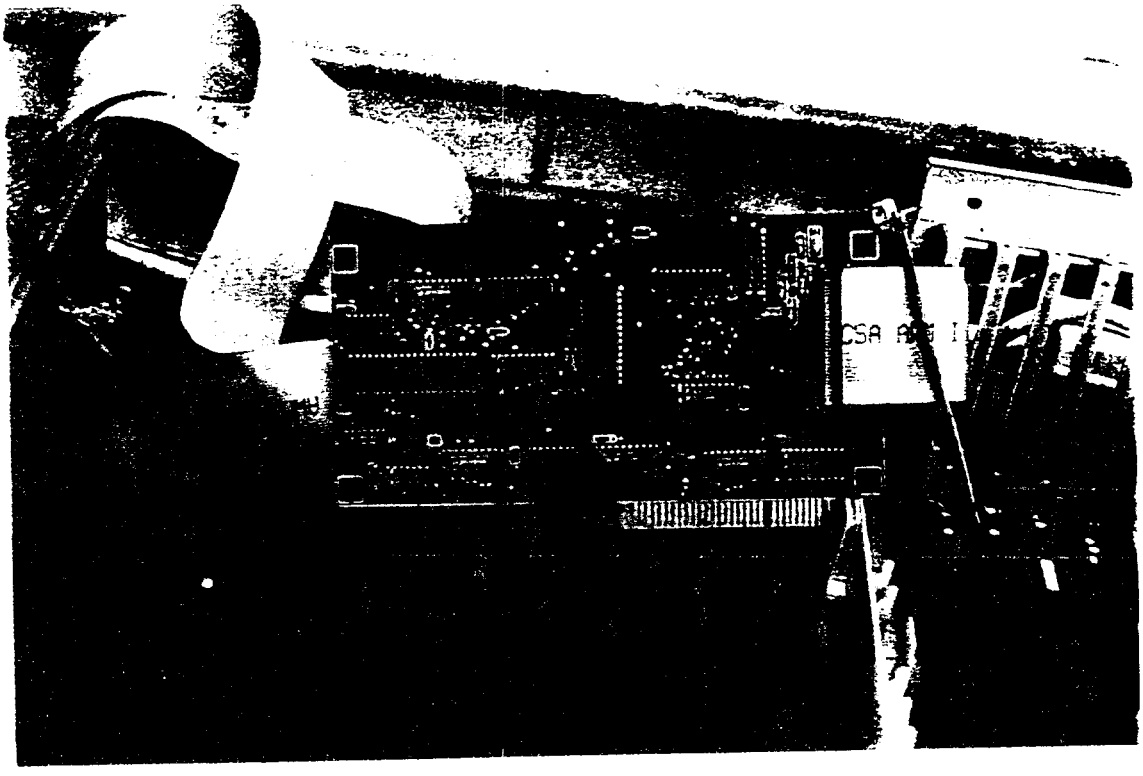


Fig. 1a. Simulador SIROM-UAMI listo para insertarse

como resultado del presente proyecto, se obtuvo un simulador de ROM, denominado SiROM-UAMI, Fig. 1 y 1a, que cumple con los requisitos de diseño anteriores y que se emplea en los laboratorios de Sistemas Digitales de la UAM-Iztapalapa.

Descripción del simulador

Circuitería

El simulador está construido en una tarjeta de circuito impreso que se inserta en una de las ranuras libres de cualquier computadora personal y está programado bajo el sistema operativo Windows 3.11, compatible con Windows 95. Está formado, Fig. 2, por acopladores de corriente, lógica de decodificación y control, interfaz periférica programable 8255, memoria RAM 6264(8Kbytes X 8), conector macho de 28 terminales con cable de 30 Cm. En el apéndice 1 se da un diagrama completo de la circuitería.

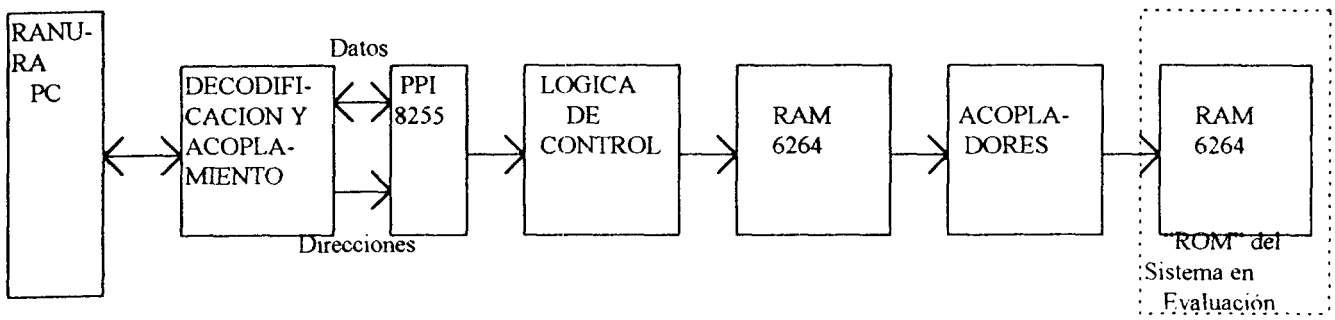
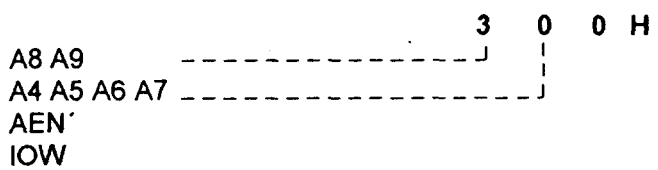


FIG. 21Diagrama a bloques del simulador de memoria ROM.

Lógica de decodificación y acopladores. La lógica de decodificación, parcial, mapea al simulador en las localidades de entrada/salida 300 a 303H, disponibles para tarjetas de prototipos, que utilizan los puertos de la interfaz 8255. La conexión al canal de la computadora se realiza, a través de una de las ranuras disponibles, mediante acopladores de corriente, 74LS244(3) y 74LS245(1), para fines de protección ya que el simulador se emplea en un medio de desarrollo en el que siempre existe la posibilidad de alguna falla. De ésta forma aun cuando se presente una falla destructiva en el sistema bajo prueba, los daños no llegarán a la computadora anfitrión.

La decodificacion de los puertos de entrada/ salida (véase el diagrama esquemático) se hace en base a compuertas OR 74LS32 y a un decodificador 74LS138. Simplificando el problema, sólo se necesita obtener la codificación de la dirección 300H de entrada/salida ya que el direccionamiento detallado de las direcciones 300H-303H se hará directamente sobre el dispositivo 8255, mediante las líneas de dirección A0 y A1.

Considerando en detalle, se necesitan las siguientes líneas para la decodificacion:



La señal AEN, habilitación de direccionamiento (address enable), activa en nivel bajo es necesaria ya que indicará cuando se trata de una dirección válida. La señal IOW´, escritura en el espacio de entrada/salida activa en nivel bajo también es necesaria ya que indica que se trata de una escritura en el espacio de entrada/salida y que es una salida ya que los datos son enviados a la SRAM 6264 para simular a la memoria EPROM.

Esta lógica, al igual que casi cualquier otra, permite la utilización de dispositivos programables PAL™ o GAL™ con la ventaja de reducir el número de componentes utilizados y el tamaño del circuito

impreso, e incluir cierto nivel de seguridad y exclusividad adicional para el diseño. En el apéndice 2 se da un programa que define las ecuaciones lógicas para ésta decodificación mediante un dispositivo GAL 16V8. Aquí no se utilizaron dispositivos programables ya que no hay toda la disponibilidad deseable para el uso del programador.

Interfaz periférica 8255. Realiza, el control y la transferencia de datos de la computadora hacia la memoria RAM 6264. Las terminales de direccionamiento son controladas por el puerto A y cinco bits (PB0-PB4) del puerto B de la interfaz periférica. Mientras que la transferencia de datos hacia la memoria RAM 6264 se lleva a cabo con los ocho bits del puerto C. Todas éstas conexiones se realizan mediante acopladores de corriente 74LS244.

Memoria RAM 6264. Mediante los acopladores de salida y el conector es la memoria "ROM" del sistema bajo prueba. Los acopladores proporcionan la corriente suficiente para manejar un conector macho con un cable de una longitud de entre 30 y 80 cm.

Lógica de control. Las terminales de datos y de direccionamiento de la memoria RAM 6264 están conectadas tanto a la computadora anfitrión, a través de la interfaz 8255, como a las terminales de la "memoria ROM" del sistema bajo prueba. Por lo tanto es necesario que, dependiendo de la función que realice el simulador, reciban los datos que les envía la computadora o que sean las terminales de la "ROM" del sistema bajo prueba. La lógica de control, 74LS04 y 74LS32 llevan las terminales de la parte que pueda causar conflicto, a alta impedancia a fin de que la RAM reciba un programa de la computadora o sea la "ROM" que tiene el programa del sistema bajo prueba, según sea necesario.

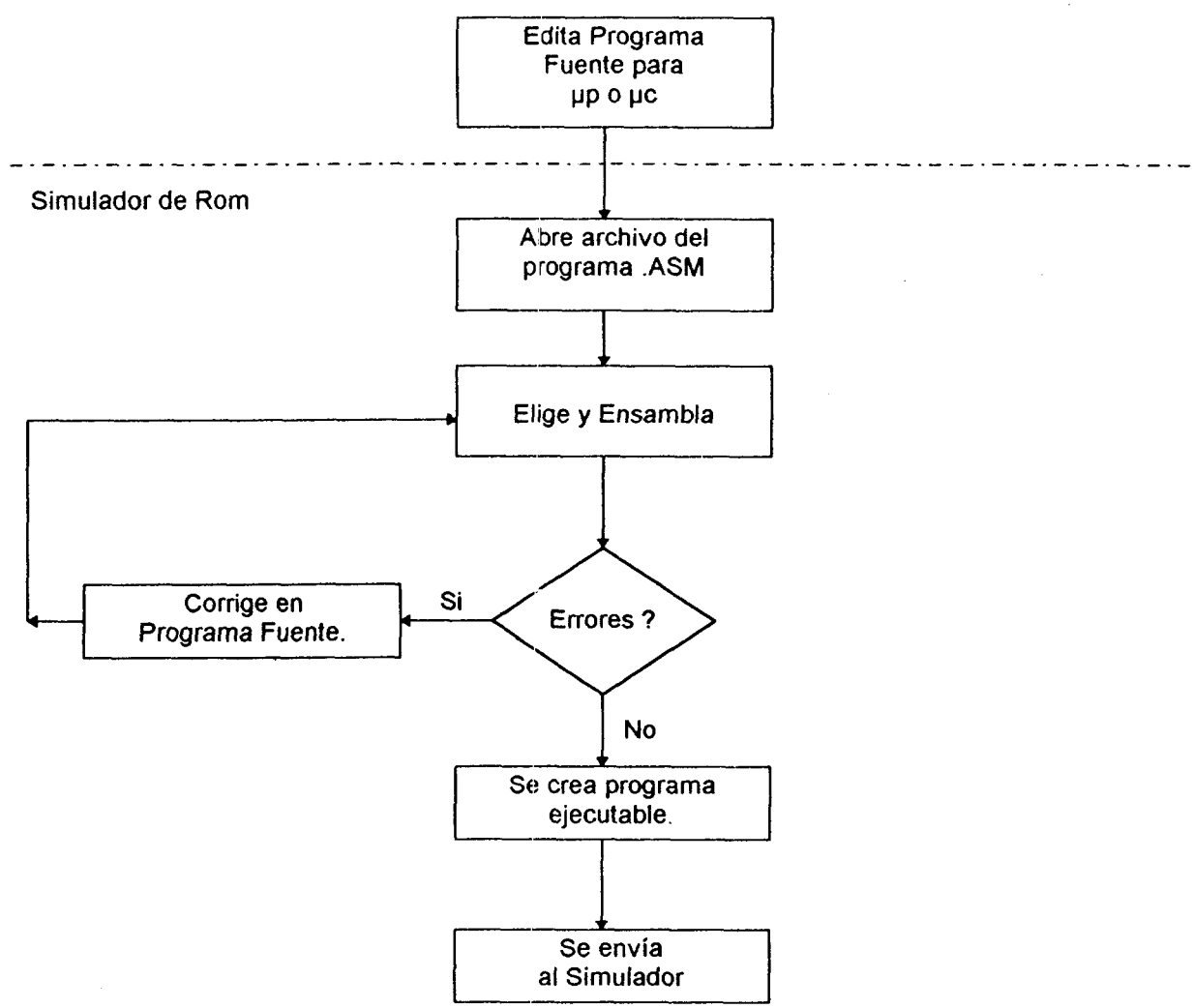


Fig. 3 Diagrama de flujo de la programación del simulador

Programación

El simulador está programado bajo el sistema operativo Windows 3.11 y cuenta con una interfase de usuario que facilita al máximo su empleo. El diagrama de flujo del programa se muestra en la Fig. 3. El diagrama de flujo consiste en lo siguiente:

1) El usuario desarrolla el programa fuente para su aplicación en lenguaje ensamblador para un microprocesador o microcontrolador de los que soporta el simulador de memoria de sólo lectura, utllizando un editor por ejemplo el del sistema operativo DOS.

2) Se invoca el simulador desde el sistema operativo Windows y se selecciona y abre el archivo del programa que se va a probar. Existe la posibilidad de abrir varias ventanas.

3) Se selecciona el ensamblador correspondiente al microprocesador/microcontrolador para el que se desarrolló el programa fuente. En la versión actual se soportan los ensambladores para: Z-80, 80X8X, 68000 y 68HC11. Si existen errores, se notifican al usuario especificando su tipo y el número de línea en que se encuentran. en este caso, los errores se pueden corregir en la ventana que abrió el archivo y repetir el ensamblado hasta que no exista error.

Si no hay errores se crea un programa ejecutable con el código del programa fuente. A continuación, se selecciona enviar y el programa se transfiere al sistema y aparece para el sistema digital bajo prueba como si estuviera grabado en ROM, pero con la ventaja de que para depurarlo no es necesario entrar al proceso programación-prueba-borrado-programación.

Así, se espera que el usuario, sólo proporcione el archivo fuente del programa en ensamblador del microprocesador/microcontrolador que esté utilizando y la programación del simulador, mediante la ejecución de programación por lotes, lo convierte en un archivo ejecutable de 8Kbytes que se carga en la RAM 6264 y que aparecerá como una memoria ROM 2764 para el sistema digital en evaluación. Ya que el programa fuente se puede desarrollar, prácticamente, con cualquier editor los cambios son muy sencillos y evitan la necesidad de grabar una memoria EPROM para cada una de las pruebas.

La mayor parte de la programación se desarrollo utilizando el lenguaje "Visual Basic", que llama a algunas rutinas que se desarrollaron en lenguaje "C". En el apéndice 1 se da un listado completo del programa.

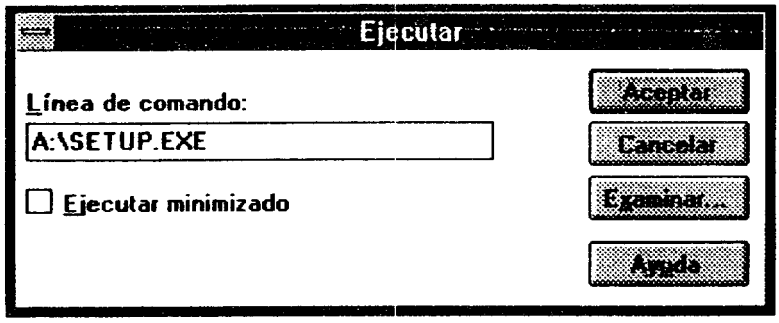
Instalación del simulador

Para el Manejo de la Tarjeta del simulador es necesario un programa de Instalación que se describe a continuación. La Instalación consiste de los siguientes pasos:

1.- Verificar que se tenga instalado un Sistema Operativo Windows versión 3.11 o superior.

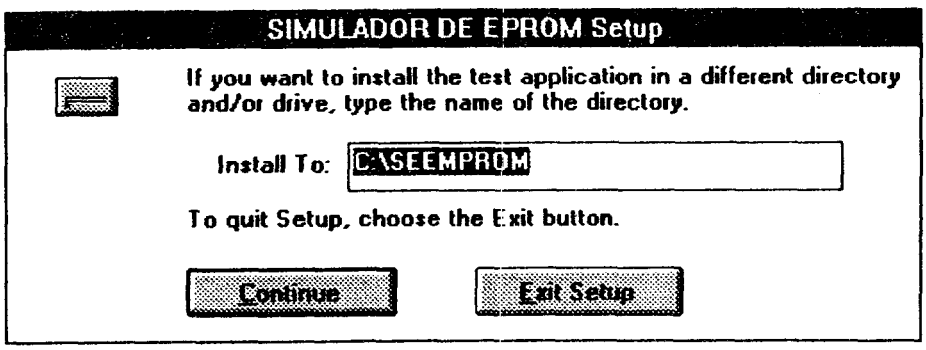
2.- Insertar el Disco de Instalación en la unidad A.

3.- Desde el "Manejador de Programas" ejecutar el programa SETUP.EXE, el cual se encuentra en el disco de instalación. Aparecerá la siguiente ventana:



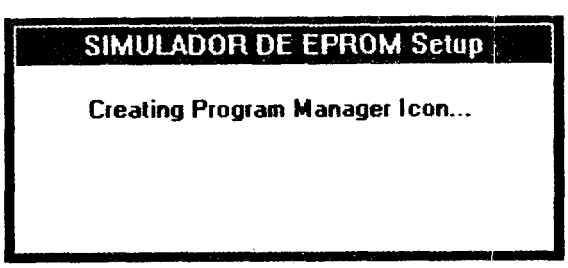
4.- Se inicia la instalación (Setup).

5.- Se le pide al usuario un directorio de instalación, se recomienda que se utilice el directorio dado por default.



6.- A continuación se le da un click a la opción de *Continue* a menos que se requiera salir del programa de instalación se activa *Exit Setup.*

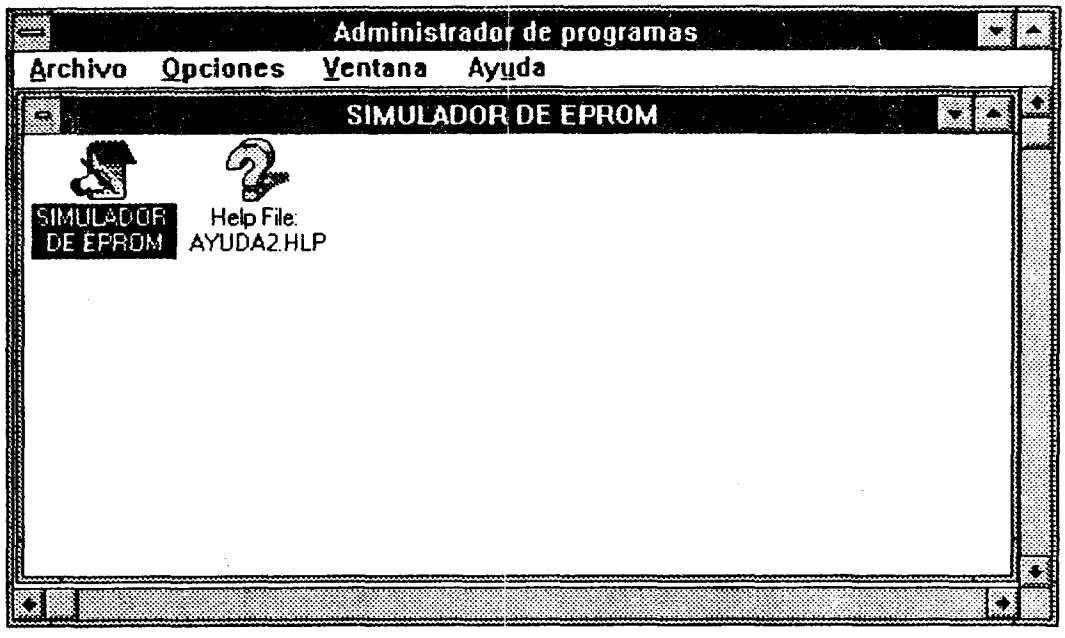
7.- Se empiezan a desempaquetar los archivos necesarios en el directorio seleccionado, y se creará un grupo de iconos.



8.- Si la instalación es exitosa se despliega el siguiente mensaje. y se presiona Aceptar.

9.- A continuación se crea el siguiente Grupo de Programas con dos Elementos de Programa. Uno de ellos es el *Simulador de Eprom* y el otro es la *Ayuda en Línea.*

10.- Finalmente se presiona la tecla de Enter sobre el Icono Simulador de Eprom o se le da un Doble Click al mismo icono, para empezar a ejecutar el programa.



Ejemplo del uso del simulador

A continuación se presenta como ejemplo, ver Fig. 4 y 4a, la utilización del simulador de ROM para un programa para el microprocesador 80188.

Se utiliza el simulador para realizar la prueba inicial del funcionamiento de la tarjeta UAMI-188(9) el programa es el siguiente:

```
                    CODE SEGMENT
                      ASSUME CS:CODE
                         ORG 1FF0H
BA A0FF             MOV   DX,0FFA0H      ; Dirección del registro UMCS=8K
B8 3EFF             MOV   AX,0FE3FH      ; dos ciclos de espera
EE                  OUT   DX,AL          ; Carga AX ahorrando un ciclo de bus
EA                  DB    0EAH
00 00               DW    0000H
00 FE               DW    00FEH          ; Brinca al inicio de la ROM

                       ORG 0000H
BA 5AFF             MOV   DX,0FF5AH      ; Dirección del registro de
B8 FF00             MOV   AX,00FFH       ; cuenta máxima A del temporizador 1
EE                  OUT   DX,AL          ; cuenta FF ciclos con f= 2 MHz.
BA 5CFF             MOV   DX,0FF5CH      ; Dirección del registro de
B8 FF00             MOV   AX,00FFH       ; cuenta máxima B del temporizador 1
EE                  OUT   DX,AL          ; cuenta FF ciclos con f= 2 MHz.
BA 5EFF             MOV   DX,FF5E        ; Dirección del registro de control
B8 03C0             MOV   AX,0C003H      ; modo continuo y alternado
EE                  OUT   DX,AL
                    CODE ENDS
                          END
```

Listado 1 del programa de prueba pr1.asm

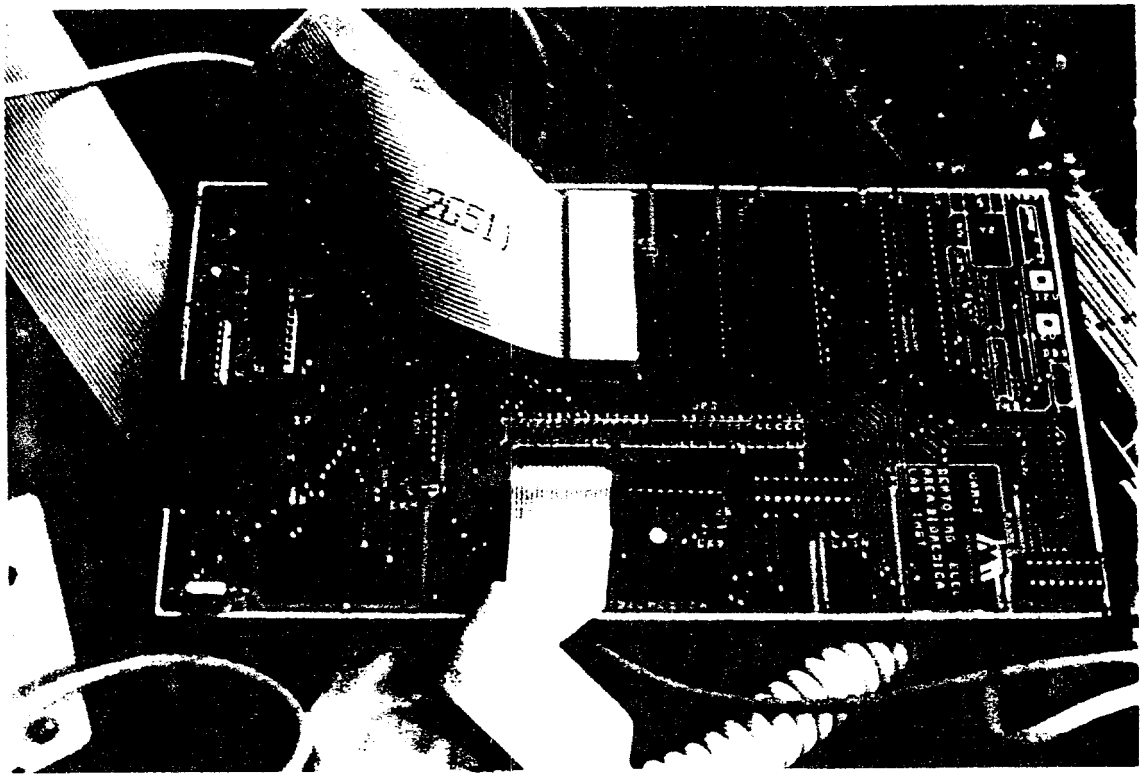


Fig. 4. Simulador conectado al sistema UAMI - 188.

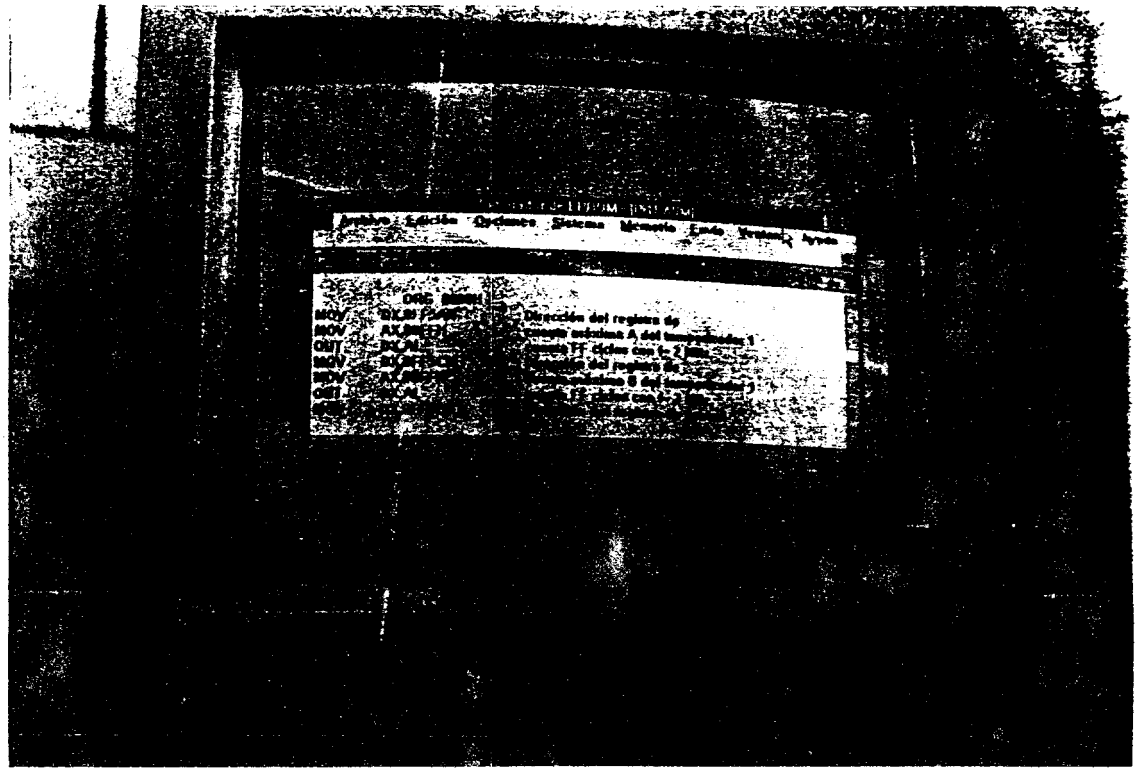
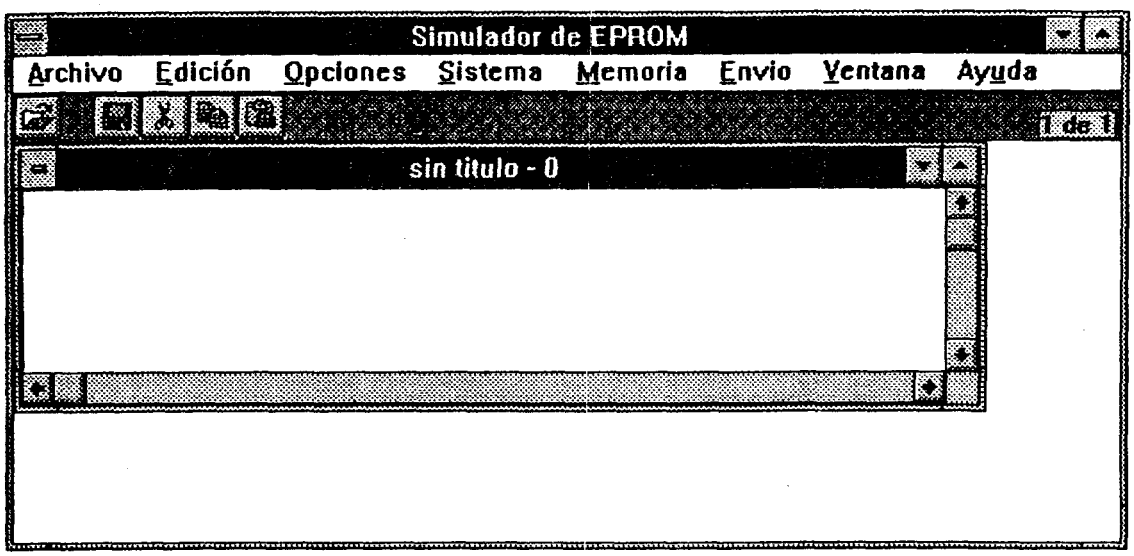


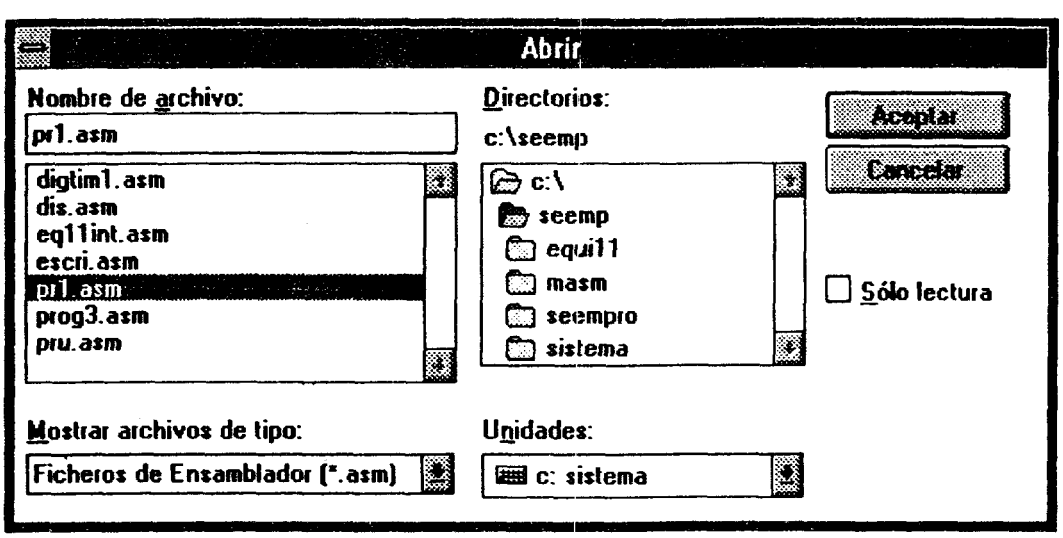
Fig. 4a. Pantalla del ejemplo.

Este programa corre sin necesidad de reubicar el bloque de control de los periféricos integrados y realiza lo siguiente:

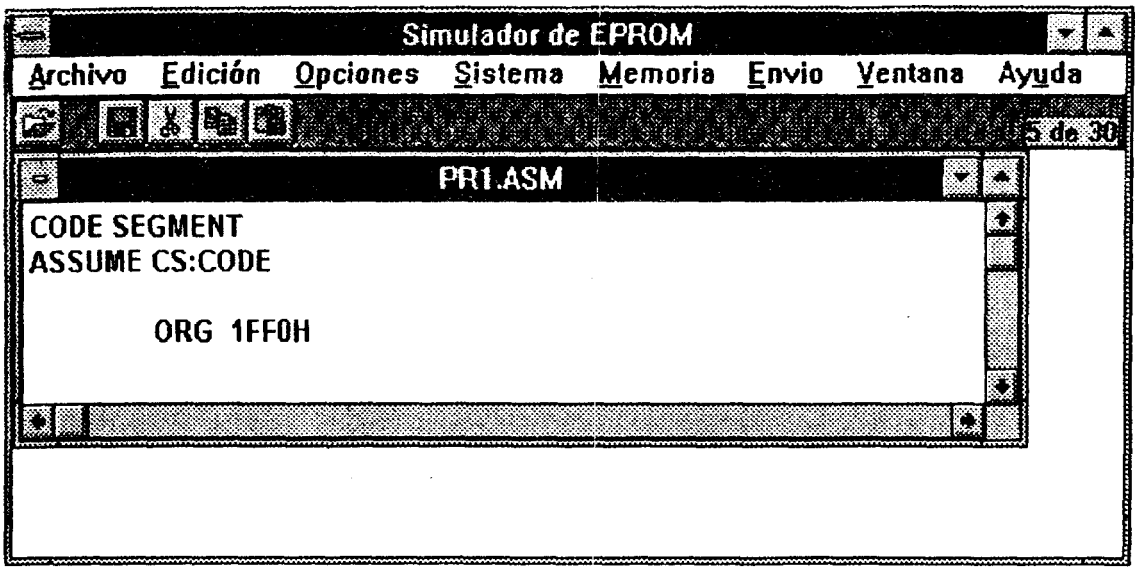
Programa la señal de habilitación de integrado que proporciona el 80188 para el espacio superior de memoria para los últimos 8Kbytes del espacio de memoria [UMCS'] y programa el temporizador integrado 1 para producir una señal cuadrada a su salida. La primera ventana que, presenta el simulador, para abrir el archivo es la siguiente:



De donde se puede observar que ya que la aplicación se encuentra bajo Windows, se pueden abrir todas las ventanas que se requieran que, en éste caso, pueden contener diferentes programas. El primer paso, es abrir el archivo que contiene el código fuente del programa que se va ensamblar que, normalmente, tiene extensión **.ASM**. Esto se puede hacer de dos formas: directamente desde el icono estándar de abrir documentos o desde el Menú Principal, en la opción de **Archivo**, y luego seleccionando **Abrir**. Si se elige la primera opción:

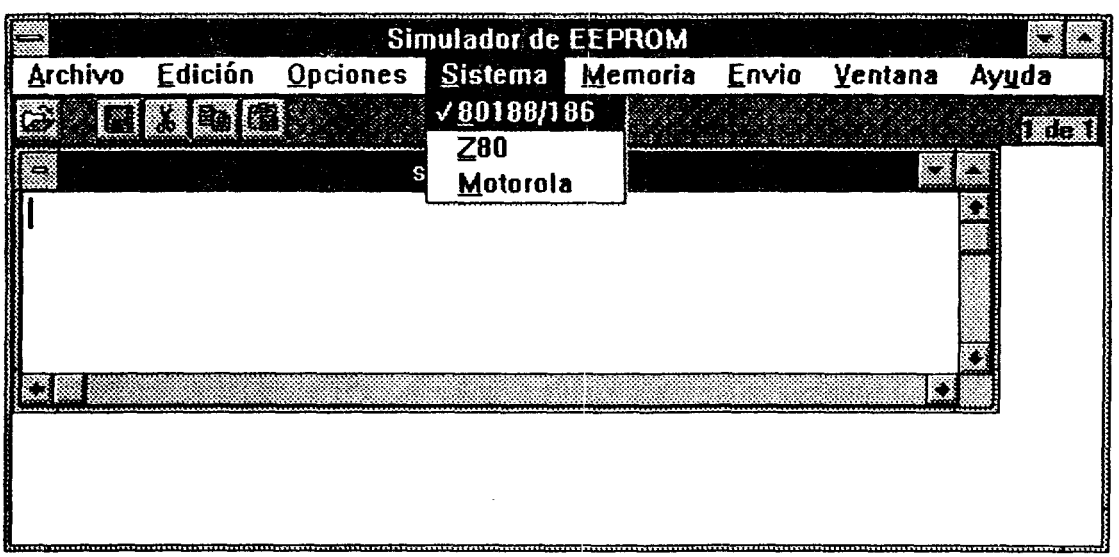


Aquí aparecen todos los archivos *.asm. En este ejemplo se selecciona el archivo pr1.asm.

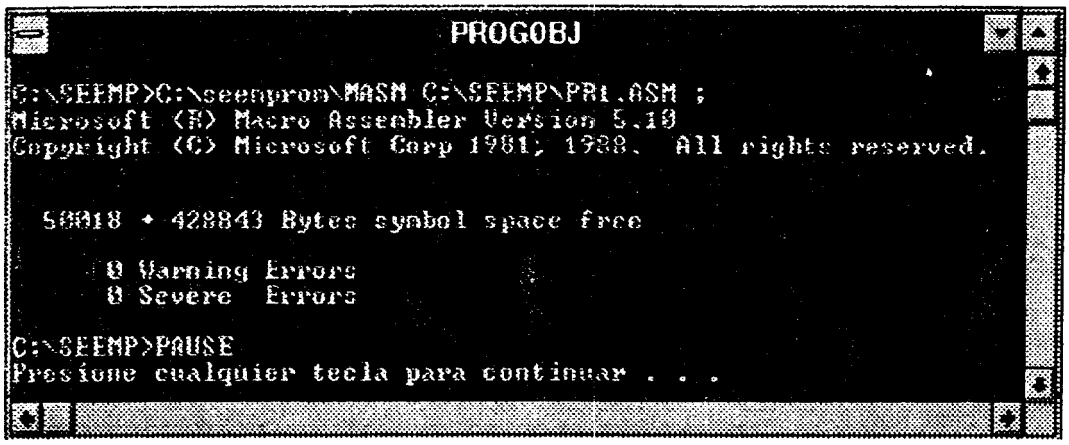


El código fuente del archivo siempre es un archivo de texto, ya que contiene los mnemónicos de las instrucciones, los directivos del ensamblador y los comentarios. Observe que en la parte superior derecha de la ventana aparece el número de la línea actual y el número de líneas totales del código, lo que resulta muy útil para encontrar errores, ya que el ensamblador indica el tipo de error y la línea en que se encuentra. Además, el código se puede editar tantas veces como sea necesario y es posible abrir tantas ventanas como se quieran, ya que la ventana principal maneja múltiples ventanas.

Después se selecciona, dentro de la opción **Sistema**, el ensamblador que corresponde al código.

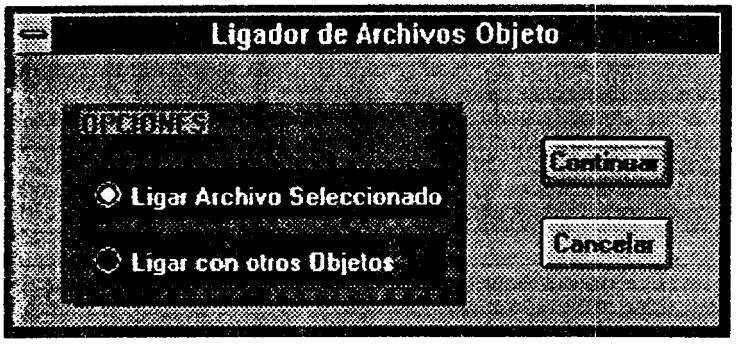


Esto es muy importante ya que en la versión actual, el simulador maneja 4 tipos diferentes de ensamblador: Intel familia 80X8X, Motorola 68000 y 68HC11, y Zilog Z-80. En este ejemplo se elige la opción de Intel (80186/188). A continuación, en la opción **Envío** se selecciona **Editor** y después **Ensamble**, que al activarse presenta la ventana de DOS en la que se ejecuta el proceso de ensamblado:

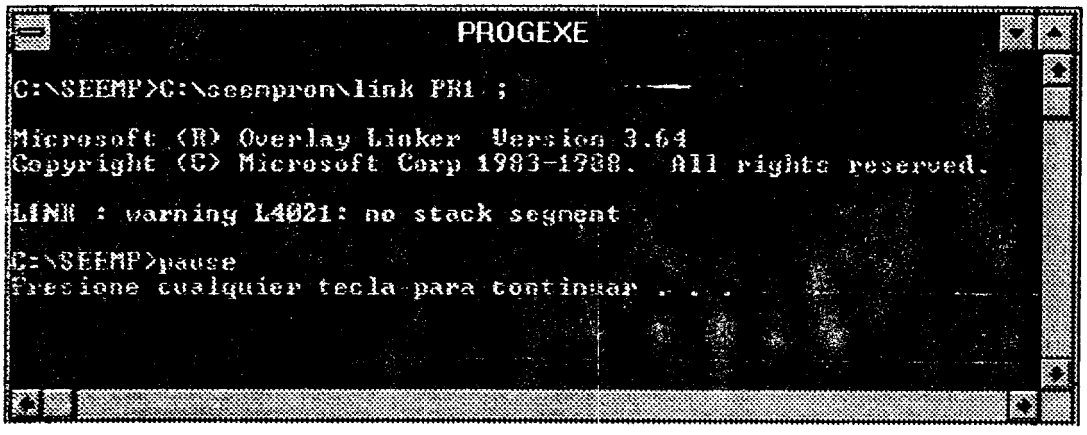


En este ejemplo se ensambló el archivo pr1.asm con el programa, del DOS, MASM.EXE, que crea un archivo con extensión **.OBJ**, que en éste caso es pr1.obj. Si al momento de compilar el programa tiene errores se identificarían sus tipos y los números de las líneas en que se presentan lo que facilita su corrección. Asumiendo que en éste caso, ya que los únicos posibles serían de "dedo", no existen errores.

El siguiente paso es ligar el archivo objeto, a fin de tener un programa ejecutable, en la opción **Envío**, mediante la ventana siguiente.



Con la selección mostrada y si se elige continuar se presenta una ventana como la siguiente.

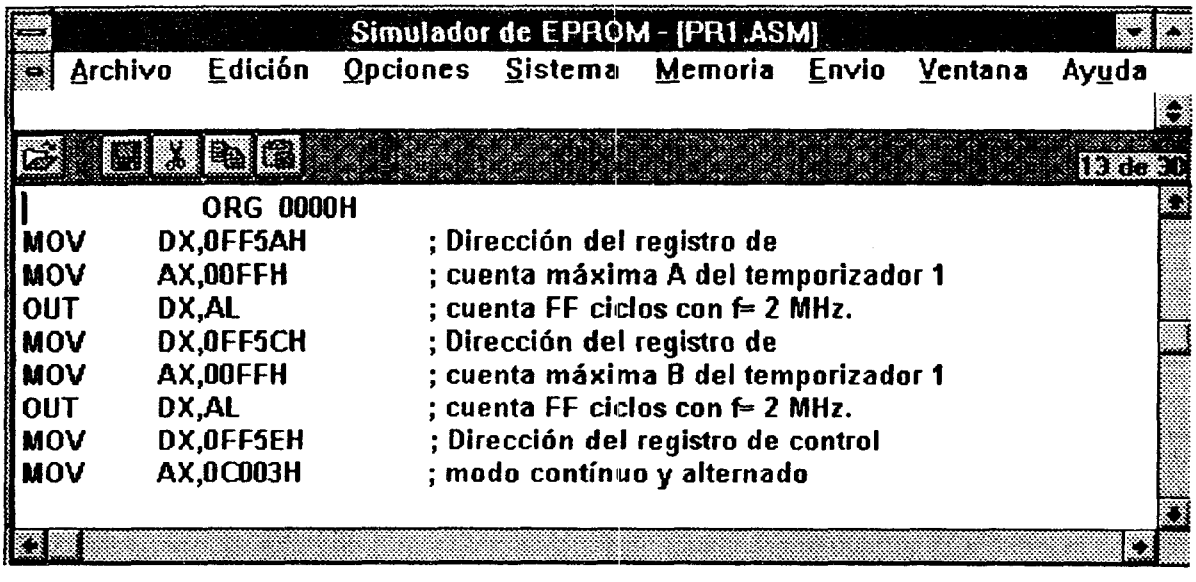


Ya que no hay errores, la prevención que se observa de que no existe el segmento de pila no es importante ya que en este caso la prueba no requiere tal segmento, se crea un archivo binario ejecutable con extensión **.EXE**, que casi está listo para enviarse por el puerto del simulador, sólo que

contiene en sus primeros bytes, código inservible para la simulación de la memoria ROM del sistema bajo prueba, ya que el sistema operativo DOS asume que los programas en ensamblador se ejecutarán en una computadora personal y bajo su control. Esto es cierto para una gran cantidad de usuarios pero no para quien, como es el caso de este ejemplo, tienen sólo un sistema de evaluación. Así, los programas del simulador deben determinar donde inicia el código del programa con el que se va a probar el Sistema Digital y desplazarlo para que sólo se tenga en la "ROM" el código útil. Así, el último paso es seleccionar, dentro de la opción **Envío**, **Proceder**, que carga el archivo **.EXE**, le quita el código innecesario y crea un archivo con extensión **.ES** (Simulador de Eprom) que es el archivo que finalmente se envía a través del puerto hacia la RAM 6264. A continuación, se presenta una ventana con el mensaje de que el envío a finalizado y que se pulse cualquier tecla para salir de esta aplicación. El simulador deja de ser utilizado por la computadora anfitrión, el sistema digital en evaluación "ve" al programa como si estuviera grabado en firme y la computadora puede utilizarse en cualquier otra tarea.

En éste ejemplo, si la prueba resultó las partes básicas de la tarjeta funcionan, el 80188, los circuitos de demultiplexión de los canales de direcciones y de datos, las señales de lectura, habilitación y dirección de los datos y el circuito de reinicialización. Sin emplear el simulador, para ejecutar este programa en la tarjeta UAMI-188 se requeriría, escribir el programa utilizando un editor, ensamblar, ligar y reubicar el programa para que la dirección inicial y la de reinicialización del microprocesador coincidieran. Con el empleo del simulador el usuario sólo tiene que seguir la interfase del programador y no tiene necesidad de reubicar su código.

Ya que ésta es la primera prueba de la tarjeta, no se fijó como requisito que la señal que se genera, mediante el temporizador, tenga una frecuencia de oscilación ni un ciclo de trabajo determinados. Supongamos que ahora se tengan como requisitos generar con el mismo temporizador una frecuencia de 1khz. con un ciclo de trabajo del 50%. En este caso, el cambio al programa pr1.asm es mínimo ya que sólo es necesario cambiar el valor de conteo de los registros A y B del temporizador que se dan en las líneas 8 y 11 del programa. El valor necesario ahora es 01F4H, ya que el cambio es realmente mínimo se vuelve abrir el archivo pr1.asm:



En esta parte se puede modificar y a partir de ahí la secuencia es la misma. Esto contrasta, con el caso en el que se utiliza realmente una ROM ya que un cambio tan simple implicaría borrar, grabar y programar lo que consumiría alrededor de media hora sólo para borrar. En éste caso, aun si el programa fuera más complicado y tuviera errores, la situación más común, es posible corregirlo en el programa fuente y repetir las pruebas cuantas veces sea necesario, sin la necesidad de borrar y

programar una memoria ROM, que sólo se grabará al tener totalmente probado y depurado el programa.

El empleo del simulador es, prácticamente, el mismo para los microprocesadores o microcontroladores restantes a los que da soporte.

Pruebas funcionales

La primera versión de la tarjeta SIROM-UAMI se construyó en una tarjeta de circuito impreso de propósito general mediante la técnica de alambrado "wire-wrap". Una vez depurada la circuitería se realizaron pruebas de la programación que consistieron en la transferencia de programas, totalmente depurados, para sistemas mínimos basados en los microprocesadores Z-80, 80188 y 68000 y en los microcontroladores 8031 y 68HC11. En todos los casos la respuesta fue satisfactoria. La tarjeta también se utilizó para evaluar las prácticas de los cursos de Sistemas Digitales que imparten los autores. Actualmente, el simulador está construido en una tarjeta de circuito impreso.

Conclusiones

1) El simulador SIROM-UAMI permite la depuración de la programación de sistemas digitales desde una computadora anfitrión, modificando sólo el programa fuente evitando el ciclo tradicional programa-grabación-prueba.

2) El simulador SIROM-UAMI permite disminuir el tiempo de desarrollo de la programación de Sistemas Digitales que es el de mayor duración en este tipo de sistemas.

3) Está diseñado a partir de una circuitería muy sencilla y cuenta con una interfase de usuario muy amable y bajo el sistema operativo Windows.

4) Su costo es menor y permite una depuración que no es posible con un programador de memorias.

El simulador se encuentra actualmente en uso y se espera para el trimestre 98-I tener una produccion piloto para alumnos de la UAMI. Se considera que las posibilidades de tener un producto atractivo para el desarrollo de Sistemas Digitales a bajo costo son buenas.

RECONOCIMIENTO

Los autores desean hacer un reconocimiento al técnico del Laboratorio de Instrumentación Médica Electrónica, del Area de Ingeniería Biomédica:

Miguel Angel Martínez Roque

Por su apoyo en el diseño y construcción del circuito impreso del simulador de memoria.

ESPECIFICACIONES

Programación

Visual Basic se proporciona disco de instalación para el sistema operativo Windows 3.1. y/o Windows 95.

Ensambladores soportados

Intel familia 80X8X, Motorola 68000 y 68HC11, Zilog Z-80.

Memoria simulada

EPROM: 8 Kbytes X 8, tipo 2764, con un tiempo de acceso de 100 n.S.

Interfase

Conector macho de 28 terminales entre 30 y 50 cm. para insertar en base de memoria.

Dimensiones:

Ancho: 11 cm.
Largo: 23 cm.

Alimentación:

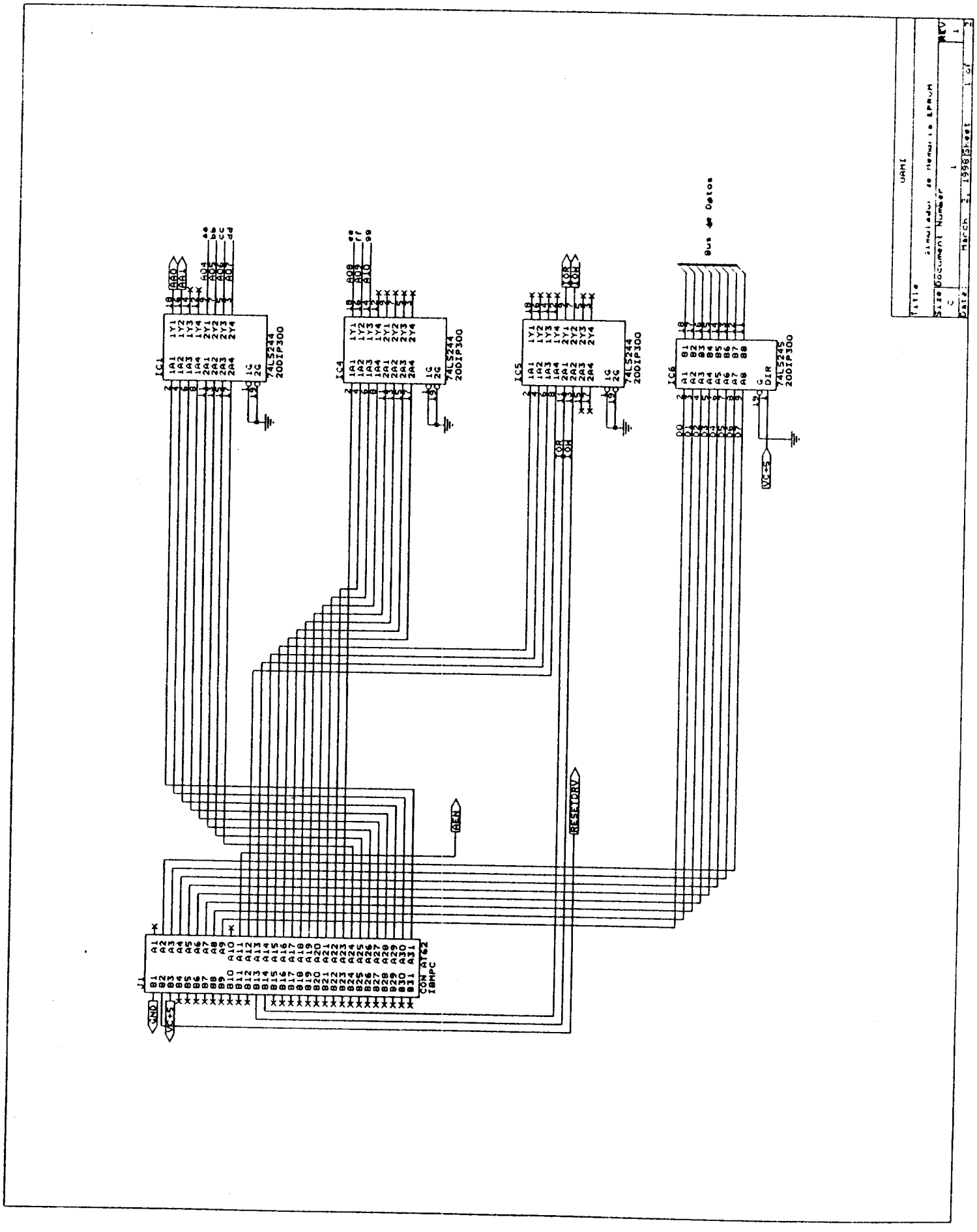
+5 Volts

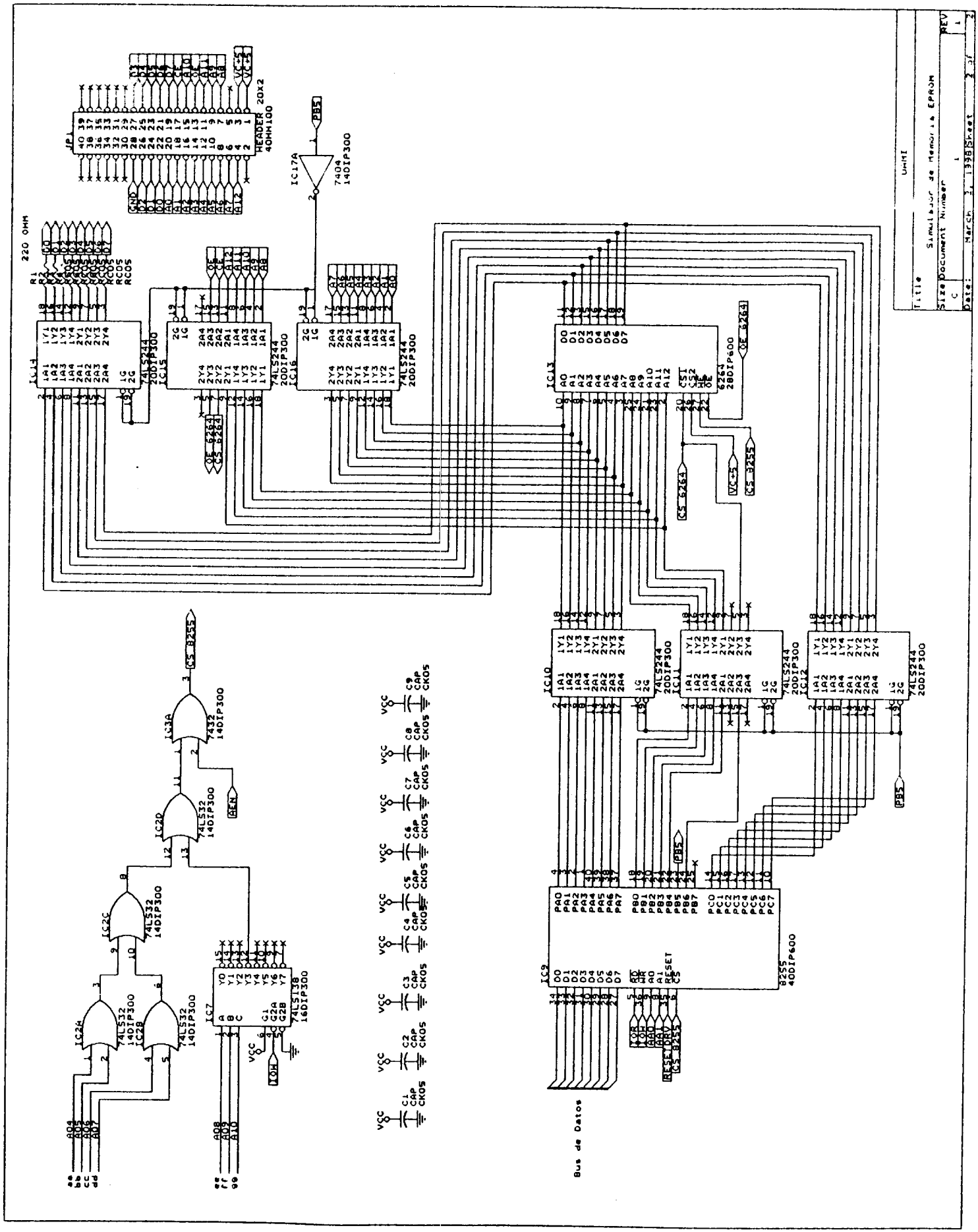
Técnica de construcción:

Dos caras sin conexión

Componentes

Acoplador de tres estados	74LS244	(9)
Transceptor de tres estados	74LS245	(1)
Registro transparente	74LS373	(1)
Compuertas Or	74LS 32	(2)
Decodificador de 3 a 1	74LS138	(1)
Inversores	74LS 04	(1)
Memoria RAM 8kbytes X 8 con un tiempo de acceso de 100 n.S.	6264	(1)
Interfaz periférica	8255	(1)
Conector para cable listón		(1)
Cable listón 30-80 Cm.		(1)
Condensadores de 0.1μ F.		(9)
Resistores de 220Ω a ¼ W.		(8)


UAMI
Bus de Datos
IC1
IC4
IC5
IC6
74LS244
20DIP300
74LS245
GEN
RESETDRV
CON AT62
IBMPC
GND
VCC5


Simulador de Memoria EPROM
Bus de Datos
HEADER 20x2
40HM100
74LS244
20DIP300
6264
28DIP600
8255
40DIP600
74LS32
14DIP300
74LS138
16DIP300
7404
14DIP300
7432
14DIP300
220 OHM
CS 8255
CS 6264
OE 6264
IOR
IOW
RESET
GEN
VCC
GND

Listados Fuente del proyecto

TARJETA DE SIMULACION DE MEMORIA DE SOLO LECTURA

Miguel A. Castro García, Silverio E. León Julio, Agustín Suárez Fernández, Miguel A. Bautista León

Dentro del código fuente se citan los siguientes objetos manejados en Visual Basic 3.0

Variables Globales.

```
Option Explicit
Global numform As Integer

Global Const HELP_CONTEXT = &H1          'Display topic in ulTopic
Global Const HELP_QUIT = &H2             'Terminate help
Global Const HELP_INDEX = &H3            'Display index
Global Const HELP_CONTENTS = &H3
Global Const HELP_HELPONHELP = &H4       'Display help on using help
Global Const HELP_SETINDEX = &H5         'Set the current Index for multi index help
Global Const HELP_SETCONTENTS = &H5
Global Const HELP_CONTEXTPOPUP = &H8
Global Const HELP_FORCEFILE = &H9
Global Const HELP_KEY = &H101            'Display topic for keyword in offabData□
Global Const HELP_COMMAND = &H102
Global Const HELP_PARTIALKEY = &H105     'call the search engine in winhelp
Declare Function winhelp Lib "User" (ByVal hWnd As Integer, ByVal lpHelpFile As String, ByVal
wCommand As Integer, dwData As Any) As Integer


Declare Function SendMessage Lib "User" (ByVal hWnd As Integer, ByVal wMsg As Integer, ByVal wParam
As Integer, lParam As Any) As Long
Declare Function GetActivewindow Lib "User" () As Integer
Declare Function Iswindow Lib "user" (ByVal hWnd As Integer) As Integer
Dim hwndact As Integer, id As Integer
'Constantes correspondientes a los mensajes que serán enviados

Const WM_USER = 1024
Const EM_LINEINDEX = WM_USER + 11
Const EM_LINEFROMCHAR = WM_USER + 25
Const EM_GETLINECOUNT = WM_USER + 10
```

Definición de Variables Generales.

```
Const SI = True
Const NO = False
Const CTRL = 2

Dim IndFuente As Integer
Dim IndTamaño As Integer

Dim id As Integer
```

```
Dim lactual As Integer
Dim estado As Integer
```

Subrutina que elige o ajusta el color del Fondo dentro de las opciones del menú.

```
Sub ColorFondo_Click (Index As Integer)
  Select Case Index
    Case 0    'color de fondo negro
      text1.BackColor = RGB(0, 0, 0)
    Case 1    'color de fondo verde
      text1.BackColor = RGB(0, 255, 0)
    Case 2    'color de fondo azul
      text1.BackColor = RGB(0, 0, 255)
  End Select
End Sub
```

Subrutina que ajusta el tipo de font o texto para desplegarse en las ventanas.

```
Sub ColorTexto_Click (Index As Integer)
  Select Case Index
    Case 0    'color de fondo blanco
      text1.ForeColor = RGB(255, 255, 255)
    Case 1    'color de fondo verde
      text1.ForeColor = RGB(0, 255, 0)
    Case 2    'color de fondo azul
      text1.ForeColor = RGB(0, 0, 255)
  End Select
End Sub
```

Subrutina que habilita las opciones de Cortar, Copia y Pegar dependiendo si hay texto seleccionado o no

```
Sub MenúEdición_Click ()
  'Activar/desactivar Cortar y Copiar si hay/no hay
  'texto seleccionado
  EdiciónCortar.Enabled = (text1.SelLength > 0)
  EdiciónCopiar.Enabled = (text1.SelLength > 0)
  'Activar/desactivar Pegar si hay texto en el portapapeles
  EdiciónPegar.Enabled = (Len(Clipboard.GetText()) > 0)
End Sub
```

Subrutina que se activa al momento de seleccionar la opción Motorola como procesador

```
Sub mot_Click ()
  z80.Checked = False
  o188.Checked = False
  mot.Checked = True
  deleditorliga.Visible = False
End Sub
```

Subrutina que se activa al momento de seleccionar la opción Z80 como procesador

```
Sub z80_Click ()
  z80.Checked = True
  o188.Checked = False
  mot.Checked = False
  deleditorliga.Visible = True
End Sub
```

Subrutina que ajusta la ventana activa en forma del mosaico tradicional.

```
Sub ventanamosaico_Click ()
  mdiform1.Arrange 1

End Sub
```

Subrutina que organiza las ventanas en forma de iconos.

```
Sub ventanaorganizariconos_Click ()
  mdiform1.Arrange 3
End Sub
```

Subrutina que ajusta la ventana activa en forma de cascada tradicional.

```
Sub ventanacascada_Click ()
  mdiform1.Arrange 0

End Sub
```

Subrutina que abre un fichero tipo texto.

```
Sub abrirfichero (fichero As String)
  Dim longitud As Long, nl As String, msg As String
  Static texto As String
  nl = Chr$(10) & Chr$(13)
  On Error GoTo rutinaerrorabrir
  screen.MousePointer = 11'reloj de arena
  Open fichero For Input As #1
  longitud = LOF(1)
  texto = Input$(longitud, #1)
  mdiform1.ActiveForm.Tag = "." + mdiform1.ActiveForm.Tag
  mdiform1.ActiveForm!Text1.Text = texto
  Close #1
salirabrir:
  screen.MousePointer = 0'restaurar puntero
  Exit Sub
rutinaerrorabrir:
   If Err = 5 Or Err = 7 Then
    msg = "el fichero es demasiado grande"
    msg = msg & nl & "para este editor"
```

```
  Else
   msg = "error: no se puede abrir el fichero"
  End If
  MsgBox msg, 48, "editor"
  Unload mdiform1.ActiveForm

  Close
  Resume salirabrir
End Sub
```

Subrutina o función que copia el texto seleccionado al Clipboard.

```
Sub edicioncopiar ()
  clipboard.SetText mdiform1.ActiveForm!Text1.SelText

End Sub
```

Subrutina o función que copia y corta el texto seleccionado al Clipboard.

```
Sub edicioncortar ()
  If TypeOf mdiform1.ActiveForm.ActiveControl Is TextBox Then
  clipboard.SetText mdiform1.ActiveForm.ActiveControl.SelText

  mdiform1.ActiveForm.ActiveControl.SelText = ""
  End If

End Sub
```

Subrutina que pega el contenido del clipboard a la ventana seleccionada.

```
Sub edicionpegar ()
 mdiform1.ActiveForm!Text1.SelText = clipboard.GetText()

End Sub
```

Subrutina que abre un archivo fuente para ser tratado, este es con extensión .ASM

```
Sub ficheroabrir ()
  Dim fichero As String
  Dim numero As Integer, longitud As Integer

  mdiform1.CMDialog1.CancelError = True
  On Error Resume Next
  mdiform1.CMDialog1.Filter = "Ficheros de Ensamblador (*.asm)|*.asm|Ficheros de Texto(*.txt)|*.txt|Todos
los Ficheros (*.*)|*.*"
  'filtro por defecto
  mdiform1.CMDialog1.FilterIndex = 1
  'visualizar la caja de dialogo abrir
  mdiform1.CMDialog1.Action = 1
  'cmdialog1.filename contiene el camino y
  'el nombre del fichero elegido
  If Err <> 32755 Then 'se pulso cancelar
```

```
fichero = mdiform1.CMDialog1.Filename
  numero = forms.Count
  longitud = Len(mdiform1.ActiveForm.Tag)
  If numero > 2 Or Left(mdiform1.ActiveForm.Tag, 9) = "si_cambio" Or
   (Left(mdiform1.ActiveForm.Tag, 9) = "no_cambio" And longitud > 9) Then
    Dim nuevodoc As Form1
    Set nuevodoc = New Form1
    nuevodoc.Caption = mdiform1.CMDialog1.Filetitle
    mdiform1.ActiveForm.Tag = mdiform1.ActiveForm.Tag + " " + fichero

  Else
     mdiform1.ActiveForm.Caption = mdiform1.CMDialog1.Filetitle
     mdiform1.ActiveForm.Tag = mdiform1.ActiveForm.Tag + " " + fichero
  End If
  abrirfichero fichero
 End If

End Sub
```

Subrutina que Guarda o salva la ventana o seleccionada y le asigna un nuevo nombre si no lo tiene.

```
Sub ficheroguardar ()
 Dim fichero As String
 mdiform1.CMDialog1.CancelError = True
 On Error Resume Next
 If Left(mdiform1.ActiveForm.Caption, 8) = "sin titu" Then
  mdiform1.CMDialog1.Filter = "Ficheros de Ensamblador (*.asm)|*.asm|Ficheros de Texto (*.txt)|*.txt|Todos
los Ficheros (*.*)|*.*"
  'filtro por defecto
  mdiform1.CMDialog1.FilterIndex = 1
  'visualizar la caja de dialogo gurdar como
  mdiform1.CMDialog1.Action = 2
  If Err = 32755 Then 'se pulso cancelar
    Exit Sub
  End If
  ' cmdialog1.filename contiene el camino y
  ' el nombre del fichero elegido
  fichero = mdiform1.CMDialog1.Filename
  ' poner el nombre del fichero como titulo
  ' del formulario
  mdiform1.ActiveForm.Caption = mdiform1.CMDialog1.Filetitle
 Else
  'el nombre del fichero es el titulo del formulario
  fichero = Mid$(mdiform1.ActiveForm.Tag, 11, 30)
 End If
 guardarfichero (fichero)
 mdiform1.ActiveForm.Tag = "no_cambio" + " " + fichero'sin cambios

End Sub
```

Subrutina que abre una nueva ventana de edición para código fuente .ASM

```
Sub ficheronuevo ()
  'crear un nuevo ejemplar de form1
  Dim nuevodoc As New Form1
  'siguiente numero para un nuevo formulario
  numform = numform + 1
  'visualizar el nuevo formulario
  nuevodoc.Show

End Sub
```

Subrutina principal e inicial, la cual inicializa mediante el programa "prueba.exe" el PIO 8255.

```
Sub MDIForm_Load ()
  Dim fichero As String

    fichero1 = "C:\seemprom\prueba.exe"

  Open "C:\seemprom\PROGpru.BAT" For Output As #1
  Print #1, fichero1
  Close #1


  id = Shell("c:\seemprom\progpru.pif", 2)

End Sub
```

Una de las 3 subrutinas principales, esta específicamente ensambla dependiendo del procesador para el cual se va se ser la Emulación Z80, Motorola o Intel.

```
Sub ensamble ()
  Dim fichero As String, fichero1 As String, fichero2 As String, direc As String, old_direc As String,
drive_actual As String
  Dim lword, rword, spcpos, nombre
  Dim id As Integer, i As Integer
  Dim nl As String

  nl = Chr$(10) & Chr$(13)
  mdiform1.CMDialog1.CancelError = True
  On Error Resume Next
  If Left(mdiform1.ActiveForm.Caption, 8) = "sin titu" Then
    Beep
    MsgBox "El archivo debe tener nombre. No se tiene ningún archivo para Ensamblar", 16, "ERROR"
    Exit Sub
  End If

  fichero2 = mdiform1.ActiveForm.Caption
  fichero = mdiform1.ActiveForm.Tag
  spcpos = InStr(11, fichero, fichero2)

  lword = Left(fichero, spcpos - 1)
```

```
If Len(lword) > 13 Then
lword = Left(fichero, spcpos - 2)   'se trata de un directorio
End If                              'que no es el raíz

old_direc = CurDir$
drive_actual = Mid$(lword, 11, 3)
ChDrive drive_actual
ChDir Mid$(lword, 11, 30)


spcpos = InStr(1, fichero2, ".")
If spcpos = 0 Then
   nombre = Mid$(lword, 11, 20) + "\" + fichero2 + "." se trata de un archivo sin extensión
Else nombre = Mid$(fichero, 11, 30)
End If

If mdiform1.ActiveForm.o188.Checked = True Then    'se trata de un 188
  fichero1 = "C:\seemprom\MASM" + " " + nombre + " ; " + Chr$(13)

ElseIf mdiform1.ActiveForm.z80.Checked = True Then
     fichero1 = "c:\seemprom\x80" + Chr$(13)'se trata de sistema Z80"
Else fichero1 = "C:\seemprom\AS11 " + nombre + Chr$(13)

End If

fichero1 = fichero1 + " PAUSE"
Open "C:\seemprom\PROGOBJ.BAT" For Output As #1
Print #1, fichero1
Close #1

id = Shell("c:\seemprom\PROGOBJ.pif", 1)


drive_actual = Mid$(old_direc, 1, 3)
ChDrive drive_actual
ChDir old_direc

End Sub
```

Otra de las 3 subrutinas principales, esta especificamente liga el archivo con extensión .OBJ y lo vuelve un archivo ejecutable .EXE dependiendo del procesador para el cual se va se ser la Emulación : Z80, Motorola o Intel.

```
Sub liga ()

Dim fichero As String, fichero1 As String, fichero2 As String, old_direc As String, drive_actual As String
Dim lword, rword, spcpos, nombre
Dim id As Integer, i As Integer
Dim nl As String
```

```
nl = Chr$(10) & Chr$(13)
mdiform1.CMDialog1.CancelError = True
On Error Resume Next
If Left(mdiform1.ActiveForm.Caption, 8) = "sin titu" Then
  Beep
  MsgBox "El archivo debe tener nombre. No se tiene ningún archivo para Ligar", 16, "ERROR"
  Exit Sub
End If

fichero2 = mdiform1.ActiveForm.Caption
fichero = mdiform1.ActiveForm.Tag
spcpos = InStr(11, fichero, fichero2)

lword = Left(fichero, spcpos - 1)

If Len(lword) > 13 Then
lword = Left(fichero, spcpos - 2)   'se trata de un directorio
End If                              'que no es el raíz


old_direc = CurDir$
drive_actual = Mid$(lword, 11, 3)
ChDrive drive_actual
ChDir Mid$(lword, 11, 30)


If mdiform1.ActiveForm.o188.Checked = True Then
 ligade.Option1.Value = True
 ligade.Show 1
 If ligade.cancelar.Tag = "cancelar" Then
  Exit Sub
 End If
 If ligade.Option1.Value Then  'se trata de solo el archivo
                    'seleccionado
 spcpos = InStr(1, fichero2, ".")
  If spcpos <> 0 Then
     nombre = " " + Left(fichero2, spcpos - 1) + " ;"
  Else nombre = " " + fichero2 + " ;"
  End If

 Else nombre = ""           ' se tratan de varios objetos a ligar
 End If
 fichero1 = "C:\seemprom\link" + nombre + Chr$(13)


Else
   fichero1 = "c:\seemprom\link2" + Chr$(13)'se trata de sistema Z80"
End If

 fichero1 = fichero1 + " pause"
 Open "C:\seemprom\PROGEXE.BAT" For Output As #1
 Print #1, fichero1
 Close #1
```

```
id = Shell("c:\seemprom\PROGexe.PIF", 1)


drive_actual = Mid$(old_direc, 1, 3)
ChDrive drive_actual
ChDir old_direc


End Sub
```

Otra de las 3 subrutinas principales, esta específicamente convierte el archivo con extensión .EXE a .SEE y envía el archivo con extensión .SEE (datos) a la memoria dependiendo del procesador para el cual se va se ser la Emulación Z80, Motorola o Intel.

```
Sub enviar ()

 Dim fichero As String, fichero2 As String
 Dim old_title As String
 Dim numero As Integer, longitud As Integer
 Dim nl As String, msg As String
 Dim old_direc As String


 nl = Chr$(10) & Chr$(13)

 old_direc = CurDir$

 mdiform1.CMDialog1.CancelError = True
 On Error Resume Next
 mdiform1.CMDialog1.Filter = "Ficheros de Seemprom (*.see)|*.see|Todos los Ficheros (*.*)|*.*"
 'filtro por defecto
 mdiform1.CMDialog1.FilterIndex = 1
 old_title = mdiform1.CMDialog1.DialogTitle
 mdiform1.CMDialog1.DialogTitle = "Envío Directo"
 'visualizar la caja de dialogo abrir
 mdiform1.CMDialog1.Action = 1

 mdiform1.CMDialog1.DialogTitle = old_title
 'cmdialog1.filename contiene el camino y
 'el nombre del fichero elegido
 If Err <> 32755 Then 'se pulso cancelar
  fichero = mdiform1.CMDialog1.Filename
     fichero2 = "c:\seemprom\proyec2.exe" + " " + fichero

     Open "C:\seemprom\PROGenv.BAT" For Output As #1
     Print #1, fichero2
     Close #1

     id = Shell("c:\seemprom\progenv.pif", 6)
```

```
  msg = "FINALIZADO" & nl & "PULSE CUALQUIER TECLA"
  MsgBox msg, 64, "ENVIO "

 End If

 ChDir old_direc

End Sub
```

Se encarga de hacer solo el envío de datos desde el archivo .SEE a la memoria.

```
Sub proceder ()
  ' se encarga de alinear los programas ejecutables
  Dim fichero As String, fichero1 As String, fichero2 As String, drive_actual As String, old_direc As String,
direc As String
  Dim lword, rword, spcpos, nombre_exe, nombre_see, nombre_hex, nombre_mot
  Dim i As Integer
  Dim nl As String, msg As String
  Dim datos As String * 1

  nl = Chr$(10) & Chr$(13)
  mdiform1.CMDialog1.CancelError = True
  On Error Resume Next
  If Left(mdiform1.ActiveForm.Caption, 8) = "sin titu" Then
    Beep
    MsgBox "El archivo debe tener nombre. No se tiene ningún archivo para Alinear", 16, "ERROR"
    Exit Sub
  End If

  fichero2 = mdiform1.ActiveForm.Caption
  fichero = mdiform1.ActiveForm.Tag
  spcpos = InStr(11, fichero, fichero2)


  lword = Left(fichero, spcpos - 1)


  old_direc = CurDir$
  drive_actual = Mid$(lword, 11, 3)
  ChDrive drive_actual
  ChDir Mid$(lword, 11, 30)
  direc = CurDir$


  lword = Left(fichero, spcpos - 2)   'se trata de un directorio
                  'que no es el raíz


  spcpos = InStr(fichero2, ".")
  fichero2 = Left(fichero2, spcpos - 1)

If mdiform1.ActiveForm.o188.Checked = True Then
```

```
  nombre_exe = Mid$(lword, 11, 30) + "\" + fichero2 + ".exe "
  nombre_see = Mid$(lword, 11, 30) + "\" + fichero2 + ".see"

  fichero1 = "c:\seemprom\exe_see.exe " + nombre_exe + nombre_see

  Open "C:\seemprom\PROGsee.BAT" For Output As #1
  Print #1, fichero1
  Close #1

  screen.MousePointer = 11

  hwndact = GetActivewindow()
  id = Shell("c:\seemprom\progsee.pif", 2)
  Do While GetActivewindow() = hwndact
    id = DoEvents()
  Loop
  hwndact = GetActivewindow()
  Do While Iswindow(hwndact)
    id = DoEvents()
  Loop

     nl = Chr$(13) + Chr$(10)
     fichero2 = mdiform1.ActiveForm.Caption
     fichero = mdiform1.ActiveForm.Tag
     spcpos = InStr(1, fichero2, ".")
     lword = Left(fichero2, spcpos - 1)
     nombre_see = lword + ".see"
     fichero2 = "c:\seemprom\proyec2.exe " + nombre_see

     Open "C:\seemprom\PROGenv.BAT" For Output As #1
     Print #1, fichero2
     Close #1

     id = Shell("c:\seemprom\progenv.pif", 2)

     msg = "FINALIZADO" & nl & "PULSE CUALQUIER TECLA"
     MsgBox msg, 64, "ENVIO DE SISTEMA 80188"

ElseIf mdiform1.ActiveForm.z80.Checked = True Then
  nombre_hex = Mid$(lword, 11, 30) + "\" + fichero2 + ".hex "
  nombre_see = Mid$(lword, 11, 30) + "\" + fichero2 + ".see"
  fichero1 = "c:\seemprom\HEX_SEE.exe " + nombre_hex + nombre_see
  Open "c:\seemprom\proghex.bat" For Output As #1
  Print #1, fichero1
  Close #1
  screen.MousePointer = 11
  hwndact = GetActivewindow()
  id = Shell("c:\seemprom\proghex.pif", 2)
  Do While GetActivewindow() = hwndact
    id = DoEvents()
  Loop
  hwndact = GetActivewindow()
  Do While Iswindow(hwndact)
    id = DoEvents()
```

```
  Loop

     fichero2 = "c:\seemprom\proyec2.exe " + nombre_see

     Open "C:\seemprom\PROGenv.BAT" For Output As #1
     Print #1, fichero2
     Close #1

     id = Shell("c:\seemprom\progenv.pif", 2)


  msg = "FINALIZADO" & nl & "PULSE CUALQUIER TECLA"
  MsgBox msg, 64, "ENVIO DE SISTEMA Z80"

Else

    nombre_mot = Mid$(lword, 11, 30) + "\" + fichero2 + ".s19"
    nombre_see = Mid$(lword, 11, 30) + "\" + fichero2 + ".see"

    fichero1 = "c:\seemprom\s19_see.exe " + nombre_mot + " " + nombre_see
  Open "c:\seemprom\progmot.bat" For Output As #1
  Print #1, fichero1
  Close #1
  screen.MousePointer = 11
  hwndact = GetActivewindow()
  id = Shell("c:\seemprom\progmot.pif", 1)
  Do While GetActivewindow() = hwndact
    id = DoEvents()
  Loop
  hwndact = GetActivewindow()
  Do While Iswindow(hwndact)
    id = DoEvents()
  Loop

     fichero2 = "c:\seemprom\proyec2.exe" + " " + nombre_see

     Open "C:\seemprom\PROGenv.BAT" For Output As #1
     Print #1, fichero2
     Close #1

     id = Shell("c:\seemprom\progenv.pif", 6)

  msg = "FINALIZADO" & nl & "PULSE CUALQUIER TECLA"
  MsgBox msg, 64, "ENVIO DE SISTEMA MOTOROLA"


End If
  screen.MousePointer = 0

  drive_actual = Mid$(old_direc, 1, 3)
  ChDrive drive_actual
  ChDir old_direc

End Sub
```

Programas secundarios llamados por el programa principal SEEMPROM.

Código fuente en ensamblador del programa "PRUEBA.EXE" el cual inicializa el PIO 8255.

```
STAC SEGMENT STACK
   DW 1024 DUP (?)
STAC ENDS

DAT SEGMENT
  CRLF DB '13,10,25 $'
DAT ENDS

COD SEGMENT
  ASSUME CS:COD,DS:DAT,SS:STAC

MAIN PROC FAR
   MOV AX,DAT
   MOV DS,AX

   MOV AL,80H
   MOV DX,303H   ; todos los puertos son de salida en modo 0
   OUT DX,AL

   MOV AH,4CH
   INT 21H

MAIN ENDP
COD ENDS
  END MAIN
```

Código Fuente en Lenguaje C del programa "HEX_SEE" el cual convierte archivos con extensión .HEX (Z80) a archivos con extensión .SEE el cual es el formato de archivo que maneja el Simulador de Memorias EPROM.

```
#include <io.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dos.h>

num_datos(FILE *in)

{char dato1, dato2;
 int  numerohex,num1,num2;


 dato1=fgetc(in);
 dato2=fgetc(in);

 switch (dato1)
```

```
{
case '0' :  {num1=0;
             break;}
case '1' :  {num1=1;
             break;}
case '2' :  {num1=2;
             break;}
case '3' :  {num1=3;
             break;}
case '4' :  {num1=4;
             break;}
case '5' :  {num1=5;
             break;}
case '6' :  {num1=6;
             break;}
case '7' :  {num1=7;
             break;}
case '8' :  {num1=8;
             break;}
case '9' :  {num1=9;
             break;}
case 'A' :  {num1=10;
             break;}
case 'B' :  {num1=11;
             break;}
case 'C' :  {num1=12;
             break;}
case 'D' :  {num1=13;
             break;}
case 'E' :  {num1=14;
             break;}
case 'F' :  {num1=15;
             break;}
default:;
}

switch (dato2)
 {
 case '0' :  {num2=0;
              break;}
 case '1' :  {num2=1;
              break;}
 case '2' :  {num2=2;
              break;}
 case '3' :  {num2=3;
              break;}
 case '4' :  {num2=4;
              break;}
 case '5' :  {num2=5;
              break;}
 case '6' :  {num2=6;
              break;}
 case '7' :  {num2=7;
```

```
                    break;}
    case '8' :  {num2=8;
                    break;}
    case '9' :  {num2=9;
                    break;}
    case 'A' :  {num2=10;
                    break;}
    case 'B' :  {num2=11;
                    break;}
    case 'C' :  {num2=12;
                    break;}
    case 'D' :  {num2=13;
                    break;}
    case 'E' :  {num2=14;
                    break;}
    case 'F' :  {num2=15;
                    break;}

    default:;
    }


  numerohex=num1*16+num2;
  return numerohex;

 }


 main(int argc,char *argv[])
 { FILE *in,*out;
  char *parametro1,*parametro2;
  char dato;
  char  num_dat,dir_low,dir_hi;
  char  tipo_reg;
  char  i,datos;
  int   directotal;
  int   j;


 parametro1=argv[1];
 parametro2=argv[2];

 in=fopen(parametro1,"rt");
 out=fopen(parametro2,"wb");

 for (j=0;j<=0x1fff;j++)
 putc(0xff,out);

 dato=getc(in);
 if (dato==':')
{


num_dat =num_datos(in);
```

```
dir_hi =num_datos(in);
dir_low =num_datos(in);
tipo_reg=num_datos(in);


while (tipo_reg!='\x3')
{

  _BL=dir_low;
  _BH=dir_hi;
  directotal=_BX;


  directotal=directotal-1;

  for (i=1; i<=num_dat; i++){

    directotal=directotal+1;
    datos=num_datos(in);
    fseek(out,directotal,SEEK_SET);
    putc(datos,out);

   } //fin del for
  datos=num_datos(in);  // se lee el checksum del archivo
  dato=getc(in);        // se lee el fin de línea
  dato=getc(in);        // se lee el código de nueva línea

  num_dat =num_datos(in);
  dir_hi =num_datos(in);
  dir_low =num_datos(in);
  tipo_reg=num_datos(in);
}

} // fin del if principal

fclose(in);
fclose(out);

return 0;
}
```

Código Fuente en Lenguaje C del programa "EXE_SEE" el cual convierte archivos con extensión .EXE (Intel) a archivos con extensión .SEE el cual es el formato de archivo que maneja el Simulador de Memorias EPROM.

```
#include "stdio.h"
#include "conio.h"
main(int argc,char *argv[])
{ FILE *in,*out;
  char car;
  int i;
```

```
 char *parametro1;
 char *parametro2;

/*clrscr();*/

parametro1=argv[1];
parametro2=argv[2];

in  = fopen(parametro1,"rb");
out = fopen(parametro2,"w+b");

for (i=0;i<=0xff;i++)
 {
  fgetc(in);
  }

do
{
 i++;
 car=fgetc(in);

} while (car==0);

 fputc(car,out);

for (i=1; i<=0x1fff; i++)
  fputc(fgetc(in),out);

fclose (in);
fclose (out);

return 0;
}
```

Código Fuente en Lenguaje C del programa "S19_SEE" el cual convierte archivos con extensión .S19 (Motorola) a archivos con extensión .SEE el cual es el formato de archivo que maneja el Simulador de Memorias EPROM.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>

num_datos(FILE *in)

{char dato1, dato2;
 int  numerohex,num1,num2;

 dato1=fgetc(in);
 dato2=fgetc(in);
```

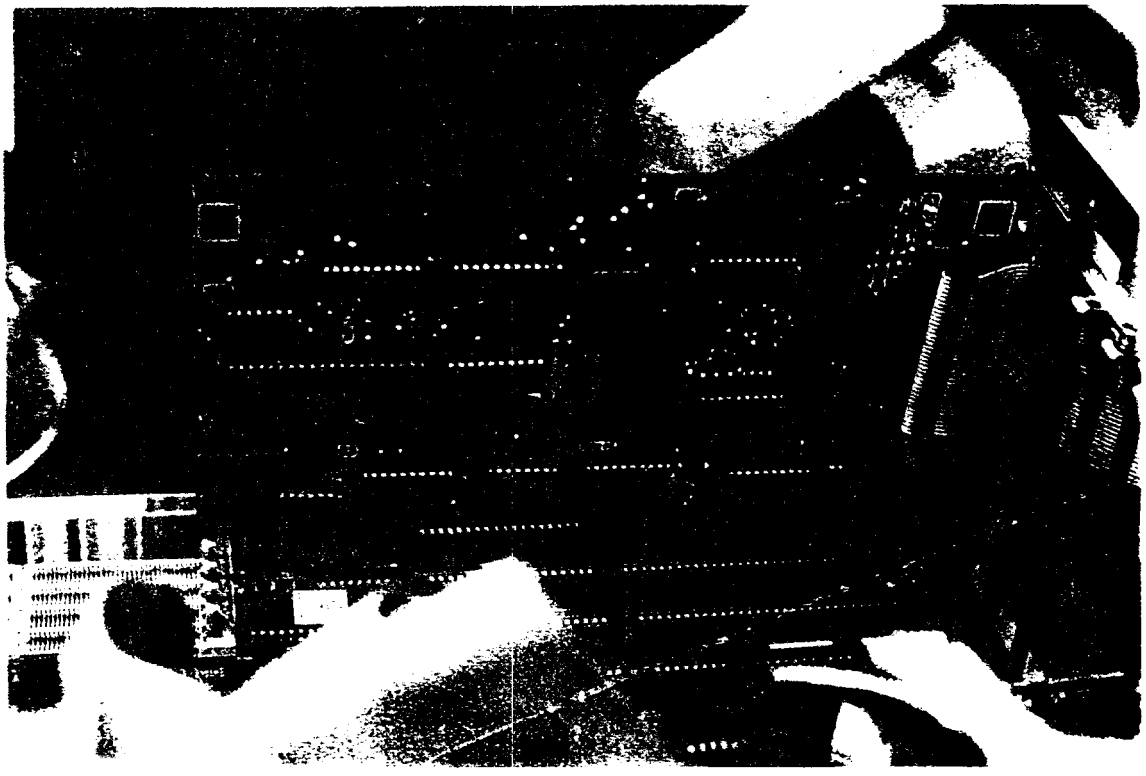
```
switch (dato1)
 {
 case '0' :  {num1=0;
              break;}
 case '1' :  {num1=1;
              break;}
 case '2' :  {num1=2;
              break;}
 case '3' :  {num1=3;
              break;}
 case '4' :  {num1=4;
              break;}
 case '5' :  {num1=5;
              break;}
 case '6' :  {num1=6;
              break;}
 case '7' :  {num1=7;
              break;}
 case '8' :  {num1=8;
              break;}
 case '9' :  {num1=9;
              break;}
 case 'A' :  {num1=10;
              break;}
 case 'B' :  {num1=11;
              break;}
 case 'C' :  {num1=12;
              break;}
 case 'D' :  {num1=13;
              break;}
 case 'E' :  {num1=14;
              break;}
 case 'F' :  {num1=15;
              break;}
 default:;
 }

switch (dato2)
 {
 case '0' :  {num2=0;
              break;}
 case '1' :  {num2=1;
              break;}
 case '2' :  {num2=2;
              break;}
 case '3' :  {num2=3;
              break;}
 case '4' :  {num2=4;
              break;}
 case '5' :  {num2=5;
              break;}
 case '6' :  {num2=6;
              break;}
```

```
 case '7' :  {num2=7;
              break;}
 case '8' :  {num2=8;
              break;}
 case '9' :  {num2=9;
              break;}
 case 'A' :  {num2=10;
              break;}
 case 'B' :  {num2=11;
              break;}
 case 'C' :  {num2=12;
              break;}
 case 'D' :  {num2=13;
              break;}
 case 'E' :  {num2=14;
              break;}
 case 'F' :  {num2=15;
              break;}

 default:;
 }
numerohex=num1*16+num2;
return numerohex;
}

num_datos1(char *in,int conta)

{char dato1, dato2;
 int  numerohex,num1,num2;

 dato1=*(in+conta);
 dato2=*(in+1+conta);

 switch (dato1)
  {
  case '0' :  {num1=0;
               break;}
  case '1' :  {num1=1;
               break;}
  case '2' :  {num1=2;
               break;}
  case '3' :  {num1=3;
               break;}
  case '4' :  {num1=4;
               break;}
  case '5' :  {num1=5;
               break;}
  case '6' :  {num1=6;
               break;}
  case '7' :  {num1=7;
               break;}
  case '8' :  {num1=8;
               break;}
  case '9' :  {num1=9;
```

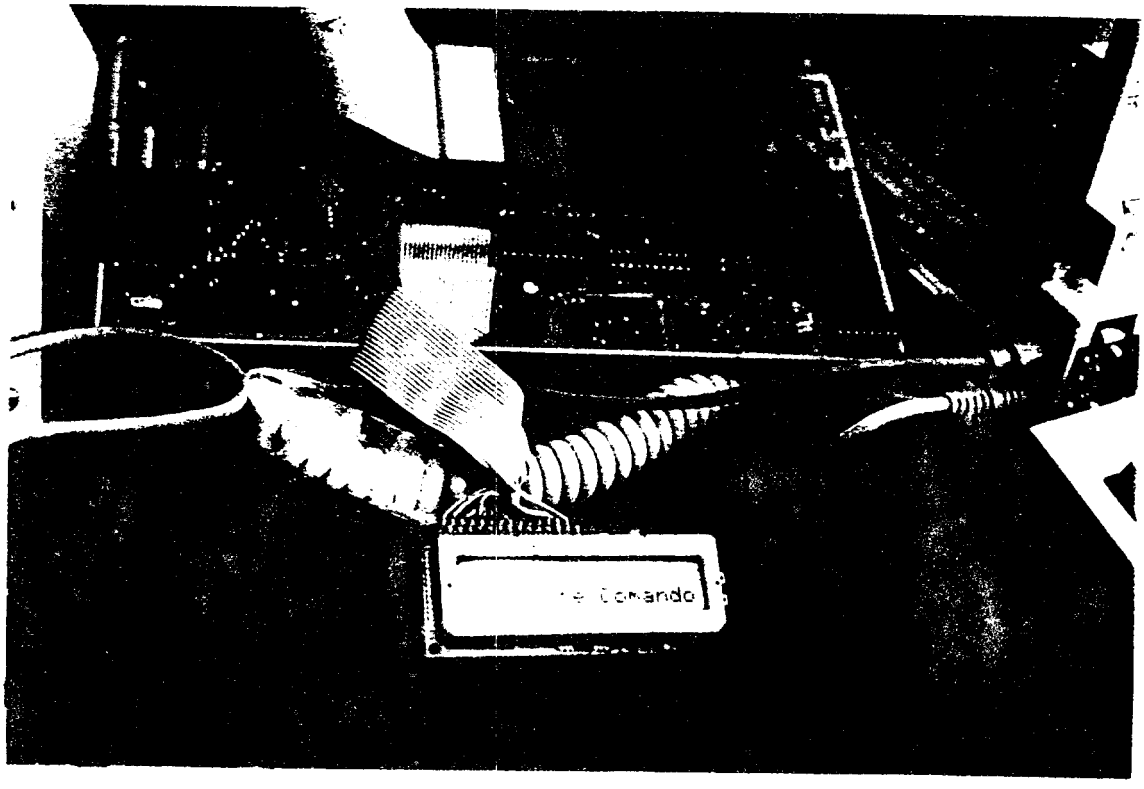
```
                    break;}
 default:;
 }

 if ( (dato1=='a') || (dato1=='A'))
        num1=10;
 if ( (dato1=='b') || (dato1=='B'))
        num1=11;
 if ( (dato1=='c') || (dato1=='C'))
        num1=12;
 if ( (dato1=='d') || (dato1=='D'))
        num1=13;
 if ( (dato1=='e') || (dato1=='E'))
        num1=14;
 if ( (dato1=='f') || (dato1=='F'))
        num1=15;

switch (dato2)
 {
 case '0' :  {num2=0;
                    break;}
 case '1' :  {num2=1;
                    break;}
 case '2' :  {num2=2;
                    break;}
 case '3' :  {num2=3;
                    break;}
 case '4' :  {num2=4;
                    break;}
 case '5' :  {num2=5;
                    break;}
 case '6' :  {num2=6;
                    break;}
 case '7' :  {num2=7;
                    break;}
 case '8' :  {num2=8;
                    break;}
 case '9' :  {num2=9;
                    break;}
 default:;
 }

 if ( (dato2=='a') || (dato2=='A'))
        num2=10;
 if ( (dato2=='b') || (dato2=='B'))
        num2=11;
 if ( (dato2=='c') || (dato2=='C'))
        num2=12;
 if ( (dato2=='d') || (dato2=='D'))
        num2=13;
 if ( (dato2=='e') || (dato2=='E'))
        num2=14;
 if ( (dato2=='f') || (dato2=='F'))
        num2=15;
```

```
 numerohex=num1*16+num2;
 return numerohex;

}


main(int argc,char *argv[])
{ FILE *in,*out;
  char *parametro1,*parametro2;
  char dato;
  char  num_dat,dir_low,dir_hi,offset_hi,offset_low;
  char  tipo_reg;
  int   directotal;
  char  i,datos;
  char  cadena[4];
  int   j;

 parametro1=argv[1];
 parametro2=argv[2];

 in=fopen(parametro1,"rt");
 out=fopen(parametro2,"wb");

 for (j=0;j<=0x1fff;j++)
 putc(0xff,out);


 dato=getc(in);
 if (dato=='S')
 {

 printf("\nIndique el Offset del Archivo en HEXADECIMAL ____");
 gotoxy(wherex()-4,wherey());
 gets(cadena);

 offset_hi= num_datos1(cadena,0);
 offset_low=num_datos1(cadena,2);

 tipo_reg=getc(in);
 num_dat =num_datos(in);
 dir_hi  =num_datos(in);
 dir_low =num_datos(in);

while  (tipo_reg!='9')
{

   _BL=dir_low;
   _BH=dir_hi;
   _AL=offset_low;
   _AH=offset_hi;

   asm sub bx,ax
   directotal=_BX;
```

```
directotal--;

for (i=1; i<=num_dat-3; i++){

  directotal++;
  datos=num_datos(in);

  fseek(out,directotal,SEEK_SET);
  putc(datos,out);


  } //fin del for
 datos=num_datos(in);  // se lee el checksum del archivo
 dato=getc(in);        // se lee el fin de linea
 dato=getc(in);        // se lee el código de nueva linea

 tipo_reg=getc(in);
 num_dat =num_datos(in);
 dir_hi  =num_datos(in);
 dir_low =num_datos(in);

}

} // fin del if principal

fclose(in);
fclose(out);

return 0;
}
```



Simulador conectado en ranura de expansión



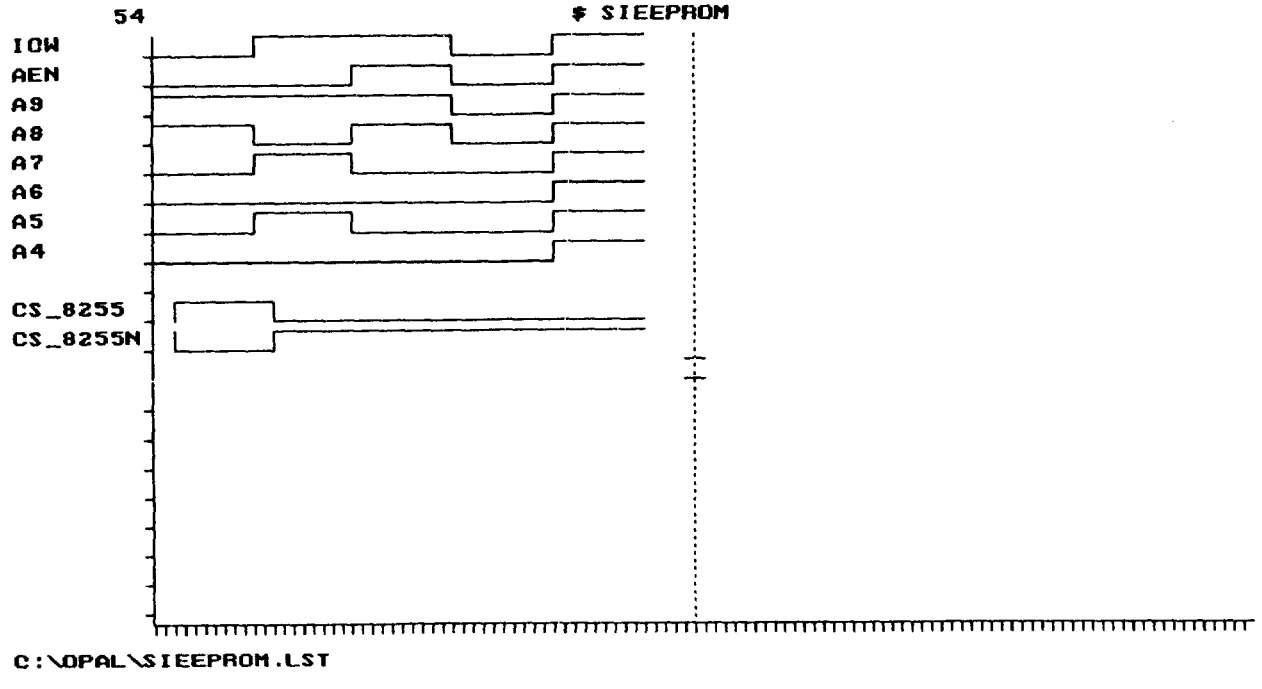
Simulador ejecutando un programa

APENDICE 2

El programa que define las ecuaciones para una GAL16V8 (16 entradas o 8 salidas) sería:

```
BEGIN HEADER
        Programa fuente utilizado para generar la parte de Decodificación del
        Simulador de Memorias Eprom.

END HEADER
BEGIN DEFINITION
 DEVICE          GAL16V8;
 INPUTS          IOW, AEN, A9, A8, A7, A6;
 INPUTS          A5, A4;
OUTPUT    (COM)   CS_8255,CS_8255N;
END DEFINITION
BEGIN EQUATION
  CS_8255  =/IOW*/AEN*/A4*/A5*/A6*/A7*A8*A9;
 /CS_8255N =/IOW*/AEN*/A4*/A5*/A6*/A7*A8*A9;
END EQUATION
BEGIN VECTOR
IOW, AEN, A9, A8, A7, A6, A5, A4;
00110000
10101010
11110000
00000000
11111111

END VECTOR
```

Aquí se notan las 8 entradas (INPUTS IOW', AEN', A9, A8, A7, A6, A5, A4) y se ven dos salidas (OUTPUTS CS_8255,CS_8255N). Se utiliza una sola salida, pero el dispositivo proporciona una adicional negada para algún otro uso no previsto. Con los 5 vectores de prueba sólo la primer condición es válida, de esto se tiene el siguiente diagrama:



El compilador utilizado es el OPAL de National Semiconductors, versión 2.2.

BIBLIOGRAFIA

1. **Brey,B.B.** *The Intel Microprocessors 8086/8088, 80186, 80286, 80386, and 80486. Architecture, programming, and Interfacing.* Third edition. Merril New York 1994.

2. **Byrd, J.S.** Architecture Prentice-Hall. New Jersey 1992.

3. **Ceballos, F. J.**. *Enciclopedia de Visual Basic.* Editorial RA-MA. México, 1991.

4. **Hall, D.V.** *Microprocessors and interfacing. Programming and hardware.* McGraw-Hill New Jersey1986.

5. **Halvorson, M.** *Microsoft Visual Basic 4 para Windows 95.* Mc-Graw Hill. México,1995.

6. **Houlette, F., White, S.E.** *Building OLE Applications with Visual Basic 4.* QUE. California 1995.

7. **Intel Corporation**. *iAPX 86/88, 186/188. User's manual programmer's reference.* California 1986.

8. **Intel Corporation**. *Embedded processors.* California 1994.

9. **León, J.S., Suárez, F.A., Hernández, M.E.** *Tarjeta de experimentación para un microprocesador 801188. Reporte de investigación.* División de Ciencias básicas e ingeniería. UAM-Iztapalapa 1994.

10. **Leventhal, A.L.** *Z80 Assembly Language Programming.* Osborne/McGraw-Hill. Berkeley California 1977.

11. **Norton, P. Socha, J.** *Assembly Language Book for the IBM PC.* Brady New York 1989.

12. **Triebel, W.A. Singh, A.** *The 8088 and 8086 Microprocessors Programming, Interfacing, Software, Hardware and applications.* Prentice-Hall New Jersey. 1997.