ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

EDITOR DE PROGRAMAS PARA LOS MICROCONTROLADORES INTEL MCS – 51/52 QUE INCLUYA VERIFICACION DE SINTAXIS DE LOS MNEMONICOS DE LAS INSTRUCCIONES

WILSON BASANTES SINCHIGUANO

Tesis previa a la obtención del titulo de Ingeniero en Electrónica y Telecomunicaciones

Quito, marzo del 2000

Certifico que el presente trabajo ha sido totalmente realizado por el Sr. Wilson Basantes Sinchiguano

Ing. Jaime Velarde
Director de Tesis

DEDICATORIA.

La presente Tesis, está dedicada muy especialmente a mis padres, quienes han sabido inculcar en mí, el anhelo de la superación basado siempre en la Educación.

AGRADECIMIENTO

Agradezco a los señores profesores de la facultad de Ingeniería Eléctrica de la Escuela Politécnica Nacional que de una u otro forma colaboraron para llevar a cabo el presente proyecto.

Agradezco también a aquellas personas que de una manera desinteresada estuvieron siempre apoyándome en el desarrollo de la presente tesis.

Y de manera muy especial agradezco al Ingeniero Jaime Velarde, Director de Tesis, sin cuya dirección y apoyo este proyecto no habría podido ser realizado.

INTRODUCCION

"La electrónica data de principios del siglo XX y durante sus primeros cincuenta años se desarrolló usando como elemento básico la válvula de vacío. Dos importantes descubrimientos abrieron una nueva etapa a la electrónica, el transistor en 1948 y el circuito integrado en 1960, los cuales constituyeron los eslabones necesarios para inundar de componentes electrónico a todos los productos y procesos del mundo."

Uno de sus mejores frutos y de mayor desarrollo ha sido el microprocesador, cuyo descubrimiento fue accidental, ya que fue un desecho de otras aplicaciones; este elemento, hoy en día es un componente muy importante en la mayoría de los procesos imaginables. Debido a su gran campo de aplicación, los microprocesadores han ido mejorando, tanto en velocidad de procesamiento, capacidad de memoria y otros aspectos que ayudan a realizar sus tareas de forma más eficiente.

Para realizar estas tareas, el microprocesador dispone del conjunto de instrucciones, que ordenadas en forma adecuada producen los programas, los cuales son un complemento a la estructura física que realiza dichas tareas. El desarrollo de estos programas requiere de herramientas computacionales que permitan traducir las instrucciones que el programador las escribe en Mnemónicos, al lenguaje que el microprocesador entiende que es el lenguaje de máquina.

Con este propósito y tomando en cuenta que al momento la Facultad de Ingeniería Eléctrica, no dispone de las herramientas necesarias para realizar este paso en un ambiente en el cual la gran mayoría de personas se halla relacionado, como es el ambiente Windows el cual nos proporciona muchas facilidades de manejo y de fácil adaptación a principiantes, se ha tomado el reto de elaborar dicho sistema.

La presente tesis cuyo título es **"Editor de programas para los microcontroladores INTEL MCS-51/52 que incluye verificación de sintaxis de los mnemónicos de las instrucciones"** es una parte de este sistema, pero de mucha importancia; pues es la que se halla más relacionada con el programador para la escritura de las instrucciones.

Esta nueva herramienta, a más de proporcionar la posibilidad de un manejo sencillo de la edición de los programas, nos dará la oportunidad de corregir la sintaxis de las instrucciones, agilitando la programación para los más expertos y ayudando a darse cuenta de los errores que cometen los principiantes en el proceso de escritura, sin la necesidad de recurrir al ENSAMBLADO y DEPURACIÓN del programa.

INDICE

INTRODUCCION

CAPITULO I

1 CARACTERISTICAS GENERALES DE LOS MICROPROCESADORES

1.1 INTRODUCCIÓN A LOS MICROPROCESADORES 1
1.1.1 Desarrollo de los microprocesadores 1
1.1.2 Definición, importancia y ventajas de utilización de los sistemas basados en microprocesadores 3
1.1.2.1 Definición. 3
1.1.2.2 Importancia. 4
1.1.2.3 Ventajas de utilización 5
1.1.3 Familia de los microcontroladores INTEL MCS-51/52 6
1.1.4 Otros microcontroladores 8
1.2 ARQUITECTURA DE LOS MICROCONTROLADORES INTEL MCS-51/52 8
1.2.1 Arquitectura del microcontrolador INTEL MCS-51/52 8
1.2.1.1 La unidad de control 9
1.2.1.2 La unidad aritmética y lógica 10
1.2.1.3 Bus interno 10
1.2.2 Descripción de terminales. 13
1.2.3 Organización de la memoria 14
1.2.3.1 Memoria de programa 14
1.2.3.2 Memoria de datos 15
1.2.4 Pórticos de E/S, interrupciones y temporizadores. 18
1.2.4.1 Pórticos de entrada/salida paralelos. 18
1.2.4.1.1 Pórtico 0 18
1.2.4.1.2 Pórtico 1 18
1.2.4.1.3 Pórtico 2 18
1.2.4.1.4 Pórtico 3 19
1.2.4.2 Pórtico serial 19
1.2.4.3 Interrupciones. 21
1.2.4.3.1 Permisos y prioridades 22
1.2.4.4 Temporizadores y contadores 23
1.2.4.4.1 Timer 0 y 1 24
1.2.4.4.2 Registros de control del modo de operación de los timers 0y1 24
1.3 SISTEMA DE DESARROLLO PARA MICROCONTROLADORES. 26
1.3.1 Equipo físico o hardware 26
1.3.1.1 Computador personal 27
1.3.1.2 Equipo de pruebas 27
1.3.1.3 Interfaz de comunicaciones 28

1.3.1.4 Programador de memorias 28
1.3.2 Sistema lógico o software 28
1.3.2.1 SIDES a51 30
1.3.2.2 Editor SIDES 30
1.3.2.3 AVMAC51 30
1.3.2.4 AVLINK 31
1.3.2.5 AVSIM51 31
1.3.2.6 PROMPRO8 31
1.3.2.7 SIDES 2000 31

CAPITULO II

2 PROGRAMACION DE LOS MICROCONTROLADORES INTEL MCS-51/52

2.1 FORMATO DE LAS INSTRUCCIONES 32
2.1.1 Elaboración de un programa 32
2.1.1.1 Definición del problema 32
2.1.1.2 Elaboración del Algoritmo 33
2.1.1.3 Escritura del programa en mnemónicos 33
2.1.2 Ciclo de una instrucción 34
2.1.2.1 Ciclo de traída de la instrucción 34
2.1.2.2 Ciclo de ejecución de la instrucción 35
2.2 MODOS DE DIRECCIONAMIENTO. 35
2.2.1 Direccionamiento Inmediato 35
2.2.2 Direccionamiento a los Registros. 35
2.2.3 Direccionamiento directo. 35
2.2.4 Direccionamiento indirecto 36
2.2.5 Direccionamiento indexado. 36
2.3 JUEGO DE INSTRUCCIONES 36
2.3.1 Instrucciones de Transferencia de Datos. 38
2.3.2 Instrucciones de Operaciones Aritméticas. 39
2.3.3 Instrucciones de Operaciones Lógicas 40.
2.3.4 Instrucciones de Control de Flujo del Programa. 41
2.3.5 Instrucciones para manipulación de Bits Específicos. 42
2.4 DESARROLLO DE LOS PROGRAMAS EN ASSEMBLER 43
2.4.1 Sintaxis de las Instrucciones y operandos. 43
2.4.1.1 Instrucciones 43
2.4.1.2 Ejemplos de Instrucciones 45
2.4.1.3 Identificadores y símbolos 45
2.4.1.4 Números 46
2.4.1.5 Caracteres constantes 47
2.4.1.6 Referencias del Contador de localidad 47

2.4.1.7 Expresiones Aritméticas. 48
2.4.2 Pseudo Opcodes del Ensamblador 49
2.4.2.1 END – El simple pseudo Opcode 49
2.4.2.2 EQU y TEQ – Igualdades 49
2.4.2.3 DS – Definir espacio. 50
2.4.2.4 DB – Definir bytes. 50
2.4.2.5 DW – Definir palabra. 51
2.4.2.6 PROC, ENDPROC y Símbolos locales. 51
2.4.2.7 INCLUDE – Otros modos para incluir archivos. 52
2.4.2.8 SEG y Segmentos 52
2.4.2.9 DEFSEG – Definición de Segmentos 53
2.4.2.10 ORG – Movimiento entre Segmentos. 54
2.4.2.11 PUBLIC Símbolos y EXTERN 54
2.4.3 Opciones del Ensamblador. 56

CAPITULO III

3 DESARROLLO DEL EDITOR INTELIGENTE
3.1 INTRODUCCIÓN A LA PROGRAMACIÓN EN VISUAL BASIC 57
3.1.1 Creación de la Interfaz. 57
3.1.2 Definición de las propiedades. 57
3.1.3 Escritura del Código. 58
3.1.4 Creación del EDITOR INTELIGENTE. 59
3.2 DIAGRAMAS DE FLUJO DEL PROGRAMA. 60
3.2.1 Diagrama de flujo Principal. 60
3.2.2 Diagrama de flujo Menú Archivo. 62
3.2.3 Diagrama de flujo Menú Edición 63
3.2.4 Diagrama de flujo Menú Ver 64
3.2.5 Diagrama de flujo Menú Insertar. 65
3.2.6 Diagrama de flujo Menú Herramientas. 66
3.2.7 Diagrama de flujo Corrección del Programa 68
3.2.8 Diagrama de flujo Corrección de Instrucción 69
3.2.9 Diagrama de flujo Ayuda 70
3.3 DESCRIPCIÓN Y DESARROLLO DEL PROGRAMA. 71
3.3.1 Descripción del programa 71
3.3.2 Desarrollo del programa. 74

CAPITULO IV

4 PRUEBAS Y RESULTADOS
4.1 EJEMPLOS DE PROGRAMACIÓN. 78
4.1.1 Ejemplo 1 (Programa sin Errores) 78
4.1.2 Ejemplo 2 (Programa con Errores) 79

4.1.3 Ejemplo 3 (Ingreso de una Instrucción) 80
4.2 MENSAJES Y CORRECCIÓN DE ERRORES 85
4.2.1 Mensajes de Error 85
4.2.1.1 Mensajes de Error en la Barra de Estado 85
4.2.1.2 Mensajes de Error al terminar una Instrucción 87
4.2.2 Corrección de Errores 89
4.3 OTRAS AYUDAS DEL EDITOR 91
4.3.1 Buscar y Reemplazar 91
4.3.2 Insertar Caracteres. 92
4.3.3 Opciones de programación. 93

CAPITULO V

5 CONCLUSIONES Y RECOMENDACIONES
5.1 CONCLUSIONES 95
5.2 RECOMENDACIONES. 96

6 ANEXOS
A Opciones del AVMAC51
B Manual del Usuario.
C Listado del programa.
D Base de datos del Editor inteligente.

7 BIBLIOGRAFIA

CAPITULO I

1 CARACTERISTICAS GENERALES DE LOS MICROPROCESADORES

1.1 INTRODUCCION A LOS MICROPROCESADORES

1.1.1 DESARROLLO DE LOS MICROPROCESADORES

Antes de entender a los microprocesadores modernos, se debe comprender primero que fue lo que paso con estos dispositivos y el punto en que se hallan actualmente.

Los antiguos babilonios empezaron a usar el **ábaco** (calculadora primitiva) alrededor del año 500 A.C. Con el tiempo esta calculadora sencilla motivo a la humanidad a desarrollar nuevas máquinas, una de ellas, la cual utilizaba engranajes y ruedas fue construida por Blas Pascal en 1642. Se continuó el progreso con las gigantescas maquinas computadoras de la década de 1940 y 1950, construidas con relevadores y tubos de vacío.

En la década de 1960 con el descubrimiento de los transistores y los componentes electrónicos de estado sólido se construyeron las poderosas computadoras.

Con el advenimiento de los circuitos integrados, se llegó al perfeccionamiento del microprocesador y de las microcomputadoras.

En 1971 INTEL desarrollo el **primer microprocesador: el 4004** que manejaba palabras o datos de 4 bits; este fue creado por petición de una compañía japonesa manufacturera de calculadoras electrónicas, llamada BUSICOM. En principio INTEL vende el 4004 en forma exclusiva a BUSICOM, pero a mediados de 1971 gana el derecho de vender los chips a otras firmas manufactureras.

Más tarde al ver que el microprocesador era un producto viable para la comercialización, INTEL produjo **el 8008, el primer microprocesador de 8 bits**; el

cual tenia un mayor tamaño de memoria y nuevas instrucciones que brindaron la oportunidad de muchas aplicaciones de mayor complejidad. Conforme se desarrollaban usos más demandantes para el microprocesador, el 8008 pronto limitó su utilidad. Es así que en 1973 INTEL introdujo el **8080**. El 8080 direccionaba más en la memoria, ejecutaba más instrucciones y era 10 veces más rápido que el 8008, pero principalmente era compatible con la tecnología TTL.

Otras empresas lanzaron su propia versión de microprocesadores de 4 y 8 bits, entre ellos podemos nombrar a: **F-8** de FAIRCHILD, **6502** de MOS TECHNOLOGY, **MC6800** de MOTOROLA, **IMP-8** de NATIONAL SEMICONDUCTOR y **PPS-8** de ROCKWELL INTERNATIONAL.

INTEL introdujo en 1977 una nueva versión del 8080, el **8085**, el cual fue ligeramente más avanzado que su similar anterior; pero su principal ventaja es que el generador de reloj y el controlador del sistema que eran componentes externos en el 8080; en esta versión fueron integrados, lo que produjo un récord de ventas. Todas estas características han hecho del 8085 uno de los microprocesadores más conocidos.

Otras empresas también crearon microprocesadores similares bajo licencia de INTEL, entre ellas tenemos a: NEC, TOSHIBA, HITACHI, AMD.

A partir de estos sucesos las distintas casas fabricantes comenzaron a desarrollar los denominados *"microcomputadoras en un solo chip"* o más comúnmente conocidos como **microcontroladores**. Estos circuitos integrados se caracterizaron por tener incorporado en un solo chip al microprocesador, a la memoria y a los pórticos de entrada/salida de datos.

Entre los microcontroladores más destacados podemos mencionar:

- La familia **MCS-48, MCS-51** y **MCS-52** de INTEL.
- Los microcontroladores **6801, 6803, 6805** de MOTOROLA.
- La familia del **Z8** de ZILOG.

En 1978 INTEL lanza el microprocesador **8086** y un año más tarde el **8088**, los cuales son **microprocesadores de 16 bits**; ofrecían mejoras en la velocidad de

ejecución y mayor capacidad de direccionamiento, lo que permitió sustituir a microcomputadoras pequeñas por estos en muchas aplicaciones.
La evolución del microprocesador de 16 bits no terminó en el 8086 y 8088 ya que continuó con la introducción del **80186**; una versión altamente integrada del 8086.
El **80286** es una versión mejorada del 8086 con una unidad administradora de memoria y mejoramiento también en la velocidad del reloj.
Otras versiones de microprocesadores de 16 bits fueron puestas en el mercado, entre ellas tenemos: **MC68000** de MOTOROLA, **Z800** de ZILOG, **16032** de NATIONAL SEMICONDUCTORS.

Las versiones de **microprocesadores de 32 bits** son: el **80386** y el **80486**.
El 80386 fue el primer microprocesador de 32 bits cuya principal ventaja fue su frecuencia de reloj mucho más alta como también su mayor tamaño de memoria.
El microprocesador 80486 contiene básicamente un 80386 mejorado, un coprocesador aritmético y una memoria cache interna.

Finalmente ingresó en el mercado en 1993 el microprocesador PENTIUM de INTEL, el cual es un **microprocesador de 64 bits**, doble vía de acceso para procesamiento superescalar que permite ejecutar más instrucciones por ciclo de reloj, doble memoria cache y un sistema de predicción de decisiones.
En la actualidad existen microprocesadores PENTIUM II y PENTIUM III con mayores capacidades que su versión anterior tanto en velocidad, capacidad de memoria y manejo de sistemas Multimedia con tendencia a mejorar estos elementos.

1.1.2 DEFINICIÓN, IMPORTANCIA Y VENTAJAS DE UTILIZACIÓN DE LOS SISTEMAS BASADOS EN MICROPROCESADORES.

1.1.2.1 DEFINICION.

El microprocesador es un circuito integrado, el cual contiene la parte más importante de la Unidad Central de Procesamiento de un computador digital. Según el tipo de

modelo varía la estructura interna, pero la mayoría contiene la Unidad de Control y la Unidad Aritmética y Lógica.

En la figura 1.1 podemos observar en forma de diagrama de bloques la estructura del microprocesador.

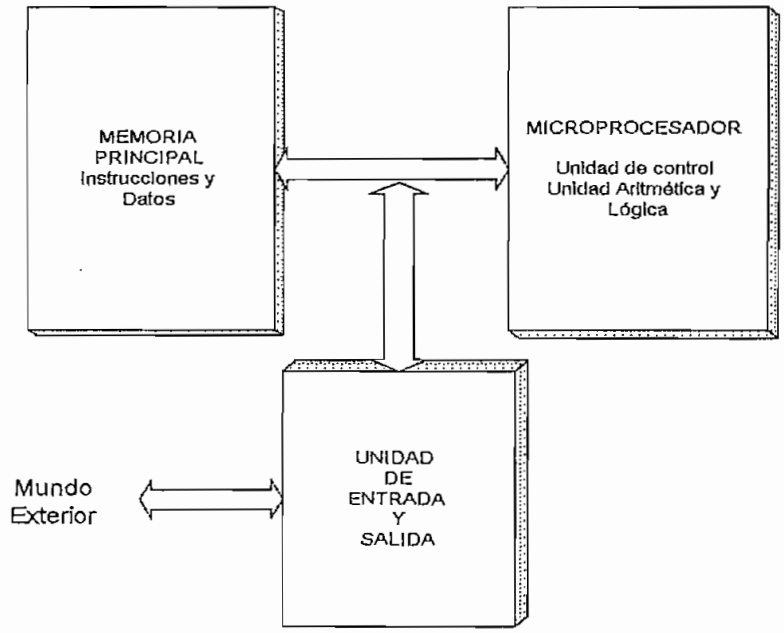

Figura 1.1 La mayor parte de los microprocesadores consisten en un circuito que contiene la U.C. y la A.L.U.

1.1.2.2 IMPORTANCIA.

- La aplicación más importante es la utilización del microprocesador para construir computadoras digitales con muy pocos circuitos integrados, esto hace que los costos cada vez sean menores y puedan ser adquiridos por empresas, tanto grandes como pequeñas, e inclusive para utilización en el hogar; pues la computadora puede ser utilizada en un muchas aplicaciones como pueden ser:

  En la empresa:

  - Llevar la contabilidad con mayor eficiencia.
  - Tener bases de datos de empleados, productos y bienes disponibles.
  - Comunicarse entre los distintos departamentos con mayor agilidad.
  - Realizar documentos de mejor presentación.
  - Otras tareas que dependen de cada tipo de empresa.

En el hogar:

- Organizar y programar tareas.
- Llevar la contabilidad de ingresos y egresos de la familia.
- Producir documentos del tipo que se requiera.
- Y otras aplicaciones.

- La posibilidad de controlar productos de uso corriente como un horno de cocina, el teléfono, o la máquina de escribir, entre otros; con todas las prestaciones que nos ofrecen, han hecho que el microprocesador tenga un gran desarrollo y una vital importancia.

- En la industria, el microprocesador ha permitido una rápida automatización de procesos, lo cual conlleva a la disminución de costos de operación y mantenimiento, como también mejorar la calidad de la producción.

1.1.2.3 VENTAJAS DE UTILIZACION

El desarrollo de aplicaciones con microprocesadores nos proporciona muchas ventajas entre las que podemos enunciar:

- *Reducción del costo del equipo diseñado,* ya que el número de circuitos integrados es menor, lo cual implica la reducción del volumen del equipo como también reduce el consumo de energía.

- *Reducción del tiempo de desarrollo del diseño,* ya que los sistemas pueden crearse en forma modular, la estructura de la información puede ser más desarrollada y los dispositivos de entrada/salida muy diversos.

- *Incremento de la confiabilidad del equipo diseñado,* dado su menor número de elementos utilizados, así como también la utilización de programas de auto diagnóstico, ayuda a la detección de fallas, mantenimiento y reparación del sistema.

- *Incremento de la flexibilidad del diseño;* las modificaciones, incremento de opciones y expansiones futuras pueden hacerse únicamente modificando el programa en la memoria.

Podemos considerar como desventaja la necesidad de disponer de un computador personal para realizar los diseños, pero las ventajas enunciadas anteriormente compensan mucho más este problema.

1.1.3 FAMILIA DE LOS MICROCONTROLADORES INTEL MCS-51/52

Los tres elementos básicos de esta familia son los microcontroladores **8031**, **8051** y **8751**, los cuales tienen una configuración muy similar.

- **8031**: No incluye memoria ROM interna, lo que lo hace útil para desarrollo de prototipos.
- **8051**: Incluye 4 KB de memoria ROM, solo puede programarse en fabrica en el proceso de construcción y configuración, reduciendo su empleo a las aplicaciones con programas fijos
- **8751**: En lugar de los 4 KB de memoria ROM interna dispone de la misma capacidad de memoria EPROM lo que permite que pueda ser programada por el usuario directamente.

Los tres microcontroladores citados están fabricados con tecnología HMOS, pero también existen en el mercado con tecnología CHMOS, entre ellos tenemos: **80C31**, **80C51** y **87C51**. La tecnología CHMOS proporciona mayor velocidad, menor consumo de energía y mayor densidad de integración lo que los hace útiles para equipos portátiles. Sus características son similares a los integrados de tecnología TTL.

La familia MCS-51/52 se complementa con una subfamilia muy interesante para determinadas aplicaciones. Esta subfamilia es **MCS-52** la cual esta compuesta por

tres componentes fundamentales: **8032, 8052** y **8752**. Estos tienen un contador adicional, mayor capacidad de memoria de programa y de datos.

- **8032**: No tiene memoria ROM.
- **8052**: Soporta 8 KB de memoria ROM.
- **8752**: Tiene memoria EPROM en lugar de la memoria ROM.

Las versiones de estos elementos fabricados en tecnología CHMOS acaban sus nombres en FA y FB. según la capacidad de memoria, esto es FA tiene 8Kbytes y FB tiene 16Kbytes.

Con el transcurso del tiempo se ha desarrollado nuevos microcontroladores dentro de esta familia, estos nuevos microcontroladores tienen mejoras tanto en la capacidad de memoria, como también incremento en el número de sus pórticos.
Uno de estos microcontroladores es el 80C51 desarrollado por PHILIPS SEMICONDUCTORS; este proporciona 64 KBytes de memoria, tanto de programa como de datos. El 80C51MX puede soportar mayor capacidad de memoria que la mencionada anteriormente.
Otra subfamilia de microcontroladores es el 51LPC, estos son microcontroladores de 8 bits de datos, los cuales incluyen un convertidor análogo – digital en el chip.
Para una mayor información sobre las nuevas familias de microcontroladores de 8 bits puede accederse a la pagina web: http://www.semiconductors.com

INTEL no es el único fabricante de microcontroladores de esta familia, como puede verse en el párrafo anterior, donde se hace una pequeña descripción de dos microcontroladores con mejoras en sus características, fabricados por PHILIPS. Otros fabricantes como: SIEMENS, DALLAS han desarrollado nuevos microcontroladores dentro de este grupo.

1.1.4 OTROS MICROCONTROLADORES

Los microcontroladores de la familia MCS – 51/52, no son los únicos existentes en el mercado. Existe también microcontroladores con características diferentes, esto es, su conjunto de instrucciones distintas, mejoras o cambios en su arquitectura, pero en definitiva su estructura funcional y su propósito de aplicación es el mismo.

Uno de estos microcontroladores son los denominados PIC; en la actualidad tienen gran aceptación, debido a su bajo costo y sencillez de manejo. Existen muchas versiones diferentes, aumentando su listado cada año; La gama más pequeña está formada por 8 pines, pasando a la gama básica con instrucciones de 12 bits, la gama media con instrucciones de 14 bits y la gama alta con instrucciones de 16 bits y 40 pines.

1.2 ARQUITECTURA DE LOS MICROCONTROLADORES INTEL MCS-51/52

1.2.1 ARQUITECTURA DEL MICROCONTROLADOR INTEL MCS-51/52

La unidad de control y la Unidad Aritmética y Lógica requieren para su funcionamiento de varios ***registros*** que a más de un bus interno conforman la arquitectura de un microprocesador. En la figura 1.2 se puede observar la arquitectura de un microprocesador.

Los registros.- Son elementos que pueden retener datos temporalmente. Estos pueden ser: de propósito general o de funciones especiales.

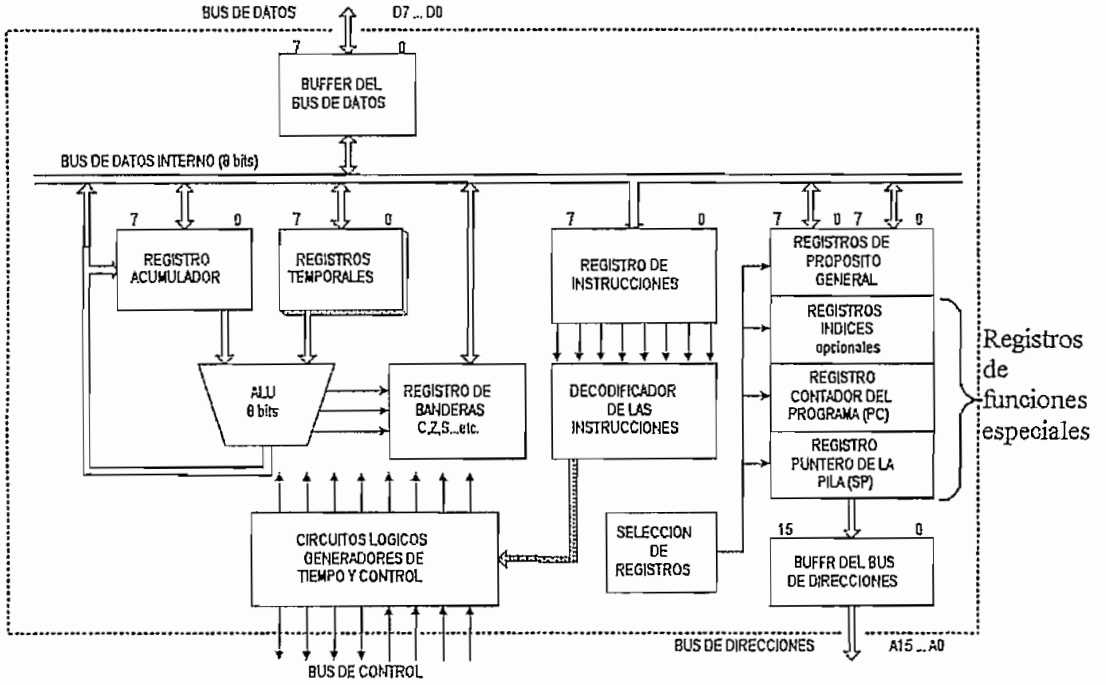

Figura 1.2 Arquitectura de un microprocesador de 8 bits.

1.2.1.1 LA UNIDAD DE CONTROL

Esta formado por: el registro de instrucciones, el decodificador de instrucciones y los circuitos lógicos generadores de tiempo y control.

- **Registro de instrucciones.-** En este registro se almacena el "código de la operación" que se está ejecutando.
- **Decodificador de instrucciones.-** Se encarga de interpretar la instrucción que se encuentra en el registro de instrucciones.
- **Circuitos lógicos generadores de tiempo y control.-** Son circuitos encargados de generar las señales internas y externas que permiten ejecutar las operaciones. También a estos circuitos les llegan señales externas como: inicialización, de espera, de interrupción, etc. Estos circuitos se hallan relacionados con el bus de control.

1.2.1.2 LA UNIDAD ARITMETICA Y LOGICA

Para su funcionamiento se asocia a los registros: acumulador, registros temporales y registros de banderas.

- **Acumulador.-** Es el registro base para las operaciones aritméticas y lógicas; puede contener operandos (antes de la operación) como también resultados (después de la operación). Es un registro fuente o destino de la transferencia de datos con la unidad de memoria y el sistema E/S.
- **Registro Temporal.-** Llamado también *registro auxiliar* sirve para almacenar momentáneamente información. Son transparentes al usuario.
- **Registro de Banderas.-** Formado por un conjunto de biestables, los cuales nos indican ciertas condiciones del resultado; como por ejemplo: el valor del signo, el carry, etc.

1.2.1.3 BUS INTERNO

Es el bus de comunicaciones entre los componentes del microprocesador, el número de líneas determina el número de bits que procesa.

Luego de conocer los elementos constitutivos del microprocesador, podemos observar en forma clara la Arquitectura del microcontrolador.

En la figura 1.3 puede observarse el diagrama de bloques de los elementos que conforman los microcontroladores de la familia INTEL MCS-51/52. En el desarrollo del capítulo se estudiará a los estos elementos.

A más de los registros que ya se señalaron para los microprocesadores; tenemos los siguientes:

Registro B.- Es utilizado principalmente en las instrucciones de multiplicación y división.

Puntero del STACK (SP).- Es un registro de 8 bits que se incrementa / decrementa con la ejecución de las instrucciones PUSH y POP.

Puntero de Datos (DPTR).- Está formado por un byte más significativo (DPTH) y un menos significativo (DPTL). Contiene una dirección de 16 bits para acceso a memoria RAM o ROM externa.

Palabra de Estado del Programa(PSW).- Es equivalente al registro de Banderas, contiene información generada por la ejecución de las instrucciones del programa. En la tabla de la figura 1.4 estas banderas, con su etiqueta y significado.

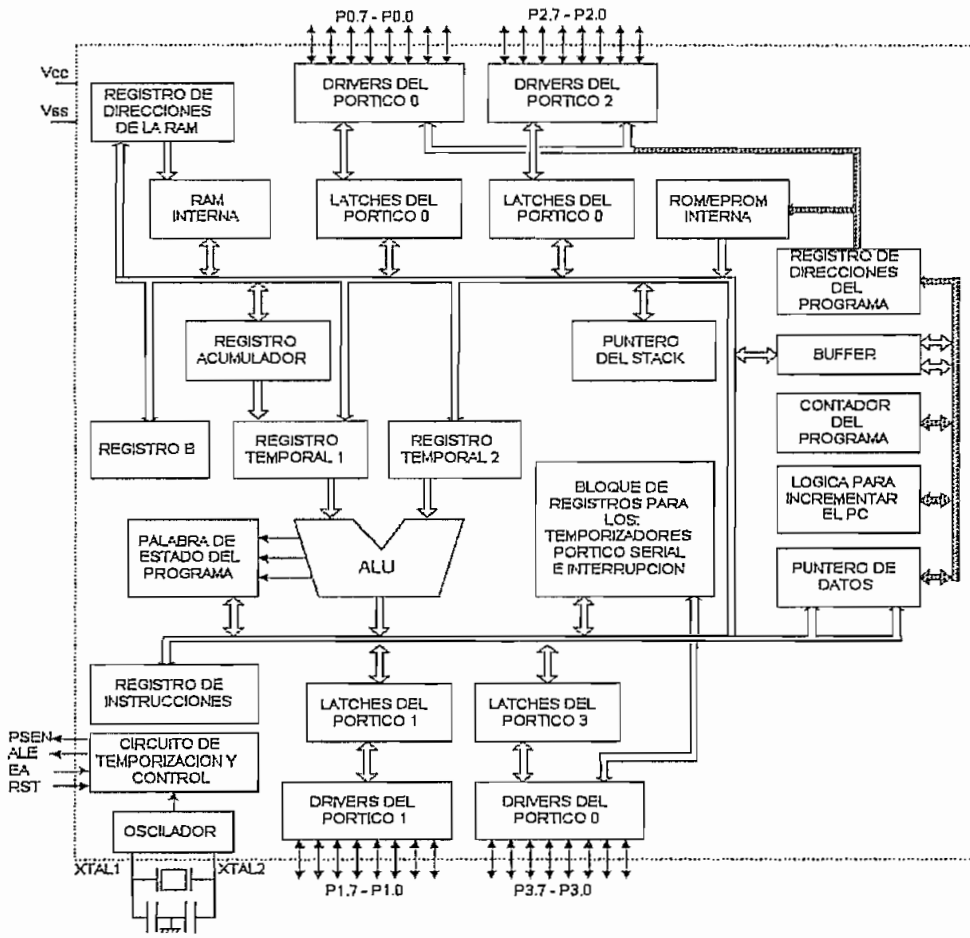

Figura 1.3 Diagrama de bloques de los Microcontroladores INTEL MCS-51/52

Etiqueta	Posición en PSW	Dirección como bit	Significado
CY	PSW.7	D7H	Bandera del Carry
AC	PSW.6	D6H	Bandera del Carry Auxiliar.
F0	PSW.5	D5H	Bandera Cero.
RS1 RS0	PSW.4 PSW.3	D4H D3H	Bits para seleccionar el banco de registros. *RS1* *RS2* *Banco Activo* 0 0 0 0 1 1 1 0 2 1 1 3
OV	PSW.2	D2H	Bandera de desbordamiento.
-	PSW.1	D1H	Reservado.
P	PSW.0	D0H	Bandera de paridad.

Figura 1.4 Tabla del significado de la Palabra de Estado del Programa (PSW).

En particular los microcontroladores de la familia INTEL MCS-51 contienen los siguientes elementos:

- CPU de 8 bits.
- RAM de 128 bytes.
- ROM de 4 KB.
- 4 puertos de E/S de 8 líneas cada uno.
- 1 puerto de E/S serie.
- 2 contadores/temporizadores de 16 bits programables.
- 1 oscilador para las señales de reloj.

Los microcontroladores de la familia MCS-52 constan de:

- CPU de 8 bits.
- RAM de 256 bytes.
- ROM de 8 KB.
- 4 puertos de E/S de 8 líneas cada uno.
- 1 puerto de E/S serie.
- 3 contadores/temporizadores de 16 bits programables.
- 1 oscilador para las señales de reloj.

1.2.2 DESCRIPCION DE TERMINALES.

Con 40 pines, el microcontrolador MCS-51/52 soporta numerosas funciones, lo que obliga a cada pin a controlar dos funciones diferentes.

A continuación se describirá las funciones de los pines.

- **4 puertos de E/S paralelos** (32 líneas bidireccionales). Se describe en forma más detallada la función de cada uno de ellos mas adelante.
- VCC. Voltaje de alimentación (+5 voltios.).
- VSS. Conexión a tierra.
- RST. (*Reset*) Reinicialización del sistema. Se lo activa con un nivel alto por un tiempo de dos ciclos de reloj.
- XTAL1. Entrada al inversor del oscilador de las señales de reloj.
- XTAL2. Salida del inversor del amplificador del oscilador. En estas dos entradas se puede colocar un cristal de cuarzo.
- ALE/PROG#. *ALE* saca un pulso que sirve para cargar el byte bajo de las direcciones durante un acceso a memoria externa.

  PROG# (el símbolo # significa que la señal es activado a nivel bajo) se usa en el proceso de la grabación de la EPROM en aquellos modelos que poseen este tipo de memoria y actúan como entrada de los impulsos de grabación.
- EA#/VPP. *VPP*. Ejecuta programas de la ROM interna, a menos que el contador de programas (PC) exceda de la dirección 1FFFH en el 8052 y 0FFFH en el 8051.

  EA#. Ejecuta el programa de la memoria externa.
- PSEN#. Esta señal sirve para leer de la memoria externa del programa. No se activa cuando se está ejecutando el programa de la ROM o EPROM interna.

En el figura 1.5 podemos observar un esquema del microcontrolador con sus terminales y la función de cada uno de ellos.

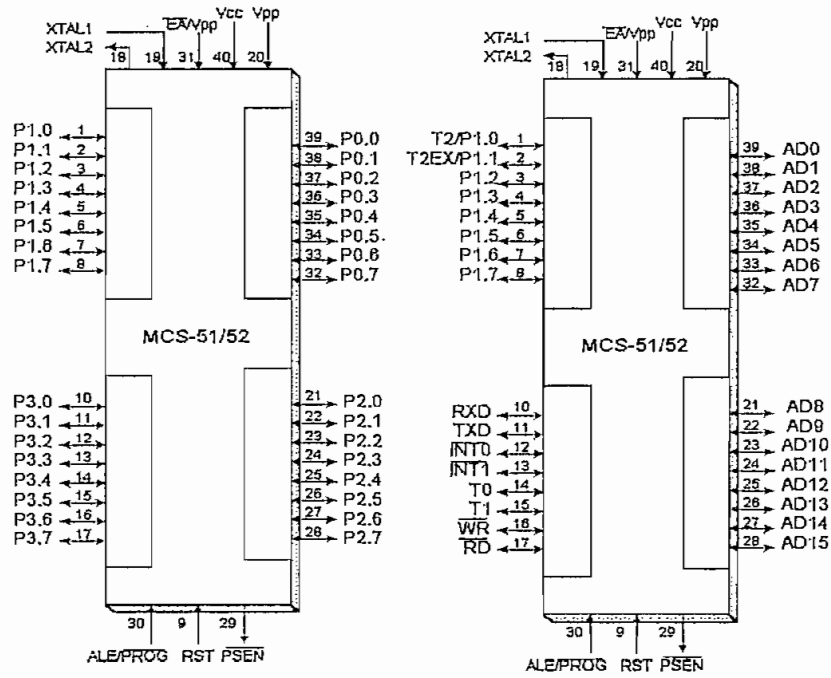

Figura 1.5 Esquema de las alternativas del microprocesador MCS-51/52 con la asignación de funciones de sus terminales
a) Asignación de los pórticos b) Función alternativa de los pines

1.2.3 ORGANIZACIÓN DE LA MEMORIA

La Unidad de memoria está implementada con memorias electrónicas que se clasifican en dos grupos:

1.2.3.1 MEMORIA DE PROGRAMA.

Constituye el espacio de memoria para el programa; es de solo lectura y de acceso aleatorio por lo que se la denomina memoria ROM. El programa se ejecuta si se mantiene la señal EA en nivel alto, de esta forma se accede a la memoria ROM interna; si AE se halla en nivel bajo ó si el contador de programa (PC) contiene un dato que excede a la dirección máxima de la ROM interna, la CPU trae las instrucciones desde la memoria ROM externa.

1.2.3.2 MEMORIA DE DATOS.

El contenido de sus posiciones puede ser leído y escrito; también son de acceso aleatorio, con el inconveniente de ser volátil, se la denomina memoria RAM.

Pueden desarrollarse sistemas que posean RAM externa de hasta 64 KB.

Para la serie 51 la RAM interna se divide en dos partes:

- 128 bytes bajos para datos.
- 128 bytes altos para el *Registro de Funciones Especiales* (SFR).

Para la serie 52 existe un tercer bloque:

- 128 bytes altos para una segunda área de datos.

En la figura 1.6 se observa el mapa de memoria de Datos y del Programa de los microcontroladores INTEL MCS-51/52.

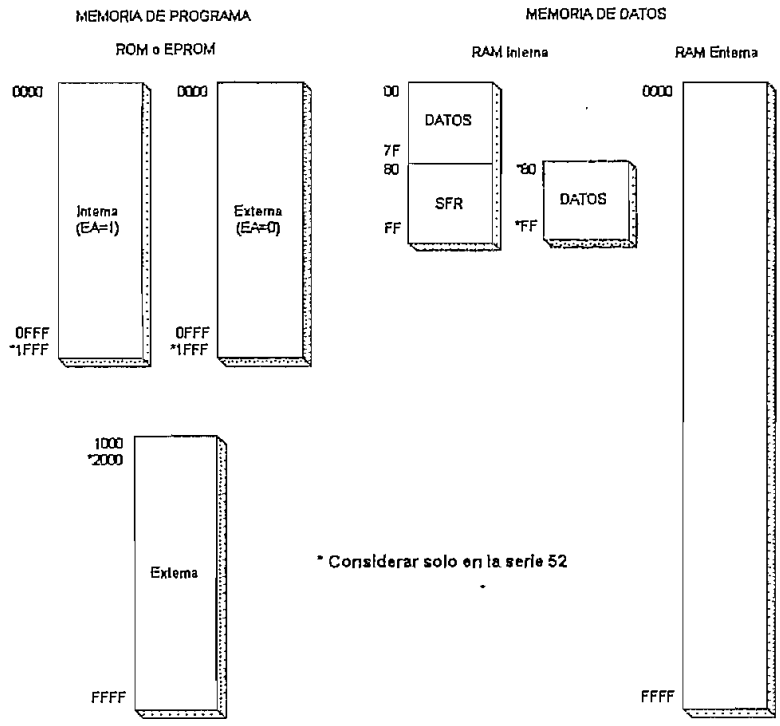

Figura 1.6 Mapa de memoria de los microcontroladores MCS-51/52

MEMORIA PARA DATOS DE LA RAM INTERNA.

- Las 32 primeras localidades (entre 00H y1FH) están ocupadas por 4 bancos de 8 registros cada uno (R0... R7).
- Las 16 localidades siguientes (entre 20H y 2FH) contienen 128 bits direccionables independientemente.
- Las restantes 80 localidades (entre 30H y 7H) son el área para datos incluido el STACK.

En la figura 1.7 puede observarse la distribución de estas localidades en un mapa de memoria.

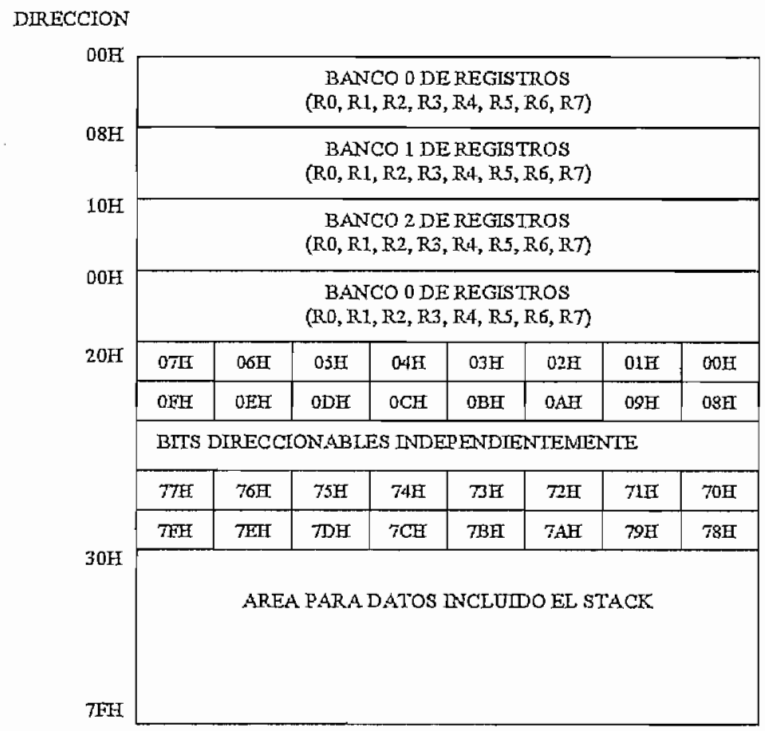

Figura 1.7 Localidades de la RAM interna entre 00H y 7FH

REGISTRO DE FUNCIONES ESPECIALES (SFR).

Todos los registros excepto el PC y los 4 bancos de registros se encuentran en el SFR. A continuación se muestra en la tabla de la figura 1.8, la etiqueta, el nombre y la dirección correspondiente a cada registro.

ETIQUETA	NOMBRE	DIRECCION
+ACC	Acumulador.	E0H
+B	Registro B.	F0H
+PSW	Palabra de Estado del Programa.	D0H
SP	Puntero del STACK.	81H
DPTR	Puntero de Datos.	82H
DPL	Byte bajo del Puntero de Datos.	82H
DPH	Byte alto del Puntero de Datos.	83H
+P0	Pórtico 0	80H
+P1	Pórtico 1	90H
+P2	Pórtico 2	A0H
+P3	Pórtico 3	B0H
+IE	Control de la habilitación de las Interrupciones.	A8H
+IP	Control de la prioridad de las Interrupciones.	B8H
+TCOM	Control de operación de los Temporizadores/Contadores 0 y 1.	88H
TMOD	Control del modo de los Temporizadores/Contadores 0 y1.	89H
*+T2COM	Control del Temporizador/Contador 2.	C8H
TL0	Byte bajo del Temporizador/Contador 0	8ªH
TH0	Byte alto del Temporizador/Contador 0	8CH
TL1	Byte bajo del Temporizador/Contador 1	8BH
TH1	Byte alto del Temporizador/Contador 1	8DH
*TL2	Byte bajo del Temporizador/Contador 2	CCH
*TH2	Byte alto del Temporizador/Contador 2	CDH
*RCAP2L	Byte bajo de Captura del Temporizador/Contador 2	CAH
*RCAP2H	Byte alto de Captura del Temporizador/Contador 2	CBH
+SCON	Control del Pórtico Serial.	89H
SBUF	Buffer de datos del Pórtico Serial.	99H
PCON	Control del consumo de Potencia para los CHMOS.	87H

*Disponible solo en la serie del 52
+Registros que pueden ser accesados como bytes o como bits.

Figura 1.8 Tabla de los Registros de Funciones Especiales.

1.2.4 PORTICOS DE E/S, INTERRUPCIONES Y TEMPORIZADORES.

1.2.4.1 PORTICOS DE ENTRADA/SALIDA PARALELOS.

Llamados también puertos, son elementos mediante los cuales el microcontrolador se comunica con el mundo exterior. Se denominan de entrada/salida paralelos ya que permitan el ingreso o la salida de varios bits de información al mismo tiempo.

La familia INTEL MCS-51/52 poseen 4 pórticos de E/S paralelos, cada uno de 8 bits. Alguno de estos pórticos poseen una segunda función; los bits pueden ser direccionados independientemente.

1.2.4.1.1 PORTICO 0

Consta de 8 líneas de E/S bidireccionales que soportan hasta 8 cargas TTL LS.

Cumple una doble función: a mas de ser el Bus de Datos Externo (D0 – D7), es el byte menos significativo del Bus de Direcciones Externo (AD0 – AD7). Esta doble función la cumple multiplexado en el tiempo y sincronizado con las salidas PSEN (Program Store Enable) y ALE (Addres Latch Enable) del Bus de Control.

1.2.4.1.2 PORTICO 1

Se trata de un puerto de E/S bidireccional capaz de soportar en sus líneas hasta 4 cargas TTL LS. No posee ninguna función alterna, excepto en los microcontroladores de la serie 52 donde los dos bits menos significativos P1.0 y P1.1 si poseen una segunda función.

1.2.4.1.3 PORTICO 2

Es un puerto de E/S paralelo, cuyas líneas soportan hasta 4 cargas TTL LS.

Tiene como función alterna la de ser el byte más significativo del Bus de Direcciones (A8–A15). Su funcionamiento también está sincronizado con las salidas PSEN y ALE.

1.2.4.1.4 PORTICO 3

Tiene características similares a los puertos anteriores cuando funciona como pórtico E/S paralelo. Además cada pin de este puerto tiene asignada una función alterna.

En la tabla de la figura 1.9 podemos ver las funciones alternas de los bits de los 4 pórticos del microprocesador.

PORTICOS	DIRECCION	MNEMONICO DEL BIT	FUNCION ALTERNA
P0	80H	P0.0 – P0.7	Bus de datos Externo y parte del Bus de Direcciones (AD0 – AD7).
P1	90H	P1.0 – P1.7	
P2	A0H	P2.0 – P2.7	Byte más significativo del Bus de Direcciones. (AD8 – AD15).
P3	B0H	P3.0 : RXD	Entrada del Pórtico Serial.
		P3.1 : TXD	Salida del Pórtico Serial.
		P3.2 : INT0	Entrada Interrupción externa 0.
		P3.3 : INT1	Entrada Interrupción externa 1.
		P3.4 : T0	Entrada del Contador del Timer 0.
		P3.5 : T1	Entrada del Contador del Timer 1.
		P3.6 : WR	Salida escritura de la RAM Externa.
		P3.7 : RD	Salida lectura de la RAM Externa.

Figura 1.9 Tabla de funciones alternativas de los pórticos del microcontrolador.

1.2.4.2 PORTICO SERIAL

Estos microcontroladores disponen de varios terminales que soportan como función alternativa, la posibilidad de transmitir (TXD: P3.1) y recibir (RXD: P3.0) datos en serie. Tanto la frecuencia de transmisión/recepción serie, como el tamaño de los datos, son programables. Además, el pórtico serie posee una interrupción dedicada.

La información manejada por el pórtico serie puede ser sincrónica o asincrónica; en el caso de ser asincrónica, cada carácter de información consta de 8 ó 9 bits; estos van precedidos por un bit de inicio *(start)* y seguido de un bit de parada *(stop)*.

Cada bit de información serie tiene una duración de 16 períodos de reloj. Los bits a transmitirse se cargan en el *buffer* de transmisión SBUF(TX), y se envían de uno en uno por el pin TX.

Por el pin RX se vigila la presencia de un flanco descendente (bit start) a partir del cual se inicia un nuevo carácter. Las muestras son tomadas a la mitad de la duración del bit y se carga en el buffer de receptor SBUF(RX).

Es posible realizar transferencia serie en modo *full-duplex.*

El puerto serie puede trabajar en diferentes modos; estos modos, así como las restantes opciones que dispone, se gobiernan a través del contenido de los bits del registro de control del PS (SCON), que se halla en la dirección 98H de los SFR.

En la figura 1.10 se observan los bits que corresponden al SCON.

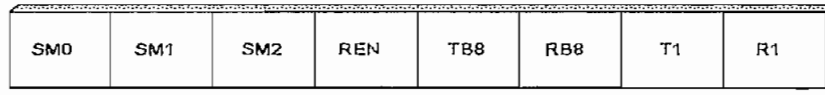

Figura 1.10 Bits del registro de control del PS

- Los bits *SM0* y *SM1*, determinan el modo en el que funcionará el puerto serial; a continuación se presenta en la tabla de la figura 1.11 los respectivos modos de operación.

SM0	SM1	MODO	FUNCION
0	0	0	Expansión E/S del Registro de desplazamiento. Transmisión sincrónica. No hay bits de start y stop.
0	1	1	UART. Transmisión/Recepción universal asincrónica, con 8 bits y velocidad variable.
1	0	2	UART con 9 bits y velocidad fija.
1	1	3	UART con 9 bits y velocidad variable.

Figura 1.11 Tabla con la descripción de los modos de operación del pórtico serie.

- El bit *SM2* Se pone en nivel alto en los modos 2 y 3 para soportar comunicaciones en multiproceso.
- *REM.* Es el bit de control encargado del permiso en la recepción.
- *TB8.* Corresponde al bit 8 (noveno) en la transmisión.
- *RB8.* Corresponde al bit 8 en la recepción.

- *TI.* Bandera de interrupción en la transmisión.
- *RI.* Bandera de interrupción en la recepción.

1.2.4.3 INTERRUPCIONES.

Se define como *interrupción,* la capacidad que tienen los microprocesadores para dejar de ejecutar momentáneamente, la secuencia normal de instrucciones de un programa y pasar a ejecutar otra secuencia de instrucciones denominada *"Rutina de Servicio a la Interrupción"*

Al producirse una interrupción, el servicio que realiza la CPU consta de las siguientes fases:

- La CPU detiene el programa en ejecución y guarda en la pila la dirección de retorno.
- El Contador de Programa se carga con una dirección en la que comienza el programa que da servicio a la interrupción. Comienza la ejecución del programa de interrupción.
- Al finalizar el programa de interrupción, la última instrucción, que es un *Retorno de Interrupción,* devuelve al PC la dirección de retorno y la CPU reanuda la ejecución del programa principal, justo en el punto donde se había interrumpido.

Los microcontroladores de la familia INTEL MCS-51/52, poseen cinco elementos que generan interrupciones en la serie 51, mientras que en la serie 52 existen siete elementos.

En cada tipo de interrupción se carga una dirección concreta en el PC, que recibe el nombre de *vector de interrupción.* En dicha dirección está situada la instrucción del programa que da servicio a esa interrupción.

A continuación se presenta en la tabla de la figura 1.12 la descripción de cada una de estas interrupciones.

FUENTE	EVENTO	BANDERA	DIRECCION Y MNEMONICO
INTERRUPCION EXTERNA 0	Nivel de 0L o transición negativa de la señal del pin INT0.	IE0	0003H EXTI0
INTERRUPCION DEL TIMER 0	Desbordamiento del Timer 0.	TF0	000BH TIMER0
INTERRUPCION EXTERNA 1	Nivel de 0L o transición negativa de la señal del pin INT1.	IE1	0013H EXTI1
INTERRUPCION DEL TIMER1	Desbordamiento del Timer 1.	TF1	001BH TIMER1
INTERRUPCION DEL PORTICO SERIAL	Finalización de la transmisión o de la recepción de un dato realizado por parte del pórtico serial.	TI o RI	0023H SINT
INTERRUPCION DEL TIMER 2 (solo en la serie 52)	Desbordamiento del Timer 2 o transición negativa de la señal del pin T2EX.	TF2 o EXF2	002BH TIMER2

Figura 1.12 Tabla con las Interrupciones de los microcontroladores.

1.2.4.3.1 PERMISOS Y PRIORIDADES

Para que la CPU acepte una interrupción y la atienda es preciso que dicha interrupción esté "permitida"; los permisos los soportan ciertos bits de los SFR agrupados en el byte de habilitación de interrupciones IE. En el cuadro de la figura 1.13 podemos observar en detalle cada uno de estos bits.

MNEMONICO	POSICION EN IE	DIRECCION COMO BIT	SIGNIFICADO
EA	IE.7	AFH	Deshabilita todas las interrupciones.
-	IE.6	AEH	Reservado.
ET2	IE.5	ADH	Habilita interrupción del Timer 2.
ES	IE.4	ACH	Habilita interrupción del pórtico serial.
ET1	IE.3	ABH	Habilita interrupción del Timer 1.
EX1	IE.2	AAH	Habilita interrupción Externa 1
ET0	IE.1	A9H	Habilita interrupción del Timer 0.
EX0	IE.0	A8H	Habilita interrupción Externa 0.

Figura 1.13 Tabla con los bits de habilitación de las interrupciones (Registro IE)

Por otra parte, cada interrupción puede ser programada como de *alta* o *baja* prioridad. Por defecto se programa como de baja prioridad.

La prioridad alta o baja de las interrupciones se programa a través del byte IP (Registro de Prioridad de Interrupciones). Con 0L define a la interrupción de menor prioridad, en cambio con 1L define a las de mayor prioridad. La tabla de la figura 1.14, detalla al byte IP.

MNEMONICO	POSICION EN IP	DIRECCION COMO BIT	SIGNIFICADO
-	IP.7	BFH	Reservado.
-	IP.6	BEH	Reservado.
PT2	IP.5	BDH	Desbordamiento del Timer 2 mayor prioridad.
PS	IP.4	BCH	El Pórtico Serial mayor prioridad.
PT1	IP.3	BBH	Desbordamiento del Timer 1 mayor prioridad.
PX1	IP.2	BAH	Interrupción Externa 1 mayor prioridad.
PT0	IP.1	B9H	Desbordamiento del Timer 0 mayor prioridad.
PX0	IP.0	B8H	Interrupción Externa 0 mayor prioridad.

Figura 1.14 Tabla con los bits del Registro de Prioridad de Interrupciones.

En caso de que varias interrupciones permitidas y con igual prioridad se produzcan simultáneamente, el orden con el que se atienden es el siguiente; donde el numero 1 el de mayor prioridad y el 6 el de menor prioridad.

1 Interrupción Externa 0.
2 Interrupción del Timer 0
3 Interrupción Externa 1.
4 Interrupción del Timer 1
5 Interrupción del Pórtico Serial.
6 Interrupción del Timer 2.

1.2.4.4 TEMPORIZADORES Y CONTADORES

Se denominan temporizadores a los circuitos lógicos que permiten determinar intervalos de tiempo con precisión.

Dentro de la estructura de los microcontroladores MCS-51/52; existen dos temporizadores en la serie 51 (Timer 0 y Timer 1) mientras que en la serie 52 existen tres (Timer 0, Timer 1 y Timer 2). Los temporizadores están constituidos por pares de registros llamados TH0/TL0, TH1/TL1 y TH2/TL2 que son localidades de los SFR, los cuales contienen dos contadores binarios ascendentes de 8 bits.

Los parámetros para el funcionamiento del Timer 0 y del Timer 1 están dados por los registros del control del modo de operación de los Timers (TMOD) y de los registros de control de los Timers (TCON), mientras que para el Timer 2 esta dado por el registro T2CON.

1.2.4.4.1 TIMER 0 Y 1

Las transiciones negativas de la señal que llegan a los registros de los Timers, producen un incremento en el valor de ellos. La fuente de impulsos que determina la frecuencia de contaje, puede provenir los ciclos de máquina generados internamente en el microcontrolador o bien de un generador externo. Si la señal de reloj proviene del oscilador del microcontrolador se dice que es un ***temporizador***; si proviene de una fuente externa que ingresa por el terminal T0 o T1, según el timer, se dice que es un ***contador*** ya que los registros están contando eventos que no necesariamente suceden a intervalos fijos.

Cuando los registros de los timers llegan a su valor máximo, en la siguiente transición regresan a cero, lo que a su vez pone un 1L en las banderas de desbordamiento de los Timers 0 ó 1; ó bits TF0 o TF1 respectivamente. De esta manera podemos programarlo para que cuente un número determinado de eventos o transcurra un cierto tiempo.

1.2.4.4.2 REGISTROS DE CONTROL DEL MODO DE OPERACIÓN DE LOS TIMERS 0 Y 1 (TMOD).

El registro TMOD contiene la información relacionada con los modos de operación de los Timer 0 y 1. Estos bits no pueden ser accesados independientemente sino en

conjunto. A continuación puede observarse en la figura 1.16 los bits del TMOD y una explicación de estos bits.

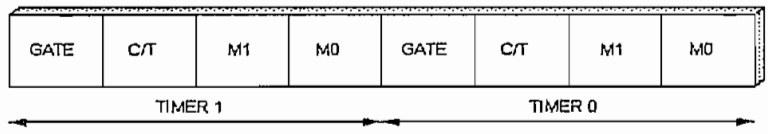

Figura 1.15 Bits del Registro de control de los modos de operación de los Timers

GATE.- Es el bit que habilita la sincronización del Timer 0 y del Timer 1 con las señales externas que llegan a los terminales INT0 y INT1, respectivamente.

C/T.- Es el bit para seleccionar entre temporizador o contador.

- C/T = 0 Temporizador. La entrada del reloj es la señal del oscilador del microcontrolador dividida para 12.
- C/T = 1 Contador. La entrada del reloj es la señal del terminal de entrada T0 y T1, respectivamente.

M0 y M1.- Son los bits para seleccionar el modo de operación.

M0	M1	MODO	OPERACIÓN
0	0	0	TH0/TL0, TH1/TL1 contadores de 13 bits.
0	1	1	TH0/TL0, TH1/TL1 contadores de 16 bits.
1	0	2	TH0/TL0, TH1/TL1 contadores de 8 bits con recarga automática.
1	1	3	TH0/TL0 doble contador de 8 bits y TH1/TL1 detenido.

REGISTROS DE CONTROL DE LOS TIMER 0 Y 1 (TCON)

El registro TCON contiene información sobre el estado de los Timers 0 y 1 y de las instrucciones externas. El detalle de los bits se muestra en la tabla de la figura 1.16.

MNEMONICO	POSICION EN IP	DIRECCION COMO BIT	SIGNIFICADO
TF1	TCON.7	8FH	Bandera de desbordamiento del Timer 1.
TR1	TCON.6	8EH	Bit de Arranque/Parada del Timer 1.
TF0	TCON.5	8DH	Bandera de desbordamiento del Timer 0.
TR0	TCON.4	8CH	Bit de Arranque/Parada del Timer 0.
IE1	TCON.3	8BH	Bandera de Interrupción Externa 1.
IT1	TCON.2	8AH	Bit de selección del tipo de señal de la INT. EXT. 1
IE0	TCON.1	89H	Bandera de Interrupción Externa 0
IT0	TCON.0	88H	Bit de selección del tipo de señal de la INT. EXT. 0

Figura 1.16 Tabla con el detalle de los bits del registro de control de los Timers 0 y 1

1.3 SISTEMA DE DESARROLLO PARA MICROCONTROLADORES.

Los elementos fundamentales que comprenden los diseños con microprocesadores son:

- **El equipo físico** que hace referencia a los elementos de tipo electrónico.
- **El sistema lógico** que está constituido por un conjunto de instrucciones o el programa.

1.3.1 EQUIPO FISICO O HARDWARE

El equipo físico que se requiere para desarrollar un proyecto con microcontroladores comprende los siguientes elementos:

a. Computador personal o Microcomputador
b. Equipo de pruebas del Programa
c. Interfaz de comunicaciones.
d. Programador de Memorias

El figura 1.17 puede verse estos componentes.

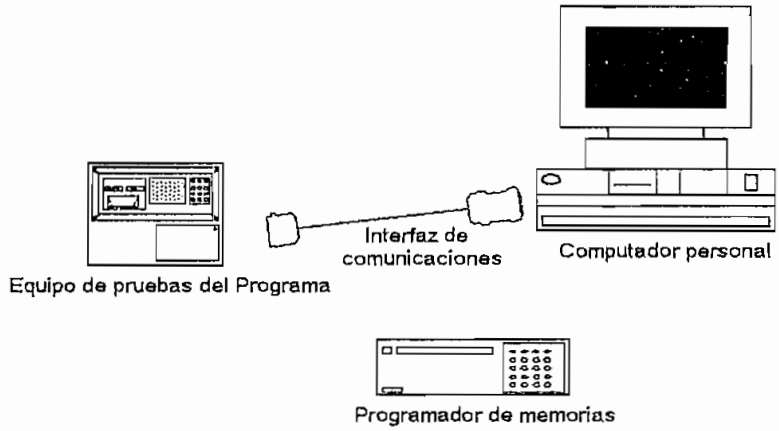

Figura 1.17 Equipo Físico

1.3.1.1 Computador Personal

El cual ayuda en el desarrollo del programa diseñado. Las características básicas necesarias que debe cumplir el computador personal (PC) dependen del tipo de programas que se utilicen para la edición, depuración y simulación del proyecto; estos aspectos lo determinarán los proveedores de dichos programas.

Es indispensable que la PC disponga de un pórtico serial, el cual se lo utilizará para comunicar o enviar datos al programador de memorias o el equipo de pruebas.

1.3.1.2 Equipo de Pruebas

El equipo de pruebas nos ayudará a observar el correcto funcionamiento del proyecto, es el paso final en el desarrollo del diseño; en este podemos ver si las condiciones requeridas se cumplen a satisfacción, para finalmente armarlo en un circuito integrado luego que el proyecto este bien elaborado.

Existen diferentes modelos de Equipos de Prueba con características diferentes, las cuales dependen del fabricante. En la facultad de Ingeniería Eléctrica se dispone del EPP MCS-31 (Equipo de Pruebas de Programa) el cual dispone de los siguientes elementos:

- Microcontrolador INTEL MCS 8051
- Memoria de Datos o RAM

- Memoria del programa o ROM
- Display con leds de 7 de segmentos.
- Teclado decimal incluido las teclas # y *
- Puerto de comunicaciones serial en la norma RS232
- Y otros dispositivos.

1.3.1.3 Interfaz de Comunicaciones

La Interfaz de comunicaciones sirve para transferir los datos ó el programa desde el computador al equipo de pruebas.

Este depende también del equipo de pruebas que sé este utilizando; en el caso del EPP MCS-31, se requiere la interfaz RS232-C, el cual es utilizado para pórtico serial. Las características tanto físicas como lógicas están definidas por el estándar RS232 de las normas internacionales.

1.3.1.4 Programador de Memorias

Este nos sirve para grabar en la memoria UVPROM el programa que procesará el microcontrolador. El programador de memorias utilizado en el área de Sistemas Microprocesados de la facultad de Ingeniería Eléctrica es el ALLMAX; el cual es un programador universal que se comunica con el computador a través de un pórtico paralelo. Este dispositivo permite: limpiar, leer, escribir y verificar las memorias; permite además realizar una prueba lógica de las memorias que se van a utilizar.

1.3.2 SISTEMA LOGICO O SOFTWARE.

El sistema lógico a pesar de ser inmaterial por estar compuesto de: ordenes, ideas, datos e información en general; es una parte muy importante en el desarrollo de sistemas basados en microcontroladores.

Un microcontrolador responde a un juego de instrucciones que el fabricante lo ha definido para sus aplicaciones. Cada instrucción está conformada por un código de tipo binario (para el caso de los microcontroladores de al familia MCS-51/52 es de 8 bits).

El proceso de desarrollo de programas para cualquier microprocesador hasta conseguir un programa ejecutable sigue los siguientes pasos:

a. ***Generación*** del PROGRAMA O MODULO FUENTE.- Que es la solución al problema, se lo desarrolla en lenguaje Ensamblador, es decir utilizando los Mnemónicos del microcontrolador para el cual se está realizando los programas. Los programas escritos en código binario o hexadecimal no tienen significado inteligible para el hombre, por lo que se diseño un lenguaje de tipo ***mnemotécnico*** en el que cada instrucción se representa con varias letras.

b. ***Ensamblado*** del PROGRAMA O MODULO OBJETO.- El cual es la solución al problema en Lenguaje de Máquina. Se llama ***Lenguaje de máquina*** a aquel que codifica las instrucciones de forma que el decodificador de la Unidad de Control las puede interpretar directamente.

c. ***Comprobación*** por algún medio manual o automático de la eficacia de la solución; en este caso un simulador de los microcontroladores MCS-51/51

d. ***Emulación o Grabación*** en memoria EPROM del programa objeto para convertirlo en el PROGRAMA EJECUTABLE por parte del microcontrolador.

Los errores serán corregidos siempre en el programa fuente.

Para realizar todo este proceso se requiere de la utilización de varios paquetes de software que a su vez se encuentran dentro de otro denominado SIDES a51. Todos estos paquetes están controlados por el Sistema Operativo MS-DOS. Cabe anotar que el SIDES a51, no es el único software que existe para realizar esta función; en el mercado existe una gran variedad de estos.

1.3.2.1 SIDES a51

El SIDES a51 (Sistema Integrado para el Desarrollo de Software para la familia de microcontroladores INTEL MCS-51/52) es un paquete de software desarrollado en el área de Sistemas Digitales de la ESCUELA POLITÉCNICA NACIONAL que simplifica el avance a través de la edición, ensamblado, simulación y depuración de programas; pudiendo inclusive realizar la ejecución en tiempo real mediante el Equipo de Pruebas de Programas.

Este sistema puede ejecutarse en computadoras IBM PC o compatibles que dispongan de una unidade de diskette, una unidad de disco duro, monitor monocromático o de color y mínimo 16 Kbytes de RAM.

El SIDES a51 esta formado por los siguientes paquetes:

1.3.2.2 EDITOR SIDES

Es un programa para la edición de texto, sirve para crear el archivo del Módulo Fuente de extensión "ASM" en el cual el programa del usuario es ingresado a través del teclado y los errores detectados en cualquier etapa del desarrollo del programa son corregidos mediante este editor.

El archivo fuente puede también escribirse en cualquier otro editor de texto; siempre tomando en consideración las normas de ingreso de una instrucción.

1.3.2.3 AVMAC51

Este programa es un Cross-Assembler programas para los microcontroladores INTEL MCS-51/52. El AVMAC51 lee el archivo del Módulo Fuente de extensión "ASM" para generar dos archivos:

a. Un archivo que contiene al Modulo Objeto con bloques relocalizables de tipo "OBJ".
b. Un archivo que contiene a los Módulo Fuente y Objeto de tipo "LST" utilizado para obtener reportes impresos.

1.3.2.4 AVLINK

Es un programa enlazador de Módulos Objeto. El AVLINK lee uno o más archivos de Módulo Objeto relocalizables y los junta para tener uno solo de extensión "HEX" que contiene el Módulo Objeto en localidades fijas o absolutas, utilizando para ello el formato INTEL de Módulos Objeto.

1.3.2.5 AVSIM51

Es un programa que permite simular en pantalla la ejecución del programa para los microcontroladores de la familiaMCS-51/52

1.3.2.6 PROMPRO8

Es un programa que permite observar el código de máquina contenido en el archivo de extensión "HEX" y poderlo enviar mediante el pórtico serial del computador hacia el programador de memorias o hacia el equipo de pruebas del programa. También tiene la posibilidad de convertir el computador en un Equipo Terminal de Datos (ETD) con la finalidad de poder comprobar programas de comunicación serial de los microcontroladores MCS-51/52.

Todos estos procesos pueden realizarse en otros paquetes computacionales, en este caso, se muestra como referencia, los nombrados anteriormente porque son los que se utilizan en la facultad.

1.3.2.7 SIDES 2000

Es una versión mejorada del SIDES a51, el cual sé esta desarrollando en el Area de Sistemas Digitales de la ESCUELA POLITECNICA NACIONAL.

Presentará varias ventajas tales como el manejo de los paquetes en ambiente WINDOWS en el cual hay mayores facilidades de manejo y es más amigable al usuario.

La presente tesis es un módulo de este sistema, en los siguientes capítulos se detallará las facilidades que tendrá esta.

CAPITULO II

2 PROGRAMACION DE LOS MICROCONTROLADORES INTEL MCS-51/52

2.1 FORMATO DE LAS INSTRUCCIONES

Un programa en lenguaje ensamblador está compuesto de una secuencia de instrucciones, cada instrucción ocupa una línea del archivo fuente. Una instrucción comprende cuatro campos, de los cuales no todos necesitan estar presentes.
Todas las instrucciones están divididas en campos iguales; para poder tener así una sintaxis común. Se permite que el contenido del campo de operandos varíe de acuerdo a la especificación de la operación, pero todos los operandos son formados de un conjunto común de elementos como: símbolos, números, operaciones aritméticas, etc.
Las operaciones y otras acciones conforman el programa; estas son: Instrucciones y Pseudo Instrucciones.

Antes de iniciar el estudio de la programación de los microcontroladores es necesario conocer la forma en que se elaborará un programa.

2.1.1 ELABORACION DE UN PROGRAMA

La elaboración de un programa para microprocesadores esta formada por varias etapas, cada una de ellas es indispensable para un optimo desarrollo del mismo; a continuación se definirá cada una de ellas.

2.1.1.1 DEFINICION DEL PROBLEMA

Es la descripción clara y concreta del problema que se quiere resolver.
En este paso debe definirse todos los parámetros necesarios para su desarrollo, así como los resultados que se quiere obtener. De la buena definición del problema

depende el tiempo requerido para realizar el sistema como también la calidad de los resultados.

Como ejemplo podemos realizar la suma de dos números: 224 y 140.

2.1.1.2 ELABORACION DEL ALGORITMO

Se define como algoritmo de un programa a la secuencia de pasos que se deben seguir, para conseguir el objetivo propuesto. Siguiendo con el ejemplo podemos definir los siguientes pasos:

1) ACUMULADOR ← 00000000 y CARRY ← 0
2) ACUMULADOR ← ACUMULADOR + 11100000 = 11100000 y CARRY ← 0
3) ACUMULADOR ← ACUMULADOR + 10001100 = 01101100 y CARRY ← 1

Los pasos del algoritmo se pueden convertir en instrucciones, cuando se definen las acciones u operaciones que se realizan en cada uno de ellos, pero también puede ser un conjunto de instrucciones dependiendo de la forma en que se define el algoritmo.

1) BORRAR EL ACUMULADOR y EL CARRY
2) SUMAR AL ACUMULADOR EL VALOR 224
3) SUMAR AL ACUMULADOR EL VALOR 140

2.1.1.3 ESCRITURA DEL PROGRAMA EN MNEMONICOS

La escritura del programa se la realiza mediante códigos, esta forma de codificar las instrucciones se denomina *ESCRITURA EN MNEMONICOS*. En esta se define claramente los elementos que constituyen las instrucciones.

Continuando con el ejemplo puede verse:

```
1) CLR     A
2) ADD     A,0E0H
3) ADD     A,8CH
```

En secciones posteriores se explicará el formato que debe cumplir cada instrucción.

2.1.2 *CICLO DE UNA INSTRUCCIÓN*

Es importante conocer como el microcontrolador ejecuta una instrucción. En la figura 2.1 se observa el diagrama de flujo del ciclo de una instrucción, la cual esta compuesta por: un ciclo de traída de la instrucción y un ciclo de ejecución de la instrucción.

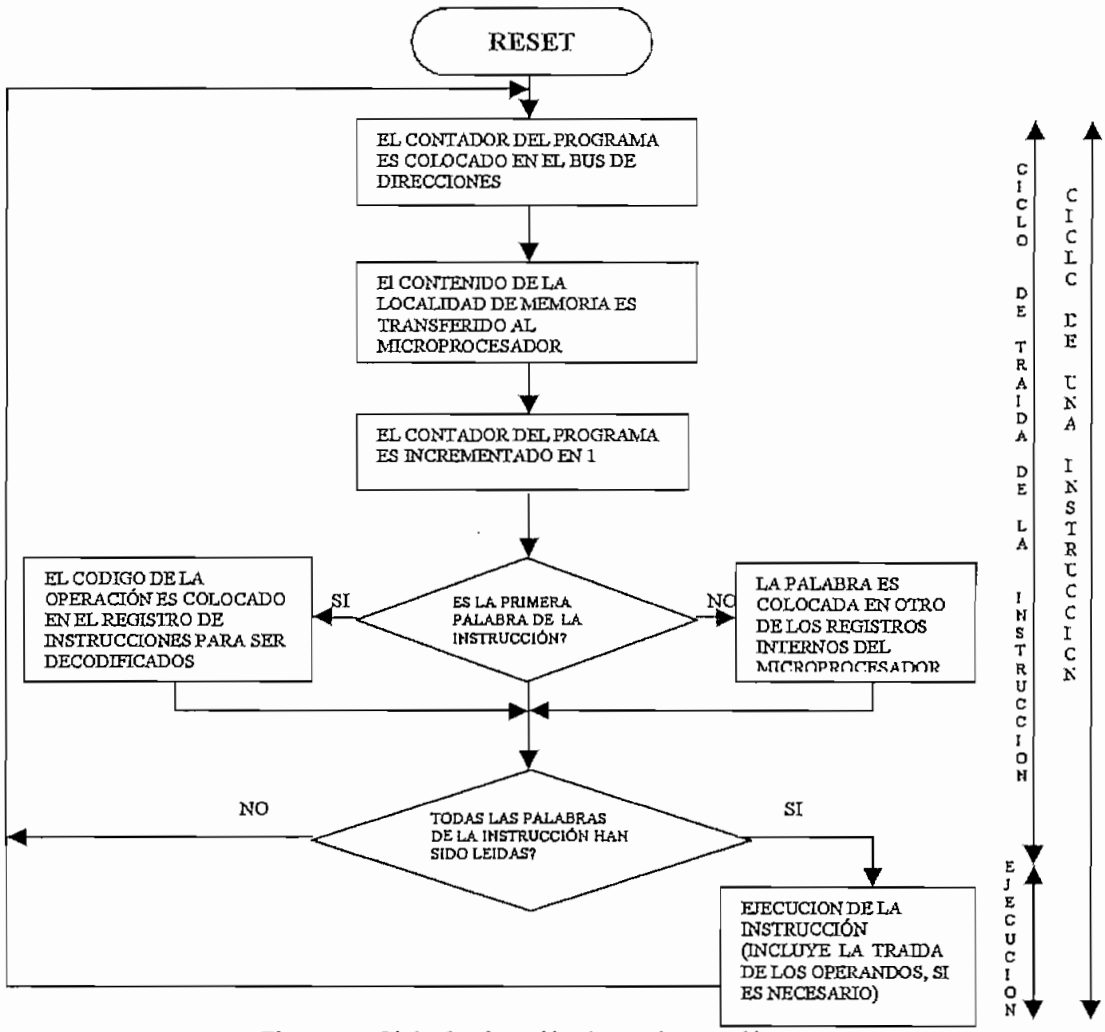

Figura 2.1 Ciclo de ejecución de una instrucción

2.1.2.1 CICLO DE TRAIDA DE LA INSTRUCCIÓN

Son operaciones de lectura, que el microcontrolador realiza a la memoria del programa. Son similares en todas las instrucciones, diferenciándose únicamente en el número de lecturas y el número de bytes que conforma la instrucción.

2.1.2.2 CICLO DE EJECUCION DE LA INSTRUCCIÓN

Son operaciones diferentes para cada tipo de instrucción, estas pueden ser internas al microcontrolador o en las que intervengan también elementos externos a este. Dependen también del *Modo de Direccionamiento* que utilice la instrucción.

2.2 MODOS DE DIRECCIONAMIENTO.

Los modos de direccionamiento son las distintas formas que se utilizan para obtener de la memoria, los operandos que intervienen en la operación.

En los microcontroladores de la familia INTEL MCS-51/52 existen básicamente cinco modos de direccionamiento.

2.2.1 DIRECCIONAMIENTO INMEDIATO.

Hace referencia a un operando que será proporciona de forma inmediata en la instrucción. Este tipo de operando se distingue porque va precedido por el signo #. Su ubicación es inmediata al opcode.

MOV A,#05 (A) ← 05

El acumulador se carga con el valor 05

2.2.2 DIRECCIONAMIENTO A LOS REGISTROS

Cuando la operación se realiza con los contenidos de los siguientes registros: R0....R7, ACC, B, C (bit del carry), DPTR. La especificación del o los registros se realiza dentro del mismo opcode de la instrucción.

XCH A,Rn (A) ↔ (Rn)

El contenido del acumulador se intercambia con el contenido del registro Rn.

2.2.3 DIRECCIONAMIENTO DIRECTO

Cuando la operación se realiza con los contenidos de la RAM interna, es decir, con los contenidos de las 128 localidades bajas (desde 00H hasta 7FH) y de los SFR (desde 80H hasta FFH). La dirección del operando se especifica como segundo byte de la instrucción, después del opcode.

MOV R4,35 (R4) ← (35)

El contenido de la dirección 35 de la RAM interna se carga en R4

2.2.4 DIRECCIONAMIENTO INDIRECTO

Cuando el operando se encuentra en área de datos de la RAM interna, las 128 localidades bajas (desde 00H hasta 7FH) en el caso de la serie 51, o en las 256 localidades (desde 00H hasta FFH) en el caso de la serie 52; la dirección del operando se especifica mediante el contenido de los registros R0, R1 o SP. También se utiliza direccionamiento indirecto cuando el operando se encuentra en la RAM externa (desde 0000H hasta FFFFH), en este caso la dirección del operando se especifica mediante los contenidos de: R0, R1 o DPTR.

A los registros que contienen la dirección se los denomina "punteros".

ANL A,@R1 (A) ← (A) ^ ((R1))

Operación AND entre el contenido del acumulador y el valor que se halla en la dirección que señala el R1.

2.2.5 DIRECCIONAMIENTO INDEXADO

Cuando la operación se realiza con los contenidos de cualquier localidad de la memoria del programa ROM (desde 0000H hasta FFFFH), la dirección del operando se especifica mediante el contenido del registro DPTR o del PC al que se le suma el contenido del ACUMULADOR, que en este caso equivale a un desplazamiento del puntero o índice.

MOVC A, @A+ DPTR (A) ← ((A) + (DPTR))

Mover hacia el acumulador el contenido de la dirección que especifica la suma del acumulador y el DPTR.

2.3 JUEGO DE INSTRUCCIONES.

Los microcontroladores de la familia MCS-51/52 poseen un conjunto de 111 instrucciones. Estas instrucciones pueden dividirse en cinco grupos:

- Instrucciones de transferencia de datos.
- Instrucciones de operaciones aritméticas.
- Instrucciones de operaciones lógicas.

- Instrucciones de control de flujo del programa.
- Instrucciones para manipulación de bits específicos.

Los mnemónicos de las instrucciones que se presentan, están formados por letras mayúsculas, minúsculas y algunos símbolos especiales. Las letras minúsculas deben ser remplazadas por algún valor numérico para definir completamente la instrucción, considerando los siguientes significados:

- *Rn.* Registro R0... R7 del banco de registros activos.
- *dirección.* dirección de 8 bits para una localidad de RAM interna.
- *@Ri.* Dirección de 8 bits para una localidad de RAM interna, direccionada indirectamente mediante el contenido de los registros R1 o R0.
- *#dato.* Constante de 8 bits incluida en la instrucción
- *#dato16.* Constante de 16 bits incluida en la instrucción
- *dirección16.* Dirección de destino de 16 bits.
- *dirección11.* Dirección de destino de 11 bits
- *rel.* Desplazamiento de 8 bits con signo en complemento de dos.
- *bit.* Dirección de 8 bits para un bit direccionable de la RAM de datos o del SFR.

Con relación al código de máquina se debe tomar en cuenta lo siguiente:

- Donde aparece "*rrr*" se reemplaza por el número binario del registro Rn que interviene en la instrucción. Así por ejemplo: R0 (rrr=000); R5 (rrr=101).
- Donde aparece "*i*" se reemplaza por el número binario del registro Ri que interviene en la instrucción. Así : R0 (i = 0), R1 (i = 1).

A continuación se presentan las instrucciones, en las que se muestra el mnemónico, la operación simbólica y el código de máquina.

2.3.1 INSTRUCCIONES DE TRANSFERENCIA DE DATOS

Mnemónico	Operación Simbólica	Código de Maquina
1. Instrucciones de movimiento de datos hacia el Acumulador		
MOV A,#dato	(A) ← dato	0111 0100 dato
MOV A,Rn	(A) ← Rn	1110 1rrr
MOV A,dirección	(A) ← dirección	1110 0101 dirección
MOV A,@Ri	(A) ← @Ri	1110 011i
2. Instrucciones de movimiento de datos hacia el Registro		
MOV Rn,#dato	(Rn) ← dato	0111 1rrr dato
MOV Rn,A	(Rn) ← (A)	1111 1rrr
MOV Rn,dirección	(Rn) ← (dirección)	1010 1rrr dirección
3. Instrucciones de movimiento de datos hacia una localidad de RAM interna con direccionamiento directo		
MOV dirección,#dato	(dirección) ← dato	0111 0101 dirección dato
MOV dirección,A	(dirección) ← (A)	1111 0101 dirección
MOV dirección,Rn	(dirección) ← (Rn)	1000 1rrr dirección
MOV dirección,dirección	(dirección) ← (dirección)	1000 0101 direcc(org)direcc(des)
MOV dirección,@Ri	(dirección) ← ((Ri))	1000 011i dirección
4. Instrucciones de movimiento de datos hacia una localidad de RAM interna con direccionamiento indirecto		
MOV @Ri,#dato	((Ri)) ← dato	0111 011i dato
MOV @Ri,A	((Ri)) ← (A)	1111 011i
MOV @Ri,dirección	((Ri)) ← (dirección)	1010 011i dirección
5. Instrucción de movimiento de datos de 16 Bits hacia el puntero de datos.		
MOV DPTR,#dato16	(DPTR) ← dato de 16 bits	1001 0000 dato15 - 8 dato7 – 0
6. Instrucciones de movimiento de datos hacia le Acumulador desde localidades de ROM		
MOVC A,@A+DPTR	(A) ← ((A)+(DPTR))	1001 0011
MOVC A,@A+PC	(A) ← ((A)+(PC))	1001 0011
7. Instrucciones de movimiento de datos entre el Acumulador y localidades de RAM externa.		
MOVX A,@Ri	(A) ← ((Ri))	1110 001i
MOVX A,@DPTR	(A) ← ((DPTR))	1110 0000
MOVX @Ri,A	((Ri)) ← (A)	1111 001i
MOVX @DPTR,A	((DPTR)) ← (A)	1111 0000
8. Instrucciones de intercambio con el Acumulador		
XCH A,Rn	(A) ↔ (Rn)	1100 1rrr
XCH A,dirección	(A) ↔ (dirección)	1100 0101 dirección
XCH A,@Ri	(A) ↔ ((Ri))	1100 011i

9. Instrucción de intercambio de los 4 bits menos significativos (un digito hexadecimal o BCD) del Acumulador con una localidad de RAM interna

XCHD	A,@Ri	(A3 – 0) ↔ ((Ri)3 – 0)	1101 011i

10. Instrucciones de Push y Pop

PUSH	dirección	(SP) ← (SP) + 1 ((SP)) ← (dirección)	1100 0000 dirección
POP	dirección	(dirección) ← ((SP)) (SP) ← (SP) - 1	1101 0000 dirección

2.3.2 INSTRUCCIONES DE OPERACIONES ARITMÉTICAS.

Mnemónico		Operación Simbólica	Código de Maquina
1. Instrucciones de Suma sin llevo			
ADD	A,#dato	(A) ← (A) + dato	0010 0100 dato
ADD	A,Rn	(A) ← (A) + (Rn)	0010 1rrr
ADD	A,dirección	(A) ← (A) + (dirección)	0010 0101 dirección
ADD	A,@Ri	(A) ← (A) + ((Ri))	0010 011i
2. Instrucciones de suma con llevo			
ADDC	A,#dato	(A) ← (A) + (C) + dato	0011 0100 dato
ADDC	A,Rn	(A) ← (A) + (C) + (Rn)	0011 1rrr
ADDC	A,dirección	(A) ← (A) + (C) + (dirección)	0011 0101 dirección
ADDC	A,@Ri	(A) ← (A) + (C) + ((Ri))	0011 011i
3.Instrucciones de Resta con debo			
SUBB	A,#dato	(A) ← (A) - (C) – dato	1001 0100 dato
SUBB	A,Rn	(A) ← (A) - (C) - (Rn)	1001 1rrr
SUBB	A,dirección	(A) ← (A) - (C) - (dirección)	1001 0101 dirección
SUBB	A,@Ri	(A) ← (A) - (C) - ((Ri))	1001 011i
4. Instrucciones de Incremento			
INC	A	(A) ← (A) + 1	0000 0100
INC	Rn	(Rn) ← (Rn) + 1	0000 1rrr
INC	dirección	(dirección) ← (dirección) + 1	0000 0101 dirección
INC	@Ri	((Ri)) ← ((Ri)) + 1	0000 011i
5.Instrucciones de Decremento			
DEC	A	(A) ← (A) – 1	0001 0100
DEC	Rn	(Rn) ← (Rn) – 1	0001 1rrr
DEC	dirección	(dirección) ← (dirección) - 1	0001 0101 dirección
DEC	@Ri	((Ri)) ← ((Ri)) – 1	0001 011i
6. Instrucción de incremento del puntero de datos			
INC	DPTR	(DPTR) ← (DPTR) + 1	1010 0011

7. Instrucciones de Multiplicación y División

MUL	AB	(A) ← (A) * (B) 7 – 0 (B) ← (A) * (B) 3 – 0	1010 0100
DIV	AB	(A) ← Cuociente A/B (B) ← Residuo A/B	1000 0100

8. Instrucción de Ajuste decimal

DA	A	Ajuste decimal del acumulador para la suma	1101 0100

2.3.3 INSTRUCCIONES DE OPERACIONES LOGICAS.

Mnemónico		Operación Simbólica	Código de Maquina
1.Instruciones de la operación lógica AND entre bytes			
ANL	A,#dato	(A) ← (A) ∧ dato	0101 0100 dato
ANL	A,Rn	(A) ← (A) ∧ (Rn)	0101 1rrr
ANL	A,dirección	(A) ← (A) ∧ (dirección)	0101 0101 dirección
ANL	A,@Ri	(A) ← (A) ∧ ((Ri))	0101 011i
ANL	dirección,#dato	(dirección) ← (dirección) ∧ (dato)	0101 0101 direcc dato
ANL	dirección,A	(dirección) ← (dirección) ∧ (A)	0101 011i dirección
2. Instrucciones de la operación lógica OR entre bytes			
ORL	A,#dato	(A) ← (A) ∨ dato	0100 0100 dato
ORL	A,Rn	(A) ← (A) ∨ (Rn)	0100 1rrr
ORL	A,dirección	(A) ← (A) ∨ (dirección)	0100 0101 dirección
ORL	A,@Ri	(A) ← (A) ∨ ((Ri))	0100 011i
ORL	dirección,#dato	(dirección) ← (dirección) ∨ (dato)	0100 0101 dire dato
ORL	dirección,A	(dirección) ← (dirección) ∨ (A)	0100 011i dirección
2. Instrucciones de la operación lógica OR Exclusivo entre bytes			
XRL	A,#dato	(A) ← (A) ⊻ dato	0110 0100 dato
XRL	A,Rn	(A) ← (A) ⊻ (Rn)	0110 1rrr
XRL	A,dirección	(A) ← (A) ⊻ (dirección)	0110 0101 dirección
XRL	A,@Ri	(A) ← (A) ⊻ ((Ri))	0110 011i
XRL	dirección,#dato	(dirección) ← (dirección) ⊻ (dato)	0110 0101 dirección dato
XRL	dirección,A	(dirección) ← (dirección) ⊻ (A)	0110 011i dirección
4.Instrucciones para Borrar y Complementar el Acumulador			
CLR	A	(A) ← 0	1110 0100
CLP	A	(A) ← - (A)	1111 0100

5. Instrucciones para Rotación del Acumulador

RL	A	Rotación del ACC a la izquierda.	0010 0011
RLC	A	Rotación circular del ACC a la izquierda.	0011 0011
RR	A	Rotación del ACC a la derecha.	0000 0011
RRC	A	Rotación circular del ACC a la derecha.	0001 0011

6. Instrucción para intercambiar los 4 bits más significativos con los menos significativos (intercambio de dígitos hexadecimales o BCD) del Acumulador.

SWAP	A	(A3 – 0) ↔ (A7 – 4)	1100 0100

2.3.4 INSTRUCCIONES DE CONTROL DE FLUJO DEL PROGRAMA.

Mnemónico	Operación Simbólica	Código de Maquina
1. Instrucciones de llamada a Subrutinas		
ACALL dirección11	(SP) ← (SP) + 1 ((SP)) ← (PC 7 – 0) (SP) ← (SP) + 1 ((SP) ← (PC 15 – 8) (PC 10 – 0) ← dirección de 11 bits	aaa1 0001 aaaa aaaa
LCALL dirección16	(SP) ← (SP) + 1 ((SP)) ← (PC 7 – 0) (SP) ← (SP) + 1 ((SP) ← (PC 15 – 8) (PC 10 – 0) ← dirección de 16 bits	0001 0010 direc15-8 direc7-0
2. Instrucciones de salto Incondicional		
AJMP dirección11	(PC 10 – 0) ← dirección de 11 bits	aaa0 0001 aaaa aaaa
LJMP dirección16	(PC) ← dirección de 16 bits	0000 0010 direcc15-8 direcc7-0
SJMP rel.	(PC) ← (PC) + rel	1000 0000 dirección,rel
JMP @A+DPTR	(PC) ← (A) + (DPTR)	0111 0011
3. Instrucciones de retorno de Subrutina y de Interrupción		
RET	(PC 15 – 8) ← ((SP)) (SP) ← (SP) – 1 (PC 7 – 0) ← ((SP)) (SP) ← (SP) – 1	0010 0010
RETI	(PC 15 – 8) ← ((SP)) (SP) ← (SP) – 1 (PC 7 – 0) ← ((SP)) (SP) ← (SP) – 1	0011 0010

4. Instrucciones de salto relativo condicionados al valor del Acumulador

JZ	rel	Sí (A)=0 (PC) ← (PC)+ rel Caso contrario continua	0110 0000 dirección,rel
JNZ	rel	Sí (A)≠0 (PC) ← (PC)+ rel Caso contrario continua	0111 0000 dirección,rel

5. Instrucciones de salto relativo condicionados al valor del Carry

JC	rel	Sí (C) =1L (PC) ← (PC)+ rel Caso contrario continua	0100 0000 dirección,rel
JNC	rel	Sí (C) = 0L (PC) ← (PC)+ rel Caso contrario continua	0101 0000 dirección,rel

6. Instrucciones de salto relativo condicionados al resultado de la comparación

CJNE	A,#dato,rel	Si (A) ≠ dato (PC) ← (PC)+ rel Caso contrario continua	1011 0100 dato direcc,rel
CJNE	A,dirección,rel	Sí (A) ≠ (direc) (PC) ← (PC)+ rel Caso contrario continua	1011 0101 direcc direcc,rel
CJNE	Rn,#dato,rel	Si (Rn) ≠dato (PC) ← (PC)+ rel Caso contrario continua	1011 1rrr dato direcc,rel
CJNE	@Ri,#dato,rel	Si ((Ri)) ≠ dato (PC) ← (PC)+ rel Caso contrario continua	1011 011i dato direcc,rel

7. Instrucciones de decremento y salto relativo si el resultado del decremento es diferente de Cero

DJNZ	Rn,rel	(Rn) ← (Rn) – 1 Sí (Rn) ≠ 0 (PC) ← (PC) + rel Caso contrario continua	1101 1rrr direcc,rel
DJNZ	dirección,rel	(dirección) ← (dirección) – 1 Sí (direcc) ≠ 0 (PC) ← (PC) + rel Caso contrario continua	1101 0101 direcc direcc,rel

8. Instrucción de ninguna operación

NOP	Ninguna operación	0000 0000

2.3.5 INSTRUCCIONES PARA MANIPULACIÓN DE BITS ESPECÍFICOS.

Mnemónico		Operación Simbólica	Código de Maquina
1. Instrucciones para Borrar un BIT			
CLR	C	(C) ← 0L	1100 0011
CLR	bit	(bit) ← 0L	1100 0010 bit

2. Instrucciones para Poner 1 lógico en un BIT

SETB	C	(C) ← 1L,	1101	0011	
SETB	bit	(bit) ← 1L	1101	0010	bit

3.Instrucciones para Complementar un BIT

CPL	C	(C) ← -(C)	1011	0011	
CPL	bit	(bit) ← -(bit)	1011	0010	bit

4. Instrucciones de la operación lógica AND entre el Carry y un Bit

ANL	C,bit	(C) ← (C) ∧ (bit)	1000	0010	bit
ANL	C,/bit	(C) ← (C) ∧ -(bit)	1011	0000	bit

5. Instrucciones de la operación lógica OR entre el Carry y un Bit

ORL	C,bit	(C) ← (C) ∨ (bit)	0111	0010	bit
ORL	C,/bit	(C) ← (C) ∨ -(bit)	1010	0000	bit

6. Instrucciones para Mover entre el Carry y un Bit

MOV	C,bit	(C) ↔ (bit)	1010	0010	bit
CPL	bit,C	(bit) ↔ (C)	1001	0010	bit

7.Instrucciones de salto relativos condicionados al valor de un Bit

JB	bit,rel	Sí (bit) = 1L entonces (PC)← (PC) + rel. Caso contrario continua.	0010	0000	bit	direcc,rel
JNB	bit,rel	Sí (bit) = 0L entonces (PC)← (PC) + rel. Caso contrario continua.	0011	0000	bit	direcc,rel
JBC	bit,rel	Sí (bit) = 1L entonces (PC)← 0L, (PC)← (PC) + rel. Caso contrario continua.	0001	0000	bit	direcc,rel

2.4 DESARROLLO DE LOS PROGRAMAS EN ASSEMBLER

Existen varios tipos de Ensambladores, los cuales presentan una u otra ventaja en su forma de trabajo. A continuación se describirá el modo de desarrollo del AVMAC51, por ser este utilizado en la facultad.

2.4.1 SINTAXIS DE LAS INSTRUCCIONES Y OPERANDOS

2.4.1.1 INSTRUCCIONES

Cada instrucción consiste de cuatro campos opcionales:

- campo de etiquetas
- campo de operaciones

- campo de operandos
- campo de comentarios

Estos campos están arreglados de la siguiente forma:

[<Etiqueta>][<Operación> [<Operandos>.....]] [;<Comentarios>]

El campo de Operaciones; contiene un símbolo Mnemónico de las instrucciones del 8051 o una pseudo Instrucciones (directivas de ensamblaje). La interpretación exacta del campo de Etiquetas y Operandos varia de acuerdo a cual instrucción o directiva está presente.

El campo de Etiqueta; si está presente consiste de un identificador opcional seguido por dos puntos(:). El uso de un identificador en el campo de etiquetas, define a este como un símbolo de usuario y determina un valor. Generalmente este valor es la dirección del primer byte de la instrucción o dato generado por la instrucción.

El campo de operandos; está compuesto de ninguno o varios operandos, si existen dos o más estos son separados por comas. Los operandos pueden ser de varios tipos, tales como: registro, expresiones aritméticas, palabras, entre otros. Dependiendo de la operación que se requiere efectuar, serán aceptados los diferentes tipos de operandos.

El campo de comentarios; consiste de una secuencia de caracteres precedidos por un punto y coma. Estos son escritos para que el programador entienda lo que esta realizando y son siempre ignorados por el ensamblador

Los campos son separados por espacios en blanco o por una combinación de espacios en blanco o tabs. Se sugiere utilizar únicamente tabs. Esto no solo elimina espacios en el archivo fuente, sino también permite que las líneas de programación se hallen en columnas verticales; lo que hace que el programa sea más fácil de leer.

Los campos pueden empezar en cualquier columna excepto que:

- La etiqueta, si está presente, debe empezar en la primera columna.
- Por lo menos un espacio en blanco debe existir antes de la operación
- Si existen líneas en blanco estas son tratadas como comentarios.

2.4.1.2 EJEMPLO DE INSTRUCCIONES

1. *Líneas que consisten solamente de un comentario.*

   ;Este es un comentario

   Un comentario puede empezar en cualquier parte de la línea. Se requiere que se inicie este con un punto y coma (;)

2. *Instrucciones con Etiqueta, Operación, Operandos y Comentarios.*

   Salto MOV A,R1 ;Copiar en el Acumulador el contenido del registro R1

   La etiqueta debe empezar en la primera columna.

3. *Línea que consiste solamente de una Etiqueta.*

   Salto

   La Etiqueta debe empezar en la primera columna.

4. *Línea compuesta de Operación y Operandos*

   SUBB A

   Si la Etiqueta no está presente, se debe insertar espacios en blanco o un tab en el inicio de la instrucción, puesto que la primera columna está reservada para la Etiqueta.

Por compatibilidad con otros ensambladores, los símbolos en el campo de etiquetas no deben ser seguidos por dos puntos (:).

2.4.1.3 IDENTIFICADORES Y SIMBOLOS.

El identificador es una palabra o un nombre, tal como: INICIO, RT0, A002. Existen diferentes tipos de identificadores en el lenguaje ensamblador. Los Mnemónicos son los nombres de las instrucciones o pseudo - instrucciones. Los Operandos contiene

varios identificadores tales como: identificadores de funciones aritméticas, así: MOD (y el resto de funciones), también existen funciones de tipo booleana como OR y otras.

Los símbolos representan caracteres, direcciones o cantidades numéricas; los cuales pueden ser utilizados como operandos.

Los Mnemónicos, Operandos y ciertos símbolos especiales son predefinidos por el ensamblador. Todos los símbolos requeridos pueden ser definidos en el programa.

Puede definirse un identificador como un símbolo para ser usado como una etiqueta o como campo de etiquetas de una declaración de EQU O TEQ. Los identificadores pueden contener algunos de los siguientes caracteres.

A – Z a – z 0 – 9 _ ?

El primer carácter de un identificador debe empezar por una letra.

Los identificadores pueden estar compuestos por varios caracteres, pero solamente son significativos un máximo de 32 caracteres. Esto es; dos identificadores son iguales si sus primeros 32 caracteres son iguales.

Los identificadores predefinidos son reservados, a estos no se los puede redefinir como símbolos.

2.4.1.4 NUMEROS

Un número consiste de una secuencia de dígitos, incluidos los dígitos hexadecimales (A hasta F), opcionalmente precedidos o seguidos por un carácter especificando la base de numeración. El primer carácter debe ser siempre un dígito decimal (0 – 9) o un identificador de base. Los identificadores de base son los siguientes.

Base	Identificador de Inicio	Identificador final
2	%	B
8	Q	O o Q
16	$	H

Si no se define identificador, la base 10 es asumida. Los siguientes ejemplos representan el número 127_{10}

127 Q177 177Q $7F 7FH %1111111 1111111B

Ya que un número debe empezar con un dígito decimal o escribiendo el identificador de base, debemos ser cuidadosos en la escritura de los valores hexadecimales que pueden empezar con letras. El numero 255_{10} puede ser escrito como $FF o 0FFH, pero no como FFH. La letra puede interpretarse como un símbolo para el ensamblador, resultando este un Error.

2.4.1.5 CARACTERES CONSTANTES

Un carácter constante, consiste de uno o dos caracteres encerrado en simples o dobles comillas (‘ ó “). La comilla simple puede ser usada como un carácter entre dobles comillas y viceversa

Los caracteres constantes son evaluados como un entero de 16 bits, con cada carácter convertido a su valor en código ASCII. Para un carácter constante simple el byte mayor es cero y el menor es el código del carácter. Para un carácter constante doble, el byte mayor es el valor que contiene el primer código del carácter y el menor contiene el segundo código del carácter.

Los siguientes ejemplos son equivalentes.

‘A’ “A” y 41H

‘AB’ “AB” y 4142H

2.4.1.6 REFERENCIAS DEL CONTADOR DE LOCALIDAD

Se lo define con el símbolo especial $. El valor de este símbolo es la dirección del primer byte del código o del dato generado por la instrucción presente. Por ejemplo en el siguiente fragmento de programa en ambos incide $ en el primer DW (definir palabra) tiene el valor 4455_{16} y ambos inciden, el segundo DW tiene el valor 4459_{16} ya que se adelanto el contador de localidades por 2 para cada operando.

```
ORG  4455H
DW   $, $
DW   $, $
```

2.1.3.7 EXPRESIONES ARITMETICAS

Los símbolos, números, caracteres constantes y referencia del contador de localidades; todos estos son evaluados como un valor entero de 16 bits. Sin embargo, así como estos valores son permitidos, también puede usar una expresión aritmética compuesta por uno o más de estos elementos conectados por operadores (funciones) y opcionalmente agrupados por paréntesis.

Como puede esperarse, el conjunto de operandos incluye: + - * y / (adicción, sustracción, multiplicación y división). También se incluyen algunos otros, tales como: modulo, relaciones de operadores y otros. El conjunto completo de operadores esta mostrado en la tabla de la figura 2.2. Todos los operadores excepto + - * / son representados por identificadores y deben ser separados de sus argumentos por al menos un espacio en blanco o tab.

Operador	Uso	Resultado
+	x+y	x+y
+	+x	0+x
-	x-y	x-y
-	-x	0-x
*	x*y	x*y (sin signo)
/	x/y	X/y (sin signo)
MOD	x MOD y	Modulo de x/y (sin signo)
SHL	x SHL y	Mover x hacia la izquierda, y lugares ($x*2^y$)
SHR	x SHR y	Mover x hacia la derecha y ligares ($x/2^y$)
HIGH	HIGH x	Byte más significativo de x
LOW	LOW x	Byte menos significativo de x
.	Dato.bit x	Dirección del byte de la dirección del dato
NOT	NOT x	Negación o complemento del Byte
AND	x AND y	Función lógica AND de cada bit de x y y
OR	x OR y	Función lógica OR de cada bit de x y y
XOR	x XOR y	Función lógica OR exclusivo de cada bit de x y y
LT	x LT y	1si x < y, caso contrario 0
LE	x LE y	1si x <= y, caso contrario 0
EQ	x EQ y	1si x = y, caso contrario 0
NE	x NE y	1si x <> y, caso contrario 0
GE	x GE y	1si x >= y, caso contrario 0
GT	x GT y	1si x > y, caso contrario 0

Figura 2.2 Tabla de Expresiones Aritméticas

El desarrollo de los programas en lenguaje Assembler dispone de otros elementos como ayuda en el desarrollo del programa, estas permiten un mejor manejo del lenguaje de programación. Estas líneas de programación son denominadas *"Pseudo instrucciones" o "Pseudo opcodes"*

2.4.2 PSEUDO OPCODES DEL ENSAMBLADOR

A continuación se describe los Pseudo instrucciones que podemos tener en un programa para microcontroladores:

2.4.2.1 END – EL SIMPLE PSEUDO OPCODE

Todos los programas de lenguajes Ensamblador contienen un fin de instrucciones. Este no es un opcode; no es parte del conjunto de instrucciones; solamente define al ensamblador terminar el ensamblaje. La instrucción END tiene un argumento opcional; esta es la transferencia de direcciones, la cual sirve como guía.

Se lo define en el campo que corresponde a la operación, su formato es el siguiente:

END

2.4.2.2 EQU Y TEQ – IGUALDADES

Son usados para dar un símbolo a un valor numérico (EQU da un valor "permanente"; TEQ asigna un valor "temporal" el cual puede cambiar después con otro TEQ). Estos son especificados en el editor. La sintaxis es igual en los dos caso.

Toda expresión puede estar definida por su tipo como también por su valor. En el 8051, los tipos que una expresión puede tener son cinco. Estos tipos corresponden a los diferentes espacios de direcciones disponibles en el 8051.Estos son:

- *CODE* para una expresión con referencias a una dirección de código.
- *DATA* a una dirección de datos de la RAM interna.
- *XDATA* a una dirección de datos en la RAM externa.
- *BIT* a una dirección de bit.
- *NUMBER* a un tipo usado por una expresión que no son direcciones.

Los valores pueden ser asignados explícitamente:

```
INICIO      EQU  5
```

A INICIO se le da el valor de 5 y el tipo por defecto es numérico, mientras:

```
INICIO      EQU  5,Code
```

A INICIO se le da el valor de 5 y el tipo es CODE. De esta manera se especifica explícitamente el tipo, a la expresión

2.4.2.3 DS – DEFINIR ESPACIO

A menudo se puede encontrar o querer definir espacios para una variable, la cual se encontrará en la RAM. Estas variables no pueden ser inicializadas en ROM.

DS solo asigna espacio para una variable. Por ejemplo; se quiere tener un arreglo de 16 bits por palabra de 20 elementos de longitud, este arreglo se lo llamará "PLASTICO". La declaración deberá ser la siguiente:

```
PLASTICO  DS   20    ;20 bits de longitud
```

DS asigna espacios de bits en lugar de bytes

2.4.2.4 DB – DEFINIR BYTES

DB es muy parecido a DS, excepto que DB requiere que se especifique valores al inicializar la estructura de datos. Esto es muy usado si se quiere tener una tabla o un mensaje almacenado en ROM y utilizarlo luego en el programa. DB requiere un byte de inicialización; el valor de cada byte es lo que se debe definir.

El mensaje es de tipo ASCII y debe estar encerrado entre comillas. Los Valores literales (entre 128 a 255) también pueden ser definidos. Esto se lo puede observar en un ejemplo en el cual un carácter ASCII es terminado por un retorno de carro, una línea libre y un cero.

```
LUIS  DB   "El mejor estudiante Politecnico",150,120,0
```

Otro forma de definir caracteres especiales es utilizando "backslash". Un "backslash" puede preceder a otros caracteres. La siguiente es una lista de los caracteres que pueden seguir a un backslash.

BACKSLASH Y CARACTERES

Caracteres	Caracteres insertables
\n	Carácter de nueva línea.
\r	Retorno de carro.
\t	Tab.
\0	Carácter nulo o cero.
\x	Una x precede a dos números de dígitos hexadecimales. X0A por ejemplo, esto es un retorno de carro

El siguiente es otro ejemplo de una pseudo instrucción terminada por un retorno de carro, una línea libre y un cero.

```
MONEDA    DB    "Sucre \r\n\0"
```

2.4.2.5 DW – DEFINIR PALABRA.

DW es parecido a DB, excepto que DW asigna espacios de "palabras", en un bloque de 16 bits de longitud. Un ejemplo muestra localizaciones de un arreglo perteneciente a seis direcciones:

```
VECTOR    DW    ARTE,VIDA,ARTE+10H,VIDA+10H,100H,ARTE+VIDA
```

2.4.2.6 PROC, ENDPROC, Y SIMBOLOS LOCALES.

A menudo es necesario encontrar a la mano una declaración que es conocida como un "símbolo local". Esta es una etiqueta que señala el fin de un lazo, puede ser usada solamente en el código inmediato. Al usar una etiqueta global como "SALTO", quiere decir que no puede usarse esta misma etiqueta de nuevo para un fin de lazo en el programa.

Este problema es resuelto por los símbolos locales. Los símbolos locales son un conjunto diferente de símbolos por dos razones:

a) Deben empezar por dos caracteres "L?" ("L?SALTO" puede ser un símbolo local).
b) Deben ser declarados entre los pseudo opcodes: PROC y ENDPROC.

Todo bloque PROC – ENDPROC debe tener un nombre. El nombre asociado con un PROC es la etiqueta que aparece sobre la línea del pseudo opcode PROC. PROCs

permite el uso de interrupciones sobre pequeñas fracciones de código. Como ejemplo podemos observar un PROC con pocos símbolos locales.

```
COBRE       PROC
            MOV  A,60H
L?SAL:
            DEC  A
            CMP  B
            JNZ  L?SAL
            ENDPROC
```

En este ejemplo, "L?SAL" es un símbolo local; este puede volver a usarse en otros PROCs.

2.4.2.7 INCLUDE – OTROS MODOS PARA INCLUIR ARCHIVOS

En algunos casos no es necesario ver en el preprocesador el contenido de todos los archivos incluidos, o no es necesario el uso del preprocesador en todo. En este caso es donde INCLUDE es muy usado.

2.4.2.8 SEG Y SEGMENTOS.

Un archivo del lenguaje ensamblador, al ser ensamblado produce un archivo objeto. Este archivo objeto es al que nosotros podemos llamar "módulo". Cuando el enlazador, enlaza los programas, primero observamos a todos los módulos disponibles a este. Uno de los trabajos de enlazador es subdividir de esta colección cada módulo, en los llamados segmentos.

Mientras el código fuente es dividido en módulos, el código de máquina final es dividido en segmentos. Los segmentos son usualmente reunidos por el enlazador.

Son cuatro los segmentos los cuales el ensamblador reconoce, estos son: "CODE", "DATA", "XDATA" Y "BIT". Si no se especifica en el ensamblador, el programa envía todos los segmentos.

Los segmentos CODE y DATA son los más necesarios. Se puede poner en el programa código en el segmento CODE y datos en el segmento DATA.

Se puede cambiar el segmento presente con el comando SEG. El siguiente fragmento de programa muestra como se puede cambiar el segmento presente y ensamblar en otro segmento.

```
        MOV  B, MARCA
        SEG  DATA
MARCA   DW   45H
        SEG  CODE
        JMP  SALTO1
        Etc...
```

2.4.2.9 DEFSEG – DEFINICION DE SEGMENTOS

En el curso de la programación se requerirá a más de los segmentos CODE y DATA otro tipo de segmentos. En algún momento se necesitará un segmento para datos de ROM (llamado ROMDATA), un segmento para datos de RAM (RAMDATA) y un segmento para el STACK. Se puede definir estos nuevos segmentos usando el pseudo opcode DEFSEG.

DEFSEG también permite definir segmentos los cuales pueden ser de dos tipos: *absolutos y relocalizables*. Los segmentos absolutos no son utilizados en su totalidad por el enlazador. Se debe definir en un segmento absoluto, la dirección absoluta en la cual va a iniciar.

DEFSEG también permite definir la dirección de un segmento. La dirección especificará, donde el segmento inicia.

Otra atribución del segmento es la definición del tamaño del bloque. Este manda al enlazador a que pueda residir en un cierto tamaño de área.

También podemos definir en el segmento inclusive si este es recubierto, o a cual clase el segmento pertenece.

En el 8051, son permitidos cuatro tipos de segmentos. El tipo del nuevo segmento puede ser especificado con el interruptor "CLASS=". Los tipos permitidos son: CODE, DATA XDATA y BIT.

A continuación se muestra como DEFSEG es usado:

```
DEFSEG STACK,CLASS=DATA          ;STACK es un segmento normal
                                 ;relocalizable.
DEFSEG INSTALAR                  ;por defecto es CODE.
DEFSEG BLOQUE,START=200H,ABSOLUTE    ;BLOQUE es un segmento
                                 ;absoluto, el cual inicia en 200H.
DEFSEG PRIMERO,ALIGN=PAGE        ;PRIMERO es relocalizable, con alineación
;de página.(PRIMERO puede iniciar en una dirección la cual es un múltiplo de
;800H).
DEFSEG SEGUNDO,OVERLAID          ;SEGUNDO es recubierto.
DEFSEG NUEVO,START=50H,CLASS=BIT ;NUEVO es un segmento nuevo en
;el espacio bit , inicia en la dirección 50H.
```

2.4.2.10 ORG – MOVIMIENTO ENTRE SEGMENTOS.

ORG permite mover segmentos, toma un solo argumento. Sí se halla en un segmento relocalizable, ORG cambia el valor del contador de localidades a la dirección de inicio del segmento presente. ORG debe ser usado solo en segmentos "Absolutos", o segmentos que son "Recubiertos" y tener un "START=DIRECCIÓN" especifica.
Si se halla en un segmento absoluto, ORG cambia el contador de localidades a la dirección absoluta dada. La siguiente sección de código muestra como sé operar:

```
DEFSEG VECTOR, START=50H    ;VECTOR es absoluto, inicia en 50H
       SEG VECTOR           ;Ahora estamos dentro del segmento VECTOR
       ORG  1000H           ;Ahora estamos en al dirección absoluta 1000H
VET1   DW   VECT            ;Defino la etiqueta VEC1, dándole un valor.
VET2   DW   VECT+2          ;Defino VET2
VET3   DW   VECT+4          ;Defino VET3
       ORG  2000H           ;Salto a la dirección 2000H
VECT   DS   6               ; Defino VECT
       SEG CODE             ; Vamos al segmento CODE para realizar el proceso
```

2.4.2.11 PUBLIC SÍMBOLOS Y EXTERN

Supongamos que tenemos una subrutina (quizá un PROC) la cual se quiere tomar como referencia de otro modulo objeto. En el curso normal de los eventos, es

molestoso tomar como referencia algo que esta definido en otro archivo, este puede producir un mensaje de error. Sin embargo esto puede hacerse con los pseudo opcodes PUBLIC y EXTERN.

Para crear símbolos públicos primero debe usarse el pseudo opcode PUBLIC en el archivo en el cual el símbolo es definido.

Después que el símbolo fue hecho publico en el archivo principal, este debe ser declarado como externo en los archivos a los cuales se está refiriendo. Un pequeño ejemplo muestra dos archivos los cuales muestran más claramente esto:

El archivo PRINCIPAL.ASM contiene la siguiente sección:

```
        PUBLIC SALTO,SALTO1,SALTO2,FIN
SALTODEC    A
        NOP
                RET
SALTO1          CALL        SALTO
        RET
SALTO2:
                INC     A
        NOP
                RET
FIN:
                CALL        SALTO2
                RET
```

El archivo SECUNDARIO.ASM puede ser:

```
                EXTERN SALTO,SALTO1,SALTO2,FIN
                CALL        SALTO
                CALL        SALTO1
                CALL        SALTO2
                CALL        FIN
```

Los símbolos externos pueden ser definidos con sus tipos. Por ejemplo, si se quiere declarar un símbolo como "HOLA" EXTERNAL, y queremos definirlo como tipo BIT se debe escribir: EXTERNAL HOLA(BIT)

2.4.3 OPCIONES DEL ENSAMBLADOR

El ensamblador tiene varias opciones las cuales pueden ser especificadas por el usuario. Las opciones deben ser dadas sobre la línea de comando o en el archivo fuente. Cuando se especifica una opción en el archivo fuente, esta debe estar sobre una línea de opciones. Las lías de opciones deben definirse así:

$opción, opción, (etc...)

Es importante saber que las líneas de opciones deben empezar con un signo de dólar en la primera columna.

Las opciones del ensamblador pueden ser de dos tipos. Llamadas *primeras opciones* deben ser especificadas en la línea de comandos o antes del inicio del archivo fuente. Otras opciones pueden aparecer en cualquier parte.

Las opciones requieren argumentos. Por ejemplo, la opción *pagelength* (la cual especifica la longitud de una pagina en la lista de archivo) requiere un número como un argumento. En este caso, el conjunto de longitud de pagina de 60 líneas puede especificarse como: "PAGELENGTH=60". Algunas otras opciones como *litle* requiere caracteres de texto como argumentos. Caracteres de texto son encerrados por paréntesis después del nombre de la opción. Por ejemplo "TITLE (Programa Principal)". Un tipo final de opciones son las opciones que toman valores booleanos. Estas opciones no requieren argumentos. Un ejemplo es la opción *paginate* , la cual controla la paginación de la lista del archivo.

La definición "PAGINATE" hace que la paginación se active, mientras que definir "NO PAGINATE" desactiva la paginación.

En el anexo A se presenta un listado de todas las opciones disponibles para el ensamblador AVMAC51.

CAPITULO III

3 DESARROLLO DEL EDITOR INTELIGENTE

3.1 INTRODUCCION A LA PROGRAMACION EN VISUAL BASIC

En la actualidad la mayoría de las funciones de programación, tienden a desarrollarse en el ambiente *Windows*, el cual es un entorno de desarrollo visual. Este ambiente ofrece innumerables ventajas, las que permiten al usuario realizar sus trabajos de manera más fácil.

Visual Basic es un lenguaje de desarrollo para *Windows*; este es una herramienta adecuada para la creación de programas que van a correr en dicho ambiente.

La creación de aplicaciones en *Visual Basic* comprende tres pasos principales, los cuales son:

1. Creación de la interfaz
2. Definición de las propiedades
3. Escritura del código

3.1.1 CREACION DE LA INTERFAZ

La interfaz es la parte visual de la aplicación con la que el usuario va a interactuar; está compuesta por formularios y controles.

Los formularios son la base para la creación de la interfaz, estos son contenedores de los elementos que formarán parte de la aplicación, como: cuadros de dialogo, elementos no visibles y controles. Tanto los formularios como los controles tienen su propio conjunto de propiedades, métodos y eventos, los cuales los hacen adecuados para una finalidad determinada.

3.1.2 DEFINICION DE LAS PROPIEDADES

Las Propiedades se puede considerar como atributos del objeto.

En este paso se debe especificar la forma como se desea que la interfaz se presente al usuario. Esto es; definir él titulo del formulario, establecer el tamaño del objeto, establecer la posición en la que se ubicará el objeto, definir el nombre de los controles, determinar el color de fondo de los objetos entre otros.

Si se va a escribir texto se debe definir el tamaño, la fuente y el color.

Entre las propiedades podemos encontrar:

- Name.- Determina el nombre asignado al control o formulario.
- *Caption.-* Este determina él titulo que se muestra en un formulario.
- *Height y width.-* Establecen el tamaño del objeto.
- *Lef y Top.-* Establecen la ubicación del objeto.

Estas son algunas de las propiedades de los objetos. Dependiendo del tipo de control insertado podemos tener propiedades específicas para este elemento.

Las propiedades también pueden ser definidas o cambiadas mediante código.

3.1.3 ESCRITURA DEL CODIGO.

Otras características de los formularios y los controles, son los métodos y eventos.

Los métodos son las acciones que se efectúan sobre los objetos y ***Los Eventos*** son las respuestas a estas acciones.

Los métodos pueden ser generados tanto por el usuario, el programa o el sistema.

Como ejemplos de eventos generados por el usuario tenemos:

- *Click.-* La acción de pulsar un botón del ratón.
- *KeyPress.-* La acción de presionar una tecla del teclado.
- *MouseMove.-* La acción de mover el ratón sobre un objeto, entre otros.

Como ejemplos de eventos generados por el programa puede nombrarse al evento *Change* el cual se produce cuando el texto en un cuadro de texto a sido modificado.

Entre los eventos generados por el sistema tenemos: *Initialize* el cual se produce antes que el formulario sea cargado.

Al escribir el código se dará respuesta a la acción que se produjo. Por tanto, se deberá escribir código en cada procedimiento (en cada acción) que se genere ó en la acción que se quiera responder.

Los procedimientos están definidos por el objeto sobre el cual se realiza una acción. Por ejemplo al hacer un click sobre un botón de nombre *Cancelar* el programa se situará en el procedimiento Cancelar_Click.
Las respuestas se definirán mediante la escritura del código en un procedimiento.
Los eventos pueden ser funciones definidas por el programa y funciones o procedimientos creadas por el programador. Los eventos también pueden ser modificaciones de las propiedades definidas anteriormente.

Al hacer referencia a una propiedad debemos especificar la procedencia de esta, es decir, si se quiere cambiar el tamaño del texto de un cuadro de texto de nombre txtEditor, el cual esta colocado en el formulario frmPrincipal, deberá escribirse de la siguiente forma:
frmPrincipal.txtEditor.FontSize = 6.
Donde 6 es un valor numérico que define el tamaño de la letra.

Cada formulario del proyecto tiene su propio conjunto de procedimientos. Si se requiere crear funciones o procedimientos que se vayan a utilizarse en más de un formulario, se puede definirlas en módulos. Los módulos son cuadros de texto en los cuales podemos crear funciones o procedimientos propias del programador.

Luego de haber realizado estos tres pasos se puede probar la aplicación.

3.1.4 CREACION DEL EDITOR INTELIGENTE

Para poder crear el proyecto "EDITOR INTELIGENTE" se tomó ciertas consideraciones respecto a la estructura de las instrucciones del microcontrolador MCS-51/52.

- El Editor debe tener características similares a los editores de texto comunes como son: WordPad o el Bloque de Notas.
- Las líneas de programación pueden ser de dos tipos: Instrucciones y Pseudo instrucciones.

- Cada instrucción esta formada por cuatro campos, los cuales son: Etiqueta, Opcode, Operandos y Comentarios, los cuales pueden o no estar presentes.
- Si una instrucción está bien escrita se pintará de color negro y si existen comentarios esté, deberá pintarse de color verde.
- Si una pseudo instrucción está bien escrita se pintará de color azul y si existen comentarios estos se pintarán de color verde.
- Si una instrucción o pseudo instrucción está mal definida se pintará la línea de color rojo.
- Se presentará mensajes del estado de la línea de programación.
- Debe permitirse hacer una corrección de la sintaxis de las líneas de programación.
- Los campos existentes deben estar alineados.

Hechas estas consideraciones se presenta el proceso de análisis que llevo a la creación del EDITOR INTELIGENTE.

3.2 DIAGRAMAS DE FLUJO DEL PROGRAMA

A continuación se presenta los diagramas de flujo que indican de una manera general la forma en que el proyecto está desarrollado. También se presenta una breve descripción de estos diagramas de flujo.

3.2.1 DIAGRAMA DE FLUJO PRINCIPAL (Inicio del Programa).

En este diagrama se describe el proceso que sigue el programa cuando se inicia, hasta mostrar la pantalla principal. En este procedimiento se realizan varias acciones como son:

- Mostrar una pantalla de presentación
- Cargar los valores que se requieren en el desarrollo de la edición; estos valores son obtenidos de la base de datos llamada "Editor Inteligente", la cual contiene información acerca del formato de las instrucciones.

Una vez que este proceso ha concluido el usuario podrá escoger entre: utilizar la barra de menú o editar el programa.

Cabe anotar que si inicia con un archivo abierto este tardará algún tiempo hasta realizar los procesos de alineación y pintado de las instrucciones existentes en dicho archivo.

En la figura 3.1 se ilustra dicho diagrama.

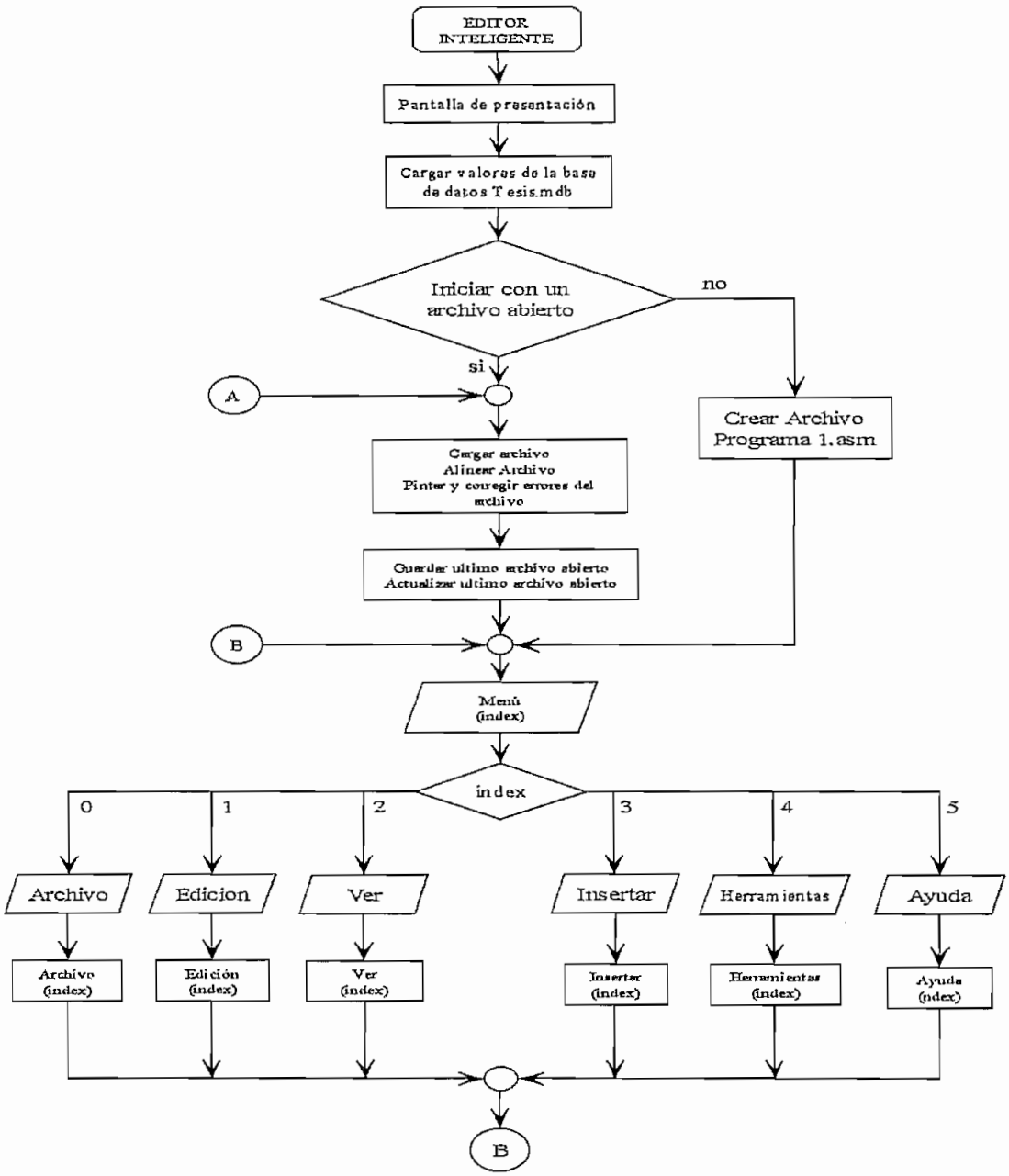

Figura 3.1 Diagrama de flujo Principal

La barra de Menú está formada por 6 opciones, las cuales son:

- *Archivo.*- Esta relacionada con la manipulación de archivos.

- *Edición.*- Se relaciona con la manipulación del programa que existe en el cuadro de texto
- *Ver.*- Permite mantener visible o no las barras de ayuda del programa.
- *Insertar.*- Permite insertar caracteres en el programa.
- *Herramientas.*- Se relaciona con la corrección y forma de presentación del Editor.
- *Ayuda.*- Presenta información sobre el Editor.

3.2.2 DIAGRAMA DE FLUJO MENU ARCHIVO

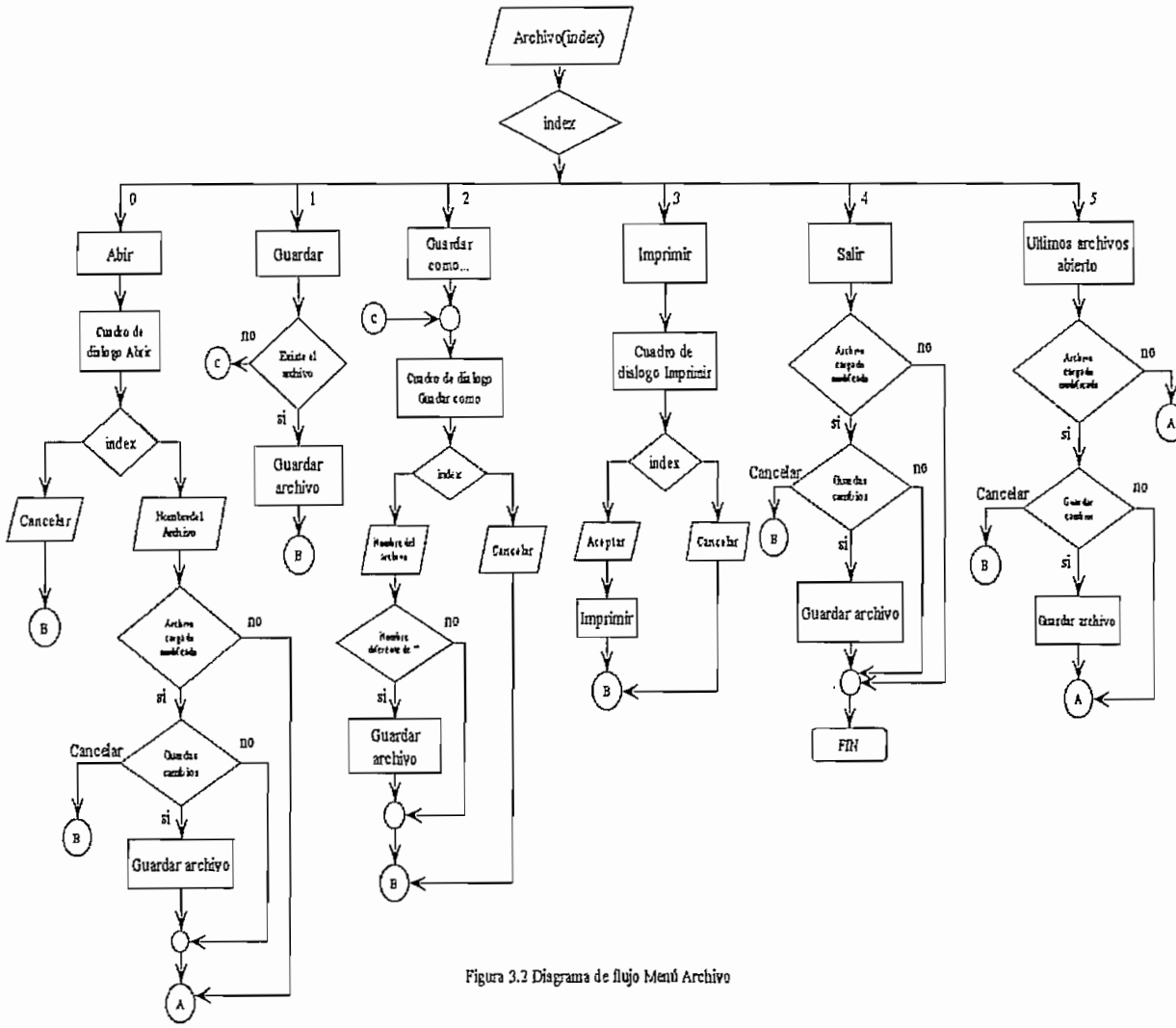

Figura 3.2 Diagrama de flujo Menú Archivo

Si se selecciona el menú Archivo, se puede escoger entre las opciones:

- *Abrir.*- Permite abrir un archivo de tipo ASM. existente.
- *Guardar.*- Guarda el archivo que se está editando.
- *Guardar como.*- Permite darle un nombre al archivo que el programa procesa.
- *Imprimir.*- Imprime el texto del programa que se está editando.
- *Salir.*- Finaliza la ejecución del Editor.
- *Ultimo Archivo.*- Abre el archivo cuya dirección se observa en este item, esta dirección corresponde a un archivo utilizado anteriormente.

Estas opciones corresponden a las opciones ya conocidas en un editor de texto. En la figura 3.2 puede verse el diagrama de flujo del Menú Archivo.

3.2.3 DIAGRAMA DE FLUJO MENU EDICION

Al ubicarse sobre el menú Edición, se puede acceder a las siguientes opciones:

- *Cortar.*- Corta el texto seleccionado y lo guarda en el portapapel.
- *Copiar.*- Copia el texto seleccionado y lo almacena el portapapel
- *Pegar.*- Pegar el texto que fue almacenado en el portapapel.
- *Borrar.*- Borra el texto seleccionado.
- *Seleccionar todo.*- Selecciona todo el texto que existe en el editor.
- *Buscar.*- Presenta el formulario Buscar, en el cual se ingresará la palabra que se requiere buscar en el texto.
- *Buscar siguiente.*- Busca en el texto la siguiente palabra que se estaba buscando.
- *Reemplazar.*- Presenta el formulario Reemplazar, en el que se debe ingresar la palabra a buscar y la palabra por la cual se reemplazará.

Como se puede apreciar el menú Edición también se lo puede encontrar en un editor de texto común. En la figura 3.3 se observa este diagrama de flujo.

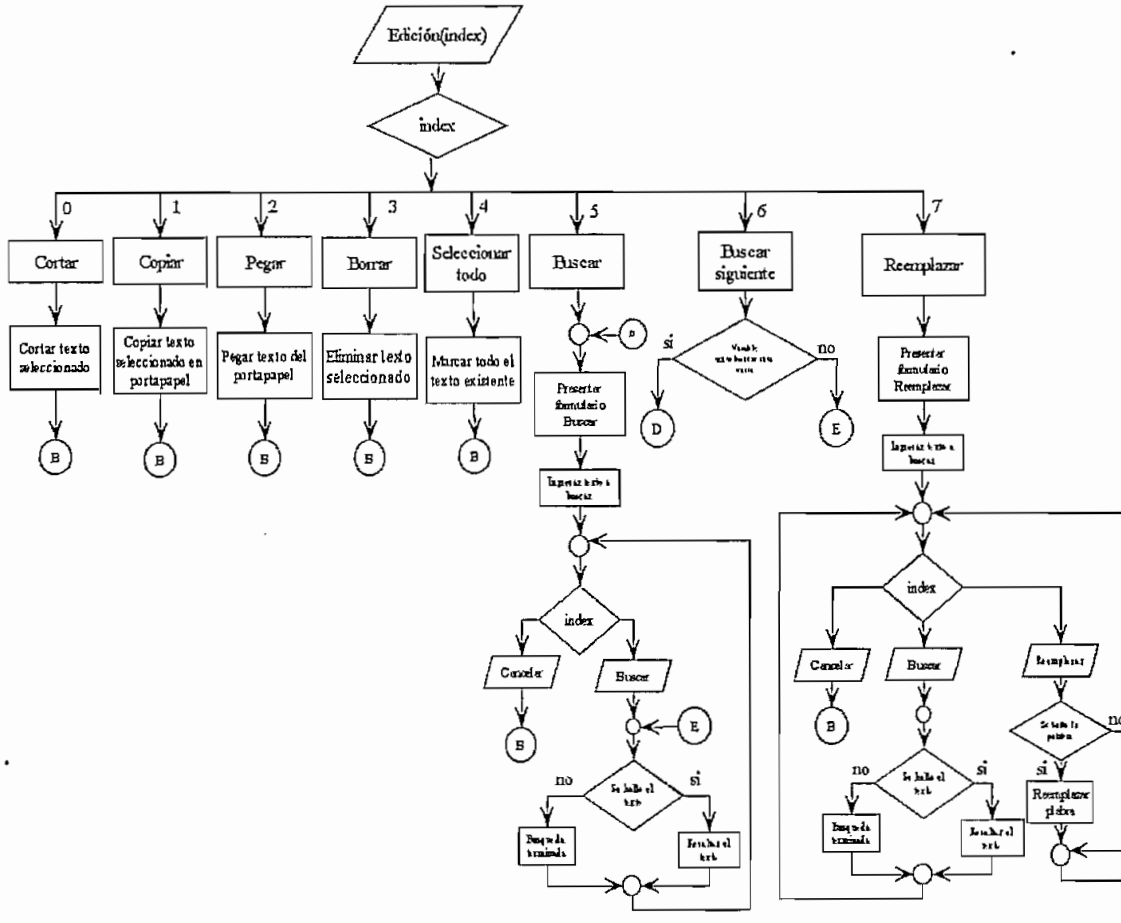

Figura 3.3 Diagrama de flujo del Menú Edición

3.2.4 DIAGRAMA DE FLUJO MENU VER

Al seleccionar el menú Ver podemos acceder a las opciones:

- *Barra de Herramientas.*- Permite hacer visible o no la barra de Herramientas.
- *Barra de Estado.*- Permite hacer visible o no la barra de Estado.
- *Barra de Posición.*- Permite hacer visible o no la barra de Posición.

En la figura 3.4 se presenta el diagrama de flujo de este sub menú.

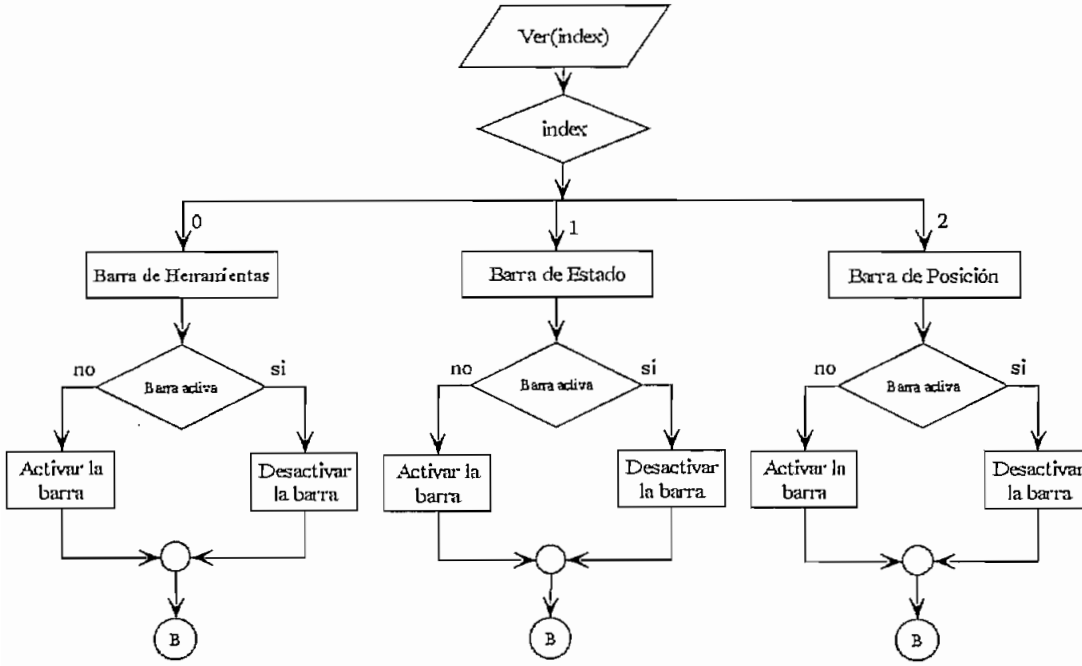

Figura 3.4 Diagrama de flujo del Menú Ver

3.2.5 *DIAGRAMA DE FLUJO MENU INSERTAR*

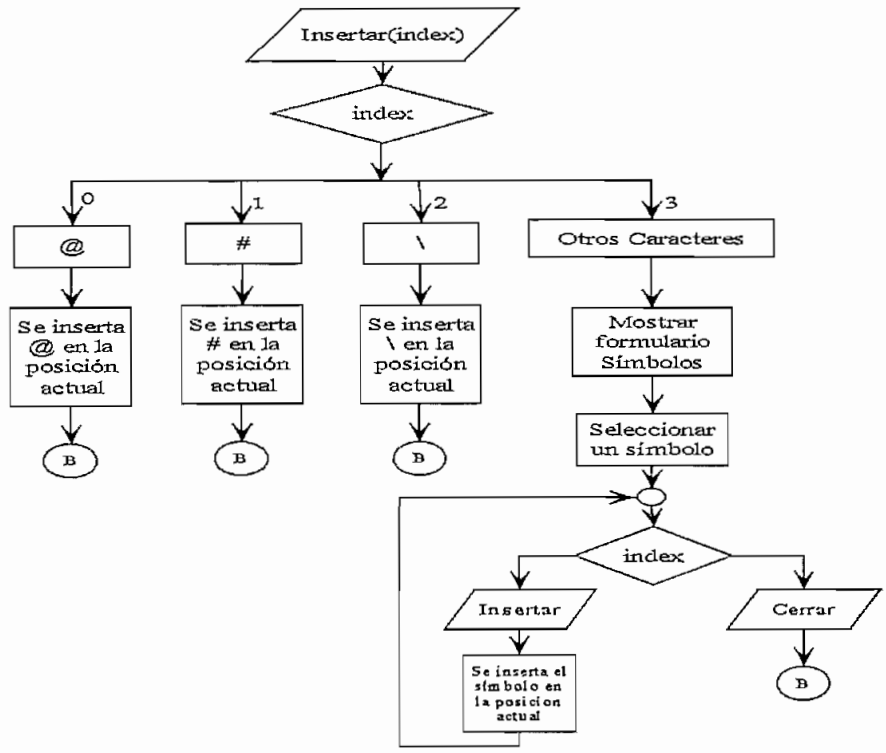

Figura 3.5 Diagrama de flujo del Menú Insertar

Si se selecciona el menú insertar se puede tener acceso a las opciones:

- @.- Permite insertar el carácter arroba (@) en el punto donde se halla el cursor.
- #.- Permite insertar el carácter número (#) en el punto donde se halla el cursor.
- \.- Permite insertar el carácter "backslash" (\) en el punto donde se halla el cursor.
- *Otros Caracteres.*- Presenta el formulario Símbolos, el cual permite insertar caracteres que corresponden al código ASCII, en el sitio en el que se halla el cursor

En la figura 3.5 puede observarse el diagrama de flujo.

3.2.6 DIAGRAMA DE FLUJO MENU HERRAMIENTAS

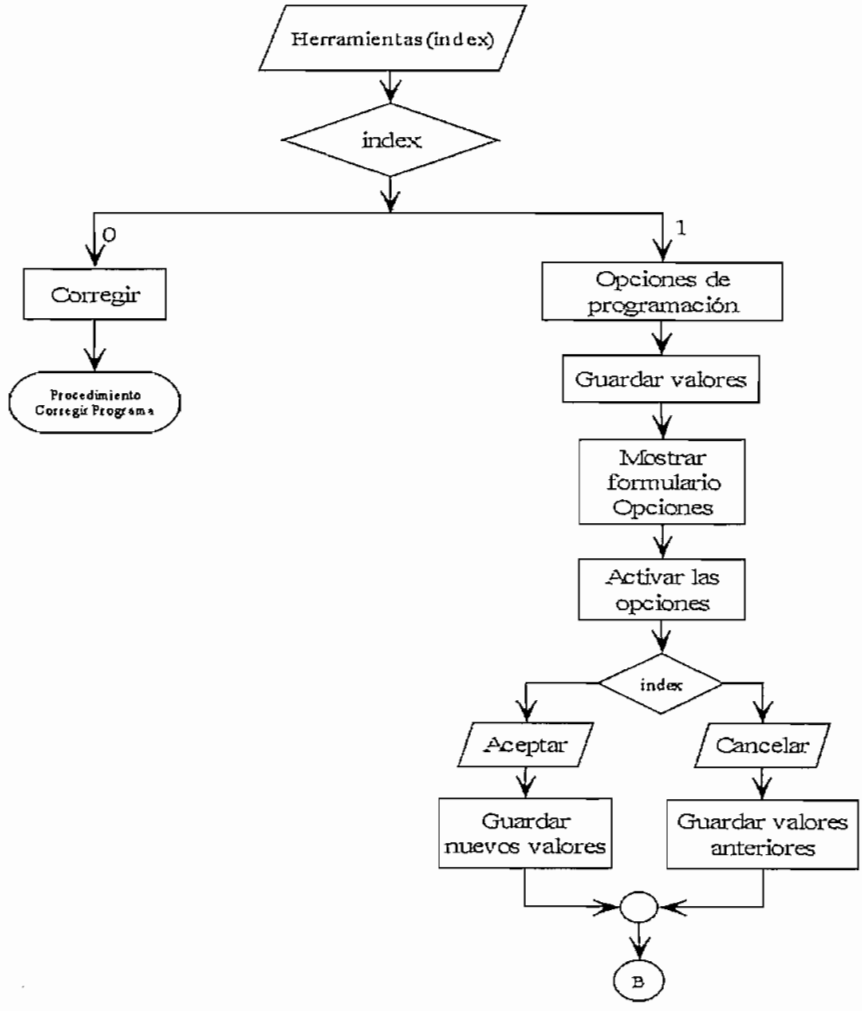

Figura 3.6 Diagrama de flujo del Menú Herramientas

Al seleccionar el menú Herramientas puede accederse a las opciones:

- *Corregir Errores.*- Permite corregir los errores de sintaxis que existen en las instrucciones del programa editado. Al encontrar algún error se presenta el formulario Corregir, el cual nos informa del tipo de error y nos permite corregirlo.
- *Opciones de programación.*- Muestra el formulario Opciones, en el cual se puede escoger entre varias opciones, como son:
  a) Mostrar ayuda en pantalla, si se presenta o no la ayuda.
  b) Uso del Ratón y teclado, si solo se utiliza el ratón y el ratón y el teclado a la vez.
  c) Y corrección de errores en el programa, si la corrección se lo realiza en: el transcurso de la definición de la instrucción, al finalizar la instrucción o al terminal de editar el programa.

En la figura 3.6 puede verse el diagrama de flujo del Menú Herramientas.

El diagrama de flujo del proceso de corrección de errores se muestra en la figura 3.7 En este diagrama, pode verse la forma como se procede a corregir todos los errores que existen en las líneas de programación, hasta el final del documento.

En la figura 3.8 y 3.9 se observa la forma como se procede a corregir una línea de programación. Este permite identificar si la línea ingresada corresponde a una pseudo instrucción o una instrucción; como también observa si la instrucción o pseudo instrucción esta bien escrita (en su forma), es decir, sí cumple con las condiciones requeridas.

3.2.7 DIAGRAMA DE FLUJO CORRECCION DEL PROGRAMA

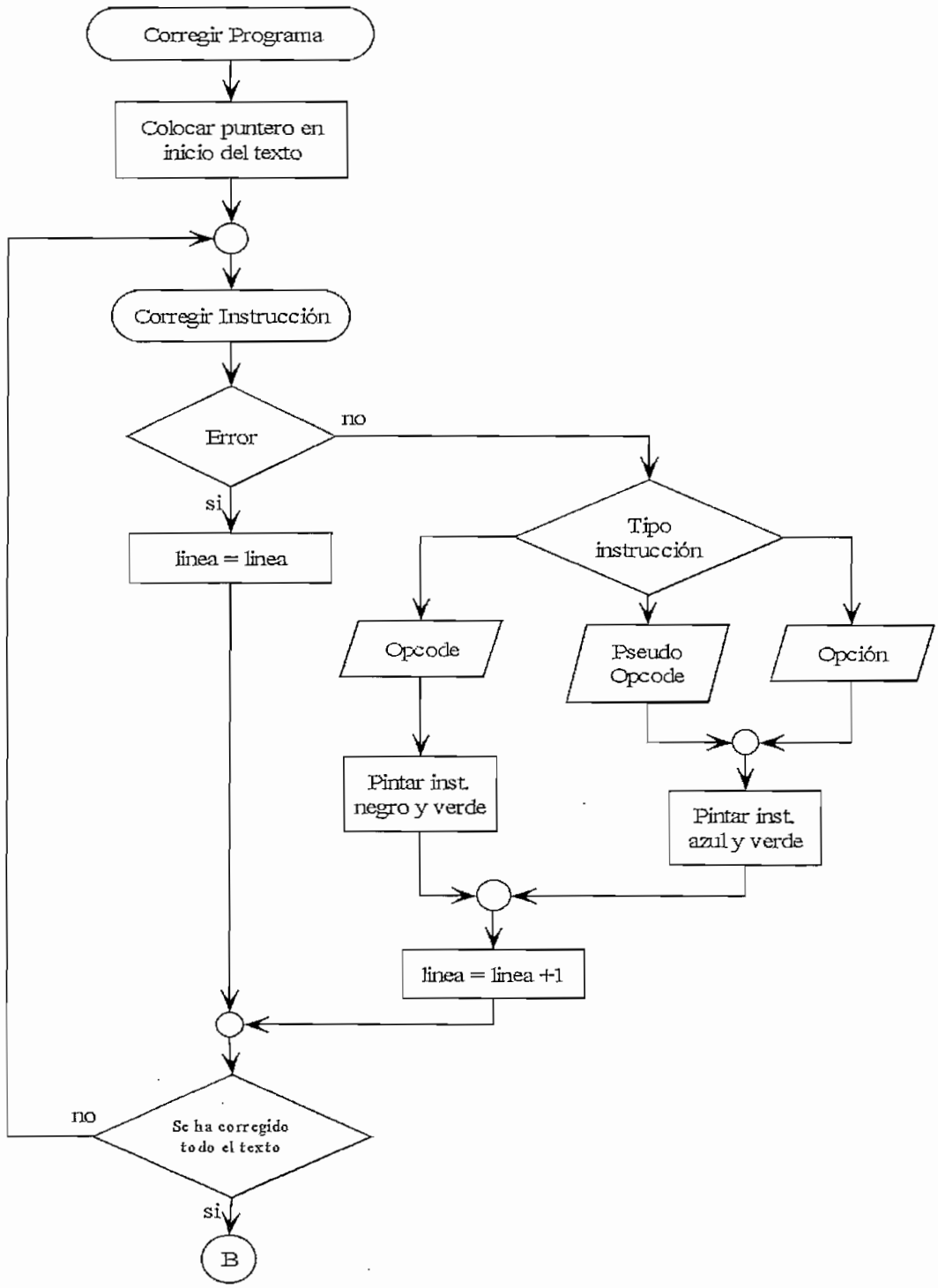

Figura 3.7 Diagrama de Flujo Corrección del Programa

3.2.8 *DIAGRAMA DE FLUJO CORRECCION DE INSTRUCCIÓN*

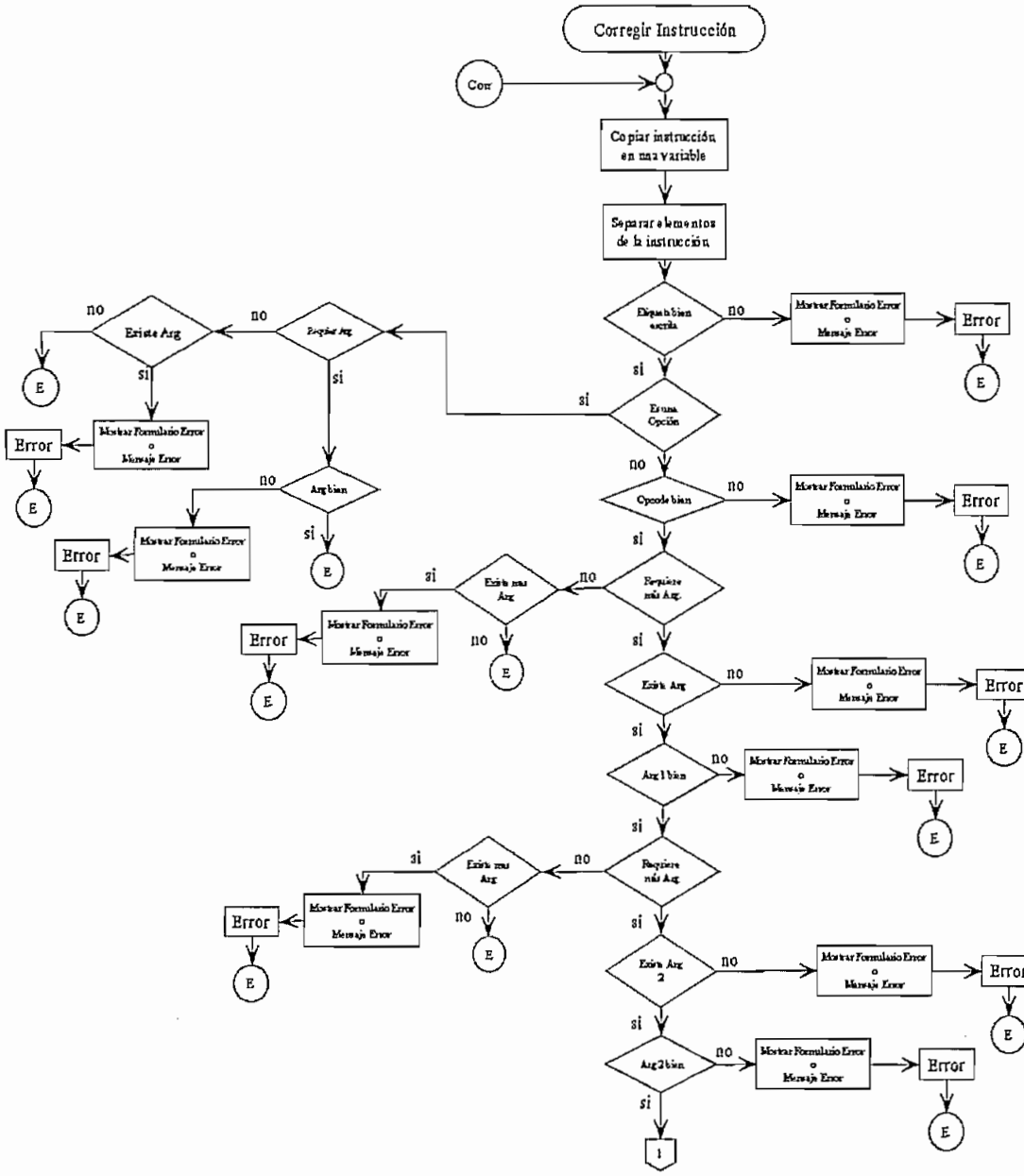

Figura 3.8 Diagrama de flujo Corrección de Instrucción

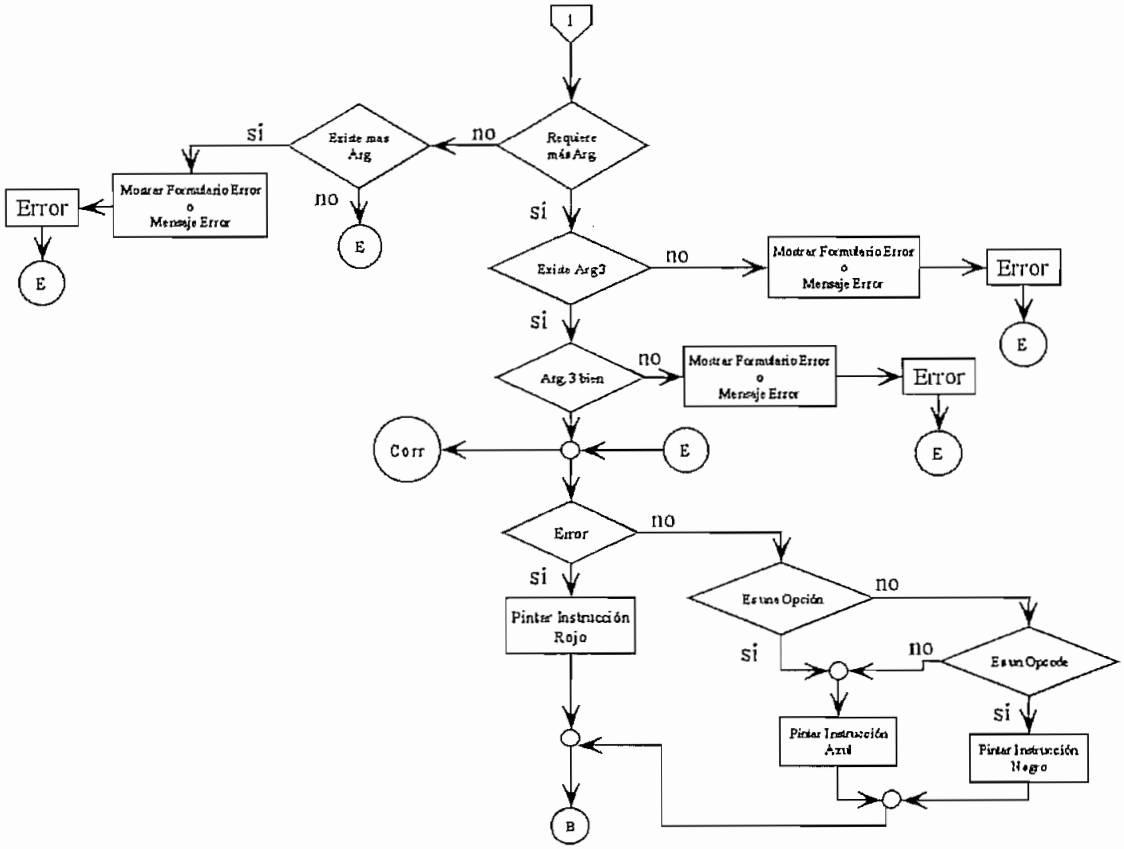

Figura 3.9 Diagrama de flujoCorrección de Instrucción (continuación)

3.2.9 DIAGRAMA DE FLUJO AYUDA

Al ubicarse sobre el menú Ayuda se puede observar las opciones: Acerca de. Esta opción permiten tener acceso al formulario Acerca de, donde se puede ver la descripción del programa. En la figura 3.10 puede verse este diagrama.

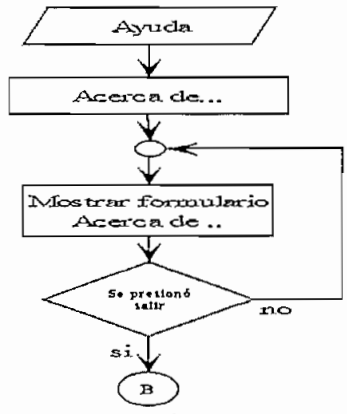

Figura 3.10 Diagrama de flujo Menú Ayuda

3.3 DESCRIPCION Y DESARROLLO DEL PROGRAMA.

3.3.1 DESCRIPCION DEL PROGRAMA

El programa tiene como base la forma de un editor de texto. Al ejecutar el programa se muestra una pantalla de presentación, la cual muestra información sobre el EDITOR INTELIGENTE. En la figura 3.11 puede verse dicha pantalla.

Figura 3.11 Pantalla de Presentación

Luego, cuando el programa está disponible al usuario, aparece la pantalla principal. Esta pantalla muestra la interfaz con la que el usuario interactuará en la edición del programa, en la figura 3.12 se muestra a la pantalla principal.

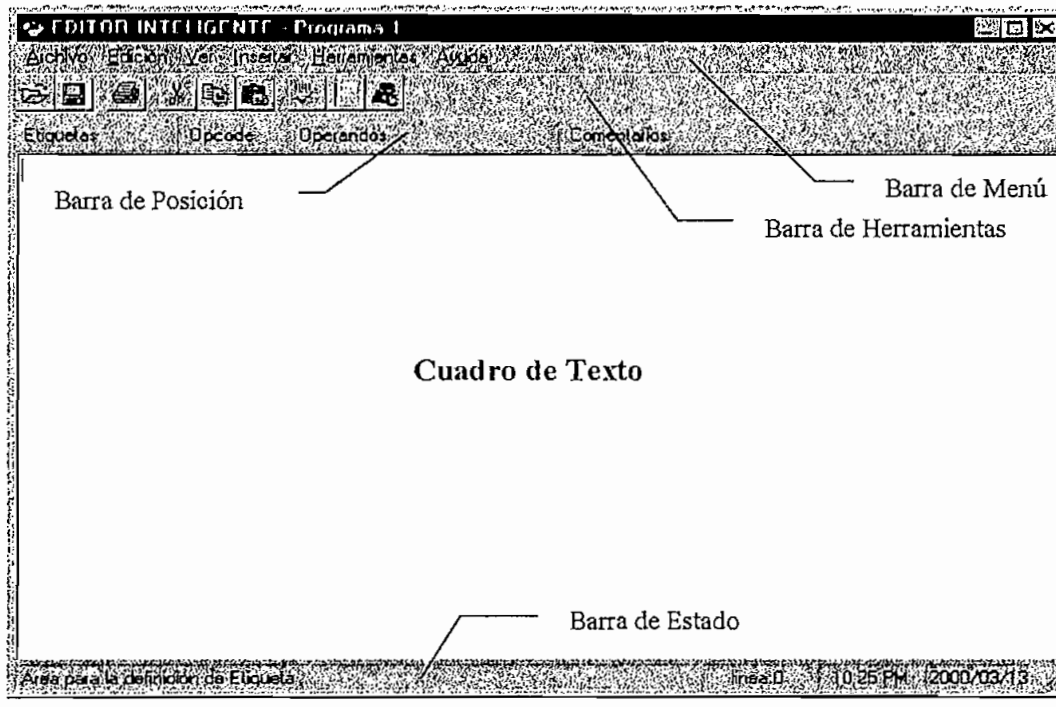

Figura 3.12 Pantalla Principal

Como se puede observar, en la pantalla principal existe cinco elementos necesarios para el desarrollo del programa. Aquí se encuentra: la Barra de Menú, la Barra de Herramientas, la Barra de Estado, la Barra de Posición y el Cuadro de Texto.

A excepción de la barra de Menú, las demás barras pueden o no estar presentes.

Sobre el cuadro de texto se editarán las instrucciones y pseudo instrucciones, que formarán el programa editado.

La Barra de Menú que permite acceder a varios submenús, estos son:

- *Submenú Abrir.*- Con las opciones: Abrir, Guardar, Guardar como, Imprimir, Salir, Ultimos Archivos.
- *Submenú Edición.*- Con las opciones: Cortar, Copiar, Pegar, Borrar, Seleccionar todo, Buscar, Buscar siguiente, Reemplazar.

- *Submenú Ver.*- Con las opciones: Barra de Herramientas, Barra de Estado y Barra de Posición.
- *Submenú Insertar.*- Con las opciones: @, #, \ y Otros caracteres.
- *Submenú Herramientas.*- Con las opciones: Corregir y Opciones de programación.
- *Submenú Ayuda.*- Con la opción: Acerca de...

En la figura 3.13 puede verse la Barra de Menú, mostrando las opciones de un submenú.

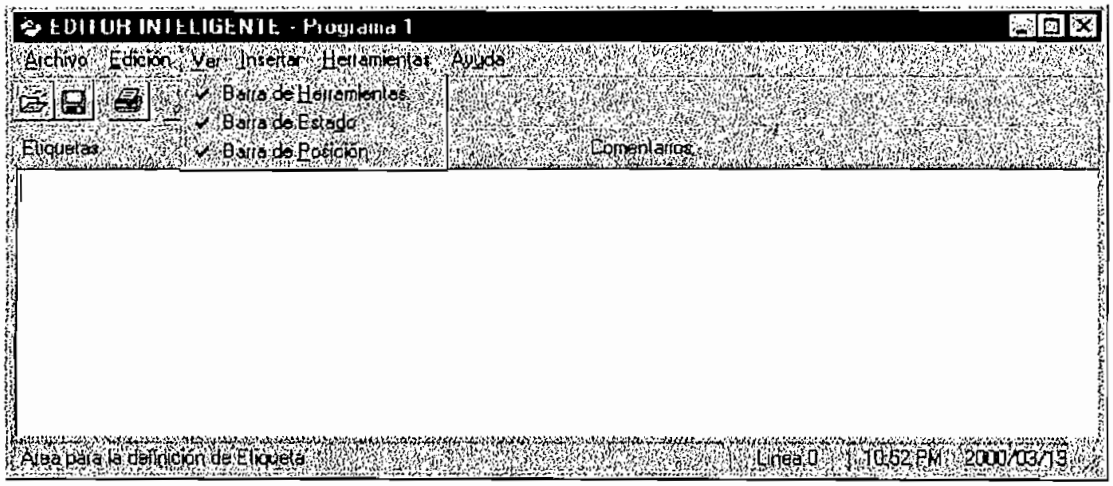

Figura 3.13 Barra de Menú

Existen tres barras adicionales a la barra de menú, estas se crearon con el fin de ayudar al usuario en la utilización del Editor. Estas barras son:

- *Barra de Herramientas.*- Muestra algunas de las opciones (las más comunes) que existen, en al barra de Menú, con el fin de agilitar el acceso a estas. Estas opciones estas definidas como botones sobre los cuales se debe hacer un clic.
- *Barra de Estado.*- Proporciona información sobre la edición de programa, así como otras ayudas como son: la fecha, la hora y la posición de línea del cursor.
- *Barra de Posición.*- Muestra los campos que existen en una instrucción así como el sitio en el que se ubican.

En la figura 3.12 puede observarse estas barras.

Durante la edición de las instrucciones, se presenta ayudas sobre los elementos que posiblemente pueden formar parte de una instrucción. Si la instrucción esta ingresándose en forma correcta, el Editor Inteligente sigue presentando esta ayuda caso contrario desaparece. Se puede seleccionar uno de estos elementos haciendo un clic sobre este.

3.3.2 DESARROLLO DEL PROGRAMA.

El Editor Inteligente está formado por 9 formularios, 5 módulos, una base de datos y un archivo de texto.

Los formularios se basan en la implementación de procedimientos, los cuales permiten tener una organización y optimización al máximo del programa.

Los formularios contienen procedimientos que dan respuesta, a los eventos que se generan, a continuación se hace una descripción de cada uno de ellos.

1. *Formulario Editor.frm.-* Es el más importante, pues, en el se producen todas las acciones que el usuario realiza. Esta formado por 42 procedimientos, los cuales, detectan todas las acciones que el editor puede efectuar. Estos procedimientos invocan a otros procedimientos o funciones que se han definido en los módulos.
2. *Formulario Presentar.frm.-* Es una pantalla de presentación, la cual, se muestra durante un tiempo, hasta que el Editor Inteligente se haya cargado. Presenta información referente al programa. No contiene procedimientos.
3. *Formulario Opciones.frm.-* Presenta las opciones que el usuario dispone, en la edición del programa. Está formado por 7 procedimientos, que determinan las acciones que el usuario realiza.
4. *Formulario Corregir.frm .-* Presenta mensajes sobre los errores que se producen, en la corrección del programa editado. También presenta la posible solución a los errores. Esta conformado por 7 procedimientos.
5. *Formulario Simbolos.frm.-* Presenta los caracteres ASCII, así como su valor en formato decimal. Esta formado por 5 procedimientos.

6. *Formulario Buscar.frm.*- Presenta un cuadro de diálogo, con el que se puede buscar alguna palabra en el texto. Está formado por 6 procedimientos que definen las acciones realizadas por el usuario.
7. *Formulario Reemplazar.frm.*- Es similar al formulario buscar, con la diferencia que, este contiene un botón y un cuadro de texto que sirve para reemplazarlo por en texto resaltado. Está formado por 8 procedimientos.
8. *Formulario AcercaDe.frm.*- Se encarga de presentar información sobre el programa. Esta formado por 1 procedimiento.

Los módulos contienen procedimientos y funciones, los cuales son invocados por los procedimientos de formulario; los procedimientos se los ha definido con un prefijo Proc para poder diferenciarlos de las funciones; las funciones no tienen prefijo. Los nombres de cada función o procedimiento, tienen relación al proceso que efectúan. Por ejemplo la función CorregirInstruccion, realiza el proceso de corregir la instrucción, devolviendo un valor que determina la existencia o no de errores.
Los módulos están definidos de forma tal que, las funciones o procedimientos que se encuentran en estos, tienen relación entre sí o con el procedimiento de formulario que la invoca. A continuación se detalla cada formulario.

1. *Modulo Principal.bas.*- Contiene los procedimientos y funciones que hacen relación a las opciones de la barra de Menú, está formado por 17 procedimientos y 3 funciones.
2. *Módulo Caracteres.bas.*- Contiene los procedimientos y funciones que hacen relación a los caracteres que se ingresan. Esta formado por 4 funciones y un procedimiento.
3. *Módulo Corregir.bas.*- Esta compuesto por una función y un procedimiento: La función es: *CorregirInstrucción,* realiza el proceso de corregir la instrucción, devolviendo un valor que especifica el tipo de línea ingresada. El procedimiento tiene el nombre: ProcCorregirErrores, este permite corrige los errores en todo el programa.

4. *Módulo Ayuda.bas.*- Contiene los procedimientos y funciones relacionados a la obtención de parámetros de la base de datos, para la comparación con los valores ingresados, como también para la presentación de la ayuda en pantalla. Está compuesto por 9 funciones y 1 procedimiento.
5. *Módulo Auxiliar.bas.*- Contiene procedimientos y funciones relacionados con todo el programa. Estos procedimientos o funciones; son sub funciones o sub procedimientos que realizan acciones pequeñas y son invocados por procedimientos de formulario. Esta formado por 13 procedimientos y 8 funciones.

Hacer una descripción de la acción que realiza cada una de las funciones o de los procedimientos sería muy extensa por lo cual se presenta en el Anexo C, el código del programa el cual contiene comentarios para el seguimiento de cada uno de ello.

El Editor Inteligente también contiene una base de datos, cuyo nombre es: Editor Inteligente. En esta se almacena, todas las instrucciones y pseudo instrucciones que puede definirse en la edición del programa. Está formada por 2 tablas:

1. *Tabla Instrucciones.*- Contiene información sobre las instrucciones del microcontrolador INTEL MCS-51/52. Esta tabla está formada por 8 campos. Cinco de estos campos contienen información sobre la estructura de las instrucciones. Dos campos sirven como ayuda para el Editor Inteligente. Y un campo de numeración de las filas. Como ejemplo se presenta 3 filas de esta tabla en la figura 3.14.

ID	ID2	Etiqueta	Opcode	Argumento1	Argumento2	Argumento3	Tipo
1	77	Opcional	MOV	A	#Dato		Opcode
106	67	Opcional	CJNE	A	#Dato	Etiqueta	Opcode
255	69	Etiqueta	EQU	Dirección			Seudoopcod

Figura 3.14 Ejemplos de la Tabla Instrucciones de la Base de datos.

2. *Tabla Argumentos.*- Contiene información sobre los tipos de argumentos que existen en una instrucción o pseudo instrucción. Esta formada por 4 campos.

Esta tabla sirve para determinar si es necesario mostrar un cuadro de texto en el caso de que sea argumento variable y pegar directamente el argumento, si es de tipo fijo. En la figura 3.15 se muestra ejemplos de esta tabla.

ID	TIPO	ARGUMENTO	INICIAL
0	Constante	AB	
3	Constante	@R1	
21	Variable	#Dato	#
26	Variable	Etiqueta	:

Figura 3.15 Ejemplos de la tabla Argumentos de la base de datos

El campo inicial, determina si el argumento se define con un carácter inicial.
En el Anexo D se muestra las tablas de la base de datos del Editor Inteligente.

Finalmente se tiene un archivo de texto, el cual contiene el Manual de Usuario del Editor Inteligente. Este archivo se lo puede encontrar en los discos de instalación del EDITOR INTELIGENTE.

Cabe señalar que en el proyecto, también se definieron *variables y constantes*. Las variables definidas fueron de tipo globales como también de tipo locales y variables de formulario.
Las variables locales fueron definidas en cada procedimiento o función, las variables de formulario se definieron en el procedimiento *Declaration* de los formularios y las variables globales fueron definidas en procedimiento *Declaraciones* de algunos módulos.

CAPITULO IV

4 PRUEBAS Y RESULTADOS

4.1 EJEMPLOS DE PROGRAMACION

A continuación se presenta ejemplos de cómo se realiza la programación, y el formato que estas tendrán.

4.1.1 EJEMPLO 1 (Programa sin errores)

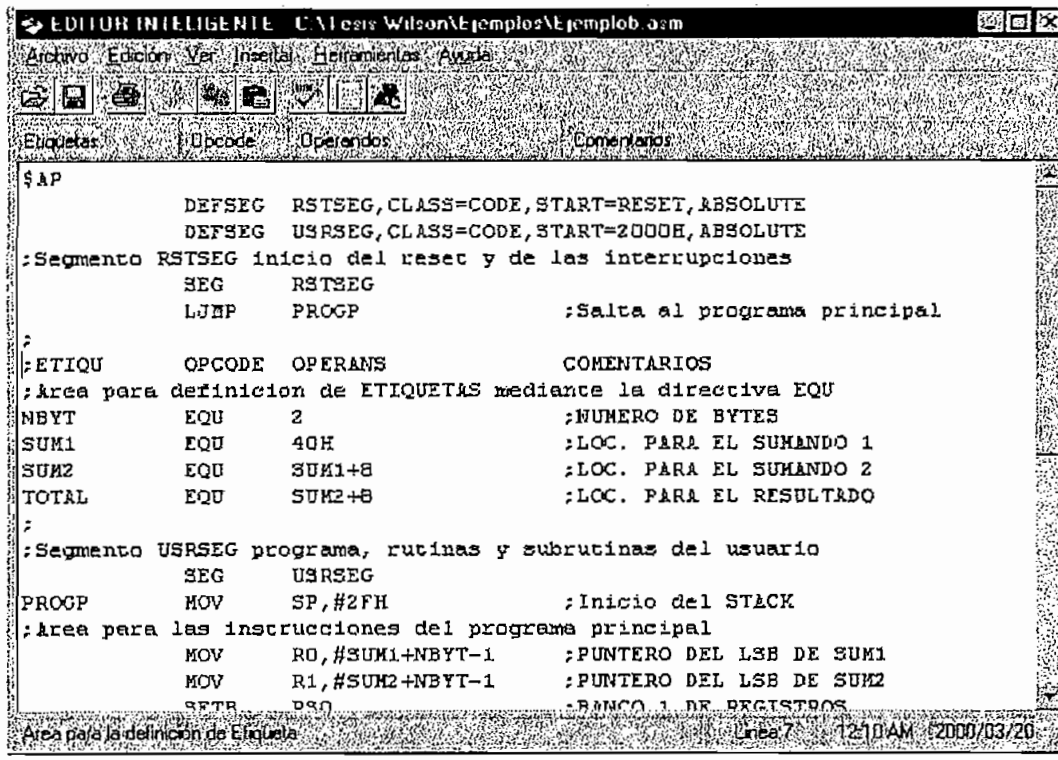

Figura 4.1 Programa sin Errores

El Editor Inteligente, el momento de cargar el programa, revisa cada una de las líneas de programación y si no halla errores de sintaxis; pinta las líneas dependiendo del tipo de línea que le corresponda.

En la tabla de la figura 4.2 puede observarse el tipo de línea de programa y el color con que se pinta.

Tipo de Línea	Color
Instrucción	Negro
Pseudo instrucción y Opciones	Azul
Comentario	Verde

Figura 4.2 Tabla de Tipo de línea de programación

Como pude verse, un programa esta formado por instrucciones y pseudo instrucciones, etiquetas y comentarios.

Las líneas de instrucción y de pseudo código, también pueden contener comentarios; estos comentarios se pintan de color verde.

Las Etiquetas toman el color de la línea a la que pertenecen o se pintan de negro sí están solas.

4.1.2 EJEMPLO 2 (Programa con Errores)

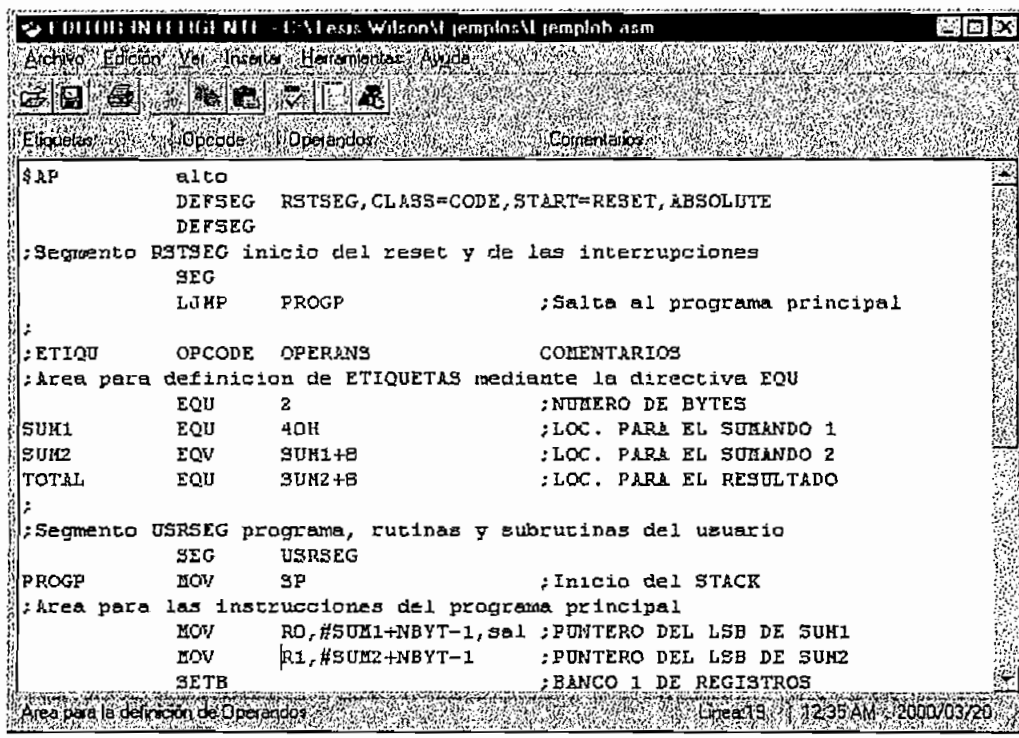

Figura 4.3 Programa con Errores de sintaxis

Cuando el EDITOR INTELIGENTE, al revisar la línea de programación, encuentra que no cumple con las reglas de sintaxis, pinta la línea de color rojo.

Como puede observarse en la figura 4.3, los errores de sintaxis pueden producirse por múltiples causa. Se nombrará el tipo de error producido en el orden en el que se halla en el gráfico.

- La opción no requiere más argumentos
- La pseudo instrucción requieres de argumentos.
- La línea de comentario debe empezar con punto y coma.
- La pseudo instrucción requiere de argumentos.
- La pseudo instrucción requiere etiqueta
- El pseudo opcode está mal escrito.
- La instrucción requiere más argumentos.
- La instrucción tiene muchos argumentos.
- La instrucción no tiene opcode escrito.
- La instrucción requiere argumentos.

4.1.3 EJEMPLO 3 (Ingreso de una Instrucción)

A continuación se detalla, paso a paso el proceso de edición de una línea de programación.

a) *Ingreso de la Etiqueta*

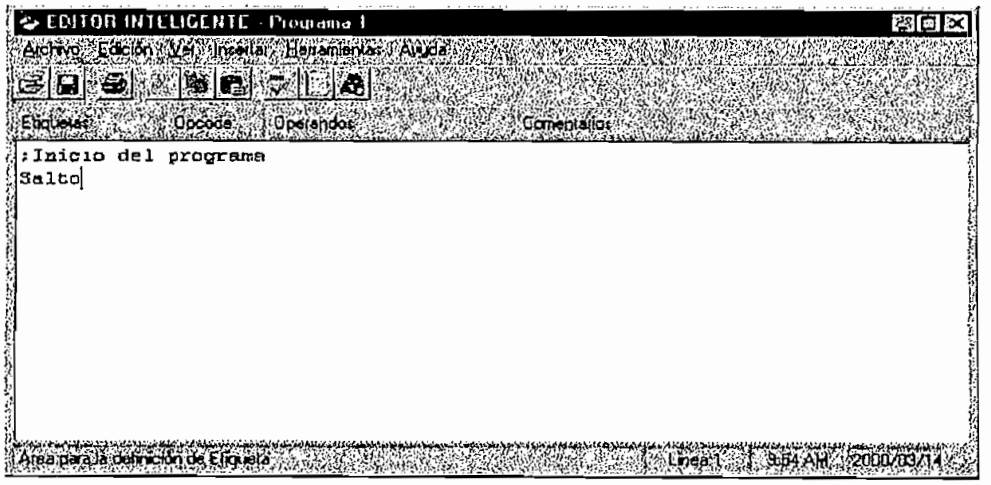

Figura 4.4 Ingreso de una Etiqueta

Una etiqueta puede o no estar presente en la línea de programación. A medida que se ingresa, se detecta si existe o no errores. Con el tabulador se pasa al siguiente campo. La barra de Estado presenta mensajes sobre el avance de la edición de la línea de programación. En la figura 4.4 se presenta un ejemplo de ingreso de Etiqueta.

b) *Ingreso del Opcode*

Al ingresar el primer carácter en el campo establecido para los Opcodes; el Editor Inteligente busca en el campo Opcode de la base de datos, si existen elementos que empiecen con este carácter, si hay elementos, se muestran todos estos en una lista. En la barra de estado puede observarse el mensaje correspondiente a la acción que se debe realizar.

A continuación se puede observar el resultado este proceso en al figura 4.5.

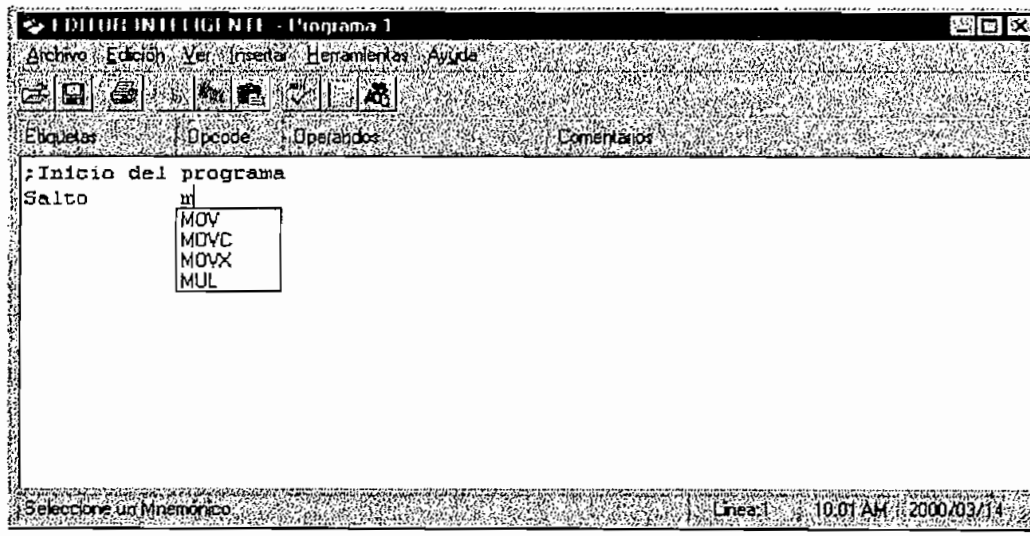

Figura 4.5 Ingreso del Opcode

c) *Elección o ingreso de un Opcode.*

En la lista se puede escoger uno de estos elementos, haciendo un clic sobre el Opcode que se desea pegar. El Opcode también puede ser ingresado por teclado, en este caso, el Editor Inteligente revisa si el Opcode está bien escrito para mostrar la siguiente ayuda. El presionar la Tecla Tab determina el fin del Opcode y salto al campo de Argumentos. En la figura 4.6 se muestra seleccionado un elemento de la lista

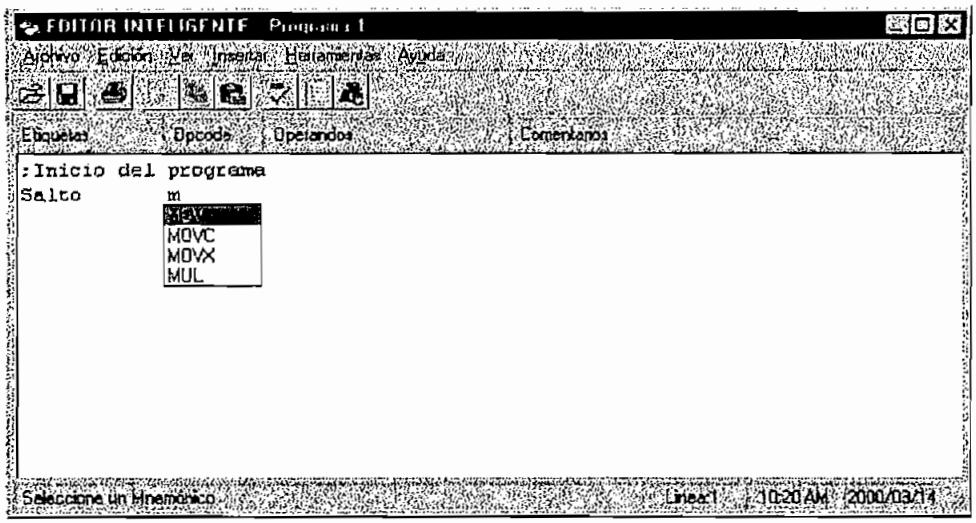

Figura 4.6 Elección de un Opcode

d) *Ingreso del Primer Argumento*

Si el Opcode ingresado requiere más parámetros, se presenta una nueva lista con los Argumentos que pueden ingresarse. Como en el caso de los Opcodes, este también puede seleccionarse un elemento o ingresar por teclado.

Luego de ingresar el Argumento si se requiere más parámetros, el Editor Inteligente detecta esto y presenta una nueva lista con Argumentos. En la figura 4.7 se presenta el resultado de ingresar un Opcode. Note que en todos los casos siempre existe un mensaje sobre la acción que debe realizar, la cual se muestra en la barra de Estado

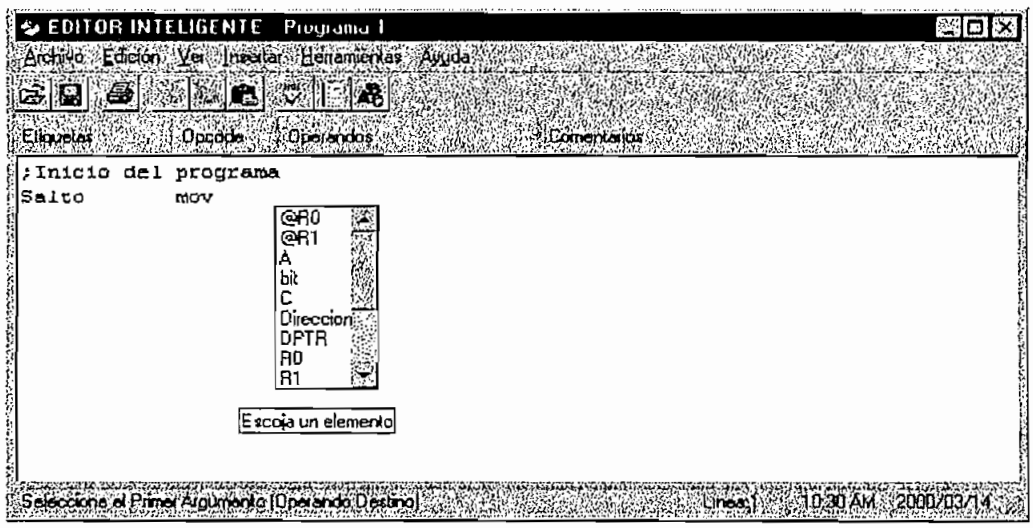

Figura 4.7 Ingreso del primer argumento.

e) *Ingreso del Segundo Argumento.*

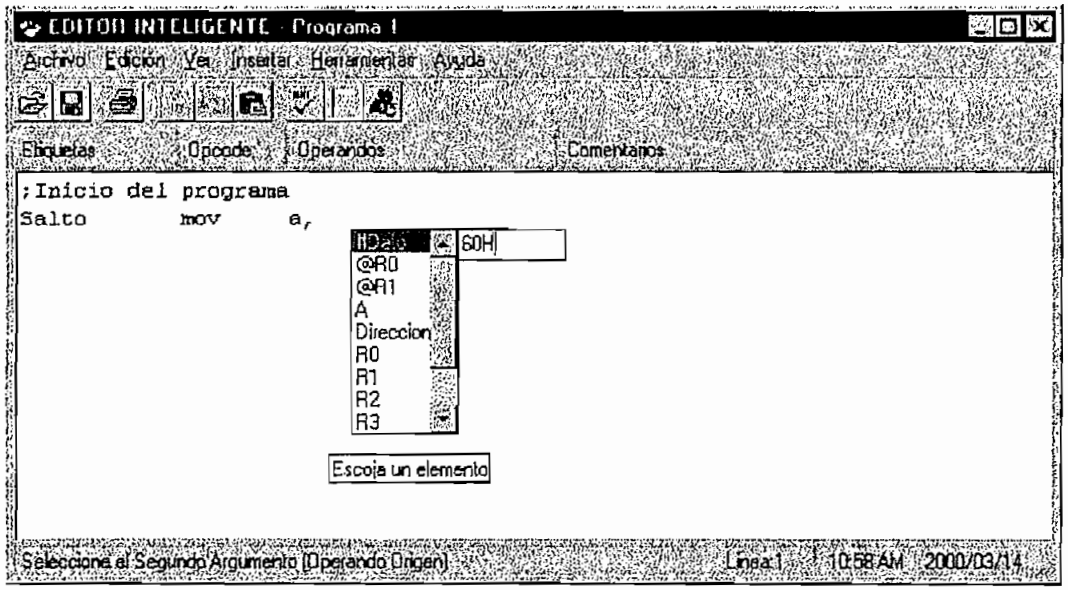

Figura 4.8 Ingreso del Segundo Argumento

Igual que en el caso del primer Argumento, se escoge un elemento o se ingresa por teclado.

Cabe señalar que existen algunos argumentos que requieren del ingreso de algún valor específico, por lo que se presentará un cuadro de texto adicional para el ingreso de este valor. Esto puede observarse en la figura 4.8.

f) *Ingreso de Comentarios*

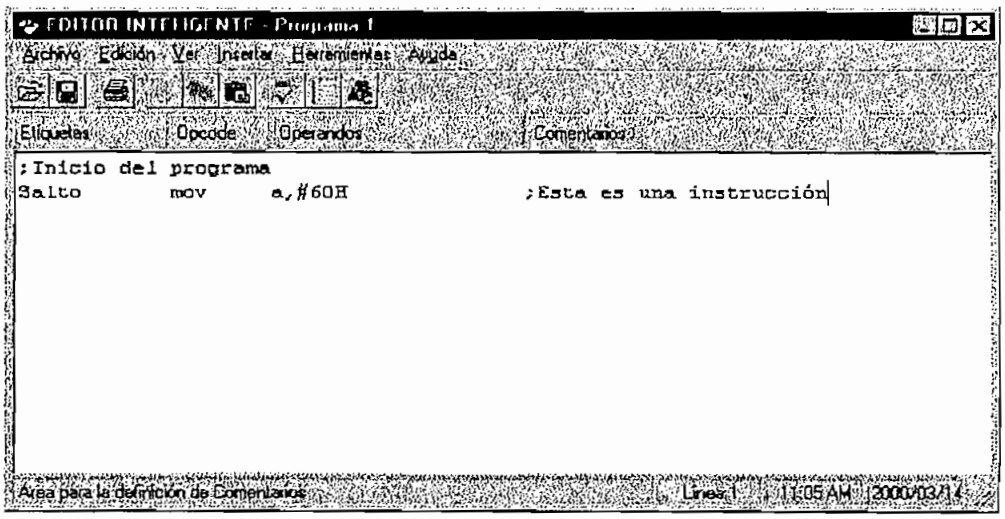

Figura 4.9 Ingreso de comentarios

El Editor Inteligente, luego de ingresar los argumentos, sitúa al cursor en el área de comentarios. Un comentario debe empezar por punto y como (;)
Luego de ingresar todos estos elementos, se debe presionar Enter para saltar a la nueva línea. En este proceso se vuelve a revisar la instrucción y se cambian los colores de esta, en base a los criterios descritos anteriormente.

En la figura 4.10 se puede ver la línea que se ha editando, luego de presionar Enter.

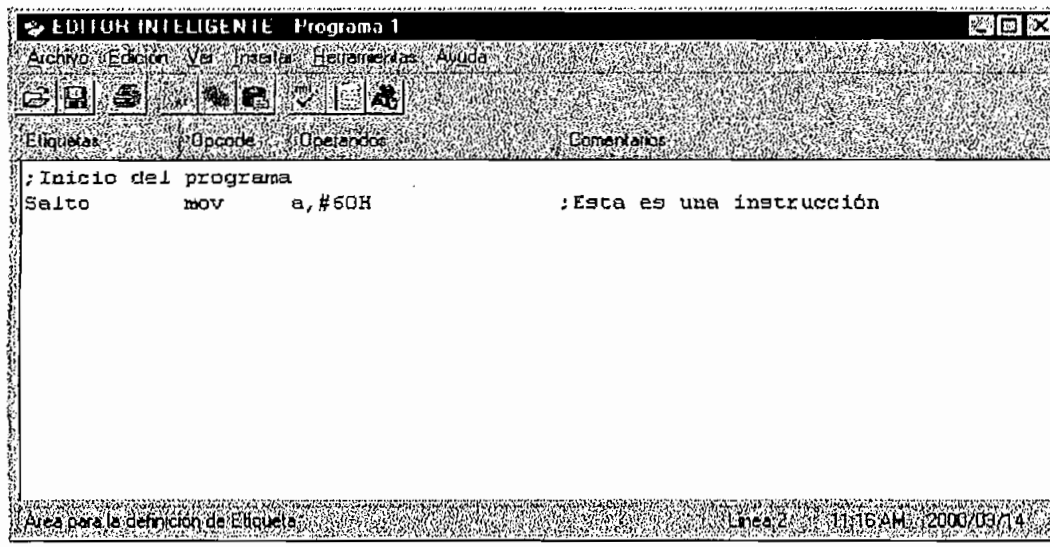

Figura 4.10 Instrucción completa

Si en cualquier momento se ingresa algún carácter o palabra mal escrita, el Editor Inteligente automáticamente deja de mostrar la ayuda.
El resultado de ingresar un opcode mal escrito se muestra en la figura 4.11

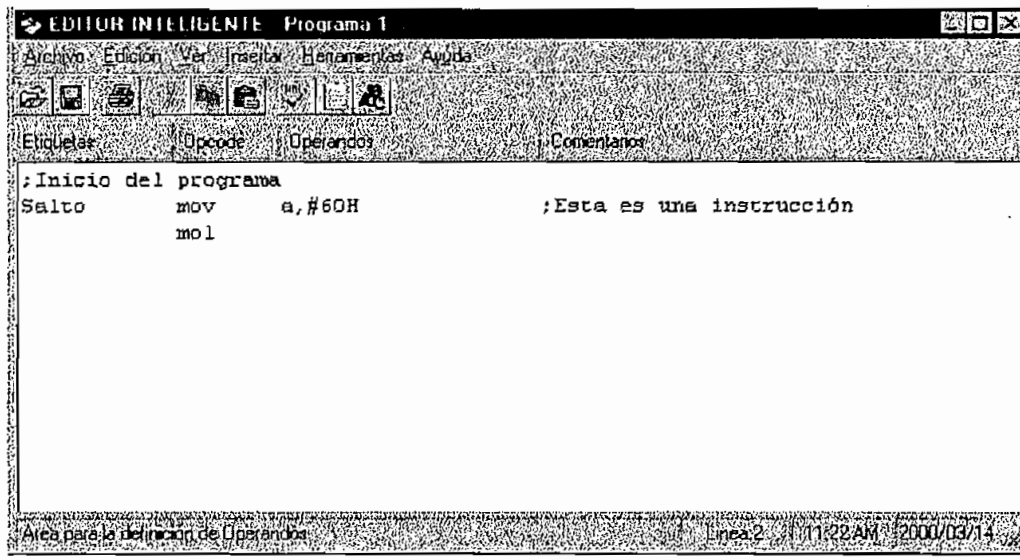

Figura 4.11 Instrucción con Opcode mal escrito

4.2 MENSAJES Y CORRECCION DE ERRORES

4.2.1 MENSAJES DE ERRORES.

El Editor Inteligente presenta dos formas de mostrar Mensajes de Error:

a) Mensajes de Error en la Barra de Estado

b) Mensajes de Error al terminar una instrucción.

4.2.1.1 MENSAJES DE ERROR EN LA BARRA DE ESTADO

Durante la edición de una instrucción se puede producir errores, estos son presentados en la Barra de Estado, por varios segundos, para luego continuar con la ejecución de la línea.

Estos mensajes sirven para eliminar el error antes de continuar editando la línea de programación.

A continuación se presenta varios mensajes de error que pueden producirse en el ingreso de una instrucción.

- *Error en la Etiqueta.*

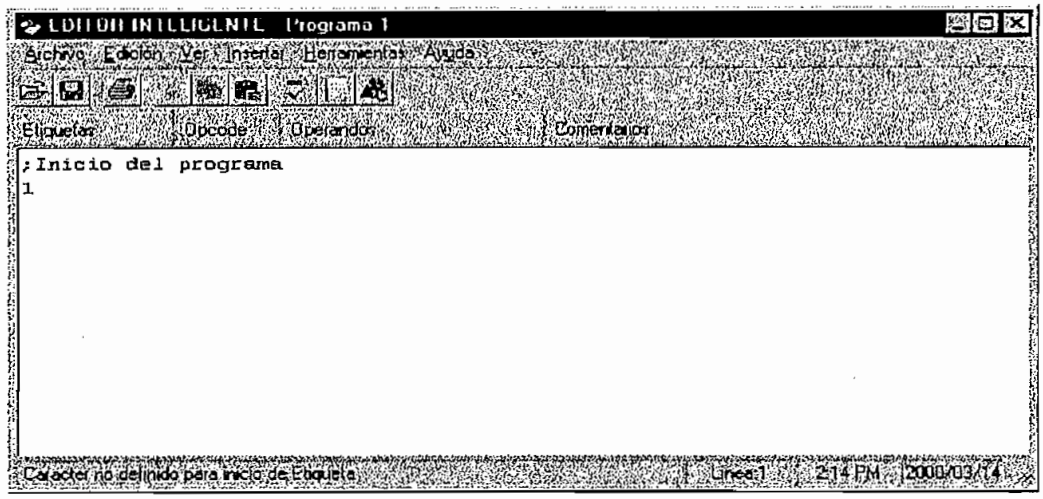

Figura 4.12 Error en la Etiqueta

En la barra de estado puede observarse el mensaje generado.

- *Error en la definición del Opcode*

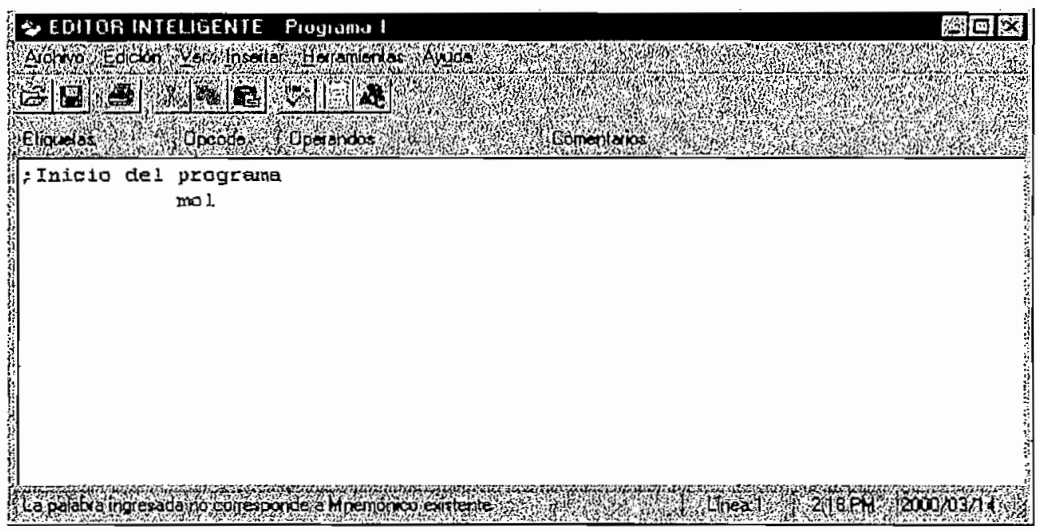

Figura 4.13 Error en el Opcode

Igual que en caso anterior, en la barra de estado se presenta el mensaje del error generado.

- *Error en la definición de los Argumentos*

En la figura 4.14 se ve el mensaje de error generado.

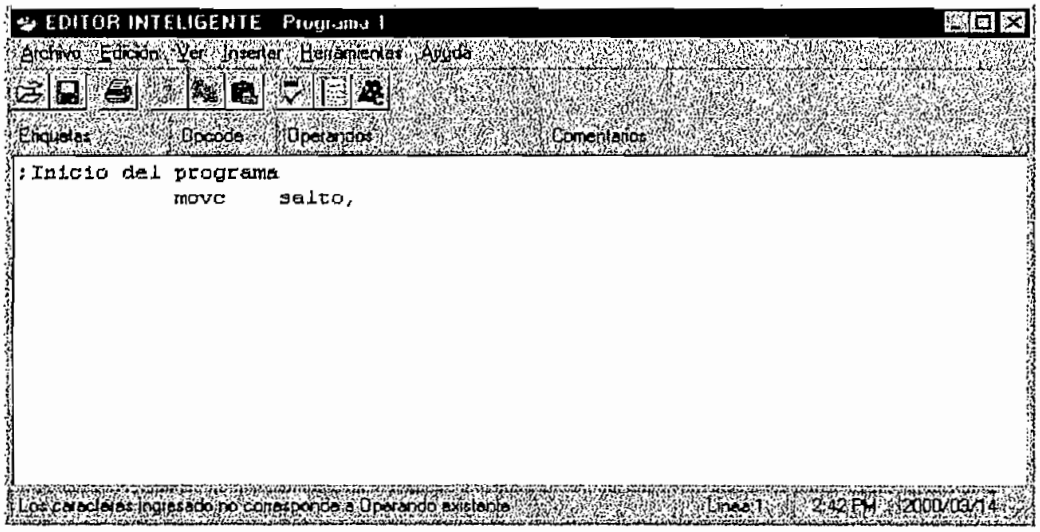

Figura 4.14 Error en los Argumentos

4.2.1.2 MENSAJES DE ERROR AL TERMINAR UNA INSTRUCCIÓN.

Una de las opciones del Editor Inteligente es: "Corregir al terminar la instrucción". Al seleccionar esta Opción; si sé a producido errores en el transcurso de la edición de la instrucción, se presenta un cuadro con el mensaje de error correspondiente. El error se resaltará mientras se muestra el mensaje. Cabe señalar que solo se presentará este cuadro de mensaje luego de presionar ENTER.

A continuación se muestra dos ejemplos:

a) Error en la Etiqueta

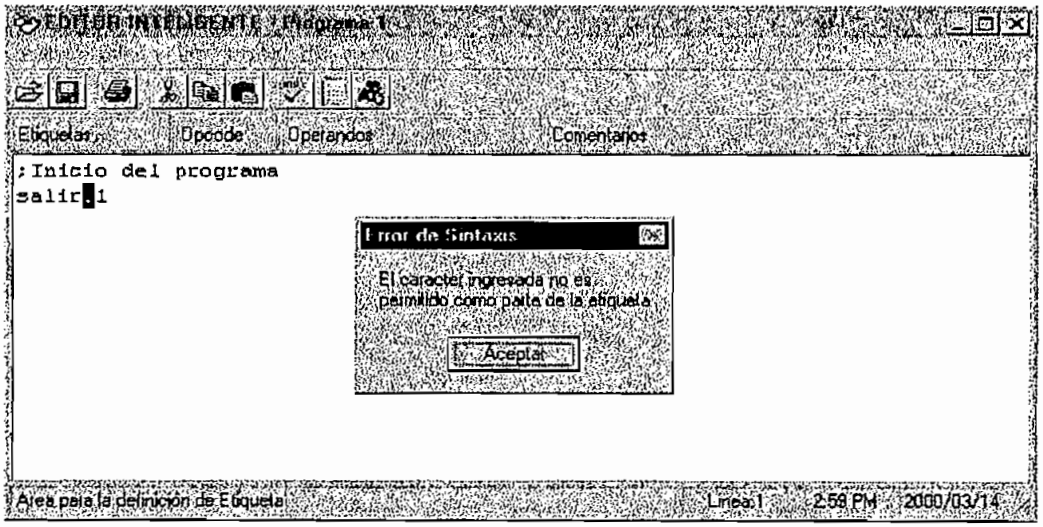

Figura 4.15 Ejemplo de Mensajes de Error

b) Instrucción incompleta

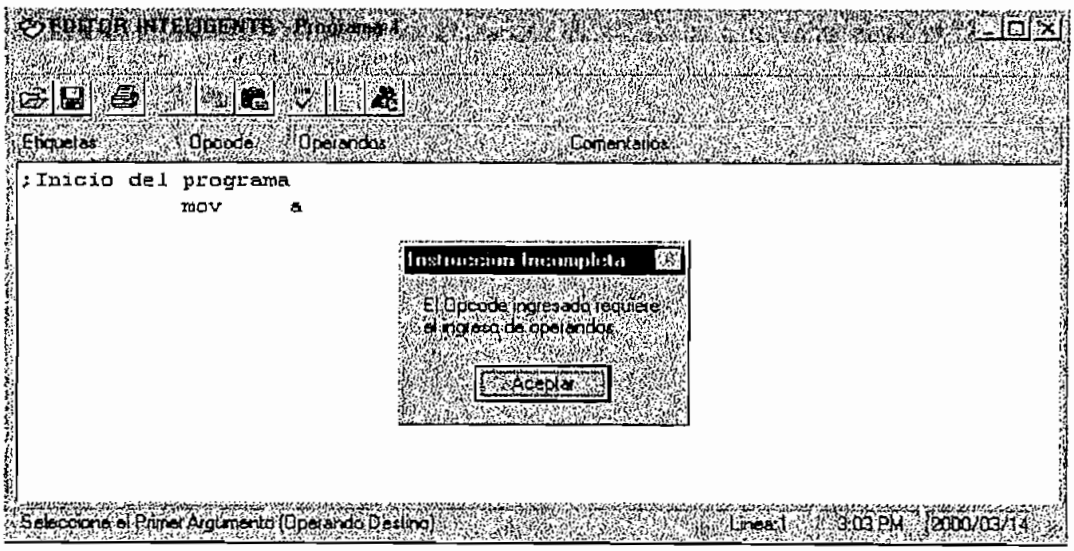

Figura 4.16 Ejemplo de Mensajes de Error

4.2.2 CORRECCION DE ERRORES.

El Editor Inteligente ofrece también la posibilidad de corregir los errores que se han producido en la edición del programa.

Esta opción permite identificar el error existente y la posibilidad de corregirlo.

Presenta un cuadro de dialogo con varios botones y otros elementos que pueden ayudar a corregir la instrucción.

A continuación se mostrará varios ejemplos que determinan la forma como se corrige una instrucción.

a) *Línea con Error en una pseudo instrucción*

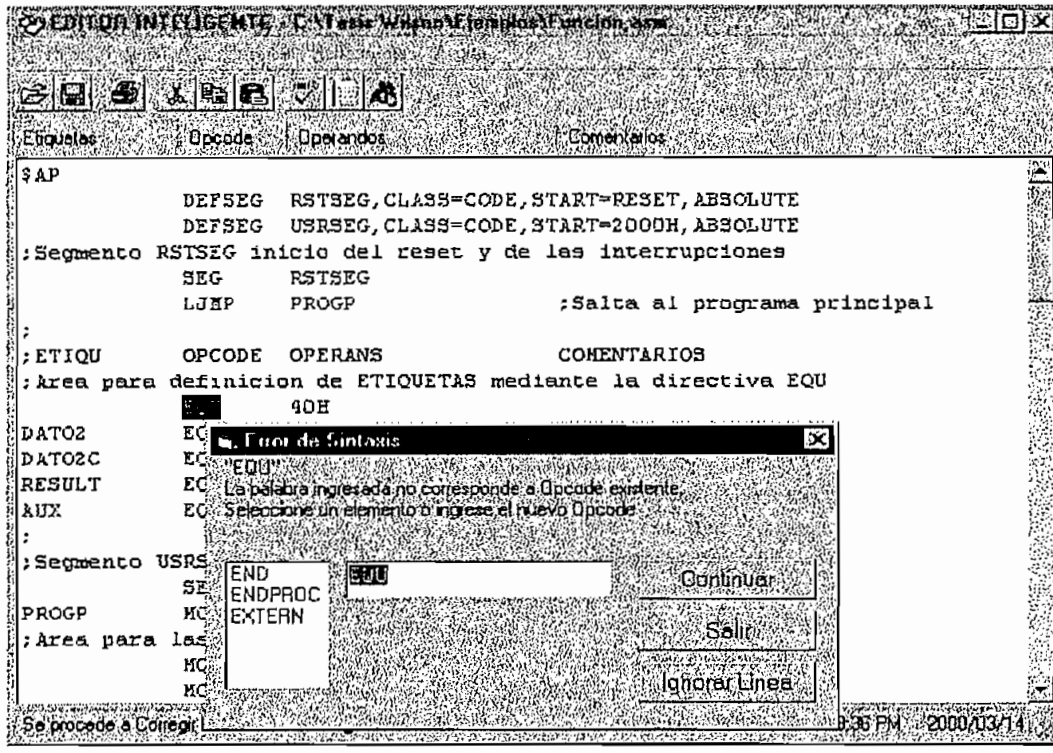

Figura 4.17 Ejemplo de Corrección de Errores

"EQU" es un Pseudo Opcode. Por lo que en este caso se deberá ignorar la línea para luego definir la etiqueta que esta pseudo instrucción requiere.

b) *Linea con Error en el pseudo Opcode*

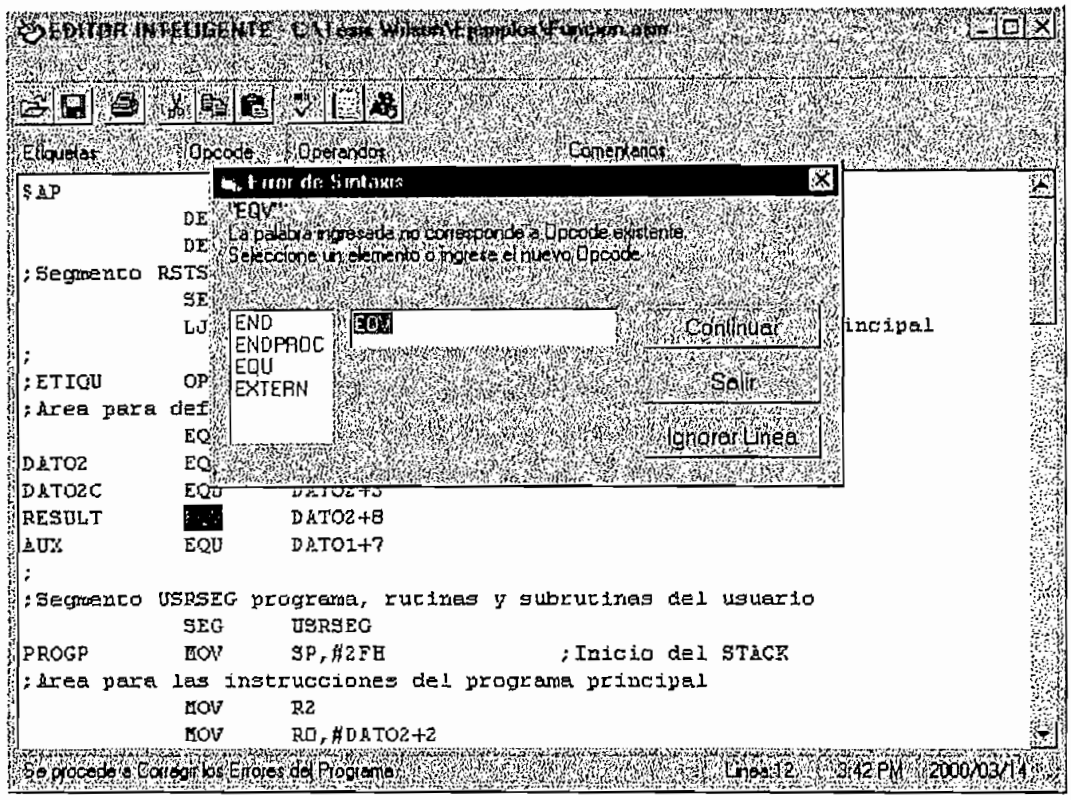

Figura 4.18 Ejemplo de Corrección de Errores

"EQV" no es un pseudo opcode, como puede verse, se ha cometido error al escribir "V" por "U". Puede seleccionarse en la lista que aparece la función correcta. Esta reemplazará a la palabra seleccionada y continuará la corrección.

c) *Línea con Error en una instrucción.*

En este caso, la instrucción requiere el ingreso de más parámetros; por lo que se puede seleccionar entre los elementos que existen en la lista y pegarlos o ingresar el valor requerido en el cuadro de texto y presionar continuar. En la figura 4.19 Se observa este tipo de error.

Estos son algunos de los errores que pueden producirse en la edición del programa.

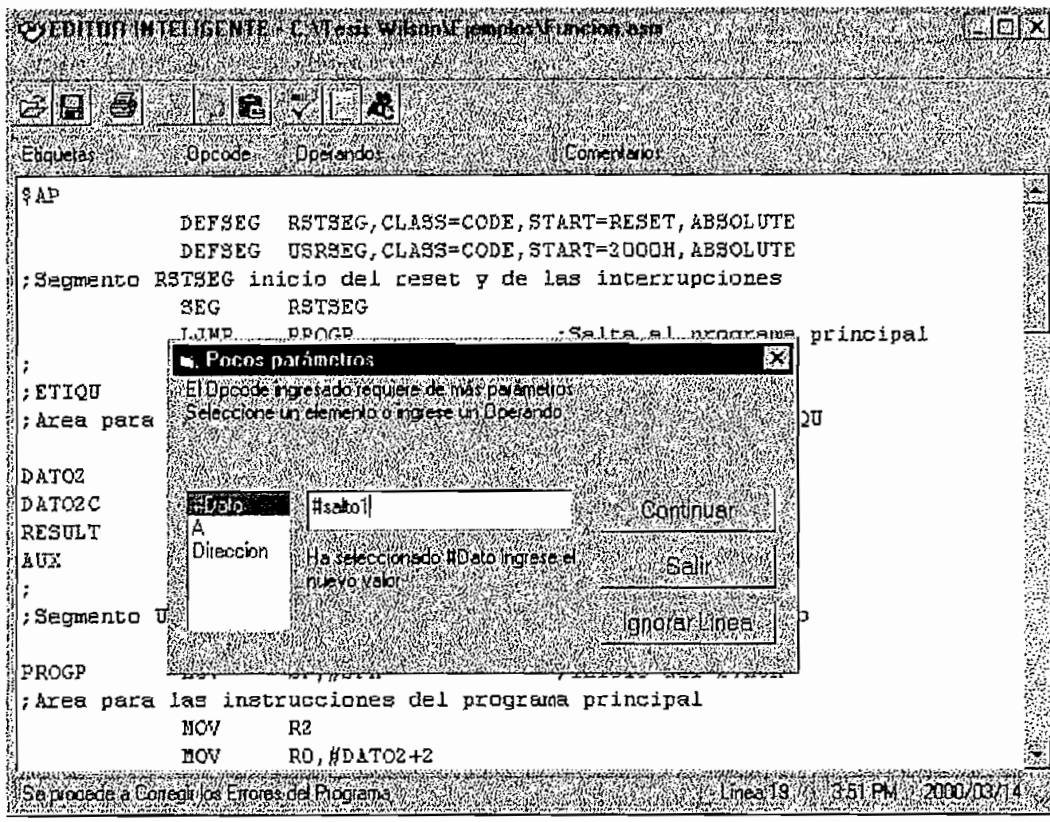

Figura 4.19 Ejemplo de Corrección de Errores

4.3 OTRAS AYUDAS DEL EDITOR.

El Editor Inteligente posee otras ayudas que facilitan el uso de este. Entre estas ayudas tenemos:

- Buscar y Reemplazar.
- Insertar Caracteres
- Opciones de programación.

4.3.1 BUSCAR Y REEMPLAZAR.

Esta opción permite al usuario buscar y también reemplazar, algún conjunto de caracteres o palabras que existen en el texto que se edita.

Se trata de dos opciones separadas, pero están relacionadas entre sí, pues cumplen funciones parecidas.

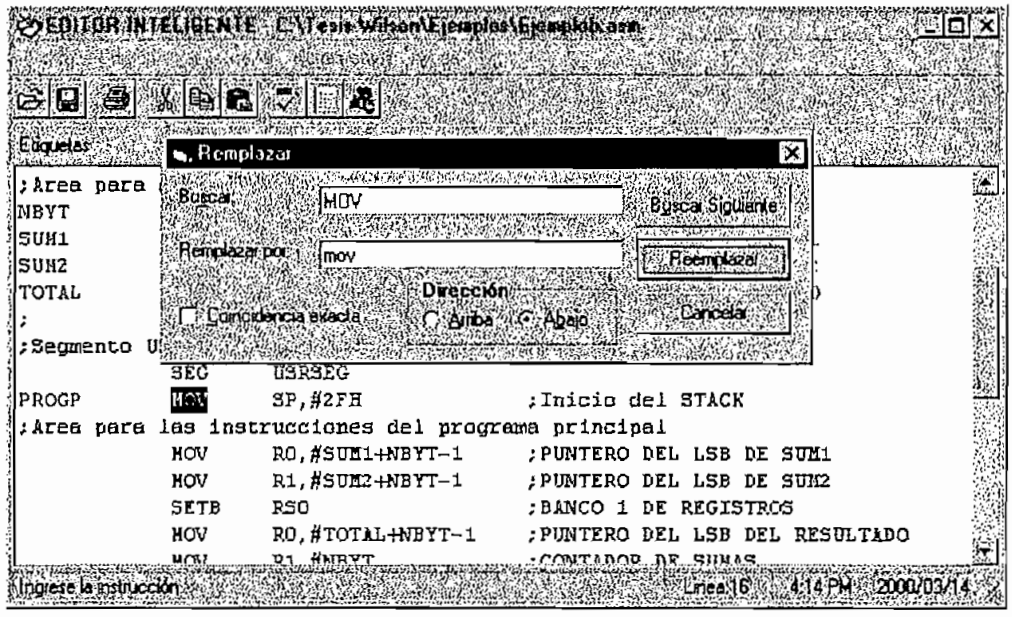

Figura 4.20 Opción Reemplazar

En la figura 4.20 se muestra ejemplo de la utilización de está herramienta.

Puede observarse que cuando encuentra la palabra, la resalta y puede reemplazarse o a su vez continuar con la búsqueda.

4.3.2 INSERTAR CARACTERES.

En ciertas ocasiones es necesario insertar algunos caracteres especiales, o conocer el valor en código ASCII de los caracteres imprimibles. Pensando en ello, se creó en el Editor Inteligente un formulario que contiene un conjunto de caracteres que pueden ser insertados en el Cuadro de texto.

A continuación en el gráfico de la figura 4.21 se observa este formulario.

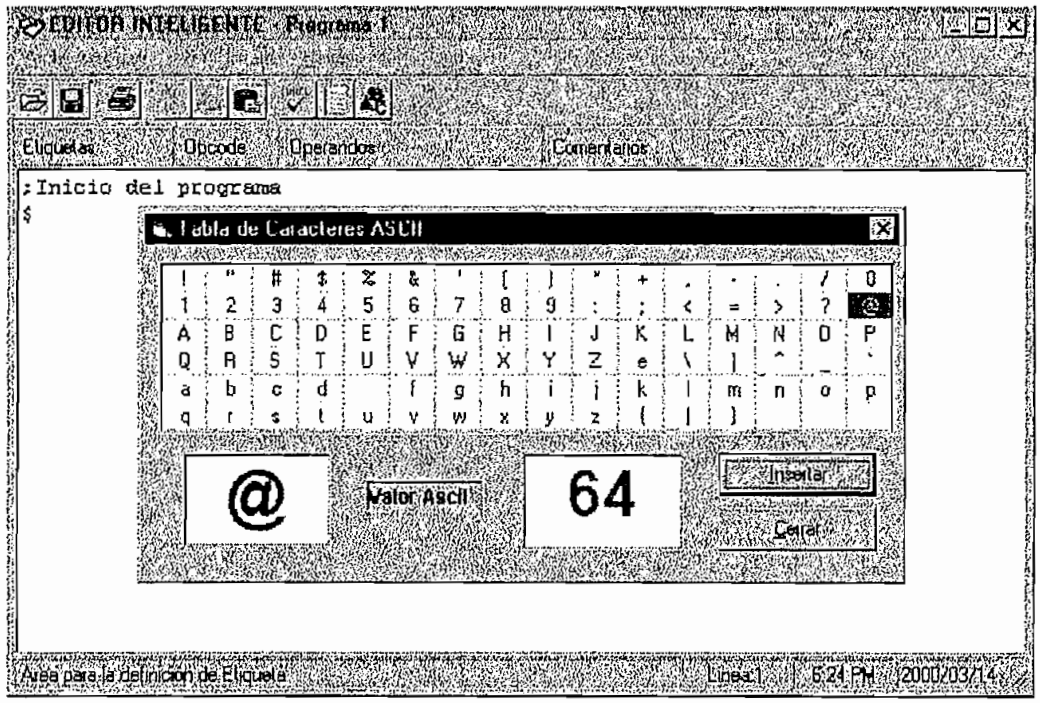

Figura 4.21 Insertar caracteres

4.3.3 OPCIONES DE PROGRAMACION

Finalmente tenemos el formulario Opciones de Programación.

En este formulario se encuentran 3 campos, los cuales permiten al usuario definir la forma en la que desea trabaja en el programa.

Estos campos son:

a) *Mostrar ayuda en pantalla.*- Hace referencia ha si se presentan las listas que contiene los elementos relacionados con la instrucción.

b) Uso del Ratón y teclado.- Permite utilizar solamente el Ratón o el Ratón y el teclado, se define de esta forma para prestar mayor agilidad en el uso del ratón.

c) Corregir Errores del Programa.- Hace referencia, a sí se permite ingresar solo caracteres válidos (es decir, si se produce un error se lo elimina), corregir al final de la instrucción (muestra el mensaje de error y no permite saltar a la siguiente

línea hasta no corregir el error) y corregir al terminar el programa (se puede ingresar cualquier carácter).

En la figura 4.22 se muestra este cuadro de opciones.

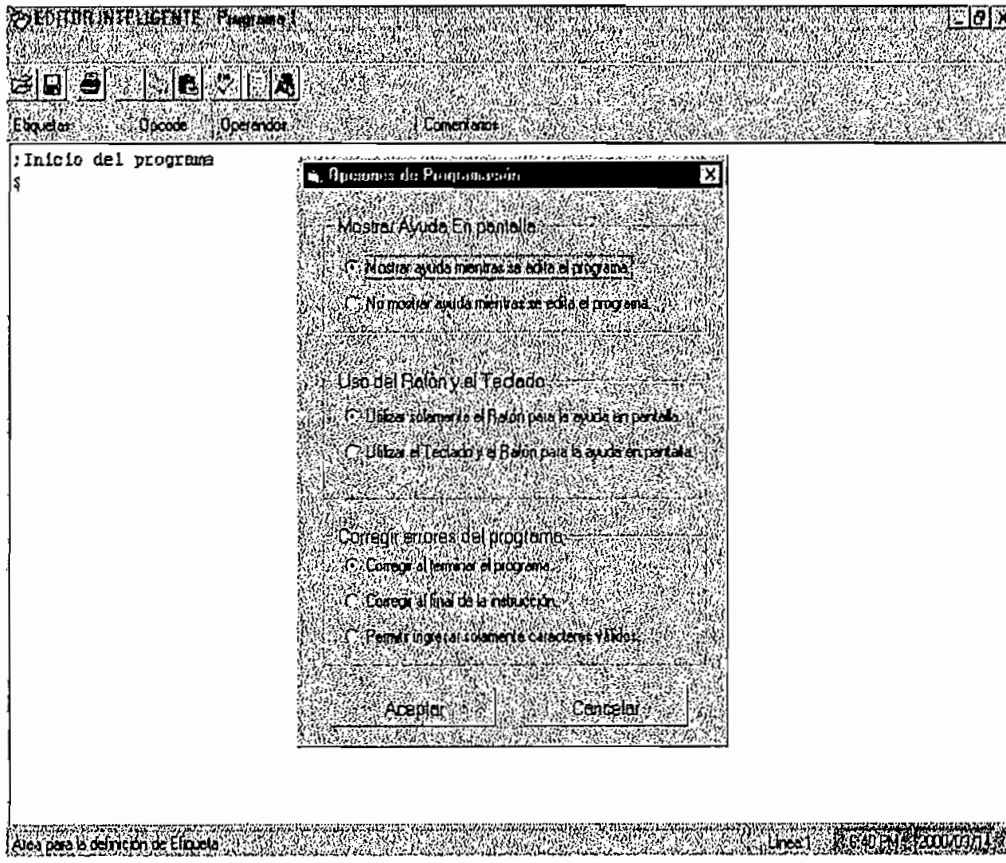

Figura 4.22 Opciones de programación

CAPITULO V

5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

En el proceso de cargar un archivo, el EDITOR INTELIGENTE se demora algunos minutos, esto se debe a que realiza varias acciones tales como: alineación, pintado y cheque de las instrucciones del archivo. Se vio en la necesidad de realizar estas acciones debido a las siguientes causas:

- El EDITOR INTELIGENTE permite abrir archivos de extensión ASM., los cuales son archivos de texto, pudiendo estos haber sido escritos en cualquier Editor de texto, por lo que las características de tabulación pueden ser diferentes; por esta razón fue necesario incorporar en el Editor un procedimiento de alineación del archivo.
- El proceso de pintado de las instrucciones se debe a que para el usuario es más fácil identificar los errores de sintaxis o el tipo de línea escrita, mediante la visualización de estas.
- El cheque de las instrucciones se lo realiza para poder pintar las instrucciones.

En la edición de una línea de programación, el EDITOR INTELIGENTE está continuamente chequeando la instrucción, esto hace que en ciertas ocasiones el cursor este en movimiento o se demore un poco en responder.

En el EDITOR INTELIGENTE se procuró cubrir la gran mayoría de errores de sintaxis, mediante pruebas continuas de edición de programas; sin dejar de lado que pueden producirse errores que estuvieron fuera de mi imaginación.

El juego de instrucciones se almacenó en una base de datos, con el fin de permitir expandir la edición de programas del EDITOR INTELIGENTE, hacia otros tipos de microcontroladores que tengan características similares en el formato de sus instrucciones.

5.2 RECOMENDACIONES

El EDITOR INTELIGENTE se desarrollo con el propósito de ser una guía en el proceso de escritura de una instrucción, como también presentar ayudas mientras se edita dicha línea, por lo que es aconsejable continuar en esta misma línea para otros tipos de microcontroladores, ya que facilita el manejo y permite una detección rápida de errores de sintaxis.

En el EDITOR INTELIGENTE no se utiliza para nada el botón derecho del ratón por lo que se recomienda utilizarlo para presentar otro tipo de ayudas como mostrar la acción que realizará una determinada instrucción.

Una buena idea sería, extender al EDITOR INTELIGENTE con el propósito de incorporar un ensamblador, lo cual agilitaría aún más la creación de proyectos para microcontroladores como también la incorporación de un simulador de instrucciones en un solo paquete.

ANEXO A

OPCIONES DEL AVMAC51

AVMAC51 Options				
Listing Control Options				
Name	Abbr	Primary?	Arg. Type	Description
PRINT	PR	Yes.	Filename	Sets the name of the list file. The default is the name of the source file with the suffix '.PRN'.
PAGINATE	PG	Yes.	Boolean	Paginates the list file, providing headers for each page. PAGEWIDTH and PAGELENGTH only work if PAGINATE is on. The page headers include the date, the chipname, the title, the subtitle, and the page number.
PAGEWIDTH	PW	Yes.	Number	Sets the width of a page in the list file. The default is 79 characters wide.
PAGELENGTH	PL	Yes.	Number	Sets the length of a page in the list file. The default is 60.
LIST	LI	No.	Boolean	Turns the listing of the program on or off. The default, of course, is on.
EJECT	EJ	No.	Number	Starts a new page in the list file if there are fewer than the specified number of lines left on the current page. For instance, 'EJ 20' ejects if there are fewer than 20 lines left. 'EJ' by itself is an unconditional eject.

Figure 4.1: AVMAC51 Options: Listings

AVMAC51 Options - Continued				
Listing Control Options				
Name	Abbr	Primary?	Arg. Type	Description
SHOWMACS	SM	No.	Boolean	Shows the evaluation of macros in the list file. When a macro is referenced, it is followed by its expansion. Ordinarily the expansion is not shown.
SHOWCOMMENTS	SC	Cmd. Line	Boolean	Typically, when macros are expanded by the preprocessor, any lines which will not generate code are left out. Specifying 'SHOWCOMMENTS' on the command line causes all non-code-generating lines to be shown.
SHOWINCS	SI	No.	Boolean	Shows the text of include files in the list file. Ordinarily, these files are not shown.
INDENT	IN	Yes.	Number	Sets the number of tabstops to indent (in the list file) if a line must be broken and continued on the next line of output.
ERRONLY	ER	Yes.	Boolean	Specifies that the list file is to list only lines which contained errors.

Figure 4.2: AVMAC51 Options: Listings - Continued

ther Options

Name	Abbr	Primary?	Arg. Type	Description
UIET	QU	No.	Boolean	Shuts the assembler up. No messages are printed on the terminal at all. (Except error messages).
ERBOSE	VB	No.	Boolean	Makes the assembler even more talkative. Various information about the assembly is printed on the console as the program is assembled.
EFINE	DF	No.	String	Defines a symbol. Its arguments consist of a symbolname followed by an absolute value. The given value is associated with the symbol. For example: 'DF(BLOAT,45H)' gives BLOAT the value 69.
ACEVAL	ME	Cmd line	Boolean	Enables or disables macro preprocessing. The preprocessor is avoided entirely if NOMACEVAL is specified. If no evaluation is desired, NOMACEVAL *must be typed on the command line.*
ROCESS	PC	Cmd line	Boolean	Enables or disables assembly. Specifying 'NOPROCESS' means that the assembly program will be preprocessed, but not assembled. The default, of course, is 'PROCESS'.
ACFILE	MF	Cmd line	Filename	Sets the name of the macro expansion intermediate file. The default is the source filename with the suffix '.MXP'.
LENGTH	IL	Yes.	Int.	Sets the number of significant characters in identifiers. The maximum, and the default, is 32. The minimum is 6

Figure 4.4: AVMAC51 Options: Misc.

Object File and Information Options

Name	Abbr	Primary?	Arg. Type	Description
OBJECT	OJ	Yes.	Filename	Sets the name of the object file. The default is the name of the source file with '.OBJ' appended.
VERSION	VS	Yes.	String	Sets the version stamp in the object module.
TITLE	TL	Yes.	String	Sets the title of the program.
SUBTITLE	SB	Yes.	String	Sets the subtitle.
MODNAME	MN	Yes.	String	Sets the name of the object module. This name is included in the object file which will be passed to the linker. MODNAMEs appear in the load map.
COPYRIGHT	CR	Yes.	String	Sets the copyright message in the object module.
DATE	DT	No.	String.	Sets the date stamp in the object module. If left unset, today's date is used.

Figure 4.3: AVMAC51 Options: Object File

Other Options - Continued				
Name	Abbr	Primary?	Arg. Type	Description
DEFAULT	DT	No.	Boolean	DEFAULT specifies that options set on the command line override options set in the program. NODEFAULT allows options in the program to override command line options. NODEFAULT is (chuckle) the default.
XREF	XR	Yes.	Boolean	Causes an XREF file to be generated. An XREF file is a file containing cross-reference information, to be used by the cross reference report generator (AVREF—see the description in the linker paper). If XREF is specified, a file with the suffix '.XRF' will be generated.
ALLPUBLIC	AP	No.	Boolean	Sometimes you will wish all the labels defined in your assembly program to be PUBLIC. Specifying ALLPUBLIC does just that. This is especially useful when one is using an AVSIM simulator, as the AVSIM only knows about PUBLIC symbol names.

Figure 4.5: AVMAC51 Options: Misc. Continued

ANEXO B

MANUAL DEL USUARIO

MANUAL DEL USUARIO

CONTENIDO:

1. Introducción
2. Requerimientos del Sistema
3. Instalación.
4. Descripción de los componentes del EDITOR INTELIGENTE.
5. Ingreso de una instrucción
6. Mensajes de Error
7. Otras ayudas del EDITOR INTELIGENTE

1. INTRODUCCIÓN.

El EDITOR INTELIGENTE versión 1.0, fue desarrollado en el lenguaje de programación VISUAL BASIC 5.0 como tesis previa a la obtención del titulo de "Ingeniería en Electrónica y Telecomunicaciones", en la Escuela Politécnica Nacional.

Es un software útil en el desarrollo de programas para los microcontroladores INTEL MCS – 51/52. Trabaja bajo el ambiente WINDOWS; su formato es similar al Bloque de Notas o el Wordpad, los cuales son Editores de texto.

Presenta ventajas propias para el desarrollo de programas, lo que lo hace fácil en su manejo.

2. REQUERIMIENTOS DEL SISTEMA

- Un computador personal 80486 o superior
- Disco duro con 6 Mbytes disponibles
- Sistema operativo Windows 95 ó superior
- Tarjeta gráfica VGA
- Un ratón (Mouse)

3. INSTALACION

- Localizar en el disco 1 del EDITOR INTELIGENTE (mediante el explorador de Windows), el Archivo SETUP.EXE y ejecutarlo.
- Insertar el disco 2; luego se presentará la pantalla de instalación en la cual se define la carpeta en la que se copiarán todos los archivo necesarios para correr esta aplicación.
- Insertar el disco 3 y 4, cuando el programa lo pida.
- El programa de instalación, creará un icono que identifica al EDITOR INTELIGENTE; este se encontrará en el Menú Inicio. Mediante este icono se podrá acceder a la aplicación EDITOR INTELIGENTE.EXE

4. DESCRIPCIÓN DE LOS COMPONENTES DEL EDITOR INTELIGENTE.

Mientras se carga el programa, se verá una pantalla de presentación, la cual desaparecerá automáticamente luego de que la aplicación este disponible al usuario.

Una vez ingresado en el ambiente del EDITOR INTELIGENTE, este mostrará una pantalla principal, la cual consta de:

- Una barra de Menú
- Una barra de Herramientas
- Una barra de Posición
- Una barra de Estado
- y un cuadro de texto

En la figura 1 puede observarse los elementos indicados:

A continuación se describe cada uno de los componentes de la pantalla principal del EDITOR INTELIGENTE.

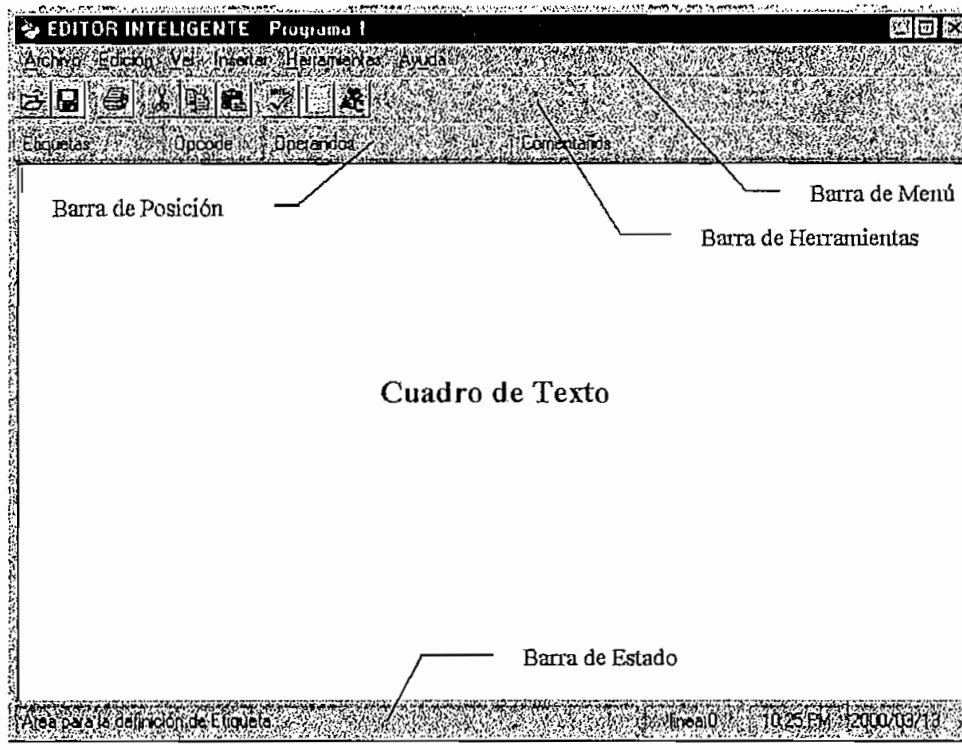

Figura 1 Pantalla principal del EDITOR INTELIGENTE

BARRA DE MENU

Esta barra consta de los submenús:

Archivo

Esta relacionado con la manipulación de los archivos. Consta de seis elementos que realizan las siguientes funciones:

a) *Abrir.*- Permite Abrir un archivo de extensión .ASM. Muestra el cuadro de diálogo Abrir, en el cual, puede seleccionarse uno de los archivos existentes en el directorio de la aplicación ó buscar en otro directorio. Sí el archivo no existe se presentará un mensaje de error indicando que no existe el archivo.

b) *Guardar.*- Guarda el programa editado en el cuadro de texto. Si el archivo aún no ha sido creado, se presenta el cuadro de diálogo Guardar Como.

c) *Guardar Como.*- Muestra el cuadro de diálogo Guardar Como, en el cual se puede definir la dirección y el nombre del archivo para luego guardarlo. Los archivos guardados tienen la extensión .ASM.

d) Imprimir.- Presenta el cuadro de diálogo Imprimir, con el fin de establecer las características de impresión del documento. El texto a imprimirse será el contenido del cuadro de texto del EDITOR INTELIGENTE.

e) Salir.- Permite finalizar la sesión de trabajo. Si sé producido algún cambio en el archivo abierto actualmente, muestra un cuadro de mensaje alertando el cambio del texto.

f) Ultimos Archivos.- Permite acceder en forma directa a los archivos abiertos últimamente, sin necesidad de mostrar el cuadro de diálogo Abrir.

Edición.

Está relacionado con el programa existente en el cuadro de texto. Consta de ocho opciones:

a) *Cortar.*- Permite cortar el texto que se ha seleccionado, guardándolo en el portapapel.

b) *Copiar.*- Permite copiar el texto seleccionado, guardarlo en el portapapel.

c) *Pegar.*- Pega el texto que se existe en el portapapel.

d) *Borrar.*- Borra el texto seleccionado.

e) *Seleccionar todo.*- Selecciona todo el texto del programa.

f) *Buscar.*- Muestra el cuadro de diálogo Buscar. El cuadro de diálogo Buscar permite ingresar un conjunto de caracteres o palabras para buscarlo en el texto del programa,.

g) *Buscar siguiente.*- Permite buscar en el programa, el texto que se halla resaltado o el texto de la última búsqueda sin necesidad de mostrar el cuadro de diálogo Buscar.

h) Reemplazar.- Presenta el cuadro de diálogo Reemplazar. El cuadro de diálogo Reemplazar permite buscar y reemplazar un conjunto de caracteres o palabras en el texto del programa.

Ver

Está relacionado con la presencia o no de las barras. Consta de tres opciones.

a) *Barra de Herramientas.*- Permite hacer visible o invisible la barra de herramientas. La presencia de esta barra en la aplicación, se la determina por el símbolo √ en el inicio de la opción.
b) *Barra de Estado.*- Permite hacer visible o invisible la barra de Estado. La presencia de esta barra en la aplicación, se la determina por el símbolo √ en el inicio de la opción.
c) *Barra de Posición.*- Permite hacer visible o invisible la barra de Posición. La presencia de esta barra en la aplicación, se la determina por el símbolo √ en el inicio de la opción.

Insertar

Está relacionado con la inserción de caracteres en el texto.

a) @.- Inserta en el texto el carácter @ en el punto donde se halla el cursor.
b) #.- Inserta en el texto el carácter # en el punto donde se halla el cursor.
c) \.- Inserta en el texto el carácter \ en el punto donde se halla el cursor.
d) Otros caracteres.- Presenta el cuadro de diálogo Símbolos. En este cuadro de diálogo existe un conjunto de caracteres, los cuales pueden ser insertados en el texto del programa.

Herramientas

Está relacionado con la presentación de la ayuda y corrección del programa. Consta de dos opciones.

a) *Corregir Errores.*- Permite corregir los errores que existen en el programa. En el caso de encontrarse un error en alguna línea del texto, se presenta un cuadro de diálogo informando sobre el tipo de error existente y la posible solución.
b) *Opciones de programación.*- Permite mostrar el cuadro de diálogo Opciones de programación. En este cuadro de diálogo se puede establecer los parámetros con los que se trabajará en la edición del texto del programa.

Ayuda

Está relacionado con la información acerca del programa. Consta de una opción.

a) *Acerca de...*- Presenta información sobre de las características del EDITOR INTELIGENTE.

Cabe anotar que el acceso a la Barra de Menú, no solamente se la puede hacer utilizando el ratón, sino también presionando las teclas alt + la letra que aparece subrayada.

BARRA DE HERRAMIENTAS

Esta barra está formada por botones de acceso directo a las opciones que se ejecutan con más frecuencia. Todos estos botones son opciones de la barra de Menú.

Se puede acceder a estas opciones haciendo un clic sobre los botones. La opción correspondiente a cada botón se muestra en la figura 2

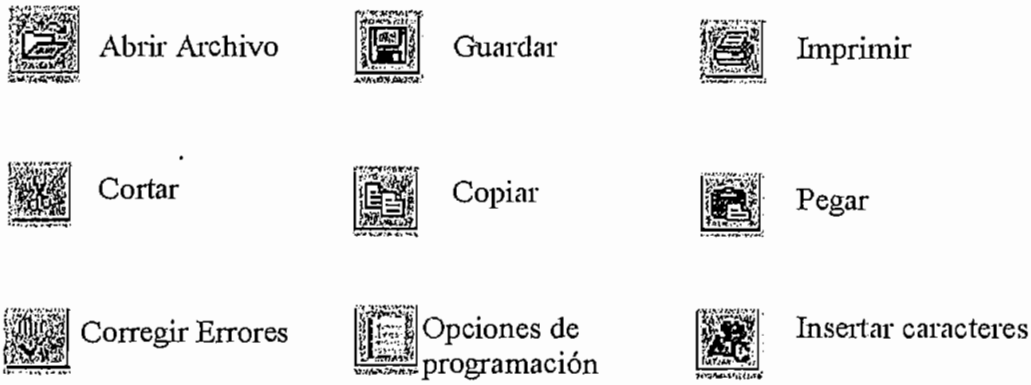

Figura 2 Botones de la Barra de Herramientas

BARRA DE ESTADO

En esta barra se presenta información que sirve como guía para el usuario, en el proceso de edición de una instrucción. También se presenta información acerca de la fecha, hora y la línea en la que se encuentra el cursor

En la figura 3, se puede observar la descripción de cada campo de la Barra de Estado.

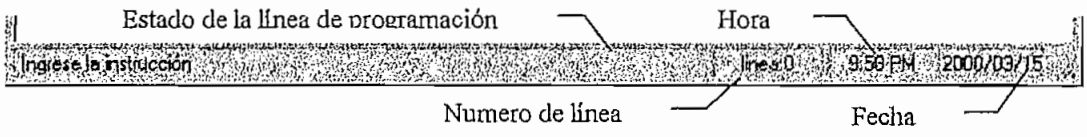

Figura 3 Barra de Estado

BARRA DE POSICION

Esta barra muestra la posición en la que se debe ubicar cada elemento de la línea de programación. También no sirve de guía para saber cuales y cuantos campos existen en una instrucción. En la figura 4 se puede observar esta barra con los elementos que la componen.

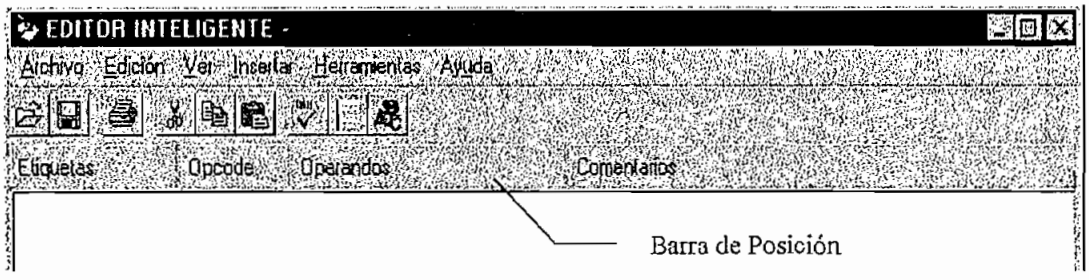

Figura 4 Barra de Posición.

CUADRO DE TEXTO.

Es el elemento más importante del EDITOR INTELIGENTE, pues sobre él, se ingresará todas las instrucciones del programa que sé este elaborando.

En la figura 1 se muestra el espacio definido para el cuadro de texto.

5. INGRESO DE UNA INSTRUCCIÓN.

Una instrucción esta formada por 4 campos, de los cuales no todos necesariamente pueden estar presentes. Estos campos son:

- *Etiqueta.*- Esta formado por un conjunto de caracteres; la etiqueta representa un punto al cual el programador puede hacer referencia desde otro punto del programa.
- *Opcode.*- Representa la operación que se quiere efectuar, esta formado por un conjunto de caracteres en forma de código, que simbolizan la acción a realizar por el microcontrolador.
- *Operandos.*- Esta formado por los argumentos que el Opcode requiere para efectuar la operación. Dependiendo del tipo de opcode, este campo puede o no estar presente.
- *Comentarios.*- Este campo es opcional; sobre él se define la operación a realizar, es decir, se escribe los comentarios que el usuario requiera como guía. Un comentario debe empezar por un punto y como (;).

En el cuadro de texto se mostrará las líneas de programación respetando el siguiente formato:

1. Una instrucción bien escrita se pintará de color negro.
2. Una pseudo instrucción bien escrita, se pintará de color azul.
3. Una instrucción o pseudo instrucción mal escrita se pintará de color rojo.
4. Todos los comentarios se pintan de color verde. A excepción de las líneas en las que existen errores, en este caso se pinta toda la línea de rojo.

6. MENSAJES DE ERROR

El EDITOR INTELIGENTE detecta los errores que se producen en la edición del programa. Existen varias formas de alertar la presencia de errores en la instrucción. A continuación se describe cada una de ellas.

a) El formato en color rojo de una línea del programa, alerta la existencia de un error en esta.

b) Durante la escritura de una línea de texto, se presentan ayudas para la edición. Si a medida que se está escribiendo dicha línea y está presente la ayuda en pantalla, al pasar al siguiente campo o argumento, la ayuda desaparece; esto es signo de que en la línea existe un error.

c) Durante la edición de una línea del programa, en la barra de estado se presentan mensajes del estado de dicha línea. Cuando se produce un error, el EDITOR INTELIGENTE lo detecta y presenta en esta barra el tipo de error cometido. Este mensaje se presentará por algunos segundos, luego de lo cual definirá únicamente el campo en el que se halla el cursor.

d) Cuando se ha seleccionado en el formulario Opciones, la alternativa: "Corregir al final de la instrucción"; al terminar de editar una línea de programación (es decir, presionar ENTER), el EDITOR INTELIGENTE, revisa si la línea ingresada está bien escrita. En el caso de encontrar un error, no permite avanzar a la línea siguiente y presenta un cuadro de mensajes, indicando el tipo de error producido, y resaltando el texto errado.

El EDITOR INTELIGENTE también permite corregir los errores existentes en el programa editado. Esto se lo hace accediendo a la opción "Corregir errores" del menú "Herramientas" de la barra de menú o su botón correspondiente en la barra de herramientas (en la figura 3 se puede observar a cual botón corresponde la opción "Corregir Errores").

El proceso de corregir errores, se inicia en la primera línea del programa, revisando si está bien escrita la instrucción; en el momento en el que se encuentra con un error, se detiene y muestra un cuadro de diálogo, en el cual se presenta el tipo de error, un mensaje sobre el error producido y la posible solución.

El cuadro de diálogo "Corregir" tiene varios botones un cuadro de texto y una lista que puede o no ser visible. En el cuadro de texto se ingresan los caracteres que va a reemplazar al texto seleccionado o el texto que se aumentará. En la lista (sí está presente) se muestra las posibles soluciones al error.

Los botones sirven para realizar varias acciones como son:

- *Botón Continuar.*- Permite corregir el error y continuar revisando el programa.
- *Botón Salir.*- Permite abandonar la corrección de errores.
- *Botón Ignorar Línea.*- Permite saltar la línea actual que contiene error.

7. OTRAS AYUDAS DEL EDITOR INTELIGENTE.

Existen otras ayudas que el EDITOR INTELIGENTE presenta, las cuales ya fueron nombradas pero en esta sección se las detallara un poco más.

Buscar y Reemplazar

Esta opción permite buscar en el primer caso, y buscar y/o reemplazar en el segundo caso, un conjunto de caracteres o letras que existen en el programa.

La opción "Buscar" presenta un cuadro de diálogo en el cual se puede ingresar los caracteres que se requiere buscar. Está formado por dos botones y dos tipos de opciones. La primera opción determina sí la búsqueda que se efectuará, coincida carácter con carácter. La segunda opción especifica la dirección de búsqueda. Los botones definen la continuación de la búsqueda o la finalización de la misma.

La opción "Reemplazar" es similar a la opción Buscar, con la diferencia que presenta dos cuadros de texto y un botón adicional. El un cuadro de texto sirve para ingresar los caracteres que se van a buscar, mientras que el otro sirve para ingresar los caracteres por los que se va a reemplazar el texto encontrado.

Si el texto que se busca no existe o por la definición de la dirección ya no hay mas elementos iguales, se presenta un mensaje de error indicando que la búsqueda a terminado.

Insertar Caracteres.

Presenta un cuadro de diálogo, el cual permite pegar caracteres de una lista existe en este. Contiene dos cuadros de texto, una lista con los caracteres insertables y dos botones. En uno de los cuadros de texto se muestra el carácter insertable en un tamaño mayor, mientras que en el otro el equivalente decimal del código ASCII del carácter. Los botones sirven para insertar el carácter en el punto donde se halla el cursor y salir de este cuadro de diálogo en el otro caso.

Opciones de programación

Determina la forma en que se editará el programa en el EDITOR INTELIGENTE. Está formado por tres campos, que son:

- *Mostrar ayuda en pantalla.*- Determina si se presentará o no la ayuda en pantalla, mientras se edita el programa.
- *Uso del Ratón y el Teclado.*- Permite agilitar el proceso de edición del programa. Sí se elige la opción "Utilizar únicamente el Ratón"; el EDITOR INTELIGENTE, permite acceder a la ayuda haciendo solo un clic sobre la opción quiere se quiere pegar. En el caso de elegir "Utilizar el Teclado y el Ratón", el EDITOR INTELIGENTE permite acceder a la ayuda presionando las teclas de desplazamiento y pegar la opción elegida con un ENTER o la tecla de desplazamiento a la derecha; también se puede utilizar el ratón con la diferencia que para acceder a una opción se requiere hacer doble clic.
- *Corregir errores de programa.*- Presenta tres tipos de opciones:
  a) *Corregir al terminar el programa.* Permite el ingreso de las instrucciones sin importar sí se comete o no un error.
  b) *Corregir al final de la instrucción.* Busca si existen errores en la línea de programación y no permite saltar a la nueva línea sin corregir dicho error. El error existente se muestra en un cuadro de mensajes y el texto errado será resaltado.
  c) *Permitir ingresar solo caracteres validos.* Revisa si se producen errores durante la escritura de la instrucción, carácter por carácter o palabra por palabra, dependiendo del campo en el que se encuentre. Sí se ingresa un carácter o palabra errada automáticamente será eliminado, no permitiendo continuar hasta que ingrese caracteres o palabras validas.

ANEXO C

LISTADO DEL PROGRAMA

```
'*** Formulario principal de la aplicación del EDITOR INTELIGENTE  ***
'*****************************************************************
Option Explicit
'Se define variables de formuario
Dim CambioLinea As Integer
Dim PosicionAnterior As Integer
Dim Linea1  As Integer
Dim Linea2  As Integer
Dim modiLinea As Boolean

Private Sub Form_Load()
  Dim i As Integer      ' Variable contador.
  Dim titulo As String
  ' La aplicación empieza aquí (evento Load del formulario Startup).
  Show
  'Permite mantener visible el texto seleccionado
  txtEdicion.HideSelection = False
  ' Siempre establece el directorio de trabajo al directorio que contiene la aplicación.
  ChDir App.Path
  'Indica que el archivo abierto no ha sido modificado
  FEstado.Modificar = False
  ' Lee el Registro del sistema y establece la lista de archivos recientes del menú Archivo.
  ProcObtenerUltimosArchivos
  ' Establece la variable pública DireccionBuscar, que determina la dirección de búsqueda
   DireccionBuscar = 1
  'Defino las posiciones específicas del Tabulador
  txtEdicion.SelTabCount = 4
  txtEdicion.SelTabs(1) = 12
  txtEdicion.SelTabs(2) = 20
  txtEdicion.SelTabs(3) = 40
  'Inicializo las variables de Opciones
  opcAyuda = True
  opcTeclado = False
  opcCorregir = False
  opcCorregirFin = False
  opcNoCorregir = True
  'Defino la dirección de la base de datos
  Data1.DatabaseName = App.Path & "\Editor Inteligente.mdb"
  Data1.RecordSource = "Instrucciones"
  Data1.Refresh
  'Inicializamos las variables necesarias
  ProcIniciarVariables
  'Cargar los valores que se requieren para correr el programa
  ProcCargarValores
  'Cargo el programa con el que se iniciará
  titulo = Command()
  If titulo <> "" Then ProcAbrirArchivo (titulo)
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
  Dim mbMensaje As String
  Dim mbComentario As String
  Dim NombreArchivo As String
  Dim mbRespuesta As Integer
  ' Comprueba si el texto ha sido modificado.
  If FEstado.Modificar Then
    NombreArchivo = Me.Caption
    mbComentario = Right(Me.Caption, Len(Me.Caption) - 21)
    mbMensaje = "El texto en [" & mbComentario & "] ha cambiado."
```

```
    mbMensaje = mbMensaje & vbCrLf
    mbMensaje = mbMensaje & "¿Desea guardar los cambios?"
    mbRespuesta = MsgBox(mbMensaje, 51, "EDITOR INTELIGENTE")
    Select Case mbRespuesta
      Case 6    ' El usuario eligió Sí.
        If Right(Me.Caption, 10) = "Programa 1" Then
          ' El archivo no ha sido guardado aún.
          NombreArchivo = "Programa1.asm"
          ' Obtiene NombreArchivo, y después llama al procedimiento de guardar.
          NombreArchivo = FuncGuardarNombreArchivo(NombreArchivo)
        Else
          ' El título del formulario contiene el nombre del archivo abierto.
          NombreArchivo = Right(Me.Caption, Len(Me.Caption) - 21)
        End If
        ' Llama al procedimiento de guardar. Si NombreArchivo está vacía, es porque
        ' el usuario eligió Cancelar en el cuadro de diálogo Guardar como;' si no, guarda el archivo.
        If NombreArchivo <> "" Then
          ProcArchivoGuardar NombreArchivo
        End If
      Case 7    ' El usuario eligió No. Descarga el archivo.
        Cancel = False
      Case 2    ' El usuario eligió Cancelar. Cancela la descarga.
        Cancel = True
    End Select
  End If
End Sub

Private Sub Form_Resize()
  ' Llama al procedimiento de cambiar el tamaño
  ProcExpandirCuadroEditor
End Sub

Private Sub Form_Unload(Cancel As Integer)
  ' Llama al procedimiento de enumerar los archivos más recientes
  ProcObtenerUltimosArchivos
End Sub

Private Sub Lista1_Click()
  'Pegar el Mnemonico seleccionado
  If Not opcTeclado Then ProcPegarPalabra
End Sub

Private Sub Lista1_DblClick()
  'Pegar el Mnemonico seleccionado
  ProcPegarPalabra
End Sub

Private Sub Lista1_GotFocus()
  'Si la lista 2 recibe el enfoque hace invisibles la lista1 y el txtOperando
  txtOperando.Visible = False
End Sub

Private Sub Lista1_KeyDown(KeyCode As Integer, Shift As Integer)
'Si presiono las teclas Enter o desplazamiento a la derecha
  If KeyCode = vbKeyReturn Or KeyCode = vbKeyRight Then
    'Pegar el Mnemonico seleccionado
    ProcPegarPalabra
    If txtOperando.Visible Then txtOperando.SetFocus
    KeyCode = vbNull
```

```
  'Si presiono la tecla Esc hace invisible la lista de Opcodes
  ElseIf KeyCode = vbKeyEscape Then
    Lista1.Visible = False
  End If
End Sub

Private Sub mnuArchivoAbrir_Click()
  ' Llama al procedimiento de abrir archivo.
  ProcNombreArchivo
End Sub

Private Sub mnuArchivoGuardar_Click()
  ' Llama al procedimiento de guardar archivo
  FuncArchivoGuardar
End Sub

Private Sub mnuArchivoGuardarComo_Click()
  Dim NombreArchivo As String
  Dim NombrePorDefecto As String
  ' Asigna el título del formulario a la variable.
  NombrePorDefecto = Right$(Me.Caption, Len(Me.Caption) - 21)
  If Me.Caption = "EDITOR INTELIGENTE - Programa 1" Then
    ' El archivo no ha sido guardado aún.
    ' Obtiene el nombre del archivo y después llama al procedimiento de guardar.
    NombreArchivo = FuncGuardarNombreArchivo("Programa1.asm")
    If NombreArchivo <> "" Then ProcArchivoGuardar (NombreArchivo)
    ' Actualiza la lista de archivos más recientes del menú Archivo.
    MenuArchivoUltimoDato (NombreArchivo)
  Else
    ' El título del formulario contiene el nombre del archivo abierto.
    NombreArchivo = FuncGuardarNombreArchivo(NombrePorDefecto)
    If NombreArchivo <> "" Then ProcArchivoGuardar (NombreArchivo)
    ' Actualiza la lista de archivos más recientes del menú Archivo.
    MenuArchivoUltimoDato (NombreArchivo)
  End If
End Sub

Private Sub mnuArchivoImprimir_Click()
 ' Llama al procedimiento imprimir
  ProcArchivoImprimir
End Sub

Private Sub mnuArchivoSalir_Click()
  ' Termina la aplicación.
  ProcArchivoSalir
End Sub

Private Sub mnuAyudaAcercaDe_Click()
  ' Muestra el formualrio Acerca De
  frmAcercaDe.Show vbModal
End Sub

Private Sub mnuUltimoArchivo_Click(Index As Integer)
  Dim mbMensaje As String
  Dim mbComentario As String
  Dim NombreArchivo As String
  Dim mbRespuesta As Integer
  ' Compnueba si el texto ha sido modificado.
  If FEstado.Modificar Then
```

```
    NombreArchivo = Me.Caption
    mbComentario = Right(Me.Caption, Len(Me.Caption) - 21)
    mbMensaje = "El texto en [" & mbComentario & "] ha cambiado."
    mbMensaje = mbMensaje & vbCrLf
    mbMensaje = mbMensaje & "¿Desea guardar los cambios?"
    mbRespuesta = MsgBox(mbMensaje, 51, "EDITOR INTELIGENTE")
    Select Case mbRespuesta
      Case 6    ' El usuario eligió Sí.
        If Right(Me.Caption, 10) = "Programa 1" Then
          ' El archivo no ha sido guardado aún.
          NombreArchivo = "Programa1.asm"
          ' Obtiene NombreArchivo, y después llama al procedimiento de guardar.
          NombreArchivo = FuncGuardarNombreArchivo(NombreArchivo)
        Else
          ' El título del formulario contiene el nombre del archivo abierto.
          NombreArchivo = Right(Me.Caption, Len(Me.Caption) - 21)
        End If
        ' Llama al procedimiento de guardar. Si NombreArchivo está vacía, es porque
        ' el usuario eligió Cancelar en el cuadro de diálogo Guardar como;
        ' si no, guarda el archivo.
        If NombreArchivo <> "" Then
          ProcArchivoGuardar NombreArchivo
        End If
      Case 2    ' El usuario eligió Cancelar. Cancela la descarga.
        Exit Sub
    End Select
  End If
  ' Llama al procedimiento de abrir archivos, pasando una
  ' referencia al nombre del archivo seleccionado
  ProcAbrirArchivo (mnuUltimoArchivo(Index).Caption)
  ' Actualiza la lista de archivos más recientes del menú Archivo.
  ProcObtenerUltimosArchivos
End Sub

Private Sub mnuEdicionCopiar_Click()
  ' Llama al procedimiento de copiar
  ProcEdicionCopiar
End Sub

Private Sub mnuEdicionCortar_Click()
  ' Llama al procedimiento de cortar
  ProcEdicionCortar
End Sub

Private Sub mnuEdicionEliminar_Click()
  ' Si el puntero del mouse no se encuentra al final del Editor Inteligente ...
  If txtEdicion.SelStart <> Len(Screen.ActiveControl.Text) Then
    ' Si no hay texto seleccionado, extiende la selección en uno.
    If txtEdicion.SelLength = 0 Then
      txtEdicion.SelLength = 1
      ' Si el puntero del mouse se encuentra en una línea en blanco, extiende la selección en dos.
      If Asc(txtEdicion.SelText) = 13 Then
        txtEdicion.SelLength = 2
      End If
    End If
    ' Elimina el texto seleccionado.
    txtEdicion.SelText = ""
  End If
End Sub
```

```
Private Sub mnuEdicionPegar_Click()
  ' Llama al procedimiento de pegar.
  'ProcEdicionPegar
End Sub

Private Sub mnuEdicionSeleccionarTodo_Click()
  ' Utiliza SelStart y SelLength para seleccionar el texto.
  txtEdicion.SelStart = 0
  txtEdicion.SelLength = Len(txtEdicion.Text)
End Sub

Private Sub mnuEdicionBuscar_Click()
  ' Si el cuadro de texto contiene algo de texto, lo asigna al
  ' cuadro de texto del formulario Buscar, si no, asigna
  ' el último valor de búsqueda.
  If txtEdicion.SelText <> "" Then
    frmBuscar.txtBuscar.Text = txtEdicion.SelText
  Else
    frmBuscar.txtBuscar.Text = TextoBuscar
  End If
  ' Establece la variable pública de inicio al principio.
  busPosicionInicial = True
  ' Establece la casilla de verificación de distinción entre mayúsculas y minúsculas
  ' de acuerdo con la variable pública
  If (MAYminBuscar) Then
    frmBuscar.chkCaso = 1
  End If
  ' Presenta el formulario Buscar.
  frmBuscar.Show vbModal
End Sub

Private Sub mnuEdicionBuscarSiguiente_Click()
  'Si existe texto seleccionado permite buscar este en el programa
  If frmEditor.txtEdicion.SelText <> "" Then
    TextoBuscar = frmEditor.txtEdicion.SelText
  End If
  ' Si la variable pública no está vacía, llama al
  ' procedimiento de búsqueda, si no llama al menú de búsqueda
  If Len(TextoBuscar) > 0 Then
    ProcBuscarEsto
  Else
    mnuEdicionBuscar_Click
  End If
End Sub

Private Sub mnuEdicionReemplazar_Click()
  ' Si hay texto en este cuadro de texto, lo asigna al
  ' cuadro de texto del formulario Remplazar. Si no, asigna el valor de la
  ' última búsqueda.
  If Me.txtEdicion.SelText <> "" Then
    frmRemplazar.txtBuscar.Text = Me.txtEdicion.SelText
  Else
    frmRemplazar.txtBuscar.Text = TextoBuscar
  End If
  ' Establece la variable pública de inicio al principio.
  busPosicionInicial = True
  ' Establece la casilla de verificación de distinción entre mayúsculas y minúsculas
  ' según la variable pública
```

```
  If (MAYminBuscar) Then
    frmRemplazar.chkCaso = 1
  End If
  ' Presenta el formulario Remplazar.
  frmRemplazar.Show vbModal
End Sub

Private Sub mnuVerBarraEstado_Click()
  ' Llama al procedimiento de la barra de Estado, pasando una referencia
  ' a esta instancia de formulario.
  ProcVerBarraEstado
  ' Llama al procedimiento de cambiar el tamaño
  ProcExpandirCuadroEditor
End Sub

Private Sub mnuVerBarraPosicion_Click()
  ' Llama al procedimiento de la barra de Estado, pasando una referencia
  ' a esta instancia de formulario.
  ProcVerBarraPosicion
  ' Llama al procedimiento de cambiar el tamaño
  ProcExpandirCuadroEditor
End Sub

Private Sub mnuVerBarraHerramientas_Click()
  ' Llama al procedimiento de la barra de herramientas, pasando una referencia
  ' a esta instancia de formulario.
  ProcVerBarraHerramientas
  ' Llama al procedimiento de cambiar el tamaño
  ProcExpandirCuadroEditor
End Sub

Private Sub mnuInsertarArroba_Click()
  'Pegar el caracter "@" en la posición que se halla el cursor
  txtEdicion.SelText = "@"
End Sub

Private Sub mnuInsertarBarra_Click()
  'Pegar el caracter "\" en la posición que se halla el cursor
  txtEdicion.SelText = "\"
End Sub

Private Sub mnuInsertarNumero_Click()
  'Pegar el caracter "#" en la posición que se halla el cursor
  txtEdicion.SelText = "#"
End Sub

Private Sub mnuInsertarOtrosCaracteres_Click()
  'Presento el formulario que contiene los caracteres ASCII
  frmSimbolos.Show vbModal
End Sub

Private Sub mnuHerramientasCorregirErrores_Click()
  'Llama al procedimiento que permite corregir los errores
  'existentes en el probrama.
  ProcCorregirErrores
End Sub

Private Sub mnuHerramientaOpciones_Click()
  'Llama al cuadro de dialogo que permite definir el modo
```

```
  'de uso del programa.
  frmOpciones.Show vbModal
  txtEdicion.SetFocus
End Sub

Private Sub tbBarHerramientas_ButtonClick(ByVal Button As ComctlLib.Button)
  Select Case Button.Key
  Case "Abrir"
    ' Llama al procedimiento de abrir archivo.
    ProcNombreArchivo
  Case "Guardar"
    ' Llamar al procedimiento guardar
    FuncArchivoGuardar
  Case "Cortar"
    ' Llama al procedimiento de cortar
    ProcEdicionCortar
  Case "Copiar"
    ' Llama al procedimiento de copiar
    ProcEdicionCopiar
  Case "Pegar"
    ' Llama al procedimiento de pegar
    ProcEdicionPegar
  Case "Imprimir"
    ' Llama al procedimiento imprimir
    ProcArchivoImprimir
  Case "Corregir"
    ' Llama al procedimiento corregir
    ProcCorregirErrores
  Case "Insertar"
    'Presento el formulario que contiene los caracteres ASCII
    frmSimbolos.Show vbModal
  Case "Opciones"
    'Llama al cuadro de dialogo que permite definir el modo
    'de uso del programa.
    frmOpciones.Show vbModal
    txtEdicion.SetFocus
  End Select
End Sub

Private Sub Timer1_Timer()
  'Muestra los mensajes generados
  ProcMensajeError (Observacion)
  Timer1.Enabled = False

End Sub

Private Sub txtEdicion_Change()
  ' Establece la variable pública para mostrar que el texto ha sido modificado.
  If txtEdicion.Text <> "" Then
    FEstado.Modificar = True
    tbBarHerramientas.Buttons(2).Enabled = True
    tbBarHerramientas.Buttons(4).Enabled = True
    tbBarHerramientas.Buttons(10).Enabled = True
    mnuHerramientasCorregirErrores.Enabled = True
  End If
End Sub

Private Sub txtEdicion_KeyDown(KeyCode As Integer, Shift As Integer)
  Dim posicion As Integer
```

```
  Dim letra As String
  'Anula  la entrada si es una tecla F2, F4
  If KeyCode = vbKeyF2 Or KeyCode = vbKeyF4 Then
    KeyCode = vbNull 'Anula la entrada
    Exit Sub
  End If
  'Defino en que línea se halla el cursor antes de pegar el caracter ingresado.
  Linea1 = txtEdicion.GetLineFromChar(txtEdicion.SelStart)
  'Guardo en una variable la posición anterior
  PosicionAnterior = txtEdicion.SelStart
  'Paso el enfoque al cuadro de lista que mostraran la ayuda
  If opcTeclado Then
    If Lista1.Visible = True Then
    'Presiono la tecla de desplazamiento hacia abajo o presiono F1
      If KeyCode = 39 Or KeyCode = 40 Then
        Lista1.ListIndex = 0
        Lista1.SetFocus
        KeyCode = vbNull
      ElseIf KeyCode = 38 Or KeyCode = 37 Then
        Lista1.ListIndex = Lista1.ListCount - 1
        Lista1.SetFocus
        KeyCode = vbNull
      End If
    End If
  Else
    If Lista1.Visible Then
      If KeyCode = 37 Or KeyCode = 38 Or KeyCode = 39 Or KeyCode = 40 Then
        KeyCode = vbNull
      End If
    End If
  End If
  'Si presiono la tecla "Esc" desaparece la ayuda en pantalla actual
  If KeyCode = 27 Then
    Lista1.Visible = False
    txtOperando.Visible = False
  End If
End Sub

Private Sub txtEdicion_KeyPress(KeyAscii As Integer)
  Dim Caracter As Integer
  'Cambio el espacio en blanco por el Tab
  If Not indComentario Then
    If Campo <= 1 Then
    If KeyAscii = vbKeySpace Then KeyAscii = vbKeyTab
    End If
  End If
  Caracter = KeyAscii
  'Llamo a la función que detecta errores en el ingreso de datos
  If CaracterErrado(Caracter) Then KeyAscii = vbNull
End Sub

Private Sub txtEdicion_KeyUp(KeyCode As Integer, Shift As Integer)
  Dim posicion As Integer
  Dim ValorAnterior As Integer
  Dim inicio As Integer
  Dim fin As Integer
  Dim Numero As Integer
  'Defino en que línea se halla el cursor despues de pegar el caracter ingresado.
  Linea2 = txtEdicion.GetLineFromChar(txtEdicion.SelStart)
```

```
  'Si presiono la tecla Enter redefino parámetros
  If KeyCode = vbKeyReturn Then
    CambioLinea = 0
    modiLinea = False
  End If
  'Si se produce cambio de linea en el proceso de presionar una tecla
  If Linea1 <> Linea2 Then
    CambioLinea = CambioLinea + 1
    'Si se produce el segundo cambio de linea
    If CambioLinea = 2 Then
      'Si se modificó la línea anterior
      If modiLinea Then
        'Guardo la ultima posicion del cursor
        posicion = txtEdicion.SelStart
        ValorAnterior = PosicionAnterior
        'Defino el punto final de la instruccion
        fin = FinInstruccion(ValorAnterior)
        'Ubico el punto inicial de la instruccion
        inicio = InicioInstruccion(ValorAnterior)
        'Separo la instruccion en sus componentes
        Call ProcSepararInstruccion(inicio, fin)
        'Verifico si existen errores en al instruccion
        Numero = CorregirInstruccion(inicio)
        Select Case Numero
        Case -1
          Call CambiarColorComentarios(inicio, fin)
        Case -2
          Call CambiarColorSeudoOpcode(inicio, fin)
        Case Else
          Call CambiarColorError(inicio, fin)
        End Select
        CambioLinea = 0
        modiLinea = False
        txtEdicion.SelStart = posicion
      End If
    End If
  Else
    'Si se produce el primer cambio de línea modilinea es true
    If CambioLinea <> 2 Then
      If Not modiLinea Then
        'Si luego de un cambio de línea se modifica la línea
        Select Case KeyCode
        'Si presiono las teclas de funciones
        Case 112 To 127: modiLinea = False
        'Si presiono la tecla Cancelar
        Case 27 To 40: modiLinea = False
        'Si se presiona cualquier otra tecla
        Case Else:  modiLinea = True: CambioLinea = 1
        End Select
      End If
    End If
  End If
  'Si se presionó las teclas de desplazamiento defino el sitio en el que se halla
  Select Case KeyCode
  Case 33 To 40
    ProcPosicionCursor
  End Select
End Sub
```

```
Private Sub txtEdicion_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
  'Defino en que posición se halla el cursor
  ProcPosicionCursor
End Sub

Private Sub txtEdicion_SelChange()
  ' Permite activar o desactivar las teclas: Cortar, Copiar, Pegar,Corregir
  If txtEdicion.SelLength >= 1 Then
    mnuEdicionCopiar.Enabled = True
    mnuEdicionCortar.Enabled = True
    mnuEdicionEliminar.Enabled = True
    With frmEditor.tbBarHerramientas
      .Buttons(6).Enabled = True
      .Buttons(7).Enabled = True
    End With
  ElseIf mnuEdicionCopiar.Enabled = True Then
    mnuEdicionCopiar.Enabled = False
    mnuEdicionCortar.Enabled = False
    mnuEdicionEliminar.Enabled = False
    With frmEditor.tbBarHerramientas
      .Buttons(6).Enabled = False
      .Buttons(7).Enabled = False
    End With
  End If
  'Permite definir en que línea se halla el punto de inserción
  NumeroLinea = txtEdicion.GetLineFromChar(txtEdicion.SelStart)
  sbBarraDeEstado.Panels(2).Text = "Linea:" & NumeroLinea
End Sub

Private Sub txtOperando_KeyDown(KeyCode As Integer, Shift As Integer)
  'Si presiono la tecla de desplazamiento a la izquierda regreso a la lista 1
  If KeyCode = 37 Then
    Lista1.SetFocus
    txtOperando.Visible = False
  End If
  'Si se presiona la tecla Enter se pega el texto que hay en txtOperando
  If KeyCode = 13 And txtOperando.Text <> "" Then TextoOperandoAuxiliar
  'Si se presiona la tecla Esc se hace invisible el txtOperando
  If KeyCode = 27 Then txtOperando.Visible = False
End Sub



'*** Formulario de Opciones de Corrección del EDITOR INTELIGENTE  ***
'*********************************************************************
Option Explicit
'Se define variables de formulario
Private opcAyuda1 As Boolean
Private opcTeclado1 As Boolean
Private opcCorregir1 As Boolean
Private opcCorregirFin1 As Boolean
Private opcNoCorregir1 As Boolean

Private Sub cmdAceptar_Click()
  Unload frmOpciones
End Sub

Private Sub cmdCancelar_Click()
  'Restablezco los valores anteriores
```

```
  opcCorregir = opcCorregir1
  opcCorregirFin = opcCorregirFin1
  opcNoCorregir = opcNoCorregir1
  opcAyuda = opcAyuda1
  opcTeclado = opcTeclado1
  Unload frmOpciones
End Sub

Private Sub Form_Load()
  'Restablezco los valores anteriores para poder mostrarlos en
  'formulario opciones
  If opcAyuda Then opbAyuda(0) = True
  If Not opcAyuda Then opbAyuda(1) = True
  If opcTeclado Then opbTeclado(1) = True
  If Not opcTeclado Then opbTeclado(0) = True
  If opcNoCorregir Then opbCorregir(0) = True
  If opcCorregirFin Then opbCorregir(1) = True
  If opcCorregir Then opbCorregir(2) = True
  'Deshabilito el uso del Ratón o teclado
  If Not opcAyuda Then
    opbTeclado(0).Enabled = False
    opbTeclado(1).Enabled = False
  End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
  'Guardo los valores ultimamente definidos
  opcAyuda1 = opcAyuda
  opcCorregir1 = opcCorregir
  opcCorregirFin1 = opcCorregirFin
  opcNoCorregir1 = opcNoCorregir
  opcTeclado1 = opcTeclado
End Sub

Private Sub opbAyuda_Click(Index As Integer)
  'Determino cual valor se ha escogido
  If Index = 0 Then opcAyuda = True
  If Index = 1 Then opcAyuda = False
  'Habilito o deshabilito el uso del Ratón o teclado
  If opcAyuda Then
    opbTeclado(0).Enabled = True
    opbTeclado(1).Enabled = True
  Else
    opbTeclado(0).Enabled = False
    opbTeclado(1).Enabled = False
  End If
End Sub

Private Sub opbCorregir_Click(Index As Integer)
  'Determino cual valor se ha escogido
  Select Case Index
  Case 0
    opcNoCorregir = True
    opcCorregirFin = False
    opcCorregir = False
  Case 1
    opcNoCorregir = False
    opcCorregirFin = True
    opcCorregir = False
```

```
  Case 2
    opcNoCorregir = False
    opcCorregirFin = False
    opcCorregir = True
  End Select
End Sub

Private Sub opbTeclado_Click(Index As Integer)
  'Determino cual valor se ha escogido
  If Index = 0 Then opcTeclado = False
  If Index = 1 Then opcTeclado = True
End Sub



'*** Formulario Corregir, sirve para corregir Errores del programa ***
'***              Forma parte del EDITOR INTELIGENTE          ***
'**********************************************************************

Private SelPalabra1 As String
Private MasParametros As Integer

Private Sub cmdContinuar_Click()
  With frmEditor
    If MasParametros = 1 Then
      .txtEdicion.SelText = Chr(9) + txtCambiar.Text
    ElseIf MasParametros = 2 Then
      .txtEdicion.SelText = "," + txtCambiar.Text
    ElseIf MasParametros = 3 Then
      .txtEdicion.SelText = ""
    Else
      .txtEdicion.SelText = txtCambiar.Text
    End If
  End With
  TerminarCorregir = False
  Unload frmCorregir
End Sub

Private Sub cmdIgnorar_Click()
  'Salto a corregir la siguiente linea
  IgnorarLinea = True
  Unload frmCorregir
End Sub

Private Sub Form_Load()
  Dim mensaje As Variant
  'Define la posición en la que se mostrará los formularios de ayuda
  If NumeroLinea < 10 Then
    frmCorregir.Top = 4500
  Else
    frmCorregir.Top = 1500
  End If
  'Muesta el mensaje de Error producido
  frmCorregir.Caption = "Error de Sintaxis"
  MasParametros = 4
  Select Case TipoError
  Case 1
    txtCambiar.Left = 1
    txtCambiar.Width = 20
```

```
    mensaje = Chr(34) + LetraError + Chr(34)
    mensaje = mensaje + Chr(13) + Chr(10) + "El caracter ingresada no puede ser"
    mensaje = mensaje + " inicio de etiqueta."
    mensaje = mensaje + Chr(13) + Chr(10) + "Ingrese la nueva Etiqueta"
  Case 2
    txtCambiar.Left = 1
    txtCambiar.Width = 20
    mensaje = Chr(34) + LetraError + Chr(34)
    mensaje = mensaje + Chr(13) + Chr(10) + "El caracter ingresada no puede ser"
    mensaje = mensaje + " parte de la etiqueta."
    mensaje = mensaje + Chr(13) + Chr(10) + "Ingrese la nueva Etiqueta"
  Case 3
    txtCambiar.Left = 1
    txtCambiar.Width = 20
    mensaje = Chr(34) + PalabraError + Chr(34)
    mensaje = mensaje + Chr(13) + Chr(10) + "La palabra ingresada no corresponde "
    mensaje = mensaje + "a Opcode existente."
    mensaje = mensaje + Chr(13) + Chr(10) + "Ingrese el nuevo Opcode o SeudoOpcode"
  Case 4
    For i = 0 To frmEditor.Lista1.ListCount - 1
      lstOpciones.AddItem frmEditor.Lista1.List(i)
    Next
    lstOpciones.Visible = True
    mensaje = Chr(34) + PalabraError + Chr(34)
    mensaje = mensaje + Chr(13) + Chr(10) + "La palabra ingresada no corresponde "
    mensaje = mensaje + "a Opcode existente."
    mensaje = mensaje + Chr(13) + Chr(10) + "Seleccione un elemento o ingrese el nuevo Opcode "
  Case 5
    For i = 0 To frmEditor.Lista1.ListCount - 1
      lstOpciones.AddItem frmEditor.Lista1.List(i)
    Next
    lstOpciones.Visible = True
    mensaje = Chr(34) + PalabraError + Chr(34)
    mensaje = mensaje + Chr(13) + Chr(10) + "La palabra ingresada no corresponde "
    mensaje = mensaje + "a Operando permitido."
    mensaje = mensaje + Chr(13) + Chr(10) + "Seleccione un elemento o ingrese el nuevo Operando "
  Case 6
    For i = 0 To frmEditor.Lista1.ListCount - 1
      lstOpciones.AddItem frmEditor.Lista1.List(i)
    Next
    lstOpciones.Visible = True
    mensaje = Chr(34) + PalabraError + Chr(34)
    mensaje = mensaje + Chr(13) + Chr(10) + "La palabra ingresada no corresponde "
    mensaje = mensaje + "a Operando permitido."
    mensaje = mensaje + Chr(13) + Chr(10) + "Seleccione un elemento o ingrese el nuevo Operando "
  Case 7
    MasParametros = 1
    frmCorregir.Caption = "Pocos parámetros"
    For i = 0 To frmEditor.Lista1.ListCount - 1
      lstOpciones.AddItem frmEditor.Lista1.List(i)
    Next
    PalabraError = ""
    lstOpciones.Visible = True
    mensaje = "El Opcode o Pseudo Opcode ingresado requiere de más parámetros "
    mensaje = mensaje + Chr(13) + Chr(10) + "Seleccione un elemento o ingrese un Operando "
  Case 8
    MasParametros = 2
    frmCorregir.Caption = "Pocos parámetros"
    For i = 0 To frmEditor.Lista1.ListCount - 1
```

```
      lstOpciones.AddItem frmEditor.Lista1.List(i)
    Next
    PalabraError = ""
    lstOpciones.Visible = True
    mensaje = "El Opcode ingresado requiere de más parámetros "
    mensaje = mensaje + Chr(13) + Chr(10) + "Seleccione un elemento o ingrese un Operando "
  Case 9
    txtCambiar.Left = 1
    txtCambiar.Width = 20
    frmCorregir.Caption = "Pocos parámetros"
    mensaje = "En la instrucción no se a definido Opcode "
    mensaje = mensaje + Chr(13) + Chr(10) + "Ingrese un Opcode "
    PalabraError = ""
  Case 10
    MasParametros = 3
    txtCambiar.Left = 1
    txtCambiar.Width = 20
    frmCorregir.Caption = "Muchos parámetros"
    mensaje = "El Opcode que ingresado no requiere de mas Operandos "
    mensaje = mensaje + Chr(13) + Chr(10) + "Se eliminará el operando ingresado "
  Case 11
    MasParametros = 3
    txtCambiar.Left = 1
    txtCambiar.Width = 20
    frmCorregir.Caption = "Muchos parámetros"
    mensaje = "El Opcode que ingresado no requiere de mas Operandos"
    mensaje = mensaje + Chr(13) + Chr(10) + "o un comentario se inicio con (;) "
    mensaje = mensaje + Chr(13) + Chr(10) + "Se eliminará el Operando ingresado "
  Case 12
    MasParametros = 1
    frmCorregir.Caption = "Pocos parámetros"
    For i = 0 To frmEditor.Lista1.ListCount - 1
      lstOpciones.AddItem frmEditor.Lista1.List(i)
    Next
    PalabraError = ""
    lstOpciones.Visible = True
    mensaje = "La Opción ingresado requiere de más parámetros "
    mensaje = mensaje + Chr(13) + Chr(10) + "Seleccione un elemento o ingrese un Argumento "
  Case 13
    MasParametros = 3
    txtCambiar.Left = 1
    txtCambiar.Width = 20
    frmCorregir.Caption = "Muchos parámetros"
    mensaje = "La Opcion ingresada no requiere de mas parámetros"
    mensaje = mensaje + Chr(13) + Chr(10) + "Se eliminará el parámetro ingresado "
  End Select
  lblMensaje.Caption = mensaje
  txtCambiar.Text = PalabraError
  If TipoError = 7 Or TipoError = 8 Then PalabraError = ""
  txtCambiar.SelStart = 0
  txtCambiar.SelLength = Len(PalabraError)
End Sub

Private Sub cmdSalir_Click()
  TerminarCorregir = True
  Unload frmCorregir
End Sub

Private Sub lstNumeros_Click()
```

```
Dim palabra As String
  a = lstNumeros.ListIndex
  palabra = SelPalabra1 + lstNumeros.List(a)
  frmPrincipal.ActiveForm.txtEdicion.SelText = palabra
  TerminarCorregir = False
  Unload frmCorregir
End Sub

Private Sub lstOpciones_Click()
  Dim Argumento As String
  Dim i As Integer
  Dim a As Integer
  Dim letra As String
  Dim salto As Boolean
  Dim tipo As Integer
  'Esta funcion permite determinar el tipo de operando a ingresar así como
  'permitir seleccionar entre varios elementos
  a = lstOpciones.ListIndex
  Argumento = lstOpciones.List(a)
  salto = False
  For i = 0 To NumConstante - 1
    If Argumento = Constante(i) Then
      tipo = 1
      salto = True
      Exit For
    End If
  Next
  If Not salto Then
  For i = 0 To NumVariable - 1
    If Argumento = Variable(i) Then
      tipo = 2
      If Indice(i) <> ":" Then inicio = Indice(i)
      Exit For
    End If
  Next
  Else
    tipo = 1
  End If
  Select Case tipo
  Case 0 To 1
    txtCambiar.Text = Argumento
    lblEscoger.Caption = "Ha seleccionado " + Argumento
    lblEscoger.Visible = True
    cmdContinuar.SetFocus
  Case 2
    txtCambiar.Text = inicio
    lblEscoger.Caption = "Ha seleccionado " + Argumento + " ingrese el nuevo valor"
    lblEscoger.Visible = True
    txtCambiar.SetFocus
  End Select
End Sub

Private Sub txtCambiar_Change()
  'Habilita el cuadro de texto
  txtCambiar.Enabled = True
End Sub
```

```
'***  Cuadro de diálogo Buscar para búsquedas de texto.     ***
'***              Forma parte del EDITOR INTELIGENTE        ***
'*************************************************************

Option Explicit

Private Sub chkCaso_Click()
  'Asigna un valor a la variable pública.
  MAYminBuscar = chkCaso.Value
End Sub

Private Sub cmdCancelar_Click()
  'Guarda los valores en las variables públicas.
  TextoBuscar = txtBuscar.Text
  MAYminBuscar = chkCaso.Value
  'Descarga el cuadro de diálogo de búsqueda.
  Unload frmBuscar
End Sub

Private Sub cmdBuscar_Click()
  ' Asigna la cadena de texto a una variable pública.
  TextoBuscar = txtBuscar.Text
  ProcBuscarEsto
End Sub

Private Sub Form_Load()
  'Deshabilita el botón de búsqueda - todavía no hay texto que buscar.
  cmdBuscar.Enabled = False
  'Lee la variable pública y establece el botón de opción.
  optDireccion(DireccionBuscar).Value = 1
End Sub

Private Sub optDireccion_Click(Index As Integer)
  'Asigna un valor a la variable pública.
  DireccionBuscar = Index
End Sub

Private Sub txtBuscar_Change()
  ' Establece la variable pública.
  busPosicionInicial = True
  ' Si el cuadro de texto está vacío, deshabilita el botón de búsqueda.
  If txtBuscar.Text = "" Then
    cmdBuscar.Enabled = False
  Else
    cmdBuscar.Enabled = True
  End If
End Sub



'*** Formulario Cuadro de diálogo Buscar para reemplazar texto. ***
'***              Forma parte del EDITOR INTELIGENTE        ***
'*************************************************************

Option Explicit

Private Sub chkCaso_Click()
  'Asigna un valor a la variable pública.
  MAYminBuscar = chkCaso.Value
```

```
End Sub

Private Sub cmdCancelar_Click()
  'Guarda los valores en las variables públicas.
  TextoBuscar = txtBuscar.Text
  MAYminBuscar = chkCaso.Value
  'Descarga el cuadro de diálogo de búsqueda.
  Unload frmReplazar
End Sub

Private Sub cmdBuscar_Click()
  ' Asigna la cadena de texto a una variable pública.
  TextoBuscar = txtBuscar.Text
  ProcBuscarEsto
End Sub

Private Sub cmdReplazar_Click()
  ' Asigna la cadena de texto a una variable pública.
  frmEditor.txtEdicion.SelText = txtReplazar.Text
  ProcBuscarEsto
End Sub

Private Sub Form_Load()
  'Deshabilita el botón de búsqueda - todavía no hay texto que buscar.
  cmdBuscar.Enabled = False
  cmdReplazar.Enabled = False
  'Lee la variable pública y establece el botón de opción.
  optDireccion(DireccionBuscar).Value = 1
End Sub

Private Sub optDireccion_Click(Index As Integer)
  'Asigna un valor a la variable pública.
  DireccionBuscar = Index
End Sub

Private Sub txtBuscar_Change()
  ' Establece la variable pública.
  busPosicionInicial = True
  ' Si el cuadro de texto está vacío, deshabilita el botón de búsqueda.
  If txtBuscar.Text = "" Then
    cmdBuscar.Enabled = False
  Else
    cmdBuscar.Enabled = True
  End If
End Sub

Private Sub txtReplazar_Change()
  ' Si el cuadro de texto está vacío, deshabilita el botón reemplazar.
  If txtReplazar.Text = "" Then
    cmdReplazar.Enabled = False
  Else
    cmdReplazar.Enabled = True
  End If
End Sub


'*** Formulario Símbolos, permite insertar simbolos en el texto ***
'***          Forma parte del EDITOR INTELIGENTE          ***
```

```
'*****************************************************************

'Numero de caracteres ASCII a desplegar
Public Limite As Integer

Private Sub cmdCerrar_Click()
  Unload Me
End Sub

Private Sub cmdInsertar_Click()
  MSFlexGridSimbolos_DblClick
End Sub

Private Sub Form_Load()
  Dim i As Integer, filas As Integer
  Dim j As Integer, columnas As Integer
  Dim contador As Integer
  Dim Simbolos(222) As String
  '************************************************
  'Inicialización del arreglo de caracteres ASCII
  'Limite para el arreglo
  '************************************************
  Limite = 126 - 33
  'Dimensionamiento de las filas
  filas = 15
  columnas = 16
  'Inicio del contador de caracteres
  contador = 0 '32
  '************************************************
  'Carga de caracteres ASCII
  '************************************************
  Simbolos(1) = "!"  '33
  Simbolos(2) = """"
  Simbolos(3) = "#"
  Simbolos(4) = "$"
  Simbolos(5) = "%"
  Simbolos(6) = "&"
  Simbolos(7) = "'"
  Simbolos(8) = "("
  Simbolos(9) = ")"
  Simbolos(10) = "*"
  Simbolos(11) = "+" '43
  Simbolos(12) = ","
  Simbolos(13) = "-"
  Simbolos(14) = "."
  Simbolos(15) = "/"
  Simbolos(16) = "0"
  Simbolos(17) = "1"
  Simbolos(18) = "2"
  Simbolos(19) = "3"
  Simbolos(20) = "4"
  Simbolos(21) = "5" '53
  Simbolos(22) = "6"
  Simbolos(23) = "7"
  Simbolos(24) = "8"
  Simbolos(25) = "9"
  Simbolos(26) = ":"
  Simbolos(27) = ";"
  Simbolos(28) = "<"
```

```
Simbolos(29) = "="
Simbolos(30) = ">"
Simbolos(31) = "?" '63
Simbolos(32) = "@"
Simbolos(33) = "A"
Simbolos(34) = "B"
Simbolos(35) = "C"
Simbolos(36) = "D"
Simbolos(37) = "E"
Simbolos(38) = "F"
Simbolos(39) = "G"
Simbolos(40) = "H"
Simbolos(41) = "I" '73
Simbolos(42) = "J"
Simbolos(43) = "K"
Simbolos(44) = "L"
Simbolos(45) = "M"
Simbolos(46) = "N"
Simbolos(47) = "O"
Simbolos(48) = "P"
Simbolos(49) = "Q"
Simbolos(50) = "R"
Simbolos(51) = "S" '83
Simbolos(52) = "T"
Simbolos(53) = "U"
Simbolos(54) = "V"
Simbolos(55) = "W"
Simbolos(56) = "X"
Simbolos(57) = "Y"
Simbolos(58) = "Z"
Simbolos(59) = "["
Simbolos(60) = "\"
Simbolos(61) = "]" '93
Simbolos(62) = "^"
Simbolos(63) = "_"
Simbolos(64) = "`"
Simbolos(65) = "a"
Simbolos(66) = "b"
Simbolos(67) = "c"
Simbolos(68) = "d"
Simbolos(59) = "e"
Simbolos(70) = "f"
Simbolos(71) = "g" '103
Simbolos(72) = "h"
Simbolos(73) = "i"
Simbolos(74) = "j"
Simbolos(75) = "k"
Simbolos(76) = "l"
Simbolos(77) = "m"
Simbolos(78) = "n"
Simbolos(79) = "o"
Simbolos(80) = "p"
Simbolos(81) = "q" '113
Simbolos(82) = "r"
Simbolos(83) = "s"
Simbolos(84) = "t"
Simbolos(85) = "u"
Simbolos(86) = "v"
Simbolos(87) = "w"
```

```
  Simbolos(88) = "x"
  Simbolos(89) = "y"
  Simbolos(90) = "z"
  Simbolos(91) = "{" '123
  Simbolos(92) = "|"
  Simbolos(93) = "}" '125
  '*********************************
  'Llenado de la tabla de símbolos '
  '*********************************
  With MSFlexGridSimbolos
  .Cols = columnas
  .Rows = filas
  'Seteo de las columnas
  For i = 0 To columnas - 1
    .ColWidth(i) = TextWidth("AAAA")
    .ColAlignment(i) = flexAlignCenterCenter
  Next i
  'Llenado del cuadro de caracteres
  For j = 0 To filas - 1
    For i = 0 To columnas - 1
      contador = contador + 1
      .Col = i
      .Row = j
      If contador > Limite Then Exit Sub
      .Text = Simbolos(contador)
    Next i
  Next j
  MSFlexGridSimbolos.SetFocus
  End With
End Sub

Private Sub MSFlexGridSimbolos_DblClick()
  'Copia el contenido de la celda actual en el documento activo
  frmEditor.txtEdicion.SelRTF = MSFlexGridSimbolos.Text
End Sub

Private Sub msflexgridsimbolos_RowColChange()
  Dim valor As Integer
  'Muestra la letra seleccionada de forma ampliada
  lblCodigo.Caption = ""
  valor = 0
  With MSFlexGridSimbolos
    lblSeleccionado.Caption = .Text
    valor = .Cols * .Row + .Col + 1 + 32
    If valor < 127 And valor <= Limite + 32 + 1 Then
      lblCodigo.Caption = valor
    ElseIf (valor >= 127 And valor < 255) And valor <= Limite + 32 + 1 Then
       valor = valor + 1
      lblCodigo.Caption = valor
    Else
      lblCodigo.Caption = ""
    End If
  End With
End Sub


'*** Cuadro de diálogo Acerca de forma parte del EDITOR INTELIGENTE ***
'*********************************************************************
```

```
Private Sub cmdAceptar_Click()
  'Descarga el formulario
  Unload frmAcercaDe
End Sub



'*** Formulario de Presentación del EDITOR INTELIGENTE ***
'*********************************************************

Option Explicit



'*** Módulo global de la aplicación del EDITOR INTELIGENTE. ***
'**************************************************************
Option Explicit

' Tipo definido por el usuario para almacenar información sobre los formularios hijos
Type EstadoDelFormulario
  Borrar As Integer
  Modificar As Boolean
End Type
Public FEstado As EstadoDelFormulario  ' Matriz de tipos definidos por el usuario
Public TextoBuscar As String          ' Almacena el texto de búsqueda.
Public MAYminBuscar As Integer        ' Búsqueda sensible a mayúsculas y minúsculas
Public DireccionBuscar As Integer     ' Dirección de búsqueda.
Public busPosicionCursor As Integer   ' Almacena la posición del cursor
Public busPosicionInicial As Integer  ' Posición inicial.
Public Const EstaAplicacion = "Editor Inteligente"   ' Constante del Registro del sistema.
Public Const EstosArchivos = "Ultimos Archivos"      ' Constante del Registro del sistema.

Sub Main()
  'Inicializo el programa presentando el formulario presentar
  frmPresentar.Show
  frmPresentar.Refresh
  'Cargo el formulario principal
  Load frmEditor
  Unload frmPresentar
  frmEditor.Visible = True
End Sub

Sub ProcEdicionCopiar()
   ' Copia el texto seleccionado en el Portapapeles.
   Clipboard.SetText frmEditor.txtEdicion.SelText
End Sub

Sub ProcEdicionCortar()
   ' Copia el texto seleccionado en el Portapapeles.
   Clipboard.SetText frmEditor.txtEdicion.SelText
   ' Elimina el texto seleccionado.
   frmEditor.txtEdicion.SelText = ""
End Sub

Sub ProcEdicionPegar()
   ' Coloca el texto del Portapapeles en el control activo.
   frmEditor.txtEdicion.SelText = Clipboard.GetText()
End Sub
```

```
Sub ProcArchivoImprimir()
  Dim Texto As String
  With frmEditor
  On Error GoTo salir
  .ComDialogo1.CancelError = True
  .ComDialogo1.Flags = cdlPDReturnDC + cdlPDNoPageNums
  If .txtEdicion.SelLength = 0 Then
    .ComDialogo1.Flags = .ComDialogo1.Flags + cdlPDAllPages
  Else
    .ComDialogo1.Flags = .ComDialogo1.Flags + cdlPDSelection
  End If
  .ComDialogo1.ShowPrinter
  Texto = .txtEdicion.Text
  Printer.Print Texto
  Printer.EndDoc
  Exit Sub
salir:
  'Si se presioo Cancelar
  Exit Sub
  End With
End Sub

Sub ProcVerBarraHerramientas()
  ' Alterna la comprobación
  frmEditor.mnuVerBarraHerramientas.Checked = Not frmEditor.mnuVerBarraHerramientas.Checked
  ' Alterna la visibilidad de la barra de herramientas según este valor.
  If frmEditor.mnuVerBarraHerramientas.Checked Then
    frmEditor.tbBarHerramientas.Visible = True
  Else
    frmEditor.tbBarHerramientas.Visible = False
  End If
End Sub

Sub ProcVerBarraEstado()
  ' Alterna la comprobación
  frmEditor.mnuVerBarraEstado.Checked = Not frmEditor.mnuVerBarraEstado.Checked
  ' Alterna la visibilidad de la barra de estado según este valor.
  If frmEditor.mnuVerBarraEstado.Checked Then
    frmEditor.sbBarraDeEstado.Visible = True
  Else
    frmEditor.sbBarraDeEstado.Visible = False
  End If
End Sub

Sub ProcVerBarraPosicion()
  ' Alterna la comprobación
  frmEditor.mnuVerBarraPosicion.Checked = Not frmEditor.mnuVerBarraPosicion.Checked
  ' Alterna la visibilidad de la barra de estado según este valor.
  If frmEditor.mnuVerBarraPosicion.Checked Then
    frmEditor.sbBarraDeMensajes.Visible = True
  Else
    frmEditor.sbBarraDeMensajes.Visible = False
  End If
End Sub

Sub ProcBuscarEsto()
  Dim busInicio As Integer
  Dim busUltimaPosicion As Integer
```

```
  Dim busPalabra As String
  Dim busFuente As String
  Dim mbMensaje As String
  Dim mbRespuesta As Integer
  Dim intOffset As Integer
  ' Establece la variable de desplazamiento de acuerdo con la posición del cursor.
  If (busPosicionCursor = frmEditor.txtEdicion.SelStart) Then
    intOffset = 1
  Else
    intOffset = 0
  End If
  ' Lee la variable pública de la posición de inicio.
  If busPosicionInicial Then intOffset = 0
  ' Asigna el valor de inicio.
  busInicio = frmEditor.txtEdicion.SelStart + intOffset
  ' Si no distingue entre mayúsculas y minúsculas, convierte la cadena en mayúsculas
  If MAYminBuscar Then
    busPalabra = TextoBuscar
    busFuente = frmEditor.txtEdicion.Text
  Else
    busPalabra = UCase(TextoBuscar)
    busFuente = UCase(frmEditor.txtEdicion.Text)
  End If
  ' Busca la cadena.
  If DireccionBuscar = 1 Then
    busUltimaPosicion = InStr(busInicio + 1, busFuente, busPalabra)
  Else
    For busUltimaPosicion = busInicio - 1 To 0 Step -1
      If busUltimaPosicion = 0 Then Exit For
      If Mid(busFuente, busUltimaPosicion, Len(busPalabra)) = busPalabra Then Exit For
    Next
  End If
  ' Si se encuentra la cadena...
  If busUltimaPosicion Then
    frmEditor.txtEdicion.SelStart = busUltimaPosicion - 1
    frmEditor.txtEdicion.SelLength = Len(busPalabra)
    frmRemplazar.cmdRemplazar.Enabled = True
  Else
    mbMensaje = "La busqueda de " & Chr(34) & TextoBuscar & Chr(34) & " ha termindo"
    mbRespuesta = MsgBox(mbMensaje, 0, " EDITOR INTELIGENTE ")
    frmRemplazar.cmdRemplazar.Enabled = False
  End If
  ' Restablece las variables públicas
  busPosicionCursor = frmEditor.txtEdicion.SelStart
  busPosicionInicial = False
End Sub

Sub ProcObtenerUltimosArchivos()
  'Este procedimiento devuelve una matriz de valores desde el Registro del sistema.
  'El Registro del sistema contiene los archivos más recientemente abiertos.
  Dim i As Integer
  Dim varArchivos As Variant ' Variable para almacenar la matriz devuelta.
  'EstaAplicacion y EstosArchivos son constantes definidas en este módulo.
  If GetSetting(EstaAplicacion, EstosArchivos, "UltimoArchivo1") = Empty Then Exit Sub
  varArchivos = GetAllSettings(EstaAplicacion, EstosArchivos)
  For i = 0 To UBound(varArchivos, 1)
    frmEditor.mnuUltimoArchivo(0).Visible = True
    frmEditor.mnuUltimoArchivo(i + 1).Caption = varArchivos(i, 1)
    frmEditor.mnuUltimoArchivo(i + 1).Visible = True
```

```
  Next i
End Sub

Sub ProcExpandirCuadroEditor()
  ' Expande el cuadro de texto para que ocupe toda el área interna del formulario.
  With frmEditor
    If .ScaleHeight <> 0 Then
      'Si la barra de herramientas es visible el cuadro de texto se extiende hasta el
      'fin de la barra de herramientas
      If .tbBarHerramientas.Visible And .sbBarraDeMensajes.Visible Then
        .txtEdicion.Height = .ScaleHeight - .tbBarHerramientas.Height - .sbBarraDeMensajes.Height
        .txtEdicion.Top = .tbBarHerramientas.Height + .sbBarraDeMensajes.Height
        .txtEdicion.Width = .ScaleWidth
      ElseIf Not .tbBarHerramientas.Visible And .sbBarraDeMensajes.Visible Then
        .txtEdicion.Height = .ScaleHeight - .sbBarraDeMensajes.Height
        .txtEdicion.Top = .sbBarraDeMensajes.Height
        .txtEdicion.Width = .ScaleWidth
      ElseIf .tbBarHerramientas.Visible And Not .sbBarraDeMensajes.Visible Then
        .txtEdicion.Height = .ScaleHeight - .tbBarHerramientas.Height
        .txtEdicion.Top = .tbBarHerramientas.Height
        .txtEdicion.Width = .ScaleWidth
      Else
        .txtEdicion.Height = .ScaleHeight
        .txtEdicion.Top = 0
        .txtEdicion.Width = .ScaleWidth
      End If
      'Si la barra de estado es visible el cuadro de texto se 3xtiende hasta el
      'incio de la barra de estado
      If .sbBarraDeEstado.Visible Then
        .txtEdicion.Height = .txtEdicion.Height - .sbBarraDeEstado.Height
      End If
    End If
  End With
End Sub

Sub ProcEscribirUltimoArchivo(UltimoArchivoAbierto)
  ' Este procedimiento utiliza la instrucción SaveSettings para escribir los nombres
  ' de los archivos más recientes en el Registro del sistema.
  Dim i As Integer
  Dim strFile As String
  Dim strKey As String
  ' Copia UltimoArchivo1 en UltimoArchivo2, y así sucesivamente.
  For i = 3 To 1 Step -1
    strKey = "UltimoArchivo" & i
    strFile = GetSetting(EstaAplicacion, EstosArchivos, strKey)
    If strFile <> "" Then
      strKey = "UltimoArchivo" & (i + 1)
      SaveSetting EstaAplicacion, EstosArchivos, strKey, strFile
    End If
  Next i
  ' Escribe al archivo abierto como primer archivo más reciente.
  SaveSetting EstaAplicacion, EstosArchivos, "UltimoArchivo1", UltimoArchivoAbierto
End Sub

Sub ProcNombreArchivo()
  Dim bolrespuesta As Boolean
  Dim NombreArchivo As String
  Dim mensaje As String
  Dim Respuesta As Integer
```

```
  Dim Comentario As String
  On Error GoTo salir
  ' Si el archivo ha sido modificado, lo guarda
  If FEstado.Modificar = True Then
    NombreArchivo = frmEditor.Caption
    Comentario = Right(frmEditor.Caption, Len(frmEditor.Caption) - 21)
    mensaje = "El texto en [" & Comentario & "] ha cambiado."
    mensaje = mensaje & vbCrLf
    mensaje = mensaje & "¿Desea guardar los cambios?"
    Respuesta = MsgBox(mensaje, 51, "EDITOR INTELIGENTE")
    Select Case Respuesta
    Case 6     ' El usuario eligió Sí.
      bolrespuesta = FuncArchivoGuardar
      If bolrespuesta = False Then Exit Sub
    Case 2     ' El usuario eligió Cancelar. Cancela la descarga.
      Exit Sub
    End Select
  End If
  frmEditor.ComDialogo1.CancelError = True
  frmEditor.ComDialogo1.filename = ""
  frmEditor.ComDialogo1.Filter = "Archivos de Programa (*.asm)|*.asm|Archivos de Texto
(*.txt)|*.txt|"
  frmEditor.ComDialogo1.ShowOpen
  NombreArchivo = frmEditor.ComDialogo1.filename
  'Permite abrir el archivo seleccionado en el cuadro de dialogo abrir
  ProcAbrirArchivo (NombreArchivo)
  MenuArchivoUltimoDato (NombreArchivo)
salir:
  'Si se presionó Cancelar
  Exit Sub
End Sub

Sub ProcAbrirArchivo(NombreArchivo)
  On Error Resume Next
  frmEditor.txtEdicion.LoadFile NombreArchivo, 1
  ' Abre el archivo seleccionado.
  If Err Then
    MsgBox "Imposible abrir el archivo: " + NombreArchivo
    Exit Sub
  End If
  ' Cambia el puntero del mouse a reloj de arena.
  Screen.MousePointer = 11
  ' Modifica el título del formulario y presenta el nuevo texto.
  frmEditor.Caption = "EDITOR INTELIGENTE - " & NombreArchivo
  frmEditor.txtEdicion.SelLength = Len(frmEditor.txtEdicion.Text)
  frmEditor.txtEdicion.SelColor = QBColor(0)
  frmEditor.txtEdicion.SelStart = 0
  ' Restablece el puntero del mouse.
  Screen.MousePointer = 0
  'Cargo el programa en el editor
  ProcCargarPrograma
  If Not Alinear Then FEstado.Modificar = False
  If Alinear Then FEstado.Modificar = True
End Sub

Sub ProcArchivoGuardar(NombreArchivo)
  Dim ContenidoArchivo As String
  On Error Resume Next
  ' Abre el archivo.
```

```
  Open NombreArchivo For Output As #1
  ' Coloca el contenido del Editor Inteligente en una variable.
  ContenidoArchivo = frmEditor.txtEdicion.Text
  ' Presenta el puntero reloj de arena.
  Screen.MousePointer = 11
  ' Escribe el contenido de la variable en un archivo.
  Print #1, ContenidoArchivo
  Close #1
  ' Restablece el puntero del mouse.
  Screen.MousePointer = 0
  ' Establece el título del formulario.
  If Err Then
    MsgBox Error, 48, App.Title
  Else
    frmEditor.Caption = "EDITOR INTELIGENTE - " & NombreArchivo
    ' Restablece el indicador Modificar.
    FEstado.Modificar = False
  End If
End Sub

Sub MenuArchivoUltimoDato(NombreArchivo)
  Dim bolrespuesta As Boolean
  ' Comprueba si el archivo abierto se encuentra en la matriz de controles del menú Archivo.
  bolrespuesta = FuncListaUltimosArchivos(NombreArchivo)
  If Not bolrespuesta Then
    ' Escribe el nombre del archivo abierto en el Registro del sistema.
    ProcEscribirUltimoArchivo (NombreArchivo)
  End If
  ' Actualiza la lista de los archivos abiertos recientemente en la matriz de controles
  ' del menú Archivo.
  ProcObtenerUltimosArchivos
End Sub

Sub ProcCargarPrograma()
  Dim posicion As Integer
  Dim inicio As Integer
  Dim fin As Integer
  Dim Mitad As Integer
  Dim Numero As Integer
  Dim Puntofin As Integer
  Dim Ultimalinea As Integer
  Dim Finprograma As Boolean
  Dim mensaje As String
  'Nos permite cambiar de color las instrucciones del programa
  With frmEditor
  'Primero se alinea el programa
  Call ProcAlinear
  Screen.MousePointer = 11
  .txtEdicion.Visible = False
  posicion = .txtEdicion.SelStart
  .txtEdicion.SelStart = 0
  mensaje = .sbBarraDeEstado.Panels(1).Text
  .sbBarraDeEstado.Panels(1).Text = "Espere mientras se carga el programa, esto puede tardar varios
minutos"
  Mitad = 0
  inicio = Mitad
  Finprograma = False
  'Determino cual es la ultima linea del programa
  Ultimalinea = .txtEdicion.GetLineFromChar(Len(.txtEdicion.Text))
```

```
  Do While Not Finprograma = True
    'Ubico el puntero en el inicio de la instrucción
    inicio = Mitad + 1
    'Ubico el puntero en el fin de la instrucción
    fin = FinInstruccion(Mitad)
    'fin = fin - 1
    'Separo la instruccion en sus componentes
    Call ProcSepararInstruccion(inicio, fin)
    'Verifico si existen errores en al instruccion
    Numero = CorregirInstruccion(inicio)
    Select Case Numero
    Case -1
      Call CambiarColorComentarios(inicio, fin)
    Case -2
      Call CambiarColorSeudoOpcode(inicio, fin)
    Case Else
      Call CambiarColorError(inicio, fin)
    End Select
    Mitad = fin + 2
    Puntofin = .txtEdicion.GetLineFromChar(Mitad + 1)
    'Si el puntero se halla en el fin del programa
    If Puntofin = Ultimalinea Then Finprograma = True
    If Puntofin <> Ultimalinea Then Finprograma = False
  Loop
  Screen.MousePointer = 0
  .txtEdicion.Visible = True
  .txtEdicion.SelStart = posicion
  .txtEdicion.SetFocus
  .txtEdicion.SelColor = QBColor(0)
  .sbBarraDeEstado.Panels(1).Text = mensaje
  End With
End Sub

Sub ProcArchivoSalir()
  Dim Comentario As String
  Dim mensaje As String
  Dim Respuesta As Integer
  Dim NombreArchivo As String
  Dim res As Boolean
  ' Si el archivo ha sido modificado, lo guarda
  If FEstado.Modificar = True Then
    NombreArchivo = frmEditor.Caption
    Comentario = Right(frmEditor.Caption, Len(frmEditor.Caption) - 21)
    mensaje = "El texto en [" & Comentario & "] ha cambiado."
    mensaje = mensaje & vbCrLf
    mensaje = mensaje & "¿Desea guardar los cambios?"
    Respuesta = MsgBox(mensaje, 51, "EDITOR INTELIGENTE")
    Select Case Respuesta
    Case 6     ' El usuario eligió Sí.
      res = FuncArchivoGuardar
    Case 2     ' El usuario eligió Cancelar. Cancela la descarga.
      Exit Sub
    End Select
  End If
  'Salir del programa
  End
End Sub

Function FuncArchivoGuardar() As Boolean
```

```
  Dim NombreArchivo As String
  If frmEditor.Caption = "EDITOR INTELIGENTE - Programa 1" Then
    ' El archivo no ha sido guardado aún.
    ' Obtiene el nombre del archivo, y después llama al procedimiento de guardar.
    NombreArchivo = FuncGuardarNombreArchivo(NombreArchivo)
  Else
    ' El título del formulario contiene el nombre del archivo abierto.
    NombreArchivo = Right(frmEditor.Caption, Len(frmEditor.Caption) - 21)
  End If
  ' Llama al procedimiento de guardar. Si NombreArchivo está vacía, es porque
  ' el usuario eligió Cancelar en el cuadro de diálogo Guardar como; si no,
  ' guarda el archivo.
  If NombreArchivo <> "" Then
    ProcArchivoGuardar NombreArchivo
    FuncArchivoGuardar = True
  Else
    FuncArchivoGuardar = False
  End If
End Function

Function FuncGuardarNombreArchivo(NombreArchivo As Variant)
  ' Presenta un cuadro de diálogo Guardar como y devuelve un nombre de archivo.
  ' Si el usuario elige Cancelar, devuelve una cadena vacía.
  On Error GoTo salir
  frmEditor.ComDialogo1.CancelError = True
  frmEditor.ComDialogo1.filename = NombreArchivo
  frmEditor.ComDialogo1.Filter = "Archivos de Edicion (*.asm)|*.asm|Archivos de Texto (*.txt)|*.txt"
  frmEditor.ComDialogo1.ShowSave
  FuncGuardarNombreArchivo = frmEditor.ComDialogo1.filename
  Exit Function
salir:
  FuncGuardarNombreArchivo = ""
End Function

Function FuncListaUltimosArchivos(NombreArchivo) As Boolean
  Dim i As Integer        ' Variable contador.
  For i = 1 To 4
    If frmEditor.mnuUltimoArchivo(i).Caption = NombreArchivo Then
      FuncListaUltimosArchivos = True
      Exit Function
    End If
  Next i
  FuncListaUltimosArchivos = False
End Function



'*** Módulo estándar que contiene procedimientos que detectan los caracteres ingresados ***
'***                  Forma parte del EDITOR INTELIGENTE.                  ***
'*****************************************************************************
********
'Declaración de variables públicas
Public indComentario As Boolean    'Se ha presionado " ; " para ingresar comentarios
Public ErrorInstruccion As Boolean 'Se pone en true cuando se comete un error en la instrucción
Public AyudaInstruccion As Boolean 'Permite presentar la ayuda en pantalla
Public opcAyuda As Boolean         'Presentar la ayuda en pantalla
Public opcTeclado As Boolean       'Utilizar el teclado en la ayuda
Public opcCorregir As Boolean      'Corregir en cada caracter ingresado
Public opcCorregirFin As Boolean   'Corregir al final de la instrucción
```

```
Public opcNoCorregir As Boolean    'Corregir al final del programa
Public valEtiqueta As String       'Guarda la Etiqueta ingresada
Public valOpcode As String         'Guarda el Opcode ingresado
Public valArgumento1 As String     'Guarda el primer Argumento ingresado
Public valArgumento2 As String     'Guarda el segundo Argumento ingresado
Public Campo As Integer            'Guarda el número de Tabs ingresados
Public numPalabra As Integer       'Determina en que palabra se halla el cursor
Public Observacion As Integer      'Define el comentario que se presentará en la barra de estado
Public NumeroLinea As Integer      'Define la posición actual de la línea
Public InicioArgumento As String   'Define el inicio de un argumento
Public Mnemonico(6) As String      'Guardo las palabras que forman la instruccion
Public Constante() As String       'Almacena las variables que existen en la base de datos en una matriz
Public Variable() As String        'Almacena las constantes que existen en la base de datos en una
matriz
Public Indice() As String          'Guarda el indice de las variables
Public NumConstante As Integer     'Almacena el numero de variables que existen en la base de datos
Public NumVariable As Integer      'Almacena el numero de constantes que existen en la base de datos
Public TerminarCorregir As Boolean 'Salir de la correccion del programa
Public IgnorarLinea As Boolean     'Se ignora la linea
Public Alinear As Boolean          'Determina que el programa cargo se ha alineado
Public PalabraError As String
Public TipoError As Integer
Public LetraError As String

Sub ProcCasoComentario()
  'Seteo las varibles globales para el ingreso de comentarios
  indComentario = True
  Observacion = 40
  Campo = 3
End Sub

Function CaracterErrado(Caracter As Integer) As Boolean
  'Definimos que tecla se ha presionado y si se produce errores en el
  'ingresos del caracter
  Select Case Caracter
  Case 13              'Caso ENTER
    If Not CasoEnter Then CaracterErrado = True
  Case 9               'Caso TAB
    If indComentario = False Then
      If Not CasoTab Then CaracterErrado = True
    End If
  Case 59              'Caso comentario (presiono ;)
    ProcCasoComentario
  Case 8               'Caso suprimir caracter anterior
    ProcPosicionCursor
    frmEditor.Lista1.Visible = False
    frmEditor.txtOperando.Visible = False
  Case Else            'Caso otros caracteres
    If Not indComentario Then
      'Si se permite corregir se elimiinará el caracter ingresado
      CaracterErrado = False
      If Not CasoCaracter(Caracter) Then CaracterErrado = True
    End If
  End Select
End Function

Function CasoCaracter(Caracter As Integer) As Boolean
  Dim i As Integer
  Dim indInicio As Boolean
```

```
Dim letra As String
Dim palabra As String
Dim Argumento1 As String
Dim Argumento2 As String
Dim posicion As Integer
With frmEditor
Select Case Campo
'**Parte inicial para el ingreso de etiquetas**
Case 0
  'Verifico si es el primer caracter de la instrucción
  If .txtEdicion.SelStart <> 0 Then
    posicion = .txtEdicion.SelStart
    letra = Mid(.txtEdicion.Text, posicion, 1)
  Else
    letra = Chr(10)
  End If
  If letra = Chr(10) Then indInicio = True
  If letra <> Chr(10) Then indInicio = False
  'Compruebo si los caracteres ingresados son permitidos como etiqueta
  If CaracterValido(Caracter, indInicio) Then
    CasoCaracter = True
    If Not ErrorInstruccion Then ProcMensajeError (10)
    If ErrorInstruccion Then ProcMensajeError (1)
  Else
    If indInicio Then ProcMensajeError (11)
    If Not indInicio Then ProcMensajeError (12)
    Observacion = 1
    .Timer1.Enabled = True
    ErrorInstruccion = True
    If opcCorregir Then
      CasoCaracter = False
      ErrorInstruccion = False
    Else: CasoCaracter = True
    End If
  End If
'**Parte para la definición de Mnemónicos**
Case 1
  CasoCaracter = True
  If Not ErrorInstruccion Then
    'Verifico si es el primer caracter del Opcode
    posicion = .txtEdicion.SelStart
    letra = Mid(.txtEdicion.Text, posicion, 1)
    If letra = Chr(9) Then
      If valEtiqueta = "Existe" Or valEtiqueta = "No existe" Then
        'Lleno la lista  con los Opcodes relacionados
        'Verifico si existen elementos en la lista
        palabra = ListaOpcode(Caracter, valEtiqueta)
        If palabra <> "" Then
          .Lista1.Height = .Lista1.ListCount
          'Ubico la lista en la posición que debe mostrarse
          ProcPosicionLista (1)
          If opcAyuda Then .Lista1.Visible = True
          CasoCaracter = True
          ProcMensajeError (23)
        'Si no existen elementos en la lista
        Else
          If opcCorregir Then
            CasoCaracter = False
            .Lista1.Visible = False
```

```
            Else
              CasoCaracter = True
            End If
            ProcMensajeError (21)
            Observacion = 20
            .Timer1.Enabled = True
          End If
        End If
      Else
        'Puedo ingresar cualquier caracter
        CasoCaracter = True
      End If
    Else
      ProcMensajeError (2)
    End If
  '**Parte para la definición de Argumentos**
  Case 2
    CasoCaracter = True
    If Not ErrorInstruccion Then
      Select Case numPalabra
      'Area para la definición del Argumento1
      Case 3
      If Caracter = Asc(Chr(44)) Then
        numPalabra = 4
        'Verifico cuantos caracteres han ingresado como Argumento1
        palabra = PalabraIngresada(9)
        Argumento1 = BuscarArgumento(palabra)
        'Si la palabra ingresada no corresponde a Argumento existente
        If Argumento1 = "Error" Or Argumento1 = "" Then
          .Lista1.Visible = False
          If opcCorregir = True Then
            ProcCortarPalabra (palabra)
            CasoCaracter = False
            numPalabra = 3
             If opcAyuda Then .Lista1.Visible = True
          End If
          ProcMensajeError (31)
          If opcCorregir Then Observacion = 33
          If Not opcCorregir Then Observacion = 3
          .Timer1.Enabled = True
        Else
          ProcMensajeError (33)
          valArgumento1 = Argumento1
          Numero = MostrarArgumento2(valOpcode, valArgumento1)
          CasoCaracter = False
        End If
      End If
      'Area para la definición del argumento 2
      Case 4
      If Caracter = Asc(Chr(44)) Then
        numPalabra = 5
        'Verifico cuantos caracteres han ingresado como Argumento2
        palabra = PalabraIngresada(44)
        Argumento2 = BuscarArgumento(palabra)
        'Si la palabra ingresada no corresponde a Argumento existente
        If Argumento2 = "Error" Then
          If opcCorregir = True Then
            ProcCortarPalabra (palabra)
            CasoCaracter = False
```

```
            numPalabra = 4
            If opcAyuda Then .Lista1.Visible = True
          End If
          ProcMensajeError (31)
          Observacion = 33
          .Timer1.Enabled = True
        Else
          ProcMensajeError (35)
          valArgumento2 = Argumento2
          Numero = MostrarArgumento3(valOpcode, valArgumento1, valArgumento2)
          CasoCaracter = False
        End If
      End If
    End Select
  Else
    ProcMensajeError (3)
  End If
  Case 3
    CasoCaracter = True
    'El comentario debe iniciarse con punto y coma (;)
    If Caracter <> 59 Then
      If opcCorregir Then CasoCaracter = False
      ProcMensajeError (41)
      Observacion = 4
      .Timer1.Enabled = True
    End If
  End Select
  End With
End Function

Function CasoEnter() As Boolean
  Dim fin As Integer
  Dim inicio As Integer
  Dim Numero As Integer
  Dim j As Integer
  With frmEditor
    .Lista1.Visible = False
    .txtOperando.Visible = False
    CasoEnter = True
    'Defino el punto final de la instruccion
    fin = .txtEdicion.SelStart
    'Ubico el punto inicial de la instruccion
    inicio = InicioInstruccion(fin)
    'Separo la instruccion en sus componentes
    Call ProcSepararInstruccion(inicio, fin)
    'Verifico si existen errores en al instruccion
    Numero = CorregirInstruccion(inicio)
    Select Case Numero
    Case -2
      Call CambiarColorSeudoOpcode(inicio, fin)
    Case -1
      Call CambiarColorComentarios(inicio, fin)
    Case Else
      Call CambiarColorError(inicio, fin)
      If opcCorregir Or opcCorregirFin Then CasoEnter = False
    End Select
    If CasoEnter Then ProcIniciarVariables
    .txtEdicion.SelStart = fin
    .txtEdicion.SelColor = QBColor(0)
```

```
  End With
End Function

Function CasoTab() As Boolean
  Dim i As Integer
  Dim posicion As Integer
  Dim lenPalabra As Integer
  Dim palabra As String
  Dim Argumento1 As String
  Dim Argumento2 As String
  Dim Opcode As String
  Dim ValorError As Integer
  Dim opcion As String
  Dim ValorArgumento As Integer
  With frmEditor
  CasoTab = True
  Select Case Campo
  'Paso al area de Opcode
  Case 0
    Campo = 1
    numPalabra = 2
    'Guardo la etiqueta en una variable para buscar errores
    posicion = .txtEdicion.SelStart
    'Busco el caracter salto de línea
    lenPalabra = NumeroCaracteres(10)
    .txtEdicion.SelStart = .txtEdicion.SelStart - lenPalabra
    .txtEdicion.SelLength = lenPalabra
    palabra = .txtEdicion.SelText
    .txtEdicion.SelStart = posicion
    'LLamo a la funcion ErrorEtiqueta para obtener errores
    If palabra <> "" Then
      ValorError = ErrorEtiqueta(palabra)
      'si no existe errores en la etiqueta
      If ValorError = 0 Then
        ErrorInstruccion = False
        ProcMensajeError (20)
        valEtiqueta = "Existe"
      'Si la etiqueta es una Opción
      ElseIf ValorError = -2 Then .
        ErrorInstruccion = False
        ProcMensajeError (14)
        valEtiqueta = UCase(Mid(palabra, 2))
        'Mostramos los argumentos existentes
        CasoTab = False
        ValorArgumento = MostrarOpcion(valEtiqueta)
      'Si la etiqueta esta mal definida
      Else
        If Not opcCorregir Then
          ErrorInstruccion = True
          ProcMensajeError (13)
          Observacion = 2
          .Timer1.Enabled = True
        Else
          'Elimino la palabra ingresada
          .txtEdicion.SelStart = .txtEdicion.SelStart - lenPalabra
          .txtEdicion.SelLength = lenPalabra
          .txtEdicion.SelText = ""
          ErrorInstruccion = False
          Campo = 0
```

```
            numPalabra = 1
            CasoTab = False
            ProcMensajeError (13)
            Observacion = 1
            .Timer1.Enabled = True
            Exit Function
          End If
        End If
      Else
        ErrorInstruccion = False
        valEtiqueta = "No existe"
      End If
    Case 1
      'Comparo la palabra ingresada con la base de datos
      Campo = 2
      numPalabra = 3
      .Lista1.Visible = False
      If Not ErrorInstruccion Then
        'Copio en  la variable Opcode los caracteres ingresados
        Opcode = PalabraIngresada(9)
        'Mostramos los operandos existentes
        CasoTab = False
        ValorArgumento = MostrarArgumento1(Opcode, valEtiqueta)
        Select Case ValorArgumento
        'Si el Opcode ingresado no existe
        Case -1
          ProcMensajeError (22)
          'Corto la palabra ingresada si se corrige en al escritura
          If opcCorregir Then
            ProcCortarPalabra (Opcode)
            Campo = 1
            numPalabra = 2
            Observacion = 20
            frmEditor.Timer1.Enabled = True
            CasoTab = False
          Else
            Observacion = 3
            frmEditor.Timer1.Enabled = True
            ErrorInstruccion = True
            CasoTab = True
          End If
        Case Else
          valOpcode = Opcode
        End Select
      Else
        ProcMensajeError (3)
      End If
    Case 2
    CasoTab = True
    .Lista1.Visible = False
      If Not ErrorInstruccion Then
        'Area para la definición del argumento 2
        'Verifico cuantos caracteres han ingresado como Argumento2
        If numPalabra <> 3 Then
          palabra = PalabraIngresada(44)
          Argumento2 = BuscarArgumento(palabra)
          'Si la palabra ingresada no corresponde a Argumento existente
          If Argumento2 = "Error" Or Argumento2 = "" Then
            If opcCorregir = True Then
```

```
            ProcCortarPalabra (palabra)
            CasoTab = False
            numPalabra = 4
            If opcAyuda Then .Lista1.Visible = True
          End If
          ProcMensajeError (31)
          If opccoregir Then Observacion = 34
          If Not opccoregir Then Observacion = 4
          .Timer1.Enabled = True
          Campo = 2
        Else
          ProcMensajeError (34)
          valArgumento2 = Argumento2
          Numero = MostrarArgumento3(valOpcode, valArgumento1, palabra)
          CasoTab = False
        End If
      'Si se presiona un tab luego del primer argumento
      Else
        numPalabra = 4
        'Verifico cuantos caracteres han ingresado como Argumento1
        palabra = PalabraIngresada(9)
        Argumento1 = BuscarArgumento(palabra)
        'Si la palabra ingresada no corresponde a Argumento existente
        If Argumento1 = "Error" Or Argumento1 = "" Then
          .Lista1.Visible = False
          If opcCorregir = True Then
            ProcCortarPalabra (palabra)
            CasoTab = False
            numPalabra = 3
            If opcAyuda Then .Lista1.Visible = True
          End If
          ProcMensajeError (31)
          If opcCorregir Then Observacion = 33
          If Not opcCorregir Then Observacion = 3
          .Timer1.Enabled = True
        Else
          ProcMensajeError (33)
          valArgumento1 = Argumento1
          Numero = MostrarArgumento2(valOpcode, valArgumento1)
          CasoTab = False
        End If
      End If
    Else
      ProcMensajeError (4)
    End If
  End Select
  End With
End Function


'*** Módulo estándar que contiene procedimientos para corregir el programa.  ***
'*** Forma parte de la aplicación del EDITOR INTELIGENTE.                    ***
'*********************************************************************************
Option Explicit

Sub ProcCorregirErrores()
  Dim j As Integer
  Dim i As Integer
  Dim n As Integer
```

```
  Dim posicion As Integer
  Dim inicio As Integer
  Dim Mitad As Integer
  Dim fin As Integer
  Dim Ultimalinea As Integer
  Dim tipo As String
  Dim valor As Integer
  Dim Puntofin As Integer
  Dim comparar As Integer
  Dim Caracter As Integer
  Dim letra As String
  Dim palabra As String
  Dim Opcode As String
  Dim Etiqueta As String
  Dim Argumento1 As String
  Dim Argumento2 As String
  Dim Argumento3 As String
  Dim mensaje As String
  Dim Finprograma As Boolean
  Dim Error As Boolean
  Dim indError As Boolean

  '** Procedimiento que permite correigir los errores **
  '**          en todo el texto del programa.        **
  'Guardo la ultima posición en la que se halla el cursor
  With frmEditor
  mensaje = .sbBarraDeEstado.Panels(1).Text
  .sbBarraDeEstado.Panels(1).Text = "Se procede a Corregir los Errores del Programa"
  posicion = .txtEdicion.SelStart
  .txtEdicion.SelStart = 0
  Mitad = 0
  inicio = Mitad
  TerminarCorregir = False
  Finprograma = False
  'Determino cual es la ultima linea del programa
  Ultimalinea = .txtEdicion.GetLineFromChar(Len(.txtEdicion.Text))
  Do While Not Finprograma = True
    'Ubico el puntero en el inicio de la instrucción
    inicio = Mitad + 1
    'Ubico el puntero en el fin de la instrucción
    fin = FinInstruccion(Mitad)
    Error = True    'Inicilizo la variable Error
    Do While Error = True
      'Separo la instruccion en sus componentes
      Call ProcSepararInstruccion(inicio, fin)
      Error = False
      ' ** Esta parte corrige los errores existentes en la instrucción **
      If Mnemonico(0) <> "" Or Mnemonico(1) <> "" Or Mnemonico(2) <> "" Or Mnemonico(3) <>
"" Or Mnemonico(4) <> "" Then
        ' Corrige los errores que existen en la Etiqueta
        If Mnemonico(0) <> "" Then
          'Encuentra los Errores que se producen en la Etiqueta
          valor = ErrorEtiqueta(Mnemonico(0))
          Select Case valor
          Case -2
            Etiqueta = Mid(Mnemonico(0), 2)
          Case -1
            'Resalto la palabra que tiene error
            .txtEdicion.SelStart = inicio - 1
```

```
        .txtEdicion.SelLength = Len(Mnemonico(0))
        PalabraError = Mnemonico(0)
        LetraError = Mid(Mnemonico(0), 1, 1)
        'Llamo al formulario Corregir
        Beep
        TipoError = 1
        frmCorregir.Show vbModal
        Error = True
      'La etiqueta está bien escrita
      Case 0
        Etiqueta = "Existe"
      Case 1
        'Resalto la palabra que tiene error
        .txtEdicion.SelStart = inicio - 1
        .txtEdicion.SelLength = Len(Mnemonico(0))
        PalabraError = Mnemonico(0)
        LetraError = Mid(Mnemonico(0), 1, 1)
        'Llamo al formulario Corregir
        Beep
        TipoError = 1
        frmCorregir.Show vbModal
        Error = True
      Case Else
        If Error = False Then
          .txtEdicion.SelStart = inicio - 1
          .txtEdicion.SelLength = Len(Mnemonico(0))
          PalabraError = Mnemonico(0)
          LetraError = Mid(Mnemonico(0), valor, 1)
          'Llamo la formulario Corregir
          Beep
          TipoError = 2
          frmCorregir.Show vbModal
          Error = True
        End If
      End Select
    Else
      Etiqueta = "No existe"
    End If

    'Corrige los errores existentes en la definición del Opcode
    If Error = False Then
      If Mnemonico(1) <> "" Then
        letra = Mid(Mnemonico(1), 1, 1)
        Caracter = Asc(letra)
        palabra = ListaOpcode(Caracter, Etiqueta)
        'Verifico si existen elementos en la lista
        If .Lista1.ListCount = 0 Then
          .txtEdicion.SelStart = inicio + Len(Mnemonico(0))
          .txtEdicion.SelLength = Len(Mnemonico(1))
          'Llamo al formulario Corregir si no se halló ningun Opcode parecido
          Beep
          PalabraError = Mnemonico(1)
          If Mid(Mnemonico(0), 1, 1) <> Chr(36) Then
            TipoError = 3
            frmCorregir.Show vbModal
          Else
            TipoError = 13
            frmCorregir.Show vbModal
          End If
```

```
            Error = True
          Else
            If Mid(Mnemonico(0), 1, 1) <> Chr(36) Then
              For i = 0 To .Lista1.ListCount - 1
                comparar = StrComp(Mnemonico(1), .Lista1.List(i), 1)
                If comparar = 0 Then
                  indError = False
                  Opcode = .Lista1.List(i)
                  i = .Lista1.ListCount
                Else
                  indError = True
                End If
              Next
            Else
              Opcode = palabra
              indError = False
            End If
            'Si no existe Opcode parecido llamo al formulario Corregir
            If indError = True Then
              .txtEdicion.SelStart = inicio + Len(Mnemonico(0))
              .txtEdicion.SelLength = Len(Mnemonico(1))
              Beep
              PalabraError = Mnemonico(1)
              TipoError = 4
              frmCorregir.Show vbModal
              Error = True
            End If
          End If
        Else
          palabra = ListaOpcode(Asc(" "), Etiqueta)
          If .Lista1.ListCount <> 0 Then
            .txtEdicion.SelStart = inicio + Len(Mnemonico(0))
            Beep
            TipoError = 12
            frmCorregir.Show vbModal
            Error = True
          Else
            Opcode = ""
          End If
        End If
        'Si  no existe Opcode y tenemos operandos, llamo al formulario corregir
        If Mnemonico(1) = "" And (Mnemonico(2) <> "" Or Mnemonico(3) <> "") Then
          Opcode = ""
          .txtEdicion.SelStart = inicio + Len(Mnemonico(0))
          Beep
          TipoError = 9
          frmCorregir.Show vbModal
          Error = True
        End If
      End If

      'Corrige los errores existentes en la definición del Argumento 1
      If Error = False Then
        palabra = ListaArgumento1(Opcode, Etiqueta)
        If Mnemonico(2) <> "" Then
          'Verifica si la instrucción ingresada es un SeudoOpcode
          .Data1.RecordSource = "Select Distinct Tipo From Instrucciones Where (Opcode='" +
Opcode + "' )"
          .Data1.Refresh
```

```
            Do While Not .Data1.Recordset.EOF
              If .Data1.Recordset(0) <> "" Then
                tipo = .Data1.Recordset(0)
                .Data1.Recordset.MoveNext
              Else
                tipo = "Seudoopcode"
                Error = True
              End If
            Loop
            If tipo = "Seudoopcode" Or tipo = "" Then
              Error = True
            Else
              Error = False
            End If
            If Not Error Then
              If .Lista1.ListCount <> 0 Then
                Argumento1 = BuscarArgumento(Mnemonico(2))
                If Argumento1 = "Error" Then
                  .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) + 1
                  .txtEdicion.SelLength = Len(Mnemonico(2))
                  Beep
                  PalabraError = Mnemonico(2)
                  TipoError = 5
                  frmCorregir.Show vbModal
                  Error = True
                End If
              Else
                .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) + 1
                .txtEdicion.SelLength = Len(Mnemonico(2))
                Beep
                PalabraError = Mnemonico(2)
                TipoError = 10
                frmCorregir.Show vbModal
                Error = True
              End If
            End If
          Else
            If .Lista1.ListCount <> 0 Then
              .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1))
              Beep
              TipoError = 7
              frmCorregir.Show vbModal
              Error = True
            Else
              Argumento1 = ""
            End If
          End If
        End If

        'Corrige los errores existentes en la definición del Argumento 2
        If Error = False Then
          palabra = ListaArgumento2(Opcode, Argumento1)
          If Mnemonico(3) <> "" Then
            If .Lista1.ListCount <> 0 Then
              Argumento2 = BuscarArgumento(Mnemonico(3))
              If Argumento2 = "Error" Then
                .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + 2
                .txtEdicion.SelLength = Len(Mnemonico(3))
```

```
                Beep
                PalabraError = Mnemonico(3)
                TipoError = 6
                frmCorregir.Show vbModal
                Error = True
              End If
            Else
              .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + 1
              .txtEdicion.SelLength = Len(Mnemonico(3)) + 1
              Beep
              PalabraError = Mnemonico(3)
              TipoError = 10
              frmCorregir.Show vbModal
              Error = True
            End If
          Else
            If .Lista1.ListCount <> 0 Then
              .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + 1
              Beep
              TipoError = 8
              frmCorregir.Show vbModal
              Error = True
            End If
          End If
        End If

        'Corrige los errores existentes en la definición del Argumento 3
        If Error = False Then
          palabra = ListaArgumento3(Opcode, Argumento1, Argumento2)
          If Mnemonico(4) <> "" Then
            If .Lista1.ListCount <> 0 Then
              Argumento3 = BuscarArgumento(Mnemonico(4))
              If Argumento3 = "Error" Then
                .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + Len(Mnemonico(3)) + 3
                .txtEdicion.SelLength = Len(Mnemonico(4))
                Beep
                PalabraError = Mnemonico(4)
                TipoError = 6
                frmCorregir.Show vbModal
                Error = True
              End If
            Else
              .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + Len(Mnemonico(3)) + 2
              .txtEdicion.SelLength = Len(Mnemonico(4)) + 1
              Beep
              PalabraError = Mnemonico(4)
              TipoError = 10
              frmCorregir.Show vbModal
              Error = True
            End If
          Else
            If .Lista1.ListCount <> 0 Then
              .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + Len(Mnemonico(3)) + 2
              Beep
```

```
            TipoError = 8
            frmCorregir.Show vbModal
            Error = True
          End If
        End If
      End If

      'Si no se ingresa los comentario con punto y coma
      If Not Error Then
        If Mnemonico(5) <> "" Then
          .txtEdicion.SelStart = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) +
Len(Mnemonico(2)) + Len(Mnemonico(3)) + Len(Mnemonico(4)) + 2
          .txtEdicion.SelLength = Len(Mnemonico(5))
          Beep
          PalabraError = Mnemonico(5)
          TipoError = 11
          frmCorregir.Show vbModal
          Error = True
        End If
      End If
    Else
      tipo = "Opcode"
    End If
    'Si existe cambios en la posición final de la línea
    'busco el nuevo fin de línea
    If Error Then fin = FinInstruccion(Mitad)
    If tipo = "Seudoopcode" Then Error = False
    'Salir del procedimiento si se presiona salir
    If TerminarCorregir Then
      .txtEdicion.SelStart = posicion
      .txtEdicion.SetFocus
      Exit Sub
    End If
    'Salto a la siguiente linea se se presiona Ignorar línea
    If IgnorarLinea Then Error = False
  Loop
  If Not IgnorarLinea Then
    .Data1.RecordSource = "Select Distinct Tipo From Instrucciones Where (Opcode='" + Opcode +
"' )"
    .Data1.Refresh
    Do While Not .Data1.Recordset.EOF
      If .Data1.Recordset(0) <> "" Then
        tipo = .Data1.Recordset(0)
        .Data1.Recordset.MoveNext
      Else
        Call CambiarColorSeudoOpcode(inicio, fin)
      End If
    Loop
  If tipo = "Opcode" Then Call CambiarColorComentarios(inicio, fin)
  If tipo = "Seudoopcode" Then Call CambiarColorSeudoOpcode(inicio, fin)
  If tipo = "" Then
    If Etiqueta = "Existe" Or Etiqueta = "Noexiste" Then
      Call CambiarColorComentarios(inicio, fin)
    Else
      Call CambiarColorSeudoOpcode(inicio, fin)
    End If
  End If
  End If
  'Limpio las variables utilizadas
```

```
    Opcode = ""
    Etiqueta = ""
    Argumento1 = ""
    Argumento2 = ""
    Argumento3 = ""
    tipo = ""
    IgnorarLinea = False
    Mitad = fin + 2
    Puntofin = .txtEdicion.GetLineFromChar(Mitad + 1)
    'Si el puntero se halla en el fin del programa
    If Puntofin = Ultimalinea Then Finprograma = True
    If Puntofin <> Ultimalinea Then Finprograma = False
  Loop
  .txtEdicion.SelStart = posicion
  .txtEdicion.SetFocus
  .txtEdicion.SelColor = QBColor(0)
  .sbBarraDeEstado.Panels(1).Text = mensaje
  End With
End Sub

Function CorregirInstruccion(inicio) As Integer
  Dim i As Integer
  Dim valor As Integer
  Dim posicion As Integer
  Dim comparar As Integer
  Dim Caracter As Integer
  Dim Etiqueta As String
  Dim Opcode As String
  Dim Argumento1 As String
  Dim Argumento2 As String
  Dim Argumento3 As String
  Dim letra As String
  Dim palabra As String
  Dim tipo As String
  Dim Error As Boolean
  With frmEditor
  CorregirInstruccion = -1
  ' ** Corrige los errores existentes en la instrucción **
  If Mnemonico(0) <> "" Or Mnemonico(1) <> "" Or Mnemonico(2) <> "" Or Mnemonico(3) <> ""
Or Mnemonico(4) <> "" Then
    ' Corrige los errores que existen en la Etiqueta
    If Mnemonico(0) <> "" Then
      'Encuentra los Errores que se producen en la Etiqueta
      valor = ErrorEtiqueta(Mnemonico(0))
      Select Case valor
      Case -2
        Etiqueta = Mid(Mnemonico(0), 2)
      Case -1
        posicion = inicio - 1
        If opcCorregirFin Then
          .txtEdicion.SelStart = posicion
          .txtEdicion.SelLength = 1
          ProcMensajeCorregir (1)
        End If
        CorregirInstruccion = inicio
        Exit Function
      'La etiqueta está bien escrita
      Case 0
        Etiqueta = "Existe"
```

```
    Case Else
      posicion = inicio + valor - 2
      If opcCorregirFin Then
        .txtEdicion.SelStart = posicion
        .txtEdicion.SelLength = 1
        ProcMensajeCorregir (2)
      End If
      CorregirInstruccion = posicion
      Exit Function
    End Select
  Else
    Etiqueta = "No existe"
  End If
  'Corrige los errores existentes en la definición del Opcode
  posicion = inicio + Len(Mnemonico(0)) + 1
  If Mnemonico(1) <> "" Then
    letra = Mid(Mnemonico(1), 1, 1)
    Caracter = Asc(letra)
    palabra = ListaOpcode(Caracter, Etiqueta)
   'Verifico si existen elementos en la lista
    If .Lista1.ListCount = 0 Then
      If opcCorregirFin = True Then
        .txtEdicion.SelStart = posicion - 1
        .txtEdicion.SelLength = Len(Mnemonico(1))
        ProcMensajeCorregir (10)
      End If
      CorregirInstruccion = posicion
      Exit Function
    Else
      For i = 0 To .Lista1.ListCount - 1
        If Mid(Mnemonico(0), 1, 1) = Chr(36) Then
          Opcode = palabra
        Else
          comparar = StrComp(Mnemonico(1), .Lista1.List(i), 1)
          If comparar = 0 Then
            Error = False
            Opcode = .Lista1.List(i)
            i = .Lista1.ListCount
          Else
            Error = True
          End If
        End If
      Next
      'Si no existe Opcode parecido
      If Error = True Then
        If opcCorregirFin Then
          .txtEdicion.SelStart = inicio + Len(Mnemonico(0))
          .txtEdicion.SelLength = Len(Mnemonico(1))
          ProcMensajeCorregir (3)
        End If
        CorregirInstruccion = posicion
        Exit Function
      End If
    End If
  Else
    palabra = ListaOpcode(Asc(" "), Etiqueta)
    If .Lista1.ListCount <> 0 Then
      If opcCorregirFin Then
        ProcMensajeCorregir (7)
```

```
        End If
        CorregirInstruccion = posicion
        Opcode = "No"
        Exit Function
      End If
      Opcode = "No"
    End If
    'Si  no existe Opcode y tenemos operandos se especifica como error
    If Mnemonico(1) = "" And (Mnemonico(2) <> "" Or Mnemonico(3) <> "") Then
      Opcode = ""
      If opcCorregirFin Then
        .txtEdicion.SelStart = inicio + Len(Mnemonico(0))
        .txtEdicion.SelLength = Len(Mnemonico(1))
        ProcMensajeCorregir (9)
      End If
      CorregirInstruccion = posicion
      Exit Function
    End If
    'Corrige los errores existentes en la definición del Argumento 1
    palabra = ListaArgumento1(Opcode, Etiqueta)
    posicion = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) + 1
    If Mnemonico(2) <> "" Then
      'Verifica si la  instrucción ingresada es un SeudoOpcode
      .Data1.RecordSource = "Select Distinct Tipo From Instrucciones Where (Opcode='" + Opcode +
" ')"
      .Data1.Refresh
      Do While Not .Data1.Recordset.EOF
        If .Data1.Recordset(0) <> "" Then
          tipo = .Data1.Recordset(0)
          .Data1.Recordset.MoveNext
        Else
          CorregirInstruccion = -2
          Exit Function
        End If
      Loop
      If tipo = "Seudoopcode" Or tipo = "" Then
        CorregirInstruccion = -2
        Exit Function
      End If
      'Si existe elementos en la lista
      If .Lista1.ListCount <> 0 Then
        Argumento1 = BuscarArgumento(Mnemonico(2))
        If Argumento1 = "Error" Then
          If opcCorregirFin Then
            .txtEdicion.SelStart = posicion
            .txtEdicion.SelLength = Len(Mnemonico(2))
            ProcMensajeCorregir (5)
          End If
          CorregirInstruccion = posicion
          Exit Function
        End If
      Else
       .If opcCorregirFin Then
          .txtEdicion.SelStart = posicion
          .txtEdicion.SelLength = Len(Mnemonico(2))
          ProcMensajeCorregir (6)
        End If
        CorregirInstruccion = posicion
        Exit Function
```

```
      End If
    Else
      If .Lista1.ListCount <> 0 Then
        If opcCorregirFin Then
          ProcMensajeCorregir (7)
        End If
        CorregirInstruccion = posicion
        Argumento1 = ""
        Exit Function
      End If
      Argumento1 = ""
    End If

    'Corrige los errores existentes en la definición del argumento 2
    palabra = ListaArgumento2(Opcode, Argumento1)
    posicion = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) + Len(Mnemonico(2)) + 3
    If Mnemonico(3) <> "" Then
      If .Lista1.ListCount <> 0 Then
        Argumento2 = BuscarArgumento(Mnemonico(3))
        If Argumento2 = "Error" Then
          If opcCorregirFin = True Then
            .txtEdicion.SelStart = posicion
            .txtEdicion.SelLength = Len(Mnemonico(3))
            ProcMensajeCorregir (5)
          End If
          CorregirInstruccion = posicion
          Exit Function
        End If
      Else
        If opcCorregirFin = True Then
          .txtEdicion.SelStart = posicion - 1
          .txtEdicion.SelLength = Len(Mnemonico(3))
          ProcMensajeCorregir (8)
        End If
        CorregirInstruccion = posicion
        Exit Function
      End If
    Else
      If .Lista1.ListCount <> 0 Then
        If opcCorregirFin = True Then
          ProcMensajeCorregir (7)
        End If
        CorregirInstruccion = posicion
        Argumento2 = ""
        Exit Function
      End If
      Argumento2 = ""
    End If

    'Corrige los errores existentes en la definición del argumento3
    palabra = ListaArgumento3(Opcode, Argumento1, Argumento2)
    If Mnemonico(4) <> "" Then
      posicion = inicio + Len(Mnemonico(0)) + Len(Mnemonico(1)) + Len(Mnemonico(2)) +
Len(Mnemonico(3)) + 4
      If .Lista1.ListCount <> 0 Then
        Argumento3 = Mnemonico(4)
        palabra = BuscarArgumento(Mnemonico(4))
        If palabra = "Error" Then
          If opcCorregirFin = True Then
```

```
            .txtEdicion.SelStart = posicion
            .txtEdicion.SelLength = Len(Mnemonico(4))
            ProcMensajeCorregir (5)
          End If
          CorregirInstruccion = posicion
          Exit Function
        End If
      Else
        If opcCorregirFin = True Then
          .txtEdicion.SelStart = posicion
          .txtEdicion.SelLength = Len(Mnemonico(4))
          ProcMensajeCorregir (8)
        End If
        CorregirInstruccion = posicion
        Exit Function
      End If
    Else
      If .Lista1.ListCount <> 0 Then
        If opcCorregirFin = True Then
          ProcMensajeCorregir (7)
        End If
        CorregirInstruccion = posicion
        Exit Function
      End If
    End If
    If Mnemonico(5) <> "" Then
      If opcCorregirFin = True Then
        .txtEdicion.SelStart = posicion
        .txtEdicion.SelLength = Len(Mnemonico(3))
        ProcMensajeCorregir (8)
      End If
      CorregirInstruccion = posicion
      Exit Function
    End If
  Else
    CorregirInstruccion = -1
    Exit Function
  End If
    .Data1.RecordSource = "Select Distinct Tipo From Instrucciones Where (Opcode='" + Opcode + "
')"
    .Data1.Refresh
    Do While Not .Data1.Recordset.EOF
      If .Data1.Recordset(0) <> "" Then
        tipo = .Data1.Recordset(0)
        .Data1.Recordset.MoveNext
      Else
        CorregirInstruccion = -2
        Exit Function
      End If
    Loop
    If tipo = "Opcode" Then CorregirInstruccion = -1
    If tipo = "Seudoopcode" Then CorregirInstruccion = -2
    If tipo = "" Then
      If Etiqueta = "Existe" Or Etiqueta = "Noexiste" Then
        CorregirInstruccion = -1
      Else
        CorregirInstruccion = -2
      End If
    End If
```

```
  End With
End Function


'*** Módulo estándar que contiene procedimientos para mostrar la    ***
'***    ayuda del programa; forma parte del EDITOR INTELIGENTE.    ***
'*************************************************************************

Sub ProcPegarPalabra()
  Dim i As Integer
  Dim Numero As Integer
  Dim lenPalabra As Integer
  Dim posicion As Integer
  Dim palabra As String
  Dim Opcode As String
  'Procedimiento que permite pagar en el texto la palabra seleccionada
  With frmEditor
  i = .Lista1.ListIndex
  palabra = .Lista1.List(i)
  Select Case numPalabra
  Case 2
    Campo = 2
    numPalabra = 3
    'Reemplazo los caracteres escritos por el elemento seleccionado
    posicion = .txtEdicion.SelStart
    'Verifico cuantos caracteres han ingresado
    lenPalabra = NumeroCaracteres(9)
    .txtEdicion.SelStart = posicion - lenPalabra
    .txtEdicion.SelLength = lenPalabra
    'Pego el elemento seleccionado
    .txtEdicion.SelText = palabra
    .txtEdicion.SelStart = posicion + Len(palabra) - lenPalabra
    .Lista1.Visible = False
    'Llamo al procedimiento que muestre los operando existentes
    .txtEdicion.SetFocus
    ini = MostrarArgumento1(palabra, valEtiqueta)
    valOpcode = palabra
  Case 3
    Numero = TipoArgumento(palabra, 1)
    'Verifico si el Argumento1 requiere de mas parametros
    Select Case Numero
    'No requiere más paramentros
    Case 1
      numPalabra = 4
      'Reemplazo los caracteres escritos por el elemento seleccionado
      posicion = .txtEdicion.SelStart
      'Verifico cuantos caracteres han ingresado
      lenPalabra = NumeroCaracteres(9)
      .txtEdicion.SelStart = posicion - lenPalabra
      .txtEdicion.SelLength = lenPalabra
      'Pego el elemento seleccionado
      .txtEdicion.SelText = palabra
      .txtEdicion.SelStart = posicion + Len(palabra) - lenPalabra
      .Lista1.Visible = False
      Numero = MostrarArgumento2(valOpcode, palabra)
      ProcMensajeError (33)
    'Requiere que se muestre el texto para ingresar los parámetros
    Case 3
```

```
        ProcPosicionLista (2)
        If opcAyuda Then
          .Lista1.Visible = True
          .txtOperando.Top = .Lista1.Top
          .txtOperando.Visible = True
          .txtOperando.SetFocus
        End If
      End Select
      valArgumento1 = palabra
    Case 4
      Numero = TipoArgumento(palabra, 2)
      'Verifico si el Argumento1 requiere de mas parametros
      Select Case Numero
      'No requiere más paramentros
      Case 1
        'Reemplazo los caracteres escritos por el elemento seleccionado
        posicion = .txtEdicion.SelStart
        'Verifico cuantos caracteres han ingresado
        lenPalabra = NumeroCaracteres(44)
        .txtEdicion.SelStart = posicion - lenPalabra
        .txtEdicion.SelLength = lenPalabra
        'Pego el elemento seleccionado
        .txtEdicion.SelText = palabra
        .txtEdicion.SelStart = posicion + Len(palabra)
        .Lista1.Visible = False
        Numero = MostrarArgumento3(valOpcode, valArgumento1, palabra)
      'Requiere que se muestre el texto para ingresar los parámetros
      Case 3
        .txtOperando.Top = .Lista1.Top
        If opcAyuda Then
          .txtOperando.Visible = True
          .txtOperando.SetFocus
        End If
      End Select
      valArgumento2 = palabra
    End Select
    End With
End Sub

Function ListaOpcion(palabra) As Boolean
    'Lleno la lista con la base da datos
    With frmEditor
    'Envio a la base la palabra ingresada
    .Data1.RecordSource = "Select Distinct Opcode From Instrucciones Where (Etiqueta ='" + palabra +
"')"
    .Data1.Refresh
    .Lista1.Clear
    ' Lleno la lista con datos de la base de datos
    Do While Not .Data1.Recordset.EOF
      If .Data1.Recordset(0) <> "" Then
        .Lista1.AddItem .Data1.Recordset(0)
        ListaOpcion = True
        .Data1.Recordset.MoveNext
      Else
        ListaOpcion = False
        Exit Function
      End If
    Loop
    End With
```

```
End Function

Function ListaOpcode(Caracter, Etiqueta) As String
  'Lleno la lista con la base da datos
  With frmEditor
  'Relaciono mayúsculas y minúsculas
  If Caracter <> Asc(" ") Then Caracter = Asc(UCase(Chr(Caracter)))
  'Envio a la base de datos el caracter ingresado para recuperar datos relacionados
  If Etiqueta = "Existe" Then
    .Data1.RecordSource = "Select Distinct Opcode From Instrucciones Where ID2=" & Caracter
  ElseIf Etiqueta = "No existe" Then
    .Data1.RecordSource = "Select Distinct Opcode From Instrucciones Where Etiqueta ='Opcional '
and ID2=" & Caracter
  Else
    .Data1.RecordSource = "Select Distinct Opcode From Instrucciones Where (Etiqueta='" + Etiqueta
+ " ')"
  End If
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno la lista con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
    If .Data1.Recordset(0) <> "" Then
      .Lista1.AddItem .Data1.Recordset(0)
      ListaOpcode = .Data1.Recordset(0)
      .Data1.Recordset.MoveNext
    Else
      ListaOpcode = " "
      Exit Function
    End If
  Loop
  End With
End Function

Function ListaArgumento1(Opcode, Etiqueta) As String
  'Lleno la lista con la base de datos
  With frmEditor
  If Etiqueta = "Existe" Then
    .Data1.RecordSource = "Select Distinct Argumento1 From Instrucciones Where (Opcode='" +
Opcode + " ')"
  ElseIf Etiqueta = "No existe" Then
    .Data1.RecordSource = "Select Distinct Argumento1 From Instrucciones Where (Opcode ='" +
Opcode + "' and Etiqueta='Opcional' )"
  Else
    .Data1.RecordSource = "Select Distinct Argumento1 From Instrucciones Where (Opcode ='" +
Opcode + "' and Etiqueta='" + Etiqueta + "' )"
  End If
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno la lista con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
    If .Data1.Recordset(0) <> "" Then
      .Lista1.AddItem .Data1.Recordset(0)
      ListaArgumento1 = .Data1.Recordset(0)
      .Data1.Recordset.MoveNext
    Else
      ListaArgumento1 = "No existe"
      Exit Function
    End If
  Loop
```

```
  End With
End Function

Function ListaArgumento2(Opcode, Argumento1) As String
  'Lleno la lista con la base de datos
  With frmEditor
  .Data1.RecordSource = "Select Distinct argumento2 From Instrucciones Where (Opcode ='" +
Opcode + "' and Argumento1='" + Argumento1 + "')"
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno la lista con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
    If .Data1.Recordset(0) <> "" Then
      .Lista1.AddItem .Data1.Recordset(0)
      ListaArgumento2 = .Data1.Recordset(0)
      .Data1.Recordset.MoveNext
    Else
      ListaArgumento2 = "no existe"
      Exit Function
    End If
  Loop
  End With
End Function

Function ListaArgumento3(Opcode, Argumento1, Argumento2) As String
  'Lleno la lista con la base de datos
  With frmEditor
  .Data1.RecordSource = "Select Distinct Argumento3 From Instrucciones Where (Opcode ='" +
Opcode + "' and Argumento1='" + Argumento1 + "'and argumento2 = '" + Argumento2 + "')"
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno la lista con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
    If .Data1.Recordset(0) <> "" Then
      .Lista1.AddItem .Data1.Recordset(0)
      ListaArgumento3 = .Data1.Recordset(0)
      .Data1.Recordset.MoveNext
    Else
      ListaArgumento3 = "no existe"
      Exit Function
    End If
  Loop
  End With
End Function

Function MostrarOpcion(opcion) As Integer
  Dim Numero As Integer
  Dim Etiqueta As String
  'Envio a la base de datos la Opcion seleccionado para recuperar datos relacionados
  With frmEditor
  'Verifico si existen elementos en la lista
  Etiqueta = ListaOpcion(opcion)
  If Etiqueta Then
    'Requiere más argumentos
    .txtEdicion.SelText = Chr(9)
    ProcPosicionLista (1)
    .Lista1.Height = 1
    If opcAyuda Then .Lista1.Visible = True
    .txtOperando.Top = .Lista1.Top
```

```
    .txtOperando.Left = 20
     If opcAyuda Then
       .txtOperando.Visible = True
       .txtOperando.SetFocus
     End If
     ProcMensajeError (25)
   Else
     'Si no existen más argumentos
     .txtEdicion.SelText = Chr(9) + Chr(9) + Chr(9)
     ProcMensajeError (4)
     Campo = 3
   End If
   End With
End Function

Function MostrarArgumento1(Opcode, Etiqueta) As Integer
   Dim Numero As Integer
   Dim Argumento1 As String
   'Envio a la base de datos el Opcode seleccionado para recuperar datos relacionados
   With frmEditor
   'Verifico si existen elementos en la lista
   Argumento1 = ListaArgumento1(Opcode, Etiqueta)
   'Si el opcode no existe
   If Argumento1 = "" Then
     MostrarArgumento1 = -1
     Exit Function
   End If
   'Si existen argumentos el la lista
   If .Lista1.ListCount <> 0 Then
     'Si existe un unico elemento
     If .Lista1.ListCount = 1 Then
       .Lista1.Height = 1
       Numero = TipoArgumento(Argumento1, 1)
       'Verifico si el Argumento1 requiere de mas parametros
       Select Case Numero
       Case 0
       'Se eligió un SeudoOpcode
         .txtEdicion.SelText = Chr(9)
         ProcMensajeError (8)
       'No requiere más paramentros
       Case 1
         numPalabra = 4
         .txtEdicion.SelText = Chr(9) + Argumento1
         .Lista1.Visible = False
         MostrarArgumento1 = 1
         Numero = MostrarArgumento2(Opcode, Argumento1)
       'Requiere que se muestre el texto para ingresar los parámetros
       Case 3
         .txtEdicion.SelText = Chr(9)
         ProcPosicionLista (2)
         If opcAyuda Then .Lista1.Visible = True
         .txtOperando.Top = .Lista1.Top
         If opcAyuda Then
           .txtOperando.Visible = True
           .txtOperando.SetFocus
         End If
         ProcMensajeError (37)
         Exit Function
       End Select
```

```
    ElseIf .Lista1.ListCount > 1 Then
      MostrarArgumento1 = 0
      .txtEdicion.SelText = Chr(9)
      If .Lista1.ListCount > 8 Then
        .Lista1.Height = 8
      Else
        .Lista1.Height = .Lista1.ListCount
      End If
      ProcPosicionLista (2)
      If opcAyuda Then .Lista1.Visible = True
      ProcMensajeError (34)
      Exit Function
    End If
  'Si no existen más argumentos
  Else
    .txtEdicion.SelText = Chr(9) + Chr(9)
    ProcMensajeError (4)
    Campo = 3
  End If
  End With
End Function

Function MostrarArgumento2(Opcode, Argumento1) As Integer
  Dim Numero As Integer
  Dim Argumento2 As String
  'Envio a la base de datos el indice seleccionado para recuperar datos relacionados
  With frmEditor
  Argumento2 = ListaArgumento2(Opcode, Argumento1)
  'Si el opcode no existe
  If Argumento2 = "" Then MostrarArgumento2 = -1
  If .Lista1.ListCount <> 0 Then
    'Si existe un unico elemento
    If .Lista1.ListCount = 1 Then
      .Lista1.Height = 1
      Numero = TipoArgumento(Argumento2, 2)
      'Verifico si el Argumento1 requiere de mas parametros
      Select Case Numero
      'No requiere más paramentros
      Case 1
        .txtEdicion.SelText = Chr(44) + Argumento2
        .Lista1.Visible = False
        MostrarArgumento2 = 1
        Numero = MostrarArgumento3(Opcode, Argumento1, Argumento2)
      'Requiere que se muestre el texto para ingresar los parámetros
      Case 3
        .txtEdicion.SelText = Chr(44)
        ProcPosicionLista (3)
        If opcAyuda Then .Lista1.Visible = True
        .txtOperando.Top = .Lista1.Top
        If opcAyuda Then
          .txtOperando.Visible = True
          .txtOperando.SetFocus
        End If
        ProcMensajeError (37)
        Exit Function
      End Select
    ElseIf .Lista1.ListCount > 1 Then
      MostrarArgumento2 = 0
      .txtEdicion.SelText = Chr(44)
```

```
      If .Lista1.ListCount > 8 Then
        .Lista1.Height = 8
      Else
        .Lista1.Height = .Lista1.ListCount
      End If
      ProcPosicionLista (3)
      If opcAyuda Then .Lista1.Visible = True
      Exit Function
    End If
  'Si no existen más argumentos
  Else
    .txtEdicion.SelText = Chr(9)
    ProcMensajeError (4)
    Campo = 3
  End If
  End With
End Function

Function MostrarArgumento3(Opcode, Argumento1, Argumento2) As Integer
  Dim Numero As Integer
  Dim Argumento3 As String
  'Envio a la base de datos el indice seleccionado para recuperar datos relacionados
  With frmEditor
  Argumento3 = ListaArgumento3(Opcode, Argumento1, Argumento2)
  'Si el Argumento2 no existe
  If Argumento3 = "" Then
    MostrarArgumento3 = -1
    'Exit Function
  End If
  If .Lista1.ListCount <> 0 Then
    'Si existe un unico elemento
    If .Lista1.ListCount = 1 Then
      .Lista1.Height = 1
      Numero = TipoArgumento(Argumento3, 2)
      'Verifico si el Argumento1 requiere de mas parametros
      Select Case Numero
      'No requiere más paramentros
      Case 1
        .txtEdicion.SelText = Chr(44) + Argumento3
        .Lista1.Visible = False
        MostrarArgumento3 = 1
      'Requiere que se muestre el texto para ingresar los parámetros
      Case 3
        .txtEdicion.SelText = Chr(44)
        ProcPosicionLista (3)
        If opcAyuda Then .Lista1.Visible = True
        .txtOperando.Top = .Lista1.Top
        If opcAyuda Then
          .txtOperando.Visible = True
          .txtOperando.SetFocus
        End If
        ProcMensajeError (38)
        Exit Function
      End Select
    ElseIf .Lista1.ListCount > 1 Then
      MostrarArgumento3 = 0
      .txtEdicion.SelText = Chr(44)
      If .Lista1.ListCount > 8 Then
        .Lista1.Height = 8
```

```
      Else
        .Lista1.Height = .Lista1.ListCount
      End If
      ProcPosicionLista (3)
      If opcAyuda Then .Lista1.Visible = True
      Exit Function
    End If
  'Si no existen más argumentos
  Else
    .Lista1.Visible = False
    .txtEdicion.SelText = Chr(9)
    ProcMensajeError (4)
    Campo = 3
  End If
  End With
End Function



'*** Módulo estándar que contiene procedimientos y funciones      ***
'***  utilizadas en el programa; forma parte del EDITOR INTELIGENTE. ***
'*************************************************************************

Sub ProcIniciarVariables()
  'Inicializo las varibles globales
  indComentario = False
  ErrorInstruccion = False
  frmEditor.Lista1.Visible = False
  frmEditor.txtOperando.Visible = False
  ProcMensajeError (1)
  Campo = 0
  numPalabra = 1
End Sub

Sub ProcCargarValores()
  'Cargo los valores que se requieren para el programa
  '** Valores constantes **
  With frmEditor
  .Data1.RecordSource = "Select Distinct ARGUMENTO From Argumentos Where TIPO
='Constante' "
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno las variables con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
    .Lista1.AddItem .Data1.Recordset(0)
    .Data1.Recordset.MoveNext
  Loop
  ReDim Constante(.Lista1.ListCount)
  NumConstante = .Lista1.ListCount
  For i = 0 To .Lista1.ListCount - 1
    Constante(i) = .Lista1.List(i)
  Next
  '** Valores variables **
  .Data1.RecordSource = "Select  ARGUMENTO From Argumentos Where TIPO ='Variable'order by
ID "
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno las variables con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
```

```
    .Lista1.AddItem .Data1.Recordset(0)
    .Data1.Recordset.MoveNext
  Loop
  ReDim Variable(.Lista1.ListCount)
  NumVariable = .Lista1.ListCount
  For i = 0 To .Lista1.ListCount
    Variable(i) = .Lista1.List(i)
  Next
  '** Indice de las variables **
  .Data1.RecordSource = "Select INICIAL From Argumentos Where TIPO ='Variable'order by ID  "
  .Data1.Refresh
  .Lista1.Clear
  ' Lleno las variables con datos de la base de datos
  Do While Not .Data1.Recordset.EOF
    .Lista1.AddItem .Data1.Recordset(0)
    .Data1.Recordset.MoveNext
  Loop
  ReDim Indice(.Lista1.ListCount)
  For i = 0 To .Lista1.ListCount
    Indice(i) = .Lista1.List(i)
  Next
  End With
End Sub

Sub ProcMensajeError(Observacion As Integer)
  Dim MensajeError As String
  ' ** Se define que tipo de error se a producido **
  Select Case Observacion

  ' Mensajes Principales
  Case 0
    MensajeError = "Ingrese la instrucción"
  Case 1
    MensajeError = "Area para la definición de Etiqueta"
  Case 2
    MensajeError = "Area para la definición de Opcode y SeudoOpcode"
  Case 3
    MensajeError = "Area para la definición de Operandos"
  Case 4
    MensajeError = "Area para la definición de Comentarios"
  Case 5
    MensajeError = "Se procede a corregir los Errores del Programa"
  Case 8
    MensajeError = "Seleccionó un SeudoOpcode; ingrese la Etiqueta"

  'Mensajes de pedido de parámetros
  Case 10
    MensajeError = "Ingrese la Etiqueta o Tabulador"
  Case 20
    If opcAyuda Then
      MensajeError = "Ingrese el Opcode o SeudoOpcode(Presione una letra)"
    Else
      MensajeError = "Ingrese el Opcode o SeudoOpcode"
    End If
  Case 30
    MensajeError = "Ingrese los Operandos"
  Case 40
    MensajeError = "Ingrese el Comentario"
```

```
'Errores producidos en el ingreso de la Etiqueta
Case 11
  MensajeError = "Caracter no definido para inicio de Etiqueta"
Case 12
  MensajeError = "Caracter no válido como parte de la Etiqueta"
Case 13
  MensajeError = "La Etiqueta ingresada no es válida"
Case 14
  MensajeError = "A ingresado una Seudo instrucción, defina el argumento o presione ENTER"

'Errores producidos en el ingreso del Opcode y SeudoOpcode
Case 21
  MensajeError = "El caracter ingresado no corresponde a Mnemónico existente"
Case 22
  MensajeError = "La palabra ingresada no corresponde a Mnemónico existente"
Case 23
  MensajeError = "Seleccione un Mnemónico"
Case 24
  MensajeError = "Area para la definición de Opcode y SeudoOpcode"
Case 25
  MensajeError = "Ingrese el argumento requerido"

'Errores porducidos en el ingreso de los Operandos
Case 31
  MensajeError = "Los caracteres ingresado no corresponde a Operando existente"
Case 32
  MensajeError = "El operando ingresado no es permitido para este Mnemónico"
Case 33
  If opcAyuda Then
    MensajeError = "Seleccione el Segundo Argumento (Operando Origen)"
  Else
    MensajeError = "Ingrese el Segundo Argumento (Operando Origen)"
  End If
Case 34
  If opcAyuda Then
    MensajeError = "Seleccione el Primer Argumento (Operando Destino)"
  Else
    MensajeError = "Ingrese el Primer Argumento (Operando Destino)"
  End If
Case 35
  If opcAyuda Then
    MensajeError = "Seleccione el Tercer Argumento"
  Else
    MensajeError = "Ingrese el Tercer Argumento"
  End If
Case 36
  MensajeError = "Ingrese el Segundo Argumento"
Case 37
  MensajeError = "Ingrese el Primer Argumento"
Case 38
  MensajeError = "Ingrese el Tercer Argumento"

'Errores que se producen en el ingreso de los comentarios
Case 41
  MensajeError = "Un comentario se inicia con (;)"
Case 42
  MensajeError = "La instrucción requiere de más parámetros"
Case 43
  MensajeError = "La instrucción es incorrecta"
```

```
  End Select
  frmEditor.sbBarraDeEstado.Panels(1).Text = MensajeError

End Sub

Sub ProcPosicionLista(ubicacion)
  Dim inicio As Integer
  Dim fin As Integer
  'Defino la posición en la que se debe ubicar la lista
  With frmEditor
  If ubicacion = 1 Then .Lista1.Left = 12
  If ubicacion = 2 Then .Lista1.Left = 20
  If ubicacion = 3 Then .Lista1.Left = 25
  If .sbBarraDeMensajes.Visible And .tbBarHerramientas.Visible Then
    inicio = .txtEdicion.Height - .Lista1.Height - 4
  ElseIf Not .sbBarraDeMensajes.Visible And .tbBarHerramientas.Visible Then
    inicio = .txtEdicion.Height - .Lista1.Height - 2
  ElseIf .sbBarraDeMensajes.Visible And Not .tbBarHerramientas.Visible Then
    inicio = .txtEdicion.Height - .Lista1.Height - 2
  Else
    inicio = .txtEdicion.Height - .Lista1.Height
  End If
  If .sbBarraDeEstado.Visible Then fin = .txtEdicion.Height - 2
  If Not .sbBarraDeEstado.Visible Then fin = .txtEdicion.Height
  Select Case NumeroLinea
  Case 0 To inicio - 1
  'Ubico la posición de la lista de acuerdo a la visibilidad de las barras
    If .sbBarraDeMensajes.Visible And .tbBarHerramientas.Visible Then
      .Lista1.Top = NumeroLinea + 3.2 + .sbBarraDeEstado.Height
    ElseIf Not .sbBarraDeMensajes.Visible And .tbBarHerramientas.Visible Then
      .Lista1.Top = NumeroLinea + 1.75 + .sbBarraDeEstado.Height
    ElseIf .sbBarraDeMensajes.Visible And Not .tbBarHerramientas.Visible Then
      .Lista1.Top = NumeroLinea + 1.75 + .sbBarraDeEstado.Height
    Else
      .Lista1.Top = NumeroLinea + .sbBarraDeEstado.Height
    End If
  Case inicio - 1 To fin
    If .sbBarraDeEstado.Visible Then
      .Lista1.Top = NumeroLinea - .Lista1.Height + 3.5
    Else
      .Lista1.Top = NumeroLinea - .Lista1.Height
    End If
  Case Else
    .Lista1.Top = .txtEdicion.Height - .Lista1.Height + 1.5
  End Select
  End With
End Sub

Sub TextoOperandoAuxiliar()
  'Procedimiento qie permite pegar el texto ingresado en txtOperando
  Dim Numero As Integer
  Dim posicion As Integer
  With frmEditor
  'Reemplazo los caracteres escritos por el elemento seleccionado
  posicion = .txtEdicion.SelStart
  'Verifico cuantos caracteres han ingresado
  If numPalabra = 2 Then
    lenPalabra = NumeroCaracteres(9)
  ElseIf numPalabra = 3 Then
```

```
    lenPalabra = NumeroCaracteres(9)
  Else
    lenPalabra = NumeroCaracteres(44)
  End If
  .txtEdicion.SelStart = posicion - lenPalabra
  .txtEdicion.SelLength = lenPalabra
  'Pego el elemento seleccionado
  .txtEdicion.SelText = InicioArgumento + .txtOperando.Text
  .txtEdicion.SelStart = posicion + Len(InicioArgumento + .txtOperando.Text)
  .txtEdicion.SetFocus
  .Lista1.Visible = False
  .txtOperando.Visible = False
  If numPalabra = 2 Then
    numPalabra = 5
    .txtEdicion.SelText = Chr(9) + Chr(9)
    ProcMensajeError (40)
  ElseIf numPalabra = 3 Then
    numPalabra = 4
    If .Lista1.ListCount = 1 Then valArgumento1 = .Lista1.List(0)
    Numero = MostrarArgumento2(valOpcode, valArgumento1)
  ElseIf numPalabra = 4 Then
    numPalabra = 5
    If .Lista1.ListCount = 1 Then valArgumento2 = .Lista1.List(0)
    Numero = MostrarArgumento3(valOpcode, valArgumento1, valArgumento2)
  Else
    .txtEdicion.SelText = Chr(9)
    ProcMensajeError (40)
  End If
  .txtOperando.Text = ""
  End With
End Sub

Sub ProcCortarPalabra(palabra)
  'Permite eliminar la palabra ingresada
  With frmEditor
  If opcCorregir Then
    .txtEdicion.SelStart = .txtEdicion.SelStart - Len(palabra)
    .txtEdicion.SelLength = Len(palabra)
    .txtEdicion.SelText = ""
  End If
  End With
End Sub

Sub ProcSepararInstruccion(inicio As Integer, fin As Integer)
  Dim i, j, k, n As Integer
  Dim Variable As Integer
  Dim letra As String
  Dim Buscar As String
  Dim inicio1 As Integer
  Dim Instruccion As String
  ' ** Procedimiento que separa la instrucción en sus componentes **
  ' Definimos el punto de inicio y fin de la instrución
  With frmEditor
  ' Copiamos la instrucción en una variable para poder dividirla
  If fin = 0 Or fin < inicio Then Exit Sub
  n = fin - inicio + 1
  inicio1 = inicio
  If inicio = 0 Then inicio1 = inicio + 1
  Instruccion = Mid(.txtEdicion.Text, inicio1, n)
```

```
  ' Dividimos la instrucción en sus componentes
  For i = 0 To 5
    Mnemonico(i) = ""
  Next
  j = 0
  For i = 1 To n
    letra = Mid(Instruccion, i, 1)
    Variable = Asc(letra)
    Select Case Variable
    Case 59        'Caso comentarios
      i = n
    Case 9         'Caso tabulador
      j = j + 1
      k = k + 1
      If j = 3 Then j = 5
      If j > 5 Then i = n
    Case 44        'Caso separador de operando
      If k = 2 Then j = j + 1
      If k <> 2 Then Mnemonico(j) = Mnemonico(j) & letra
    Case Else      'Caso variables de instrucción
      Mnemonico(j) = Mnemonico(j) & letra
    End Select
  Next
  End With
End Sub

Sub ProcMensajeCorregir(NumeroError)
  Dim Respuesta As Integer
  Dim titulo As String
  Dim mensaje As String
  Beep
  Select Case NumeroError
  Case 1
    titulo = "Error de Sintaxis"
    mensaje = "El caracter ingresada no puede ser" + Chr(13) + Chr(10) + "inicio de etiqueta"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 2
    titulo = "Error de Sintaxis"
    mensaje = "El caracter ingresada no es" + Chr(13) + Chr(10) + "permitido como parte de la
etiqueta"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 3
    titulo = "Error de Sintaxis"
    mensaje = "La palabra ingresada no corresponde" + Chr(13) + Chr(10) + "a Mnenonico existente"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 4
    titulo = "Instrucción Incompleta"
    mensaje = "Faltan parametros en la" + Chr(13) + Chr(10) + "instrucción"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 5
    titulo = "Error de Sintaxis"
    mensaje = "La palabra ingresada no corresponde" + Chr(13) + Chr(10) + "a Operando permitido"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 6
    titulo = "Muchos Paramentros"
    mensaje = "El Opcode ingresado no requiere de " + Chr(13) + Chr(10) + "Operandos"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 7
    titulo = "Instrucción Incompleta"
```

```
    mensaje = "El Opcode ingresado requiere" + Chr(13) + Chr(10) + "el ingreso de operandos"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 8
    titulo = "Muchos Paramentros"
    mensaje = "El Opcode ingresado no requiere de demasiados parámetros " + Chr(13) + Chr(10) + "ó
los comentarios inician con (;)"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 9
    titulo = "Instrucción Incompleta"
    mensaje = "La instruccion ingresado requiere la" + Chr(13) + Chr(10) + "especificación de Opcode
o SeudoOpcode"
    Respuesta = MsgBox(mensaje, 0, titulo)
  Case 10
    titulo = "Muchos Paramentros"
    mensaje = "La Opción ingresado no requiere de demasiados parámetros "
    Respuesta = MsgBox(mensaje, 0, titulo)
  End Select
  ProcMensajeError (43)
End Sub

Sub ProcPosicionCursor()
  Dim posicion As Integer
  Dim inicio As Integer
  Dim letra As String
  Dim k As Integer
  Dim i As Integer
  With frmEditor
  posicion = .txtEdicion.SelStart
  If .txtEdicion.SelLength = 0 Then
    If .txtEdicion.SelStart <> 0 Then
      'Definimos en que posición se encuentra el punto de inserción
      inicio = posicion
      k = 0
      letra = " "
      Do While Not letra = Chr(10)
        If .txtEdicion.SelStart = 0 Then inicio = 1
        letra = Mid(.txtEdicion.Text, inicio, 1)
        If letra = Chr(9) Then k = k + 1
        If letra = ";" Then
          Campo = 3
          ProcMensajeError (4)
          .txtEdicion.SelStart = posicion
          Exit Sub
        End If
        inicio = inicio - 1
        If inicio = 0 Then Exit Do
      Loop
      Campo = k
      If Campo = 0 Then ProcMensajeError (1): numPalabra = 1
      If Campo = 1 Then ProcMensajeError (2): numPalabra = 2
      If Campo = 2 Then
        'Definimos en que posición se encuentra el punto de inserción
        inicio = posicion
        i = 0
        letra = ""
        Do While Not letra = Chr(9)
          letra = Mid(.txtEdicion.Text, inicio, 1)
          If letra = Chr(44) Then i = i + 1
          inicio = inicio - 1
```

```
        Loop
        If i = 0 Then numPalabra = 3
        If i = 1 Then numPalabra = 4
        If i = 2 Then numPalabra = 5
        ProcMensajeError (3)
      End If
      If Campo = 3 Then ProcMensajeError (4): numPalabra = 6
    Else
      ProcMensajeError (0)
      Campo = 0
    End If
  .txtEdicion.SelStart = posicion
  End If
  End With
End Sub

Sub ProcAlinear()
  'Procedimiento que hace que la instrucción se alinee
  Dim Varios As Boolean
  Dim inicio As Integer
  Dim posicion As Integer
  Dim PosicionInicio As Integer
  Dim contador As Integer
  Dim posicionUltima As Integer
  Dim espacio As Integer
  Dim tabulador As Integer
  With frmEditor
  Screen.MousePointer = 11
  Alinear = False
  .txtEdicion.Visible = False
  .txtEdicion.SelStart = 0
  inicio = 0
  PosicionInicio = inicio
  contador = 0
  'Hacemos esto mientras no exista un salto de linea
  Do While .txtEdicion.Find(Chr(10), inicio, , 8) > -1
    'Busco un espacio en blanco
    posicion = .txtEdicion.Find(Chr(32), PosicionInicio, , 8)
    'Si encuentra el especio en blanco
    If posicion <> -1 Then
      posicionUltima = posicion + 1
      Varios = True
      contador = contador + 1
      Do While Varios
          espacio = .txtEdicion.Find(Chr(32), posicionUltima, , 8)
          tabulador = .txtEdicion.Find(Chr(9), posicionUltima, , 8)
          If espacio = posicionUltima Then
            contador = contador + 1
            posicionUltima = espacio + 1
            Varios = True
          ElseIf tabulador = posicionUltima Then
            contador = contador + 1
            posicionUltima = espacio + 1
          Else
            Varios = False
          End If
          'Guardo la posición en la que se hallo el ultimo espacio en blanco
        PosicionInicio = posicion + 1
      Loop
```

```
      inicio = posicion
      If contador > 1 Then
        .txtEdicion.SelStart = posicion
        .txtEdicion.SelLength = contador
        .txtEdicion.SelText = Chr(9)
        Alinear = True
        contador = 0
      Else
        contador = 0
      End If
    Else
      inicio = posicion
    End If
  Loop
  .txtEdicion.SelStart = 0
  Screen.MousePointer = 0
  .txtEdicion.Visible = True
  .txtEdicion.Refresh
  End With
End Sub

Sub CambiarColorComentarios(inicio As Integer, fin As Integer)
  Dim centro As Integer
  Dim Variable As String
  Dim Comentario As Boolean
  Dim posicion As Integer
  With frmEditor
  If inicio = 0 Then inicio = 1
  If fin < inicio Then Exit Sub
  'Función que permite cambiar de color los comentarios
  'Me ubico en la posicion intermedia si existe comentarios
  .txtEdicion.SelStart = inicio
  posicion = .txtEdicion.SelStart
  centro = .txtEdicion.Find(";", inicio - 1, fin, 8)
  If centro <> -1 Then Comentario = True
  'Pinto de color negro la instrucción
  .txtEdicion.SelStart = inicio - 1
  .txtEdicion.SelLength = fin - inicio + 1
  .txtEdicion.SelColor = QBColor(0)
  .txtEdicion.SelStart = posicion
  'Si existe comentarios se pintara de color verde
  If Comentario = True Then
    If centro <> 0 Then centro = centro - 1
    .txtEdicion.SelStart = centro
    .txtEdicion.SelLength = fin - centro + 1
    .txtEdicion.SelColor = QBColor(2)
    .txtEdicion.SelStart = posicion
    .txtEdicion.SelColor = QBColor(0)
  End If
  End With
End Sub

Sub CambiarColorError(inicio As Integer, fin As Integer)
  Dim posicion As Integer
  With frmEditor
  .txtEdicion.SelStart = inicio
  posicion = .txtEdicion.SelStart
  'Función que permite cambiar de color de la instrucción mal escrita
  'Pinto de color rojo la instrucción que contiene errores
```

```
  If inicio = 0 Then inicio = 1
  .txtEdicion.SelStart = inicio - 1
  .txtEdicion.SelLength = fin - inicio + 1
  .txtEdicion.SelColor = QBColor(12)
  .txtEdicion.SelStart = posicion
  .txtEdicion.SelColor = QBColor(0)
  End With
End Sub

Sub CambiarColorSeudoOpcode(inicio As Integer, fin As Integer)
  Dim posicion As Integer
  Dim centro As Integer
  Dim Comentario As Boolean
  With frmEditor
  .txtEdicion.SelStart = inicio
  posicion = .txtEdicion.SelStart
  If fin <= inicio Then Exit Sub
  If inicio = 0 Then inicio = inicio + 1
  'Función que permite cambiar de color los comentarios
  'Me ubico en la posicion intermedia si existe comentarios
  centro = .txtEdicion.Find(";", inicio, fin, 8)
  If centro <> -1 Then Comentario = True
  'Función que permite cambiar de color de la instrucción
  'Pinto de color azul la instrucción que contiene SeudoOpcode
  .txtEdicion.SelStart = inicio - 1
  .txtEdicion.SelLength = fin - inicio + 1
  .txtEdicion.SelColor = QBColor(9)
  .txtEdicion.SelStart = posicion
  .txtEdicion.SelColor = QBColor(0)
  'Si existe comentarios se pintara de color verde
  If Comentario = True Then
    If centro <> 0 Then centro = centro - 1
    .txtEdicion.SelStart = centro
    .txtEdicion.SelLength = fin - centro
    .txtEdicion.SelColor = QBColor(2)
    .txtEdicion.SelStart = posicion
    .txtEdicion.SelColor = QBColor(0)
  End If
  End With
End Sub

Function NumeroCaracteres(indCaracter As Integer) As Integer
  Dim valor As String
  Dim Caracter As String
  Dim ContadorCaracter As Integer
  Dim posicion As Integer
  'Define cuantos caracteres se han ingresado despues del último Tab
  'o despues de la coma o del salto de línea
  With frmEditor
  If .txtEdicion.SelStart <> 0 Then
    ContadorCaracter = 0
    valor = Chr(indCaracter)
    posicion = .txtEdicion.SelStart
    Do While Not Caracter = valor
      Caracter = Mid(.txtEdicion.Text, posicion, 1)
      posicion = posicion - 1
      ContadorCaracter = ContadorCaracter + 1
      If posicion = 0 Then Exit Do
    Loop
```

```
    If posicion = 0 Then
      If valor = Chr(10) Then NumeroCaracteres = ContadorCaracter
      If valor <> Chr(10) Then NumeroCaracteres = ContadorCaracter - 1
    End If
    If posicion <> 0 Then NumeroCaracteres = ContadorCaracter - 1
  Else: NumeroCaracteres = 0
  End If
  End With
End Function

Function PalabraIngresada(letra As Integer) As String
  'Permite identificar los caracteres ingresados
  Dim lenPalabra As Integer
  Dim i As Integer
  Dim posicion As Integer
  With frmEditor
  posicion = .txtEdicion.SelStart
  lenPalabra = NumeroCaracteres(letra)
  PalabraIngresada = Mid(.txtEdicion.Text, posicion + 1 - lenPalabra, lenPalabra)
  End With
End Function

Function ErrorEtiqueta(palabra) As Integer
  Dim i As Integer
  Dim Caracter As Integer
  Dim opcion As String
  With frmEditor
  'Encuentra los Errores que se producen en la Etiqueta
  For i = 1 To Len(palabra)
    Caracter = Asc(Mid(palabra, i, 1))
    Select Case i
    'Corrige errores en la primera letra de la etiqueta
    Case 1
      If Not CaracterValido(Caracter, True) Then
        ErrorEtiqueta = 1
        Exit Function
      End If
    'Corrige errores en las demás letras de la etiqueta
    Case 2 To Len(palabra)
      If Not CaracterValido(Caracter, False) Then
        ErrorEtiqueta = i
        Exit Function
      End If
    End Select
  Next
  'Verifico si la etiqueta ingresada es una Opción
  Caracter = Asc(Mid(palabra, 1, 1))
  If Caracter = 36 Then
    opcion = UCase(Mid(palabra, 2))
    .Data1.RecordSource = "Select Distinct Etiqueta From Instrucciones Where ID2=" & Caracter
    .Data1.Refresh
    .Lista1.Clear
    ' Lleno las variables con datos de la base de datos
    Do While Not .Data1.Recordset.EOF
      If opcion = .Data1.Recordset(0) Then
        'La etiqueta esta bien especificada y es una opción
        ErrorEtiqueta = -2
        Exit Function
      End If
```

```
      .Data1.Recordset.MoveNext
    Loop
    'La etiqueta es una opción mal especificada
    ErrorEtiqueta = -1
    Exit Function
  End If
  ErrorEtiqueta = 0
  End With
End Function

Function TipoArgumento(Argumento1, ValorArgumento) As Integer
  Dim i As Integer
  Dim letra As String
  Dim salto As Boolean
  'Esta funcion permite determinar el tipo de operando a ingresar así como
  'permitir seleccionar entre varios elementos
  With frmEditor
  salto = False
  InicioArgumento = ""
  .txtOperando.Visible = False
  For i = 0 To NumConstante - 1
    If Argumento1 = Constante(i) Then
      TipoArgumento = 1
      salto = True
      Exit For
    End If
  Next
  If Not salto Then
  For i = 0 To NumVariable - 1
    If Argumento1 = Variable(i) Then
      TipoArgumento = 3
      If Indice(i) <> ":" Then InicioArgumento = Indice(i)
      Exit For
    End If
  Next
  End If
  'Determino donde se mostrará el cuadro de texto
  Select Case TipoArgumento
  Case 3
    If ValorArgumento = 1 Then .txtOperando.Left = 28
    If ValorArgumento = 2 Then .txtOperando.Left = 33
  End Select
  End With
End Function

Function BuscarArgumento(Argumento As String) As String
  Dim i, j, z As Integer
  Dim comparar As Integer
  Dim comparar1 As Integer
  Dim comparador As String
  Dim cadena As String
  Dim letra As Integer
  Dim inicio As Integer
  'Comparo los caracteres ingresados con los Argumentos que existen
  With frmEditor
  'Comparo la palabra ingresada con los valores constantes
  For i = 0 To .Lista1.ListCount - 1
    comparar = StrComp(Argumento, .Lista1.List(i), 1)
    If comparar = 0 Then
```

```
      BuscarArgumento = .Lista1.List(i)
      Exit Function
    End If
  Next
  'Si no existe elemento parecido al ingresado
  letra = Asc(Argumento)
  'Comparo con los valores que tienen un indice
  For i = 0 To NumVariable - 1
    If letra = Asc(Indice(i)) Then
      inicio = letra
      comparador = Variable(i)
      For j = 0 To .Lista1.ListCount - 1
        comparar = StrComp(comparador, .Lista1.List(j), 1)
        If comparar = 0 Then
            BuscarArgumento = comparador
            Exit Function
        End If
      Next
    End If
  Next
  'Comparo con los valores que sobran
  For i = 0 To NumVariable - 1
    If Indice(i) = ":" Then
      comparador = Variable(i)
      For j = 0 To .Lista1.ListCount - 1
        comparar = StrComp(comparador, .Lista1.List(j), 1)
        If comparar = 0 Then
          If Asc(Argumento) <> inicio Then
          BuscarArgumento = comparador
          Exit Function
          End If
        End If
      Next
    End If
  Next
  'Si no coincide con ninguno
  BuscarArgumento = "Error"
  End With
End Function

Function FinInstruccion(lugar As Integer) As Integer
  Dim fin As Integer
  'Funcion que permite ubicarse en el fin de la instrucción
  With frmEditor
  fin = .txtEdicion.Find(Chr(10), lugar, , 8)
  If fin = -1 Then fin = Len(.txtEdicion.Text)
  FinInstruccion = fin - 1
  End With
End Function

Function InicioInstruccion(lugar As Integer) As Integer
  Dim inicio As Integer
  Dim posicion As Integer
  Dim Variable As String
  'Función que permite ubicarse en el inicio de la instrucción
  With frmEditor
  If lugar = 0 Then
    InicioInstruccion = 0
    Exit Function
```

```
  End If
  'Me pongo en el inicio de la instrucción
  If .txtEdicion.SelStart = 0 Then inicio = 1
    For i = lugar To 1 Step -1
    Variable = Mid(.txtEdicion.Text, i, 1)
    If Variable = Chr(10) Or i = 0 Then Exit For
    Next
  InicioInstruccion = i + 1
  End With
End Function

Function CaracterValido(Caracter As Integer, PrimerCaracter As Boolean) As Boolean
  'Procedimiento que distingue si son o no letras
  If Not PrimerCaracter Then
    Select Case Caracter
    Case 48 To 57  'Caracteres números
      CaracterValido = True
      Exit Function
    Case 95        'Caracter especial permitido "_"
      CaracterValido = True
      Exit Function
    Case 58        'Caracter especial permitido ":"
      CaracterValido = True
      Exit Function
    Case 63  'Caracter especial permitido "?"
      CaracterValido = True
      Exit Function
    End Select
  End If
  Select Case Caracter
  Case 65 To 90  'Caracteres mayúsculas
    CaracterValido = True
    Exit Function
  Case 97 To 122  'Caracteres minúsculas
    CaracterValido = True
    Exit Function
  Case 36  'Caracter especial permitido "$"
    CaracterValido = True
    Exit Function
  Case Else
    'Especifico que el caracter no es valido como parte de la etiqueta
    CaracterValido = False
  End Select
End Function
```

ANEXO D

BASE DE DATOS DEL EDITOR INTELIGENTE

Base de Datos "Editor Inteligente"

Tabla Instrucciones.

ID	ID2	Etiqueta	Opcode	Argumento1	Argumento2	Argumento3	Tipo
1	77	Opcional	MOV	A	#Dato		Opcode
2	77	Opcional	MOV	A	R0		Opcode
3	77	Opcional	MOV	A	R1		Opcode
4	77	Opcional	MOV	A	R2		Opcode
5	77	Opcional	MOV	A	R3		Opcode
6	77	Opcional	MOV	A	R4		Opcode
7	77	Opcional	MOV	A	R5		Opcode
8	77	Opcional	MOV	A	R6		Opcode
9	77	Opcional	MOV	A	R7		Opcode
10	77	Opcional	MOV	A	Direccion		Opcode
11	77	Opcional	MOV	A	@R0		Opcode
12	77	Opcional	MOV	A	@R1		Opcode
13	77	Opcional	MOV	R0	#Dato		Opcode
14	77	Opcional	MOV	R1	#Dato		Opcode
15	77	Opcional	MOV	R2	#Dato		Opcode
16	77	Opcional	MOV	R3	#Dato		Opcode
17	77	Opcional	MOV	R4	#Dato		Opcode
18	77	Opcional	MOV	R5	#Dato		Opcode
19	77	Opcional	MOV	R6	#Dato		Opcode
20	77	Opcional	MOV	R7	#Dato		Opcode
39	77	Opcional	MOV	R0	A		Opcode
40	77	Opcional	MOV	R1	A		Opcode
41	77	Opcional	MOV	R2	A		Opcode
42	77	Opcional	MOV	R3	A		Opcode
43	77	Opcional	MOV	R4	A		Opcode
44	77	Opcional	MOV	R5	A		Opcode
45	77	Opcional	MOV	R6	A		Opcode
46	77	Opcional	MOV	R7	A		Opcode
47	77	Opcional	MOV	R0	Direccion		Opcode
48	77	Opcional	MOV	R1	Direccion		Opcode
49	77	Opcional	MOV	R2	Direccion		Opcode
50	77	Opcional	MOV	R3	Direccion		Opcode
51	77	Opcional	MOV	R4	Direccion		Opcode
52	77	Opcional	MOV	R5	Direccion		Opcode
53	77	Opcional	MOV	R6	Direccion		Opcode
54	77	Opcional	MOV	R7	Direccion		Opcode
55	77	Opcional	MOV	Direccion	#Dato		Opcode
56	77	Opcional	MOV	Direccion	A		Opcode
57	77	Opcional	MOV	Direccion	R0		Opcode
58	77	Opcional	MOV	Direccion	R1		Opcode
59	77	Opcional	MOV	Direccion	R2		Opcode
60	77	Opcional	MOV	Direccion	R3		Opcode
61	77	Opcional	MOV	Direccion	R4		Opcode

62	77	Opcional	MOV	Direccion	R5		Opcode
63	77	Opcional	MOV	Direccion	R5		Opcode
64	77	Opcional	MOV	Direccion	R7		Opcode
65	77	Opcional	MOV	Direccion	Direccion		Opcode
66	77	Opcional	MOV	Direccion	@R0		Opcode
67	77	Opcional	MOV	Direccion	@R1		Opcode
68	77	Opcional	MOV	@R0	#Dato		Opcode
69	77	Opcional	MOV	@R1	#Dato		Opcode
70	77	Opcional	MOV	@R0	A		Opcode
71	77	Opcional	MOV	@R1	A		Opcode
72	77	Opcional	MOV	@R0	Direccion		Opcode
73	77	Opcional	MOV	@R1	Direccion		Opcode
74	77	Opcional	MOV	DPTR	#Dato16		Opcode
75	77	Opcional	MOVC	A	@A+DPTR		Opcode
76	77	Opcional	MOVC	A	@A+PC		Opcode
77	77	Opcional	MOVX	A	@R0		Opcode
78	77	Opcional	MOVX	A	@R1		Opcode
79	77	Opcional	MOVX	A	@DPTR		Opcode
80	77	Opcional	MOVX	@R0	A		Opcode
81	77	Opcional	MOVX	@R1	A		Opcode
82	77	Opcional	MOVX	@DPTR	A		Opcode
83	88	Opcional	XCH	A	R0		Opcode
84	88	Opcional	XCH	A	R1		Opcode
85	88	Opcional	XCH	A	R2		Opcode
86	88	Opcional	XCH	A	R3		Opcode
87	88	Opcional	XCH	A	R4		Opcode
88	88	Opcional	XCH	A	R5		Opcode
89	88	Opcional	XCH	A	R6		Opcode
90	88	Opcional	XCH	A	R7		Opcode
91	88	Opcional	XCH	A	Direccion		Opcode
92	88	Opcional	XCH	A	@R0		Opcode
95	88	Opcional	XCH	A	@R1		Opcode
96	88	Opcional	XCHD	A	@R0		Opcode
97	88	Opcional	XCHD	A	@R1		Opcode
98	65	Opcional	AJMP	Direccion11			Opcode
99	76	Opcional	LJMP	Direccion16			Opcode
100	83	Opcional	SJMP	Etiqueta			Opcode
101	74	Opcional	JMP	@A+DPTR			Opcode
102	74	Opcional	JZ	Etiqueta			Opcode
103	74	Opcional	JNZ	Etiqueta			Opcode
104	74	Opcional	JC	Etiqueta			Opcode
105	74	Opcional	JNC	Etiqueta			Opcode
106	67	Opcional	CJNE	A	#Dato	Etiqueta	Opcode
107	67	Opcional	CJNE	A	Direccion	Etiqueta	Opcode
109	67	Opcional	CJNE	@R0	#Dato	Etiqueta	Opcode
110	67	Opcional	CJNE	@R1	#Dato	Etiqueta	Opcode
111	68	Opcional	DJNZ	R0	Etiqueta		Opcode
112	68	Opcional	DJNZ	R1	Etiqueta		Opcode

113	68	Opcional	DJNZ	R2	Etiqueta	Opcode
114	68	Opcional	DJNZ	R3	Etiqueta	Opcode
115	68	Opcional	DJNZ	R4	Etiqueta	Opcode
116	68	Opcional	DJNZ	R5	Etiqueta	Opcode
117	68	Opcional	DJNZ	R6	Etiqueta	Opcode
118	68	Opcional	DJNZ	R7	Etiqueta	Opcode
119	68	Opcional	DJNZ	Direccion	Etiqueta	Opcode
120	65	Opcional	ADD	A	#Dato	Opcode
121	65	Opcional	ADD	A	R0	Opcode
122	65	Opcional	ADD	A	R1	Opcode
123	65	Opcional	ADD	A	R2	Opcode
124	65	Opcional	ADD	A	R3	Opcode
125	65	Opcional	ADD	A	R4	Opcode
126	65	Opcional	ADD	A	R5	Opcode
127	65	Opcional	ADD	A	R6	Opcode
128	65	Opcional	ADD	A	R7	Opcode
129	65	Opcional	ADD	A	Direccion	Opcode
130	65	Opcional	ADD	A	@R0	Opcode
131	65	Opcional	ADD	A	@R1	Opcode
132	65	Opcional	ADDC	A	#Dato	Opcode
133	65	Opcional	ADDC	A	R0	Opcode
134	65	Opcional	ADDC	A	R1	Opcode
135	65	Opcional	ADDC	A	R2	Opcode
136	65	Opcional	ADDC	A	R3	Opcode
137	65	Opcional	ADDC	A	R4	Opcode
138	65	Opcional	ADDC	A	R5	Opcode
139	65	Opcional	ADDC	A	R6	Opcode
140	65	Opcional	ADDC	A	R7	Opcode
141	65	Opcional	ADDC	A	Direccion	Opcode
142	65	Opcional	ADDC	A	@R0	Opcode
143	65	Opcional	ADDC	A	@R1	Opcode
144	83	Opcional	SUBB	A	#Dato	Opcode
145	83	Opcional	SUBB	A	R0	Opcode
146	83	Opcional	SUBB	A	R1	Opcode
147	83	Opcional	SUBB	A	R2	Opcode
148	83	Opcional	SUBB	A	R3	Opcode
149	83	Opcional	SUBB	A	R4	Opcode
150	83	Opcional	SUBB	A	R5	Opcode
151	83	Opcional	SUBB	A	R6	Opcode
152	83	Opcional	SUBB	A	R7	Opcode
153	83	Opcional	SUBB	A	Direccion	Opcode
154	83	Opcional	SUBB	A	@R0	Opcode
155	83	Opcional	SUBB	A	@R1	Opcode
156	73	Opcional	INC	A		Opcode
157	73	Opcional	INC	R0		Opcode
158	73	Opcional	INC	R1		Opcode
159	73	Opcional	INC	R2		Opcode
160	73	Opcional	INC	R3		Opcode

161	73	Opcional	INC	R4		Opcode
162	73	Opcional	INC	R5		Opcode
163	73	Opcional	INC	R6		Opcode
164	73	Opcional	INC	R7		Opcode
165	73	Opcional	INC	Direccion		Opcode
166	73	Opcional	INC	@R0		Opcode
167	73	Opcional	INC	@R1		Opcode
168	68	Opcional	DEC	A		Opcode
169	68	Opcional	DEC	R0		Opcode
170	68	Opcional	DEC	R1		Opcode
171	68	Opcional	DEC	R2		Opcode
172	68	Opcional	DEC	R3		Opcode
173	68	Opcional	DEC	R4		Opcode
174	68	Opcional	DEC	R5		Opcode
175	68	Opcional	DEC	R6		Opcode
176	68	Opcional	DEC	R7		Opcode
177	68	Opcional	DEC	Direccion		Opcode
178	68	Opcional	DEC	@R0		Opcode
179	68	Opcional	DEC	@R1		Opcode
180	73	Opcional	INC	DPTR		Opcode
181	77	Opcional	MUL	AB		Opcode
182	68	Opcional	DIV	AB		Opcode
183	68	Opcional	DA	A		Opcode
184	65	Opcional	ANL	A	#Dato	Opcode
185	65	Opcional	ANL	A	R0	Opcode
186	65	Opcional	ANL	A	R1	Opcode
187	65	Opcional	ANL	A	R2	Opcode
188	65	Opcional	ANL	A	R3	Opcode
189	65	Opcional	ANL	A	R4	Opcode
190	65	Opcional	ANL	A	R5	Opcode
191	65	Opcional	ANL	A	R6	Opcode
192	65	Opcional	ANL	A	R7	Opcode
193	65	Opcional	ANL	A	Direccion	Opcode
194	65	Opcional	ANL	A	@R0	Opcode
195	65	Opcional	ANL	A	@R1	Opcode
196	65	Opcional	ANL	Direccion	#Dato	Opcode
197	65	Opcional	ANL	Direccion	A	Opcode
198	79	Opcional	ORL	A	#Dato	Opcode
199	79	Opcional	ORL	A	R0	Opcode
200	79	Opcional	ORL	A	R1	Opcode
201	79	Opcional	ORL	A	R2	Opcode
202	79	Opcional	ORL	A	R3	Opcode
203	79	Opcional	ORL	A	R4	Opcode
204	79	Opcional	ORL	A	R5	Opcode
205	79	Opcional	ORL	A	R6	Opcode
206	79	Opcional	ORL	A	R7	Opcode
207	79	Opcional	ORL	A	Direccion	Opcode
208	79	Opcional	ORL	A	@R0	Opcode

209	79	Opcional	ORL	A	@R1	Opcode
210	79	Opcional	ORL	Direccion	#Dato	Opcode
211	79	Opcional	ORL	Direccion	A	Opcode
212	88	Opcional	XRL	A	#Dato	Opcode
213	88	Opcional	XRL	A	R0	Opcode
214	88	Opcional	XRL	A	R1	Opcode
215	88	Opcional	XRL	A	R2	Opcode
216	88	Opcional	XRL	A	R3	Opcode
217	88	Opcional	XRL	A	R4	Opcode
218	88	Opcional	XRL	A	R5	Opcode
219	88	Opcional	XRL	A	R6	Opcode
220	88	Opcional	XRL	A	R7	Opcode
221	88	Opcional	XRL	A	Direccion	Opcode
222	88	Opcional	XRL	A	@R0	Opcode
223	88	Opcional	XRL	A	@R1	Opcode
224	88	Opcional	XRL	Direccion	#Dato	Opcode
225	88	Opcional	XRL	Direccion	A	Opcode
226	67	Opcional	CLR	A		Opcode
227	67	Opcional	CPL	A		Opcode
228	82	Opcional	RL	A		Opcode
229	82	Opcional	RLC	A		Opcode
230	82	Opcional	RR	A		Opcode
231	82	Opcional	RRC	A		Opcode
232	83	Opcional	SWAP	A		Opcode
233	65	Opcional	ACALL	Direccion11		Opcode
234	76	Opcional	LCALL	Direccion16		Opcode
235	82	Opcional	RET			Opcode
236	82	Opcional	RETI			Opcode
237	80	Opcional	PUSH	Direccion		Opcode
238	80	Opcional	POP	Direccion		Opcode
239	78	Opcional	NOP			Opcode
240	67	Opcional	CLR	C		Opcode
241	67	Opcional	CLR	bit		Opcode
242	83	Opcional	SETB	C		Opcode
243	83	Opcional	SETB	bit		Opcode
244	67	Opcional	CPL	C		Opcode
245	67	Opcional	CPL	bit		Opcode
246	65	Opcional	ANL	C	bit	Opcode
247	65	Opcional	ANL	C	/bit	Opcode
248	79	Opcional	ORL	C	bit	Opcode
249	79	Opcional	ORL	C	/bit	Opcode
250	77	Opcional	MOV	C	bit	Opcode
251	77	Opcional	MOV	bit	C	Opcode
252	74	Opcional	JB	bit	Etiqueta	Opcode
253	74	Opcional	JNB	bit	Etiqueta	Opcode
254	74	Opcional	JBC	bit	Etiqueta	Opcode
255	69	Etiqueta	EQU	Direccion		Seudoopcode
256	69	Opcional	END			Seudoopcode

257	84	Etiqueta	TEQ	Direccion			Seudoopcode
258	68	Opcional	DS	Argumento			Seudoopcode
259	68	Opcional	DB	Argumento			Seudoopcode
260	68	Opcional	DW	Argumento			Seudoopcode
261	80	Etiqueta	PROC				Seudoopcode
262	69	Opcional	ENDPROC				Seudoopcode
263	73	Opcional	INCLUDE				Seudoopcode
264	83	Opcional	SEG	Code			Seudoopcode
265	83	Opcional	SEG	Data			Seudoopcode
266	83	Opcional	SEG	Xdata			Seudoopcode
267	83	Opcional	SEG	Bit			Seudoopcode
268	83	Opcional	SEG	Argumento			Seudoopcode
269	68	Opcional	DEFSEG	Argumento			Seudoopcode
270	79	Opcional	ORG	Argumento			Seudoopcode
271	80	Opcional	PUBLIC	Argumento			Seudoopcode
272	69	Opcional	EXTERN	Argumento			Seudoopcode
273	36	PR	Nombre				Seudoopcode
274	36	PG					Seudoopcode
275	36	PW	Numero				Seudoopcode
276	36	PL	Numero				Seudoopcode
277	36	LI					Seudoopcode
278	36	EJ	Numero				Seudoopcode
279	36	SM					Seudoopcode
280	36	SC					Seudoopcode
281	36	SI					Seudoopcode
282	36	IN	Numero				Seudoopcode
283	36	ER					Seudoopcode
284	36	OJ	Nombre				Seudoopcode
285	36	VS	Carácter				Seudoopcode
286	36	TL	Carácter				Seudoopcode
287	36	SB	Carácter				Seudoopcode
288	36	MN	Carácter				Seudoopcode
289	36	CR	Carácter				Seudoopcode
290	36	DF	Carácter				Seudoopcode
291	36	QU					Seudoopcode
292	36	VB					Seudoopcode
293	36	DF	Carácter				Seudoopcode
294	36	ME					Seudoopcode
295	36	PC					Seudoopcode
296	36	MF	Nombre				Seudoopcode
297	36	IL	Numero				Seudoopcode
298	36	DT					Seudoopcode
299	36	XR					Seudoopcode
300	36	AP					Seudoopcode
301	67	Opcional	CJNE	R0	#Dato	Etiqueta	Opcode
302	67	Opcional	CJNE	R1	#Dato	Etiqueta	Opcode
303	67	Opcional	CJNE	R2	#Dato	Etiqueta	Opcode
304	67	Opcional	CJNE	R3	#Dato	Etiqueta	Opcode

305	67	Opcional	CJNE	R4	#Dato	Etiqueta	Opcode
306	67	Opcional	CJNE	R5	#Dato	Etiqueta	Opcode
307	67	Opcional	CJNE	R6	#Dato	Etiqueta	Opcode
308	67	Opcional	CJNE	R7	#Dato	Etiqueta	Opcode

Tabla Argumentos

ID	TIPO	ARGUMENTO	INICIAL
0	Constante	AB	
1	Constante	A	
2	Constante	@R0	
3	Constante	@R1	
4	Constante	R0	
5	Constante	R1	
6	Constante	R2	
7	Constante	R3	
8	Constante	R4	
9	Constante	R5	
10	Constante	R6	
11	Constante	R7	
12	Constante	DPTR	
13	Constante	C	
14	Constante	@A+DPTR	
15	Constante	@A+PC	
16	Constante	@DPTR	
17	Constante	Code	
18	Constante	Data	
19	Constante	Xdata	
20	Constante	Bit	
21	Variable	#Dato	#
22	Variable	#Dato16	#
23	Variable	Direccion	:
24	Variable	Direccion11	:
25	Variable	Direccion16	:
26	Variable	Etiqueta	:
27	Variable	bit	:
28	Variable	/bit	/
29	Variable	Argumento	:

BIBLIOGRAFIA

ANGULO, José. "Microprocesadores y Microcontroladores 8085, MCS–51 y ST–6" Editorial Paraninfo. Madrid. 1993

VELARDE, Jaime. "Folleto de sistemas microprocesados". Escuela Politécnica Nacional. Quito. 1992.

HAMILTON, Mesías. "Programa didáctico para la visualización gráfica de la ejecución de las instrucciones del microcontrolador MCS/52". Tesis. Escuela Politécnica Nacional.

INTEL, "MCS BASIC – 52 user´s Manual". Intel. California 1988

GONZALES, José. "Introducción a los microcontroladores de 16 bits". Editorial McGRAW – HILL. Madrid. 1994

ANGULO, José. "Microcontroladores PIC". Editorial McGRAW HILL. Madrid. 1997

MICROSOFT, Visual Basic. "Manual del programador" Versión 5.0. Corporación Microsoft. 1996

PERRY, Greg. "Aprendiendo Visual Basic 6". Editorial PRENTICE HALL. México. 1999