ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA ELÉCTRICA

EDITOR INTELIGENTE DE PROGRAMAS PARA LOS MICROCONTROLADORES PICs.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES

ROMMEL GEOVANNY MAIGUA VARGAS

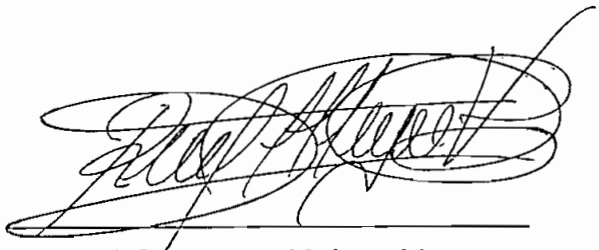
DIRECTOR: ING. JAIME VELARDE

Quito, Noviembre 2001

DECLARACIÓN

Yo, Rommel Geovanny Maigua Vargas, declaro bajo juramento que el trabajo aquí descrito es de mí autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.



Rommel Geovanny Maigua Vargas

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Rommel Geovanny Maigua Vargas, bajo mi supervisión.

Ing. Jaime Velarde

DIRECTOR DE PROYECTO

AGRADECIMIENTO

A Dios por darnos la fuerza espiritual para mirar siempre adelante.

A mis padres por todo el esfuerzo y lucha diaria en busca de mejores días para nosotros sus hijos.

A la Escuela Politécnica Nacional por acogernos en sus aulas, a sus profesores por brindarnos sus conocimientos, su pasión por las Ciencias y la Tecnología, su disciplina así como por compartir su pensamiento crítico y reflexivo, formación con la que podremos aportar con nuestro trabajo diario en pos de mejores días para nuestro Ecuador.

Mi más sincero agradecimiento al Ing. Jaime Velarde por su paciencia, ayuda permanente y valiosa dirección para la culminación de este proyecto de titulación.

A mis compañeros de Siemens en especial a Alexis, Marcelo y Jaime por su apoyo en estos meses de trabajo, a mis amigos: Jong Chiu, Carlos, Jair, Rafa, Flavio, Hugo, Antonio, Edison, Edith, Paulina, Janeth, María del Carmen, Eliana por su apoyo y su ejemplo de lucha diaria en este caminar por la vida.

A Alexandra por todo el apoyo brindado estos meses mi eterna gratitud.

Rommel Geovanny

DEDICATORIA

A mis padres María y Luis por su infatigable trabajo, por su preocupación permanente, por su ejemplo de servicio a la comunidad pero sobre todo por la formación que me brindaron a lo largo de todos estos años.

A Aurorita mi abuelita por su cariño, sus consejos y ejemplo de trabajo diario en contacto con la madre tierra.

A mis hermanas María Alexandra y Mayra por su apoyo diario y por compartir juntos estos años de estudio y trabajo en busca de sueños planteados años atrás.

A mi tierra natal Machachi, en su mes de Independencia.

Rommel Geovanny

CONTENIDO

INTRODUCCIÓN

Con el desarrollo de la tecnología, y el nacimiento del siglo XXI, las aplicaciones de los microcontroladores son ilimitadas. Dentro del campo de la Ingeniería es de vital importancia la aplicación de los microcontroladores y específicamente en la Ingeniería Electrónica es trascendente el conocimiento profundo de los mismos; es por ello que en las Universidades y Escuelas Politécnicas con carreras orientadas a la Ingeniería Eléctrónica, el estudio de los Microcontroladores es un requisito indispensable.

Existen numerosas herramientas de Software para la puesta a punto de un programa para microcontroladores, el presente trabajo pretende ser una herramienta en la edición de programas fuente para microcontroladores PIC.

El software desarrollado en el presente proyecto de titulación **"Editor Inteligente de programas para los microcontroladores PICs"**, sirve para editar programas fuente en lenguaje ensamblador de una manera didáctica, haciendo la programación fácil, guiando paso a paso al programador mientras edita el programa que se ejecutará en los microcontroladores PIC. Este software comprende un editor de texto llamado "Smart Pic Editor" y un módulo de Administración de la base de datos denominado "Admin Pic Editor".

A continuación se describe brevemente cada uno de los capítulos contenidos en el presente volumen.

En el *capitulo I* se presenta el marco teórico para la elaboración de programas para los microcontroladores PIC, enfocando el estudio del set de instrucciones del PIC 16C84 por ser uno de los más sencillos y prácticos a la hora de realizar proyectos.

En el *capitulo II* se plantean las características y herramientas de edición y corrección con que cuenta el "Smart Pic Editor" desarrollado, así como las características del módulo de administración de la base de datos "Admin Pic Editor", también se hace un análisis del modelo adoptado para la base de datos.

En el *capitulo III* se desarrolla el diseño de la interfaz del editor y del módulo de Administración de la base de datos, indicando detalladamente cada uno de los componentes, así como se presentan diagramas de flujo de la interacción del software consigo mismo y con los usuarios que lo van a utilizar.

En el *capitulo IV* se abarcan un conjunto de pruebas tanto para el editor como para el módulo de Administración de la base de datos. Para el "Admin Pic Editor - Modulo de Administración" se plantea el almacenamiento de un nuevo conjunto de instrucciones de un modelo de microcontrolador, modificaciones que se pueden realizar en la base de datos. Para el "Smart Pic Editor" se realizaron un conjunto de pruebas de edición de nuevos programas, también pruebas de corrección de programas con errores de sintáxis, así como pruebas de velocidad de respuesta en la apertura de programas, como resultado de estas pruebas se realizó la puesta a punto de la aplicación hasta obtener un producto final de calidad.

En el *capitulo V* se indican las conclusiones y recomendaciones.

En la parte final se presentan las referencias bibliográficas y se anexan el conjunto de instrucciones de los microcontroladores PIC16C84 y PIC12C5XX, el manual de Usuario del Smart Pic Editor, el manual del Administrador del módulo Admin Pic Editor, la referencia del listado del programa y las tablas de una base de datos del editor inteligente.

CAPITULO 1. ELABORACION DE PROGRAMAS PARA LOS MICROCONTROLADORES

1.1 PROCESOS EN EL DISEÑO UTILIZANDO MICROCONTROLADORES

En la actualidad casi todos los aparatos eléctricos que se encuentran a nuestro alrededor y que requieren cumplir una tarea específica que implique algún grado de inteligencia sin duda alguna poseen un microcontrolador incorporado. Aparatos como televisores, equipos de sonido, controles remotos, teléfonos, lavadoras, hornos, en la actualidad cuentan con microcontroladores, los periféricos como: el ratón, el teclado y la impresora de los computadores, también utilizan microcontroladores.

[1]"Un microcontrolador es un computador completo, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se destina a gobernar una sola tarea."

El microcontrolador se diferencia de los microprocesadores fundamentalmente en que los microprocesadores permiten el acceso a las líneas de sus buses de direcciones, datos y control que le permiten conectar a memoria externa y módulos de entrada/salida por lo que se los considera como sistemas abiertos.

[2]"Un microcontrolador es un sistema cerrado que contiene un computador completo y de prestaciones limitadas que no se puede modificar."

Para la elaboración de un proyecto que utiliza microcontroladores se presentan fases típicas, a continuación se detallan estas fases.

---

[1] ANGULO &ANGULO, Microcontroladores PIC Diseño práctico de aplicaciones, McGraW -Hill, página 1.

[2] ANGULO &ANGULO, Microcontroladores PIC Diseño práctico de aplicaciones, McGraW -Hill, página 3.

1.1.1 EDICION DEL PROGRAMA FUENTE.

Consiste en la escritura del programa en lenguaje de alto o bajo nivel por parte del programador. Para ello se utiliza un editor de texto en el que se escribirá el programa usando las instrucciones de acuerdo al lenguaje que se utilice. Este programa así escrito se conoce como *programa fuente*.

1.1.2 ENSAMBLAR O COMPILAR.

Consiste en la utilización de un ensamblador o compilador para traducir el programa fuente a código de máquina para cargarlo en la memoria de programa del microcontrolador. A este programa en código de máquina se le conoce como *programa objeto*.

1.1.3 SIMULACION POR SOFTWARE.

Pretende la simulación del programa, es decir, la ejecución de instrucciones representando en la pantalla del PC el comportamiento interno del microcontrolador así como los estados de las líneas de entrada / salida, no se trabaja en tiempo real pues corresponde a una simulación por software. Su empleo no requiere de hardware y es muy útil para realizar la depuración inicial, para la correspondiente corrección de errores lógicos del programa. No es aconsejable en aplicaciones en donde el tiempo sea un parámetro determinante, así como en aplicaciones que tengan mucha dependencia con el mundo exterior.

1.1.4 GRABACION DEL PROGRAMA EN EL MICROCONTROLADOR.

Una vez depurado el programa, se procede a grabar el mismo en la memoria de instrucciones del microcontrolador; para ello se requiere de un grabador que consiste de una tarjeta electrónica que soporta varios zócalos con diferentes números de pines en los que se introducen diversos tipos de microcontroladores, la grabación es controlada mediante un programa de comunicaciones desde un PC al que se adapta el grabador mediante uno de sus puertos.

Una vez grabado el programa en el microcontrolador se puede realizar pruebas en tiempo real para ello se puede conectar al microcontrolador grabado, los periféricos fundamentales y analizar su comportamiento, y con ello hacer una depuración media o corrección de errores de funcionamiento tanto del programa como del hardware.

1.1.5 MONTAJE DEL PROTOTIPO.

Consiste en la construcción completa del prototipo con el microcontrolador grabado y todos sus periféricos; se procede a la depuración final que consiste en la última corrección que deba realizarse a nivel de prototipo para que finalmente se realice el montaje definitivo.

1.2 LA PROGRAMACION Y SUS HERRAMIENTAS.

Para la elaboración de un programa para un microcontrolador generalmente se utilizan los lenguajes de bajo nivel, pues esto representa ahorro de código en la elaboración de programas, y esto optimiza la utilización de la memoria de programa del microcontrolador que generalmente son de capacidades limitadas, así como la ejecución es rápida.

Los lenguajes de alto nivel utilizados para la programación de microcontroladores son el C y el BASIC, en los últimos años algunas empresas han lanzado al mercado compiladores e intérpretes para diversas familias de microcontroladores.

El lenguaje más común utilizado por los profesionales para la programación es el **Ensamblador** que es el más cercano a la máquina y que permite obtener el mínimo código optimizando la memoria de programa y consiguiendo menor tiempo de ejecución.

Para el desarrollo de aplicaciones con microcontroladores se cuenta con herramientas tanto en software como en hardware, siendo las más difundidas las desarrolladas por la casa Microchip Technology. A continuación indicaremos algunas de estas herramientas.

1.2.1 HERRAMIENTAS DE SOFTWARE.

Además de los compiladores e intérpretes se utilizan simuladores software, que son programas que simulan la ejecución de instrucciones representando el comportamiento interno del microprocesador así como los estados de las líneas de entrada / salida; el único inconveniente es que no se trabaja en tiempo real.

Una de las herramientas más conocidas y difundidas es el *MPLAB-IDE (Integrated Development Environment)* que permite escribir, depurar y optimizar aplicaciones con microcontroladores PIC, pues contiene un editor de texto, simulador y administrador de proyectos. También soporta los emuladores *MPLAB -ICE y PICMASTER*, los programadores *PICSTART Plus* y *PROMATEII* y otras herramientas de desarrollo.

El MPLAB IDE integra varias herramientas como:

- MPLAB Project Manajer.- Que permite crear un proyecto y trabajar con archivos especificos relacionados al proyecto.

- MPLAB Editor.- usado para crear y editar archivos texto que corresponderán al archivos fuente.

- Simulador MPLAB -SIM.- Que permite la simulación de la ejecución de instrucciones y los estados de las E/S de los diferentes modelos de microcontroladores PIC (MCUs).

- MPLAB-ICE Emulator.- Es usado con emuladores hardware para realizar pruebas en tiempo real.

- MPASM.- corresponde a un ensamblador universal para las diversas familias de microcontroladores PIC.

En la figura 1.1 se puede observar un cuadro que indica algunas de las herramientas de software utilizadas en el desarrollo de aplicaciones con microcontroladores PIC.

HERRAMIENTA	NOMBRE	FABRICANTE o REFERENCIA
Editor	EDIT	Del Sistema Operativo DOS
Editor	C++	Lenguaje C de programación.
Editor	Basic	Lenguaje Basic de programación.
Editor	WordPad	Microsoft
Editor	MPLAB Editor	Michochip
Ensamblador	MPASM	Parallax
Ensamblador	MPASM	Microchip
Compilador de Lenguaje C	PCM	CCS
Compilador	PBASIC	MicroLab Engineering
Simulador	MPLAB SIM	Microchip
Simulador	SIMUPIC'84	Microsystems Engineering

Fig. 1.1 Herramientas de software para la programación.

1.2.2 HERRAMIENTAS DE HARDWARE.

Grabador.- Es el encargado de escribir el programa en la memoria del microcontrolador, existen desde grabadores muy completos y que responden a varios modelos de diferentes familias y muy costosos hasta grabadores específicos y de bajo costo. Microchip ha desarrollado los programadores o grabadores *PRO MATE®II. y PICSTART Plus* para este fin.

El grabador *"Micro' PIC Programmer"* diseñado por Microsystem Engineering es capaz de gravar todos los PIC de la gama media de 18,28 y 40 terminales.

Emulador.- Corresponde a un dispositivo físico que controlado por un programa desde un computador personal y disponiendo de una cabeza con terminales igual al del microcontrolador que es colocado en el sitio donde irá el microcontrolador a emular y permite mediante la pantalla del computador hacer un seguimiento del programa y la interacción con los periféricos en tiempo real. Existen emuladores

como el *PICMASTER* y el *PICMASTER-CE*, siendo uno de los más recientes el *MPLAB-ICE.*

Sistemas de desarrollo.- Corresponden a equipos que con la combinación de hardware y software permiten realizar la mayoría de las fases que implica una aplicación con microcontroladores uno de estos sistemas es el *MICRO'PIC TRAINER* de Microsystems Engineering que incorpora un grabador de PIC, hardware y software de adaptación a PC, programa de comunicaciones, tarjeta con periféricos típicos como interruptores, leds, display de siete segmentos, poteciómetros para señales analógicas y pantalla LCD.

1.3 IMPLICACIONES EN LA EDICION DE LOS MODULOS FUENTE

Para la edición del programa fuente para microcontroladores se usa cualquier editor que utilice código ASCII.

1.3.1 ESTRUCTURA DE UN PROGRAMA PARA MICROCONTROLADORES PIC

Al igual que la mayoría de programas que se elaboran para correr en un microcontrolador, un programa fuente, fundamentalmente constituye un conjunto de instrucciones y pseudoinstrucciones.

INSTRUCCIONES.- Para el lenguaje ensamblador y en formato texto constituye un conjunto de campos claramente definidos opcionales u obligatorios que al ser traducidos a código de máquina tienen una longitud definida en bits y cuya función es la de realizar una tarea definida.

PSEUDOINSTRUCCIONES.- Constituye un conjunto de campos claramente definidos que permiten al ensamblar al momento de traducir a código de máquina interactuar con el módulo fuente que contiene el programa.

1.3.2 ANALISIS DE LA SINTAXIS DE UN PROGRAMA PARA MICROCONTROLADORES PIC

Una línea de programa fuente puede estar constituida de la siguiente estructura:

[Instrucción] [Comentario]

[PseudoInstrucción] [Comentario]

Donde cada uno de los campos indicados entre corchetes pueden o no estar presentes en la línea.

Una instrucción a su vez puede considerar los siguientes campos:

[Etiqueta] [Opcode [Operandos]]

Etiqueta: Constituye un conjunto de caracteres letras, números, subraya que cumplan que el primer caracter no sea un número.

- **Opcode:** Corresponde a un conjunto de caracteres propios para cada tipo o familia de microcontroladores (mnemónicos) que indican el tipo de instrucción.

- **Operandos:** Conjunto de caracteres que con el opcode definen la operación de la instrucción, cada operando va separado por comas, y para una instrucción es claramente definido el número de operandos que acompañan al opcode así como su posición.

- **Comentario:** Corresponde a un conjunto de caracteres que el programador puede escribir como contenido aclaratorio del programa que se encuentra realizando. Este viene precedido de un indicador o signo que advierte al programador pero sobre todo al ensamblador que lo que está escrito a continuación es un comentario. Para el PIC este signo es el punto y coma ";".

Los campos son separados por una combinación de espacios en blanco o tabuladores, es aconsejable el uso de tabuladores pues a más de eliminar espacios en el archivo fuente, permite que los campos que forman una línea de programa se presenten en columnas, lo que hace que el programa sea más fácil de analizar y editar por el programador.

Una pseudoInstrucción (directiva de ensamblaje) presenta una estructura similar al de la instrucción variando el contenido del opcode y operandos a mnemónicos propios de la pseudoInstrucción.

1.3.3 INFORMACION DESCRIPTIVA DE LAS INSTRUCCIONES

Es importante destacar que la empresa fabricante de los PIC, la Microchip Technology proporciona información detallada de cada uno de los modelos fabricados donde también se puede encontrar el conjunto de instrucciones.

Dentro de la información del Conjunto de Instrucciones se puede encontrar la siguiente información:

Tipo de Instrucción: Muchas veces se suele agrupar o clasificar a las instrucciones en grupos de acuerdo a ciertos criterios; por ejemplo para los PIC 16CXX se agrupa a las instrucciones en 3 grupos así:

- Instrucciones de operación de registros orientados al byte.
- Instrucciones de operación de registros orientados al bite.
- Instrucciones de operación literal y de control.

Sintáxis: Corresponde a la forma de escritura de la instrucción

Descripción: Corresponde a información escrita de la operación que realiza la instrucción.

Operación Simbólica: Describe la operación de la instrucción usando una notación reducida de símbolos.

Ejemplo: Corresponde a un ejemplo de uso de la instrucción.

Código de Máquina: Corresponde al código de la instrucción detallando el contenido de cada bit de la palabra.

Límites de Operandos: Información del rango de datos que pueden tomar los operandos.

Banderas afectadas: Las banderas o señalizadores afectados al ejecutarse la instrucción.

Palabras: Número de palabras de n bits que ocupa la instrucción en código de máquina.

Ciclos: El número de ciclos de máquina que demora la instrucción en ejecutarse.

1.4 LA FAMILIA DE LOS PIC Y SUS INSTRUCCIONES.

1.4.1 LA FAMILIA DE LOS PIC.

El fabricante de los PIC dispone de más de 52 versiones diferentes y cada año aumenta su lista considerablemente. Microchip dispone de cuatro familias de microcontroladores de 8 bits que las enunciamos a continuación:

1. GAMA ENANA: PIC 12C(F)XXX de 8 terminales.
2. GAMA BAJA O BASICA: PIC16C5X con instrucciones de 12 bits.
3. *GAMA MEDIA: PIC16CXXX con instrucciones de 14 bits.*
4. GAMA ALTA: PIC17CXXX PIC18CXXX con instrucciones de 16 bits.

1.4.1.1 Gama Enana: PIC 12C(F)XXX de 8 terminales.

Son de reducido tamaño, de 8 terminales.

El repertorio de instrucciones es de 33 o 35 con una longitud de 12 o 14 bits por instrucción respectivamente.

1.4.1.2 Gama baja o básica: PIC16C5X con instrucciones de 12 bits.

Encapsulados con 18 y 28 terminales

Tienen un repertorio de 33 instrucciones con formato de 12 bits.

1.4.1.3 Gama Media: PIC16CXXX con instrucciones de 14 bits.

Es la gama más variada, Abarca modelos con encapsulado de 18 hasta 68 terminales, dentro de esta familia está el microcontrolador PIC 16X84.

Cuenta con 35 instrucciones de 14 bits cada una, compatible con el conjunto de instrucciones de la gama baja.

1.4.1.4 Gama alta: PIC17CXXX -PIC18CXXX con instrucciones de 16 bits.

Contiene un repertorio de 58 instrucciones para los PIC17CXXX y 72 instrucciones para los PIC 18CXXX, el formato es de 16 bits.

Son de arquitectura abierta, con la posibilidad de ampliación del microcontrolador con elementos externos.

1.4.2 REPERTORIO RISC

Todos los modelos de microcontroladores PIC responden a instrucciones RISC, que significa "Conjunto de instrucciones reducido para computador" que implica las siguientes características:

- Las instrucciones carecen de complejidad, casi todas las instrucciones tardan en ejecutarse un ciclo de instrucción.

- Tienen pocas restricciones en el uso de operandos.

- Todas las instrucciones tienen la misma longitud, por ejemplo 14 bits en los PIC 16X8X, así como todos los datos son de un byte. Con memoria de instrucciones y datos de diferente longitud de palabra debido a la arquitectura Harvard.

1.4.3 TIPOS DE FORMATOS PARA EL MODELO PIC 16X8X.

Cada instrucción de los PIC de la gama media, entre los que se encuentran los modelos PIC 16X8X tienen una longitud de 14 bits, dividida en campos de bits referentes a elementos que maneja la instrucción, así:

a) Campo del código OP (OPCODE).

b) Campo de los operandos fuente (f) y destino (d).

c) Campo de operando inmediato o literal (k).

d) Campo que referencia a un bit (b).

e) Campo de la dirección del salto.

Los diversos formatos que admiten las instrucciones de los PIC 16X8X se clasifican en cinco grandes grupos, atendiendo al tipo de operación que desempeñan así:

1.- Operaciones orientadas a manejar registros de tamaño byte.

El formato de las instrucciones de este grupo se divide en tres campos:

a) Campo del código OP de 6 bits.

b) Campo de la dirección del operando fuente (f) de 7 bits

c) Campo que define el operando destino (d) de 1 bit

Sintaxis: *[label] nemónico f, d*

Cuando d=1 el registro destino coincide con el fuente.

2.- Operaciones orientadas a manejar bits.

El formato de las instrucciones de este grupo se divide en tres campos:

d) Campo del código OP de 4 bits.

e) Campo de la dirección del operando fuente de 7 bits

f) Campo de la posición del bit en el registro de 3 bits

Sintaxis: *[label] nemónico f, b*

3. Operaciones que manejan un valor inmediato o literal

El formato de las instrucciones de este grupo tiene dos campos:

a) Campo del Código OP de 6 bits.

b) Campo del valor inmediato (k) con 8 bits.

4. Operaciones incondicionales de control del flujo del programa.

Estas instrucciones afectan al contenido del Contador de Programa (PC) y sirven para romper la secuencia ordenada de las instrucciones del programa. Su formato tiene dos campos:

a) Campo del código OP de 3 bits.

b) Campo de la dirección del salto que se carga en el PC de 11 bits.

5. Operaciones de salto condicional.

Se tienen instrucciones que cuando se cumple una condición proceden a realizar el salto de una instrucción. La condición es el estado de un bit de un registro o la puesta a cero de un registro tras un decremento o incremento.

El formato de las instrucciones de salto es:

g) Campo del código OP de 6 bits.

h) Campo de la dirección del operando fuente (f) de 7 bits.

i) Campo que define el operando destino (d) de 1 bit.

1.4.4 NOMENCLATURA Y SIMBOLOS

Se utilizará la nomenclatura y símbolos que emplea MICROCHIP para el lenguaje ensamblador MPASM.

SÍMBOLO	DESCRIPCIÓN
0xhh	Es la forma que se usa en el lenguaje MPASM para referenciar a los números hexadecimales de dos dígitos (hh). Por ejemplo 0x05.
f	Representa la dirección en la memoria RAM de datos del registro fuente. Tiene un tamaño de 7 bits, con un direccionamiento de 128 posiciones comprendidas entre la dirección 0x00 y la0x7F.
W	Registro de trabajo (acumulador).
b	Indica la dirección del bit de un registro de 8 bits.
d	Es un bit que conforma el campo del formato de una instrucción que indica el registro destino. Si d=0 el resultado se almacena en el registro "W", y si d=1 el resultado se almacena en "f".
k	Campo que contiene un valor inmediato, dato constante o etiqueta.
label	Nombre de una etiqueta.

x	Valor inmediato de un bit. El asembler generará código con x=0. Es la forma recomendada de uso para compatibilidad con todas las herramientas de software de Microchip.
( )	Contenido.
[ ]	Indicador de Opciones.
→	Sentido de flujo de la información.

Fig. 1.2 Nomenclatura utilizada

1.4.5 CONJUNTO DE INSTRUCCIONES PARA EL MODELO PIC 16X8X.

[3]A continuación se indica el conjunto de instrucciones para el modelo PIC 16X8X.

1.4.5.1 Instrucciones que manejan registros

SINTAXIS: ***[label] nemónico f, d***

Siendo *f* y *d* los dos operandos fuente y destino implementados por registros de 8 bits de la memoria de datos; el registro *f* referenciado por la dirección de 7 bits, mientras que el destino solo por 1, que si vale 0 es el W y si vale 1 es el fuente.

SINTAXIS	*OPERACIÓN*
[label] ADDWF f,d	Suma el contenido del registro de trabajo "W" (Acumulador) con el registro "f", si el segundo operando "d" de la instrucción es 0 el resultado es almacenado en W si es 1 en "f".
[label] ANDWF f,d	AND entre el contenido del registro W con el contenido de "f", si "d" es 0 se almacena en W si "d" es 1 en "f".

[3] La clasificación o grupos de instrucciones se presenta de acuerdo a los autores Angulo&Angulo en su libro MICROCONTROLADORES PIC Diseño práctico de aplicaciones, sin embargo en el sitio web www.microchip.com se puede encontrar los "Technical Description" en donde se indica con mayor detalle el conjunto de instrucciones y para diferentes modelos de microcontroladores.

[label] CLRF f	El contenido del registro "f" es borrado (se pone los bits a cero).
[label] CLRW	Borra "W".
[label] COMF f,d	El contenido del registro "f" es complementado, si "d" es 0 el resultado es almacenado en "W", si d es 1 el resultado es almacenado en el registro "f".
[label] DECF f,d	Decrementa el registro "f", si "d" es cero el resultado va a "W", si "d" es 1 el resultado es almacenado en "f".
[label] INCF f,d	Incrementa el registro "f", si "d" es cero el resultado es almacenado en "W". Si "d" es 1 el resultado es almacenado en "f".
[label] IORWF f,d	OR inclusiva del registro "W" con el registro "f", si el segundo operando es 0 el resultado es almacenado en "W" si es 1 en "f".
[label] MOVF f,d	El contenido del registro "f" es movido al destino dependiendo del estado de "d", si el segundo operando "d" es 0 el destino es "W", si es 1 el destino es "f".
[label] MOVWF f,d	Mueve datos del registro "W" al registro "f".
[label] NOP	Ninguna operación.
[label] RLF f,d	El contenido del registro "f" es rotado un bit a la izquierda a través del acarreo. Si "d" es 0 el resultado es puesto en el registro "W". Si "d" es 1 el resultado va a "f".
[label] RRF f,d	El contenido del registro "f" es rotado un bit a la derecha a través del acarreo. Si "d" es 0 el resultado es puesto en el registro "W". Si "d" es 1 el resultado va a "f".
[label] SUBWF f,d	Resta "W" de f. Si "d" es 0 el resultado es guardado en el registro "W", si es 1 el resultado es guardado en "f".

[label] SWAPF f,d	Intercambia nibbles bajos y altos del registro "f", si "d" es 0 el resultado es almacenado en "W" si es 1 se almacena en el registro "f".
[label] XORWF f,d	OR exclusivo del contenido de "W" con "f". Si "d" es 0 es almacenado en el registro "W" y si "d" es 1 el resultado es almacenado en el registro "f".

Fig. 1.3 Tabla de instrucciones que trabajan con registros.

1.4.5.2 Instrucciones que manejan bits.

SINTAXIS: ***[label] nemónico f, b***

Se tiene dos instrucciones que manejan bits.

SINTAXIS	*OPERACIÓN*
[label] BSF f,b	Pone a uno el bit "b" del contenido del registro "f".
[label] BCF f,b	Pone a cero el bit "b" del contenido registro "f".

Fig. 1.4 Tabla de instrucciones que manejan bits.

1.4.5.3 Instrucciones de salto.

SINTAXIS: ***[label] nemónico f, d***

Se presentan cuatro instrucciones de salto.

SINTAXIS	*OPERACIÓN*
[label] BTFSC f,d	Explora un bit de "f", si el bit "b" en el registro "f" es 1 entonces se ejecuta la siguiente instrucción, si el bit "b" en el registro "f" es 0 la siguiente instrucción es descartada (salta), y una instrucción NOP es ejecutada instantáneamente marcando 2 ciclos.
[label] BTFSS f,d	Explora un bit de "f", si el bit "b" en el registro "f" es 0 entonces se ejecuta la siguiente instrucción, si el bit "b" en el registro "f" es 1 entonces la siguiente

	instrucción es descartada, y un NOP es ejecutado instantaneamente marcando 2 ciclos.
[label] DECFSZ f,d	El contenido del registro "f" es decrementado, si "d" es 0 el resultado va al registro "W"; si "d" es 1 el resultado va a "f". Si el resultado es 0 salta una instrucción, si es 1 se ejecuta la siguiente instrucción.
[label] INCFSZ f,d	El contenido del registro "f" es incrementado, si "d" es 0 el resultado va a "W"; si "d" es 1 el resultado va a "f". Si el resultado es 0 salta una instrucción, si es 1 se ejecuta la siguiente instrucción.

Fig. 1.5 Tabla de instrucciones de salto.

1.4.5.4 Instrucciones que manejan operandos inmediatos

SINTAXIS: ***[label] nemónico k***

Hay seis instrucciones de este tipo.

SINTAXIS	*OPERACION*
[label] ADDLW k	El contenido del registro "W" es sumado a los 8 bits de "k" y el resultado es puesto en el registro "W".
[label] ANDLW k	El contenido del registro "W" se hace un AND con los 8 bits de "k". El resultado es puesto en "W".
[label] IORLW k	El contenido del registro "W" se hace un OR con los 8 bits de "k", el resultado es puesto en "W".
[label] MOVLW k	Mueve a "W" un valor inmediato indicado en "k".
[label] SUBLW k	Resta los bits "W" de "k". (con el método complemento de 2). El resultado es almacenado en el registro "W".
[label] XORLW k	OR exclusiva de los ocho bits de k con el contenido de W. El resultado es puesto en el registro W.

Fig. 1.6 Tabla de instrucciones que manejan operandos inmediatos.

CAPITULO 2. EL EDITOR INTELIGENTE

2.1 CARACTERISTICAS DEL "SMART PIC EDITOR"

La herramienta de software "Smart Pic Editor" se encuentra enmarcada dentro del ámbito de las actividades propias del desarrollo de proyectos con microcontroladores, siendo la etapa de la edición de programas en lenguaje de bajo nivel el pilar fundamental.

Debido a la utilización de esta herramienta de software en el ámbito investigativo, de estudio y de enseñanza de la programación con microcontroladores en las Universidades y Escuelas Politécnicas, se han considerado criterios importantes a la hora de diseñar está herramienta al igual que para escoger el software base para la implementación.

2.1.1 CARACTERISTICAS GENERALES

En la presente sección se definen las características generales de la herramienta software desarrollada que corresponde a un editor de programas para microcontroladores. Estas características son:

- Funcionamiento sobre el sistema operativo Windows 95 o superior.
- Fácil edición de los mnemónicos de las instrucciones y pseudoinstrucciones, mediante el despliegue de alternativas válidas de las cuales se pueda escoger.
- Detección de los errores de sintaxis cometidos al elaborar un programa en mnemónicos de los Microcontroladores PIC.
- Despliegue de información referente a la operación de las instrucciones, sintaxis, código de máquina, ejemplo para el uso, etc.

- Uso de color, en la visualización de las instrucciones en la pantalla de edición así como en los resultados impresos de los textos editados.

- Utilidades de edición como las de cortar, copiar, pegar, buscar, reemplazar, etc.

- Los archivos editados son del mismo formato que utilizan los programas ensambladores de estos microcontroladores como archivos de entrada, es decir, archivos que usen únicamente códigos ASCII.

- La extensión de los archivos generados por omisión es ASM es decir el archivo resultante será un ***. ASM**. Donde * (asterisco) corresponde a un nombre válido de archivo cualesquiera.

- Permitir la edición de un solo archivo a la vez, debido a que está herramienta puede ser parte de un módulo completo de software de edición, simulación, y ensamblaje de programas para microcontroladores PIC.

- Facilidad en la instalación y difusión pues es una herramienta didáctica para la enseñanza de la programación de microcontroladores PIC.

- Módulo administrador que permite modificar fácilmente el contenido de la base de datos con la que trabaja el Editor con la finalidad de actualizarla con instrucciones adicionales que poseen las nuevas versiones de los microcontroladores PIC.

2.1.2 REQUERIMIENTOS DE INTERFAZ

El editor desarrollado considera los siguientes requerimientos de interfaz:

2.1.2.1 De Interfaz de Usuario

- Interfaz visual amigable para el usuario.

- Características funcionales con que cuentan las ventanas de Windows por ejemplo: uso de menús, botones de acceso rápido, barra de estado.

- Interfaz sencilla y de fácil uso para el administrador de la base de datos.

2.1.2.2 De Interfaz de Software

El Editor trabaja bajo ambiente del sistema operativo Windows 95 o superior.

2.1.2.3 De Interfaz de Hardware

- El programa corre en un computador personal (PC); se recomienda las siguientes características de hardware mínimas para ganar en velocidad de procesamiento de las instrucciones que ejecuta el "Smart PIC Editor" y para que su desempeño sea óptimo:

    - Monitor a color.
    - Procesador Pentium II o superior.
    - 32 MB de memoria RAM o más.
    - Mínimo 3 MB disponibles en Disco Duro.
    - Sistema Operativo Windows 95 o superior.
    - Ratón de dos botones.
    - Impresora a color.

  Es decir, las características de un computador personal, disponible en el mercado actual son suficientes.

- Uso tanto del ratón como el teclado para brindar al usuario mayores ventajas en cuanto a velocidad de escritura del programa como para la selección y

navegación de las ayudas en pantalla (listas de opcodes, operandos relacionados con opcodes seleccionados, pantallas de corrección de errores, entre otras).

2.1.3 ESTRUCTURA DEL SOFTWARE DESARROLLADO

El software desarrollado está estructurado por dos grandes componentes:

El Editor.- Cuya base es la de un editor de texto normal en ambiente Windows, en este editor se implementan una serie de herramientas de edición y corrección de sintaxis de un programa para microcontroladores PIC, que le dan el carácter inteligente al Editor.

El Administrador de la base de datos.- Diseñado para que por su intermedio se pueda administrar y gestionar la información de la base de datos, es decir, el conjunto de instrucciones de varios modelos de microcontroladores que son utilizados por el editor.

En la figura 2.1 se muestra un diagrama de bloques de los componentes general del editor y del administrador de la base de datos.

HERRAMIENTAS DE EDICION Y CORRECCION

2.2.1 HERRAMIENTAS DE EDICION

A continuación indicamos cada una de las herramientas que dispone el "Smart Pic Editor".

Barra de menú.- Comprende una barra de menú típica de Windows con las opciones comunes de un editor de texto, cuenta con los siguientes menús:

- Menú Archivo.
- Menú Edición.

- Menú Ver.
- Menú Herramientas.
- Menú Ayuda.

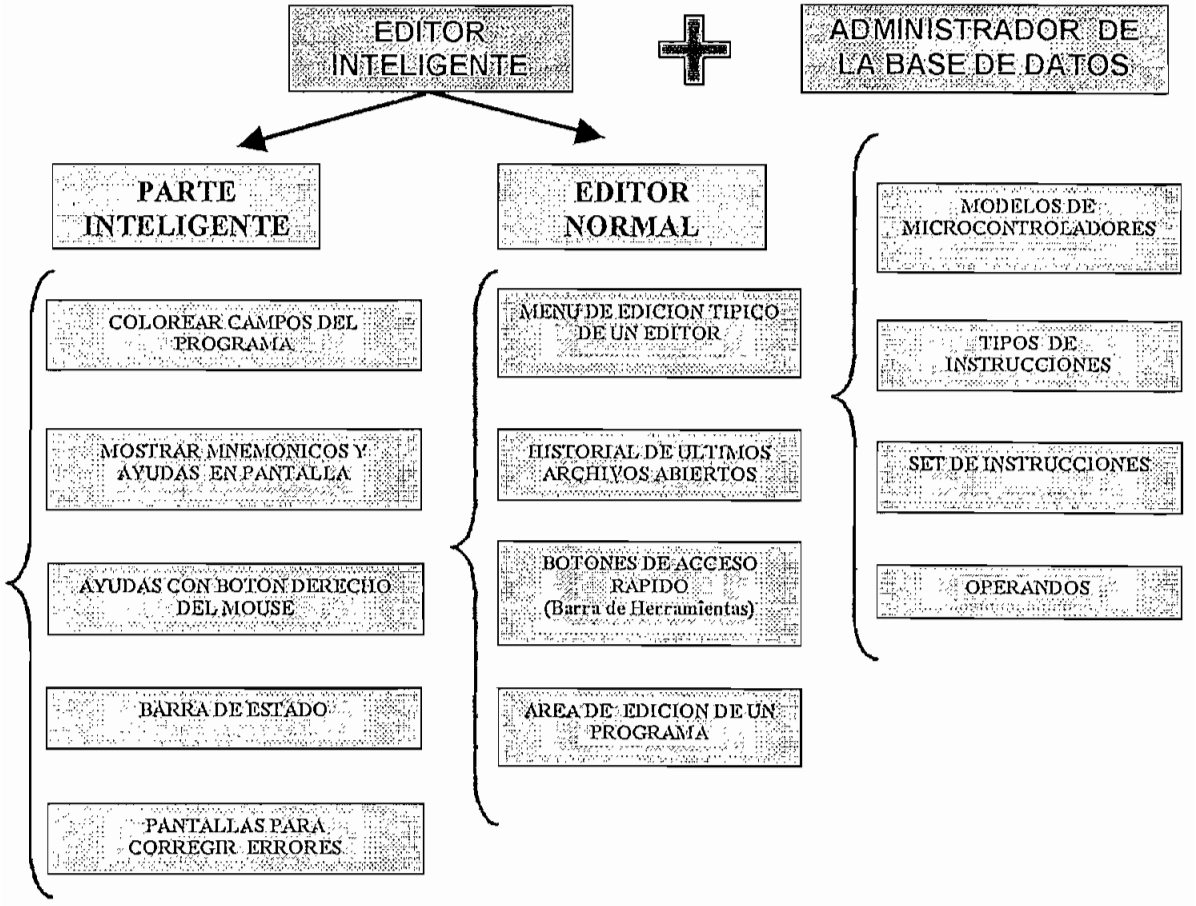


Fig. 2.1 Componentes Generales del "Smart PIC Editor" y del "Admin Pic Editor"

Barra de herramientas.- Corresponde a una barra con botones para acceso directo a las opciones usadas con mayor frecuencia, estos botones cuentan con una interfaz gráfica que permite identificar fácilmente una opción.

Así se puede listar los siguientes botones:

- Botón Abrir.

- Botón Guardar.
- Botón Imprimir.
- Botón Cortar.
- Botón Copiar.
- Botón Pegar.
- Botón Buscar.
- Botón para corregir errores del programa.
- Botón de opciones del editor.
- Botón para insertar caracteres.
- Botón de selección de modelo de microcontrolador.
- Botón Acerca del editor.

Barra de campos.- Presenta un distintivo de cada uno de los campos para una correcta tabulación del programa a editar.

Estos campos son: Etiqueta, Opcode, Operandos y Comentarios

Barra de estado.- Presenta cinco paneles de descripción de eventos durante la edición y corrección de un programa:

- Panel de mensajes que el editor genera para orientar al usuario.
- Panel de línea actual.
- Panel de fecha del PC.

- Panel de hora actual.

- Panel informativo del modelo de microcontrolador con el que se está trabajando.

Area de edición.- Comprende el espacio destinado a la edición del programa.

Listados de mnemónicos y ayudas en pantalla.- Dependiendo de la configuración, modelo de microcontrolador escogido y el campo en donde se encuentre el cursor, el editor permite presentar listas conteniendo los posibles opcodes o una vez escogidos estos, los posibles operandos, para una rápida y eficaz edición de un programa para microcontrolador.

En la figura 2.2 se presenta un esquema general que resume los componentes principales de la pantalla de edición del "Smart PIC Editor".

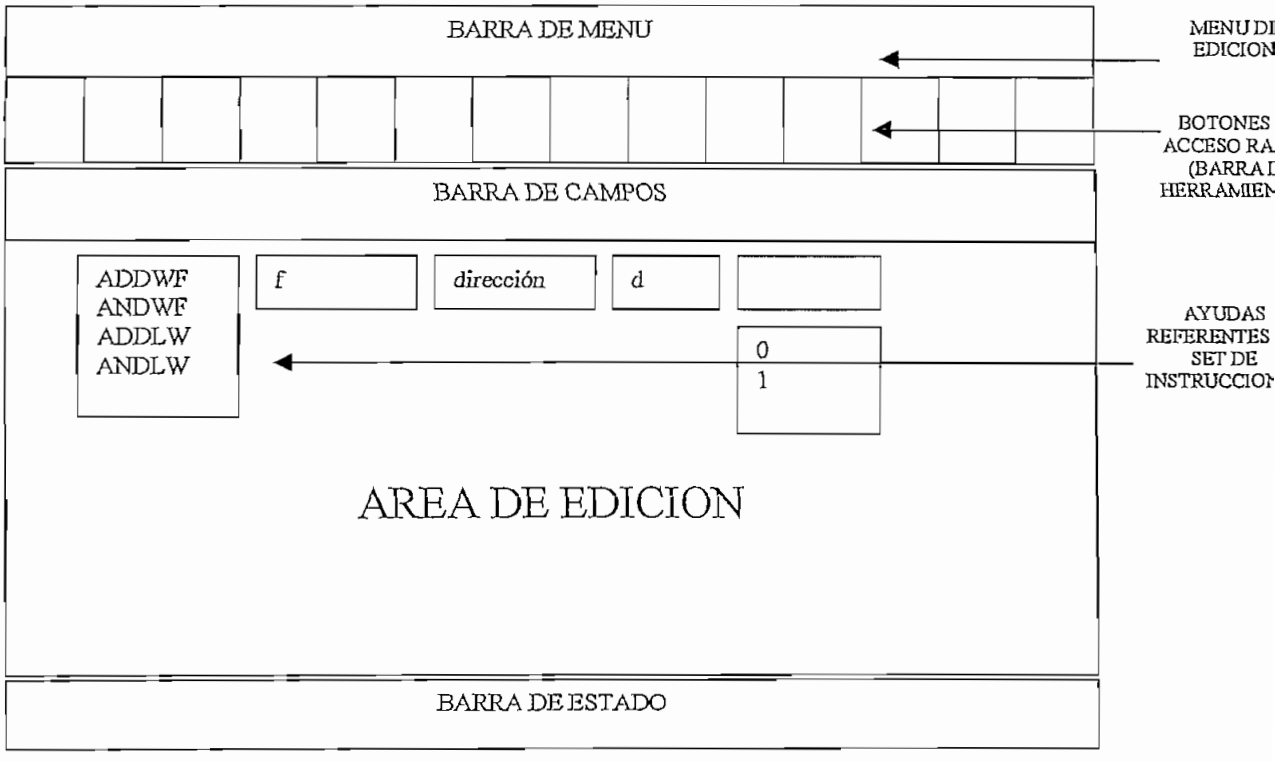


Fig. 2.2 Componentes de la pantalla de Edición

2.2.2 HERRAMIENTAS DE CORRECCION

Opciones de corrección.- El editor trabaja en dos modalidades de corrección:

- Corregir errores de sintaxis al finalizar el programa.
- Corregir errores al finalizar la edición de una línea de programa.

Colorear el programa.- Para mayor facilidad en la detección de errores durante la edición y corrección de un programa se colorea cada una de las líneas de la siguiente manera:

- Color Negro.- Para indicar una instrucción sintácticamente correcta.
- Color Azul.- Para indicar una seudoinstrucción correctamente escrita.
- Color Verde.- Para pintar la parte de una línea que corresponde a un comentario.
- Color Rojo.- Indica una instrucción o pseudoinstrucción escritas de manera incorrecta.

Esta opción de colorear se presenta: al cargar un programa, al cambiar de línea durante la edición, al cambiar el modelo de microcontrolador utilizado en la sesión.

Pantallas de corrección de errores.- Constituyen un conjunto de pantallas que se despliegan dinámicamente al analizar una línea y encontrar un error, estas pantallas tienen la finalidad de orientar al usuario en la corrección de los errores detectados.

Ayudas con botón derecho del ratón.- Con el botón derecho del ratón se puede consultar rápidamente las características principales de las instrucciones o pseudoinstrucciones o corregir un error presente en una línea.

2.3 OTRAS AYUDAS DEL EDITOR

2.3.1 EL ADMINISTRADOR DE LA BASE DE DATOS

Una característica muy importante del editor a desarrollarse es la flexibilidad en cuanto a manejo del conjunto de instrucciones de diferentes modelos de microcontroladores PIC; para brindar esta característica se desarrolló el módulo de administración de la base de datos "Admin Pic Editor".

2.3.2 COMPONENTES DEL ADMINISTRADOR

Permite la administración de los siguientes parámetros del conjunto de instrucciones de un microcontrolador:

Modelo del microcontrolador.- En vista que el editor es flexible en cuanto al uso del microcontrolador y su respectivo conjunto de instrucciones, se necesita identificar a que modelo de microcontrolador pertenece un conjunto de instrucciones.

Tipo de Instrucciones.- Tanto en el conjunto de instrucciones que aparecen en los "Technical Description" de la Microchip así como varios autores suelen clasificar a las instrucciones de un modelo específico de microcontrolador en grupos o tipos de instrucciones. Por ello, la asignación de una instrucción a un "tipo de instrucción" con el Administrador es dinámica, permitiendo el intercambio de grupo o creación de un nuevo tipo de instrucciones.

Tipo de Operandos.- Debido a que los tipos de operando son genéricos para varias instrucciones, se los podría definir para su administración.

Algunos parámetros para ser gestionados son:

- Nombre
- Descripción

- Tipo de Operando
- Valores aceptados

Instrucciones.- La parte más importante del Administrador de la base de datos es la manipulación del conjunto de instrucciones de los distintos modelos de microcontroladores.

Se pretende manipular información como:

- Opcode.
- Operandos.
- Valores de límites de Operandos.
- Sintaxis.
- Operación simbólica.
- Ejemplo de uso.
- Código de Máquina.
- Banderas afectadas
- Descripción.

Así como indicadores que permitan: orientar al editor para diferenciar una instrucción de una pseudoinstrucción, si la instrucción o pseudoinstrucción requiere de etiqueta obligatoria, también si las pseudointrucciones admiten operandos opcionales.

2.4 LA BASE DE DATOS

Para el almacenamiento de la información que va a ser utilizada por el editor se ha definido una base de datos, como se conoce, una base de datos es un conjunto de información relacionada entre si, en el siguiente numeral se indican las características que debe cumplir la base de datos.

2.4.1 CARACTERISTICAS DE LA BASE DE DATOS PARA EL EDITOR

La base de datos que contenga información del conjunto de instrucciones de los microcontroladores debe cumplir con las siguientes características:

- Lenguaje común para interactuar con la herramienta software en la que se desarrollará el editor.
- Ingreso y extracción de datos de la base de una manera sencilla.
- No requiere un software especial para su funcionamiento.
- Acepte un modelo entidad - relación es decir pueda constituirse en una base relacional.

2.4.2 ANALISIS DE LA BASE DE DATOS REQUERIDA

2.4.2.1 Modelo Entidad Relación.

La mayoría de las bases de datos comerciales que existen en el mercado son relacionales, es decir, se conforman básicamente de entidades y relaciones entre estas entidades; es por ello que para poder diseñar la base de datos del editor es necesario conocer en que consiste el modelo Entidad – Relación.

Entidad.- Es cualquier objeto o ente que está definido por un conjunto de atributos.

Para representar una entidad gráficamente utilizamos la siguiente simbología:

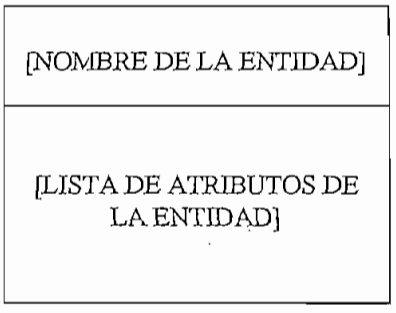


Fig. 2.3 Representación de una Entidad

Atributos.- Son características que definen una entidad; pueden haber muchos atributos para cada entidad.

Dependiendo de la naturaleza del problema, ciertos atributos de una entidad pueden ser o no relevantes.

Por ejemplo en una institución de educación un ALUMNO sería una entidad. Su nombre, dirección, fecha de nacimiento, número de cédula serían algunos de sus atributos relevantes.

Llave o clave primaria.- Se denomina clave o llave primaria de una entidad a un atributo o combinación de atributos que permiten distinguir una entidad de otra.

En el diagrama entidad -relación de la base de datos que se presenta en la figura 2.5, los atributos que conforman la llave primaria se presenta en negrilla y subrayados para diferenciarlos de los demás.

Relaciones.- Son asociaciones entre entidades y permiten determinar de que manera el conjunto de atributos de una entidad se relaciona con el conjunto de atributos de otra.

Cardinalidad.- Se denomina así al número de nexos que tiene una entidad con otra entidad. En la figura 2.4 se muestra los diferentes tipos de cardinalidad existentes.

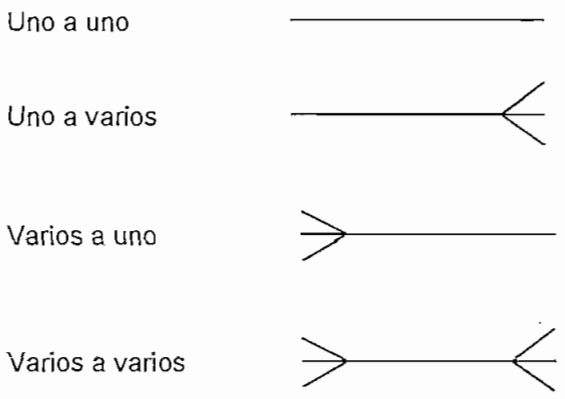


Fig.2.4 Representación de la cardinalidad de una relación

2.4.2.2 Estructura de la Base de Datos

Después del estudio y análisis de las instrucciones de los diversos modelos de microcontroladores PIC, y para cumplir con las necesidades de información que requiere el editor para su funcionamiento se ha diseñado el modelo entidad-relación de la figura 2.5.

2.4.2.3. Descripción de las Entidades y Relaciones

Entidad: ModeloMicro	Esta entidad comprende los modelos de microcontroladores por ejemplo: PIC16C84, PIC12C5XX PIC18CXX2, etc.		
Atributos	**Llave primaria**	**Obligatorio**	**Significado**
Modelo	SI	SI	Código identificador del modelo de microcontrolador.
Nombre	NO	SI	Nombre descriptivo del modelo de microcontrolador.
Descripción	NO	NO	Descripción opcional del modelo de microcontrolador.

Entidad: Tipo de Instrucción	Esta entidad comprende los diversos tipos de instrucciones en los que pueden clasificarse o agruparse las instrucciones de un microcontrolador. Por ejemplo un tipo de instrucción podría ser: Operaciones orientadas a manejar registros de tamaño byte.		
Atributos	**Llave primaria**	**Obligatorio**	**Significado**
Modelo	SI	SI	Código identificador del modelo de microcontrolador al que corresponde el tipo de instrucción.
ID	SI	SI	Código identificador del tipo de instrucción.
Nombre	NO	SI	Nombre del tipo de instrucción.
Descripción	NO	NO	Explicación adicional sobre el tipo de instrucción.

Entidad: Instrucción	Esta entidad comprende el conjunto de instrucciones que puede tener cada modelo de microcontrolador PIC.		
Atributos	**Llave primaria**	**Obligatorio**	**Significado**
Modelo	SI	SI	Código identificador del modelo de microcontrolador al que corresponde la instrucción.

Identificador	SI	SI	Corresponde a un valor autonumérico que identifica a la instrucción o directiva.
Opcode	NO	SI	Es el mnemónico de la instrucción u opcode.
Operando1	NO	NO	Constituye el primer operando en caso de que la instrucción lo requiera.
Operando2	NO	NO	Constituye el segundo operando en caso de que la instrucción lo requiera.
Operando3	NO	NO	Constituye el tercer operando en caso de que la instrucción lo requiera.
Operando4	NO	NO	Constituye el cuarto operando en caso de que la instrucción lo requiera.
Sintaxis	NO	NO	Es la forma de escritura de la instrucción. Por ejemplo: ADDWF f,d
Operación Simbólica	NO	NO	Es la operación que realiza en forma de símbolos Por ejemplo: Para la instrucción BCF f,b la operación simbólica sería: 0→(f<b>).
Ejemplo	NO	NO	Constituye un ejemplo de una instrucción así: ADDWF 06H,0.
Descripción	NO	NO	Texto descriptivo de la operación de la instrucción.

Código de Máquina	NO	NO	El código correspondiente a la instrucción después de ensamblarla.
Ciclos	NO	NO	Número de ciclos de máquina que demora en ejecutarse la instrucción.
Banderas	NO	NO	Las banderas activadas o desactivadas al ejecutarse la instrucción.
Seudo Instrucción	NO	NO	Indicativo que diferencia entre una instrucción y una seudo instrucción.
Limites de los Operandos	NO	NO	Rango de valores que pueden tomar los operandos.
Palabras	NO	NO	Número de palabras que ocupa la instrucción.
Exigir Etiqueta	NO	SI	Indica si una instrucción o pseudoinstrucción necesita una etiqueta obligatoria. Los valores que puede tomar son: "Si" y "No".
Operandos Opcionales	NO	SI	Indica si una instrucción o pseudoinstrucción puede tener uno o más operandos opcionales. Los valores que puede tomar son: "Si" y "No".

Entidad: Tipo de Operando	Permite conocer el tipo de operando que acompañan a los opcodes en las instrucciones, por ejemplo: bit, una dirección, una etiqueta, etc.		
Atributos	**Llave primaria**	**Obligatorio**	**Significado**
Modelo	SI	SI	Modelo de microcontrolador al que pertenece el tipo de operando.
Nombre	SI	SI	Nombre del tipo de operando.
Descripción	NO	NO	Explicación adicional sobre el tipo de operando.
Tipo de Valor	NO	SI	Definimos los siguientes: Lista, constante y variable

Entidad: ValorOperando	Esta entidad contiene cada uno de los valores que puede tomar, un operando de una instrucción en el caso de que necesitemos restringir a un conjunto de valores.		
Atributos	**Llave primaria**	**Obligatorio**	**Significado**
Modelo	SI	SI	Código identificador del modelo de microcontrolador.
Operando	SI	SI	Código identificador del operando al cual se está restringiendo el valor.

Valor	SI	SI	Valor que puede tomar el operando.

Relación: 1	Relación entre un Modelo de Microcontrolador y los tipos de instrucción que puede contener. Indica que una o más tipos de instrucciones se encasillan en un modelo, y viceversa.
Entidad Origen	ModeloMicro.
Entidad Destino	Tipo de Instrucción.
Cardinalidad	1 a varios.

Relación: 2	Relación entre un Modelo de Microcontrolador y los tipos de operandos que utilizan sus instrucciones. Indica que uno o más tipos de operando se agrupan en un Modelo.
Entidad Origen	ModeloMicro.
Entidad Destino	Tipo de Operando.
Cardinalidad	1 a varios.

Relación: 3	Relación entre un Modelo de Microcontrolador y su conjunto de instrucciones. Indica que un modelo tiene una o más instrucciones con sus

	características propias.
Entidad Origen	ModeloMicro.
Entidad Destino	Instrucción.
Cardinalidad	1 a varios.

Relación: 4	Relación entre un tipo de instrucción y las instrucciones que lo conforman. Indica que una instrucción puede o no ser parte de un grupo o tipo de instrucciones.
Entidad Origen	Tipo de Instrucción.
Entidad Destino	Instrucción.
Cardinalidad	1 a varios.

Relación: 5	Relación entre el tipo de operando y cada uno de los operandos que conforman la instrucción (operando1, operando2, operando3 u operando 4).
Entidad Origen	Tipo de Operando.
Entidad Destino	Instrucción.
Cardinalidad	1 a varios.

Relación: 6	Relación entre el tipo de operando y los valores que puede tomar este en caso de que dichos valores se quieran restringir Es decir que un tipo de operando puede restringirse a uno o más valores.
Entidad Origen	Tipo de Operando.
Entidad Destino	Valor Operando.
Cardinalidad	1 a varios.

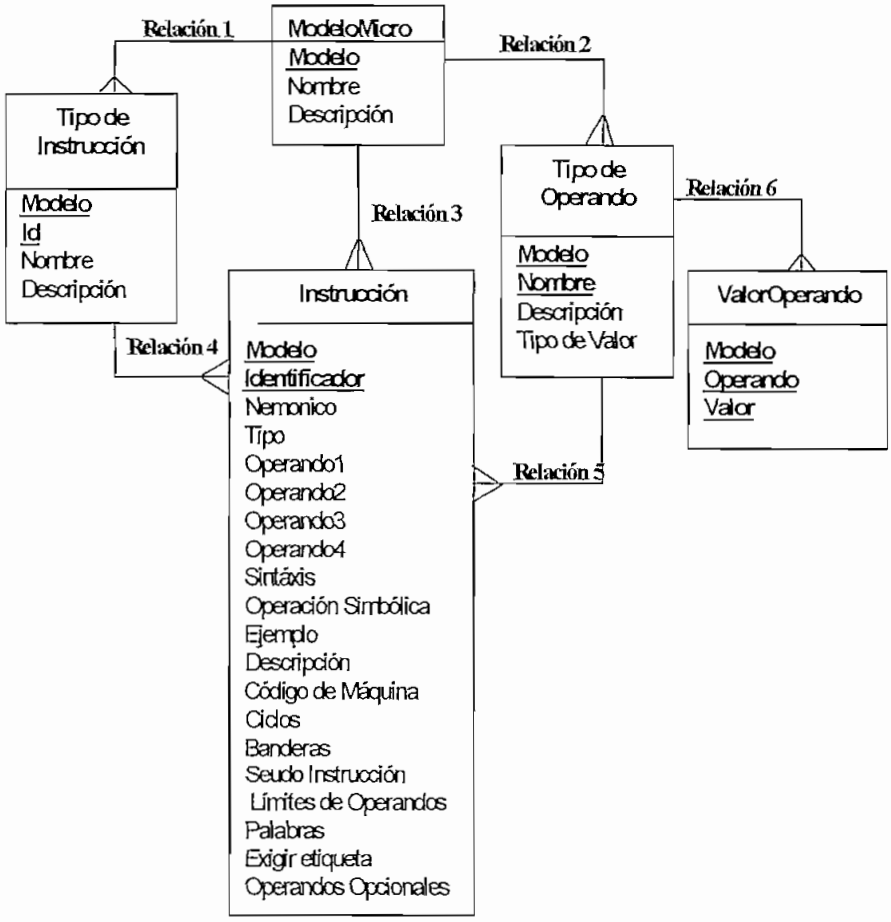


Fig. 2.5 Modelo Entidad - Relación de la Base de Datos

CAPITULO 3. DESARROLLO DEL SOFTWARE

3.1 INTERFAZ DEL EDITOR

3.1.1 SELECCIÓN DE LA HERRAMIENTA PARA DESARROLLAR EL EDITOR

Existen algunos lenguajes de programación que se usan para desarrollo de software, citaremos algunos de ellos:

- BASIC
- PASCAL
- TURBO C
- LENGUAJE C
- LENGUAJE C++
- VISUAL BASIC
- JAVA

Así mismo existen algunas herramientas software en diversos lenguajes de programación y cuyas versiones cada vez permiten resolver problemas de mayor complejidad y con mayores facilidades citaremos algunas:

- BORLAND C
- VISUAL C++
- VISUAL FOX PRO

- VISUAL BASIC

- VISUAL J++

De entre estos lenguajes de programación y herramientas se ha **seleccionado Visual Basic 6.0 de Microsoft** para el diseño e implementación de nuestra aplicación; indicamos a continuación las razones por las que fue escogida dicha herramienta:

- Debido a que es muy importante un entorno amigable y ayudas visuales para el programador.

- Visual Basic genera aplicaciones que trabajan en el sistema operativo Microsoft® Windows® muy difundido en la actualidad.

- Su aplicación terminada es un archivo .EXE que en tiempo de ejecución utiliza bibliotecas de vínculos dinámicos (DLL), este se pueden distribuir libremente.

- Visual Basic 6.0 cuenta con un asistente para empaquetado y distribución de las aplicaciones generadas en esta herramienta, lo cual facilita el trabajo tanto del desarrollador como del usuario en el momento de realizar la instalación de dichas aplicaciones.

- Incorpora funciones de acceso a datos que permiten utilizar las bases de datos más difundidas en el mercado, entre ellas Microsoft Access.

- Los requerimientos de hardware y software permiten su instalación en una PC:

    - Microsoft Windows 95, NT Workstation 4.0 o superior.

    - Procesador 486DX/66 MHz o superior (se recomienda Pentium o superior).

- Unidad de CD-ROM.

- Pantalla VGA o de mayor resolución compatible con Microsoft Windows.

- 16 MB de RAM para Windows 95, 32 MB de RAM para Windows NT Workstation.

3.1.2 INTRODUCCION A LA PROGRAMACION EN VISUAL BASIC

Visual Basic es un lenguaje "Visual" debido al método que se usa para crear la interfaz gráfica de usuario (GUI), pues sin escribir numerosas líneas de código para describir propiedades y características de los elementos de la interfaz, simplemente se puede agregar objetos prefabricados en su lugar dentro de la pantalla.

Visual Basic ha evolucionado a partir del lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code), lenguaje muy difundido en la historia de la informática o computación y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están directamente relacionadas con la interfaz gráfica de Windows.

Se puede crear aplicaciones útiles con sólo aprender unas pocas palabras clave; la eficacia del lenguaje permite a los profesionales acometer cualquier objetivo que pueda alcanzarse mediante cualquier otro lenguaje de programación de Windows.

Hay tres pasos importantes en la creación de una aplicación Visual Basic para Windows:

1. Crear la interfaz

2. Definir las propiedades

3. Escribir el código

Los formularios son la base de la creación de la interfaz de una aplicación, en el se pueden agregar ventanas, cuadros de diálogo, botones, también sirve como contenedor de elementos que no son parte visible de la interfaz.

Cada uno de los objetos tienen características que pueden ser personalizadas y los harían diferentes entre objeto de la misma especie. Se puede enlistar algunas propiedades: name, caption heigth, width, left, top, Backcolor, Datasource, etc.

Dependiendo del tipo de objeto, este cuenta con un conjunto de propiedades asociadas.

El código consiste en sentencias del lenguaje, constantes y declaraciones por medio de este código se pueden establecer procedimientos y funciones que se realizarán como respuesta a un evento que realice el usuario para de esta manera provocar una interacción del programa con el usuario.

Un módulo contiene código público, declaraciones, procedimientos Sub y funciones que pueden compartirse dentro del proyecto.

3.1.3 COMPONENTES DEL EDITOR

El Smart Pic Editor está compuesto por un conjunto de formularios y módulos:

3.1.3.1 Formularios:

Contiene los siguientes formularios:

FrmAcercade (frmAbout.frm)

Contiene información general acerca del Smart Pic Editor.

FrmBuscar(FrmBuscar.frm)

Formulario que permite buscar un texto dentro del Editor.

FrmCorreccion (FrmCorreccion.frm)

Utilizado para la presentación de la pantalla de corrección de errores del Editor.

Frm ModeloMicro (FrmModeloMicro.frm)

Permite la selección del modelo de microcontrolador cuyo conjunto de instrucciones se utiliza para la edición de un programa en el Editor.

FrmOpciones (FrmOpciones.frm)

Despliega la pantalla que contiene las opciones de configuración para trabajar con el editor inteligente.

FrmPortada (FrmPortada.frm)

Contiene la presentación general de la Herramienta de Software desarrollada "Smart PicEditor".

FrmPrincipal (FrmMain.frm)

Es la ventana que contiene a la pantalla de edición, corresponde a un formulario de tipo MDI (Multiple document interface) que corresponde al fondo de la aplicación del Editor.

FrmPrograma (Frm Programa.frm)

Corresponde a la pantalla de edición del Editor.

FrmReemplazar(FrmReemplazar.frm)

Permite buscar un texto definido dentro de un programa y reemplazarlo por otro texto indicado por el usuario en este formulario.

FrmSimbolos(Form1.frm)

Presenta un conjunto de caracteres ASCII que pueden ser insertados dentro de un programa que está siendo editado.

3.1.3.2 Modulos:

mPrincipalEditor (mPrincipalEditor.bas)

Contiene las instrucciones necesarias para iniciar una sesión con el Editor Inteligente En la subrutina de inicio (main) se implementa la lógica de Inicio de SmartPicEditor, se realizan tareas tales como:

- Establecer el directorio de gráficos como el subdirectorio gráficos.
- Establece el directorio de trabajo si aún no lo está.
- Obtiene el directorio de trabajo del Editor.
- Establece parámetros de Opciones del Editor.
- Obtener el argumento del editor en la línea de comandos.
- Muestra la pantalla de presentación del editor, o si ha indicado la ubicación y nombre de un archivo se abre directamente.

mVariablesyConstantes(mVariablesyconstantes.bas)

Contiene la declaración de constantes y de variables públicas usadas en la programación del Editor. Se puede mencionar las siguientes constantes:

- Caracter de Comentario.
- Caracter Separador de Argumentos.

- Lista de tipos de archivos que se mostrarán en los cuadros de diálogo.
- Elemento predeterminado que se mostrarán en el cuadro de diálogo.
- Constantes para la inicialización del Sistema como los archivos recientes que se guardan en el regedit (registro de configuraciones del windows).

Variables tales como:

- Referencia al formulario MDI.
- La ubicación del directorio del archivo recientemente abierto en el editor.
- Directorio de gráficos usados en el editor.
- Nombre y ruta de la base de datos PIC.
- Nombre del archivo invocado como argumento del programa.
- Identificador de modelo predeterminado a utilizar.
- Nombre del modelo predeterminado a utilizar.
- Variables para la inicialización de las opciones del editor.
- Variables usadas durante el análisis de la sintaxis de un programa.
- Variables para la busqueda de texto en un programa.

mArchivo(mArchivo.bas)

En este módulo se agrupan las funciones y subrutinas utilizadas en el menú archivo del Editor, las cuales permiten: abrir, guardar, guardar como, cerrar, imprimir, guardar el historial de los últimos archivos abiertos.

mEdicion(mEdicion.bas)

Este módulo contiene las funciones y subrutinas utilizadas en el menú edición del Editor, para cortar, copiar, pegar, borrar, seleccionar todo el texto, buscar, buscar siguiente, insertar caracteres.

mVer(mVer.bas)

En este módulo se encuentran las subrutinas para mostrar u ocultar la barra de herramientas, barra de estado, barra de campos, mostrar la posición de la línea actual del cursor, añadir mensajes en la sección de mensajes de la barra de estado, activar o desactivar las opciones del menú de opciones de acuerdo a la existencia o no de programas abiertos, activar o desactivar la barra de herramientas.

mHerramientas(mherramientas.bas)

Contiene las subrutinas para la implementación del menú de herramientas así: corregir errores del programa, corregir errores de la línea actual, información de la instrucción actual, presentación de la pantalla de opciones del editor, presentación de la pantalla para escoger el modelo de microcontrolador.

mInteligente(minteligente.bas)

Contiene funciones y subrutinas para:

Analizar la sintaxis de un programa y colorear un programa.

Analizar la sintaxis de una línea de programa y colorear.

Validar si una palabra es etiqueta, opcode u operando válido.

Mostrar Lista de Mnemónicos cuando se está editando el programa.

Obtener el número de Operandos que tiene una instrucción.

Obtener la posición donde inicia el comentario dentro de la línea actual.

Analiza los tabuladores (Tabs) y cambios de línea (Enter) para saber en que campo del programa está ubicado el cursor.

mAyuda(mAyuda.bas)

Contiene una subrutina para presentar la pantalla informativa acerca del editor.

3.2 DIAGRAMAS DEL EDITOR

A continuación se presenta los diagramas de flujo que indican de manera general el funcionamiento del editor y como este interactua con los eventos que realiza el usuario.

3.2.1 PANTALLA PRINCIPAL DEL EDITOR

En este diagrama se describe el proceso desde la ejecución del programa hasta mostrar la pantalla principal del editor para que el usuario pueda utilizarlo.

El Smart Pic Editor puede ser invocado mediante una línea de comando en la cual se puede incluir opcionalmente el path y nombre de un archivo de programa que se quiere editar, de existir se abre el archivo, caso contrario muestra la pantalla de presentación de Editor y posteriormente la pantalla principal.

La pantalla principal cuenta con los siguientes componentes:

- Barra de Menú.
- Barra de Herramientas.
- Barra de Campos.
- Barra de Estado.

- Area de Edición.

Dentro de la barra de Menú se encuentran las siguientes opciones:

Menú Archivo.- Contiene submenús que permiten al usuario trabajar con archivos.

Menú Edición.- Agrupa las opciones para la manipulación de texto durante la edición de un programa.

Menú Ver.- Permite presentar u ocultar las barras de herramientas, de estado y de campo.

Menú Herramientas.- Contiene herramientas para realizar la corrección de errores, establecer las opciones por omisión del editor y escoger o cambiar el modelo de microcontrolador con el que se está trabajando.

Menú Ayuda.- Permite mostrar el formulario informativo Acerca del Smart Pic Editor.

Dentro de la barra de herramientas se encuentran varios botones de acceso rápido para:

- Abrir.
- Guardar.
- Imprimir.
- Cortar.
- Copiar.
- Pegar.

- Buscar.
- Corregir Programa.
- Opciones de programación.
- Insertar caracteres.
- Escoger Modelo Micro.
- Acerca de...

En la figura 3.1 se indica el diagrama de flujo principal del Editor.

3.2.2 MENU ARCHIVO

El menú Archivo contiene los siguientes submenús:

Abrir.- Permite abrir un archivo, verificando que este aún no este abierto.

La primera vez que se abre un archivo el editor solicita escoger un modelo de microcontrolador a utilizar durante la sesión.

Guardar.- El editor guarda los cambios realizados al archivo de programa que se encuentre abierto.

Por omisión el Editor guarda el archivo con la extensión "asm".

Guardar como.- Permite guardar el archivo de programa que está abierto, para ello presenta un cuadro de diálogo en el cual el usuario puede seleccionar la ubicación y nombre con que se grabará el archivo.

Cerrar.- Cierra el archivo que se encuentra abierto, en caso de que el archivo haya sido modificado, se pregunta al usuario si desea guardar los cambios antes de cerrarlo.4

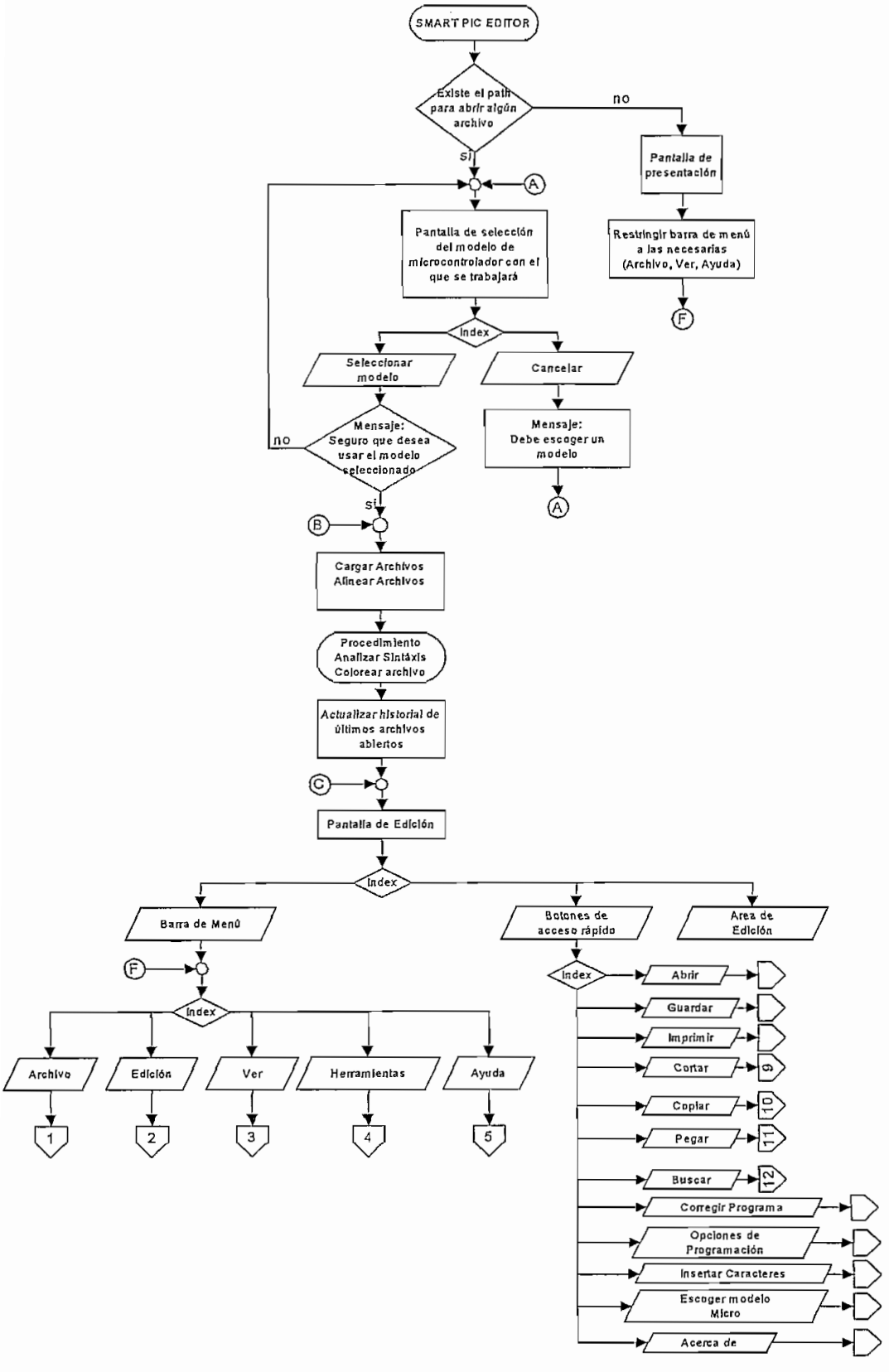


Fig. 3.1 Diagrama de flujo de la pantalla principal del Editor

Imprimir.- Permite imprimir el archivo que se encuentre abierto. El Editor presenta el cuadro de diálogo para imprimir.

Historial de últimos archivos abiertos.- Cada vez que se abren archivos, en esta sección se muestran los cinco últimos archivos recientemente editados.

Salir.- Permite cerrar el Editor previa la confirmación del usuario.

En la figura 3.2 se presenta el diagrama de flujo del menú Archivo.

3.2.3 MENU EDICION

El menú Edición contiene los siguientes submenús:

Cortar.- Mueve el texto seleccionado desde el archivo al portapapeles.

Copiar.- Coloca una copia del texto seleccionado en el portapapeles.

Pegar.- Coloca el contenido de texto del portapapeles en la posición actual del cursor o reemplaza un texto que se encuentra seleccionado.

Borrar.- Elimina el texto seleccionado.

Seleccionar todo.- Selecciona todo el texto del programa que se encuentra abierto.

Buscar.- Presenta un formulario en el que el usuario puede ingresar el texto a buscar.

Buscar siguiente.- Busca y resalta la siguiente cadena que coincida con el patrón de búsqueda especificado por el usuario.

Reemplazar.- Presenta el formulario para reemplazar en el cual el usuario puede especificar el patrón de búsqueda así como la cadena de texto con que se sustituirá en caso de que se encuentre.

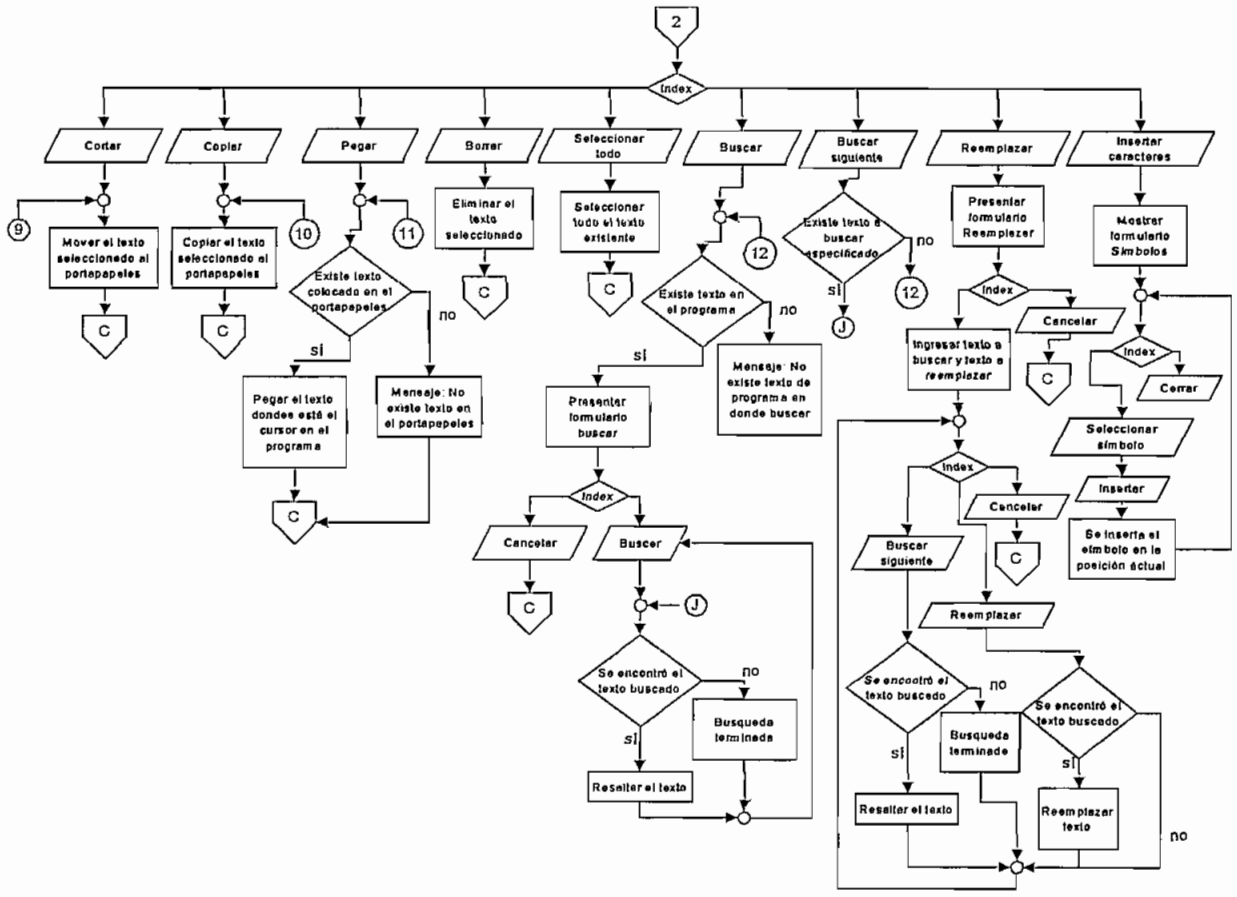


Fig. 3.3 Diagrama de flujo del menú Edición

3.2.4 DIAGRAMA DEL MENU VER

El menú Ver contiene los siguientes submenús:

Barra de herramientas.- Presenta u oculta la barra de herramientas del Editor.

Barra de estado.- Presenta u oculta la barra de estado del Editor.

Barra de campos.- Presenta u oculta la barra de campos del Editor.

En la figura 3.4 se presenta el diagrama de flujo del menú Ver.

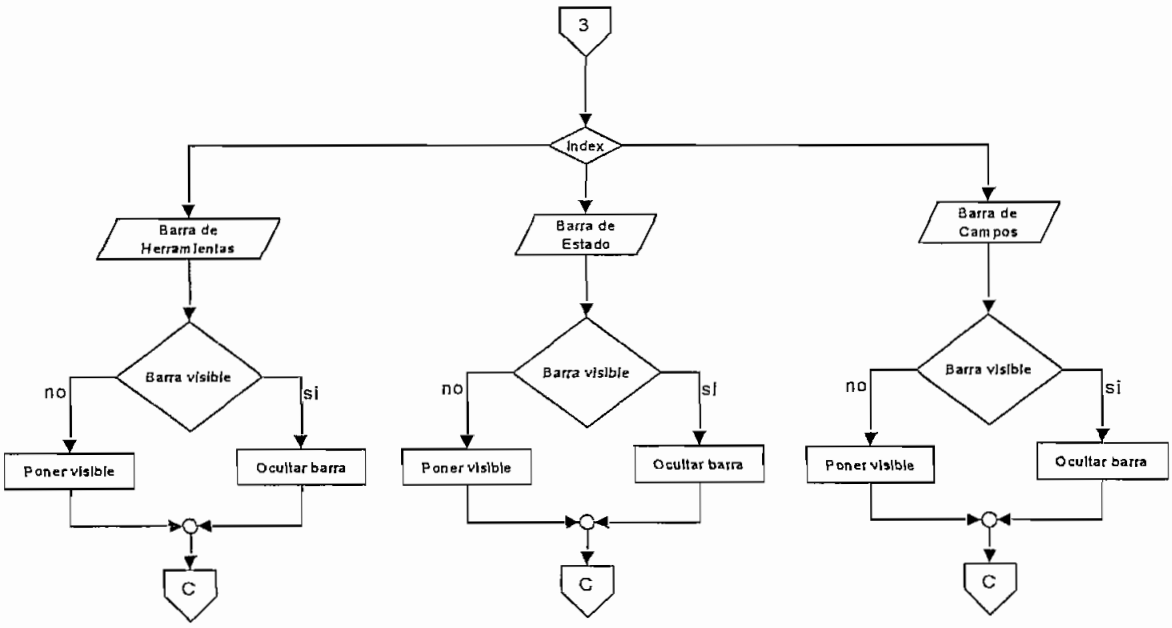


Fig. 3.4 Diagrama de flujo del menú Ver

3.2.5 MENU HERRAMIENTAS

En la figura 3.5 se decribe el diagrama de flujo del menú herramientas, que contiene los siguientes submenús:

Corregir Errores de todo el programa.- Realiza el análisis, coloreado y corrección de las instrucciones del programa, en caso de existir errores se muestra una pantalla que permite al usuario corregir dichos errores o ignorarlos.

Corregir Errores de la línea actual.- Realiza el análisis y coloreado de la línea donde el cursor se encuentra posicionado, en caso de que esta línea tenga errores, presenta la pantalla para corregir o ignorar dichos errores.

Información de la Instrucción actual.- Despliega información de la instrucción presente en la línea donde el cursor se encuentra ubicado.

Opciones.- Presenta un formulario que permite establecer las opciones del editor

Escoger Modelo.- Muestra una pantalla para cambiar el modelo con el que trabajará el usuario durante la sesión. El usuario puede escoger el modelo a partir de un listado de modelos.

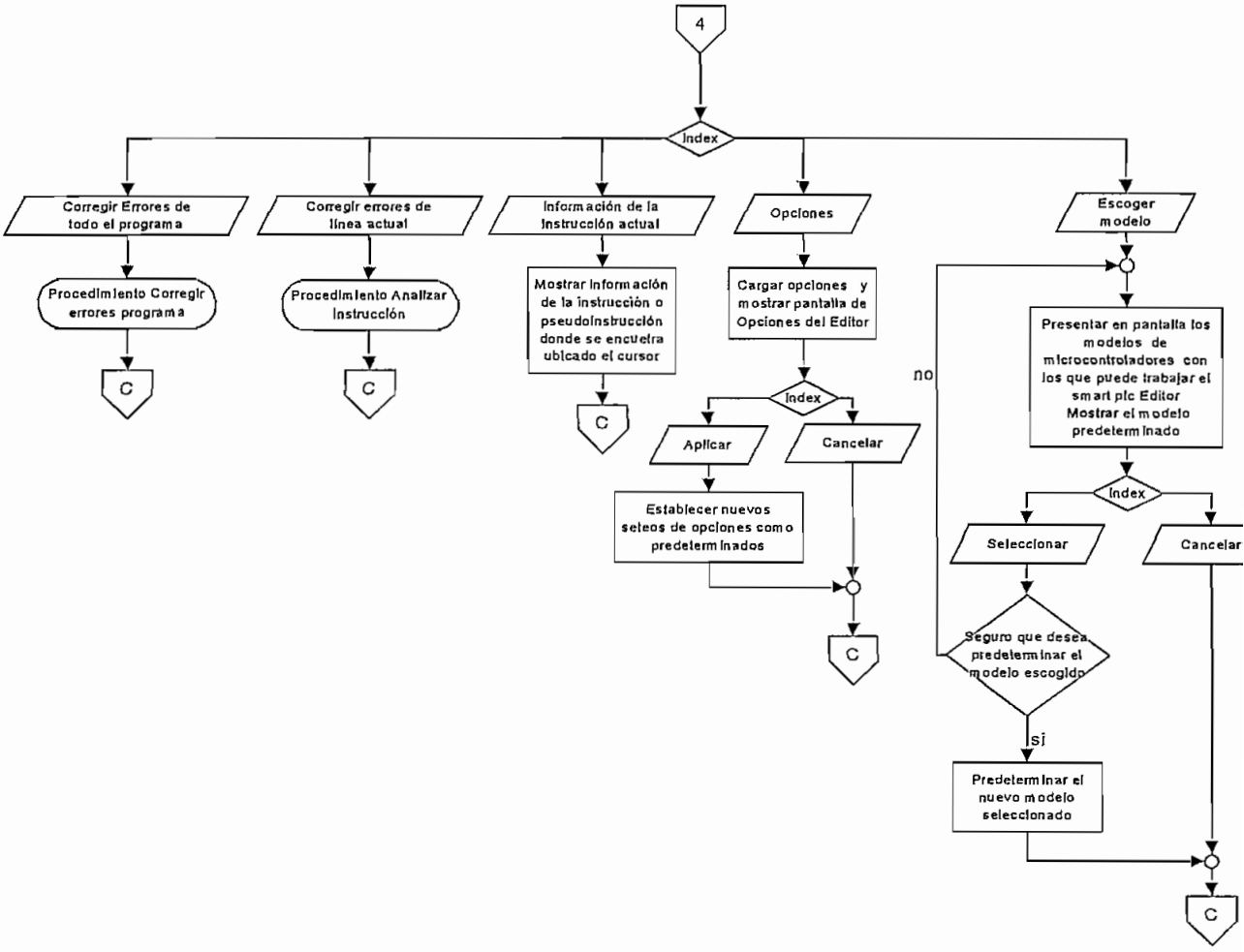


Fig. 3.5 Diagrama de flujo menú Herramientas

3.2.6 ANALISIS DE LA SINTAXIS Y COLOREADO DEL PROGRAMA

En los diagramas de las figuras 3.6 y 3.7 se indica la forma como el Smart Pic Editor analiza la sintaxis de cada una de las líneas del programa. Cada línea es coloreada de acuerdo a las siguientes consideraciones:

Color Negro.- Para indicar una instrucción sintácticamente correcta.

Color Azul.- Para indicar una pseudo instrucción correctamente escrita.

Color Verde.- Para pintar la parte de una línea que corresponde a un comentario.

Color Rojo.- Indica una instrucción o pseudo instrucción escritas de manera incorrecta.

3.2.7 DIAGRAMA DEL PROCEDIMIENTO PARA CORREGIR ERRORES DEL PROGRAMA

En el diagrama de flujo de la figura 3.8 se muestra el proceso de corrección de errores de cada una de las instrucciones que componen el programa, para ello se realiza el análisis de la sintaxis instrucción por instrucción, la instrucción analizada se colorea de acuerdo a lo indicado en el numeral anterior (3.2.6).

En el caso de encontrar una instrucción errónea el Smart Pic Editor presenta una pantalla en la que el usuario puede corregir o ignorar el error, si se ignora dicho error, la corrección continúa desde la siguiente línea.

3.2.8 DIAGRAMA DEL PROCEDIMIENTO PARA ANALIZAR UNA INSTRUCCION

Los diagrama de las figuras 3.9 y 3.10 describen el análisis que el editor realiza para una línea de programa, una vez encontrado un error se visualiza el formulario de corrección de errores y por medio de este se corrige o ignora dicho error.

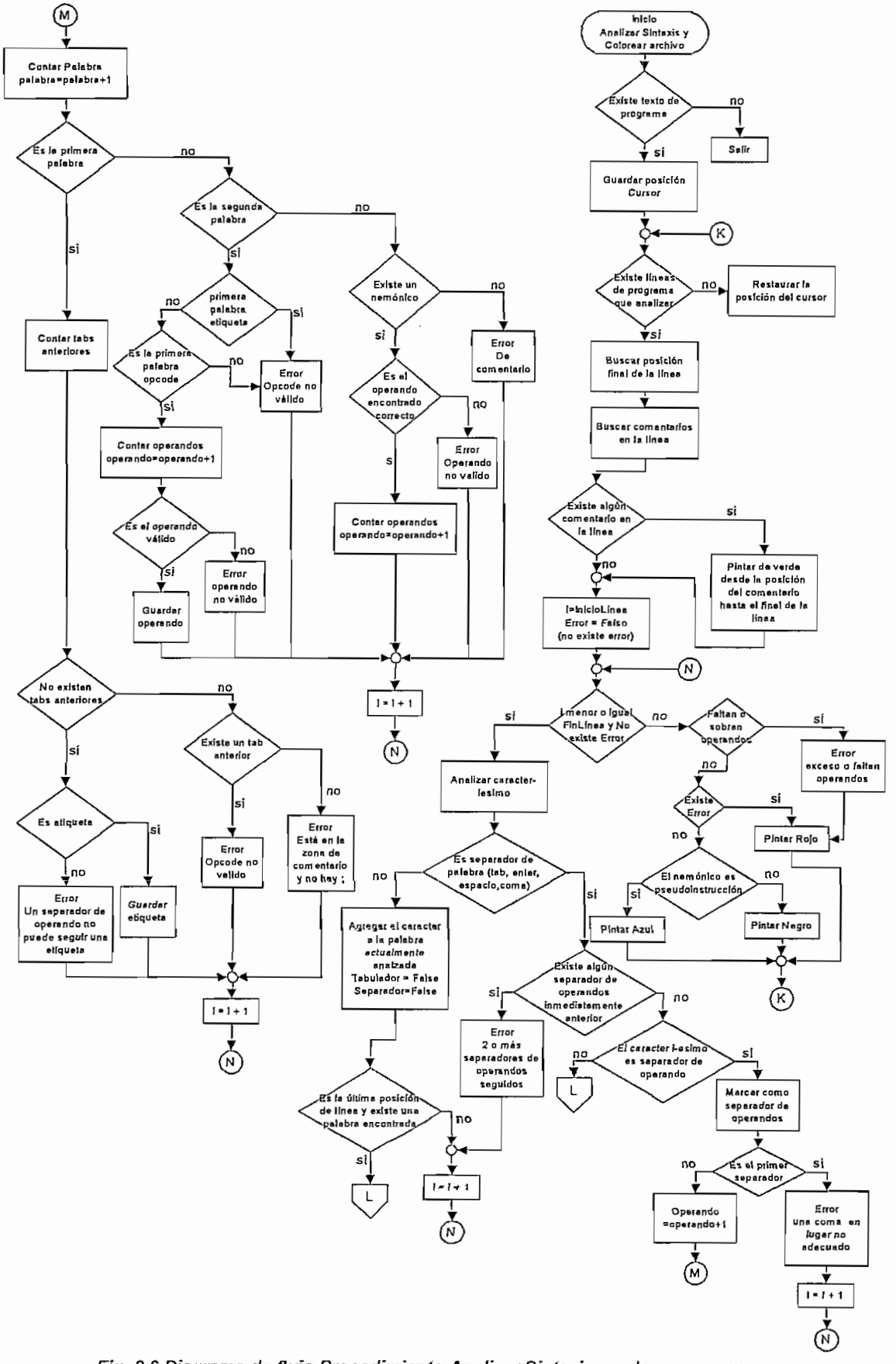


Fig. 3.6 Diagrama de flujo Procedimiento Analizar Sintaxis y colorear programa

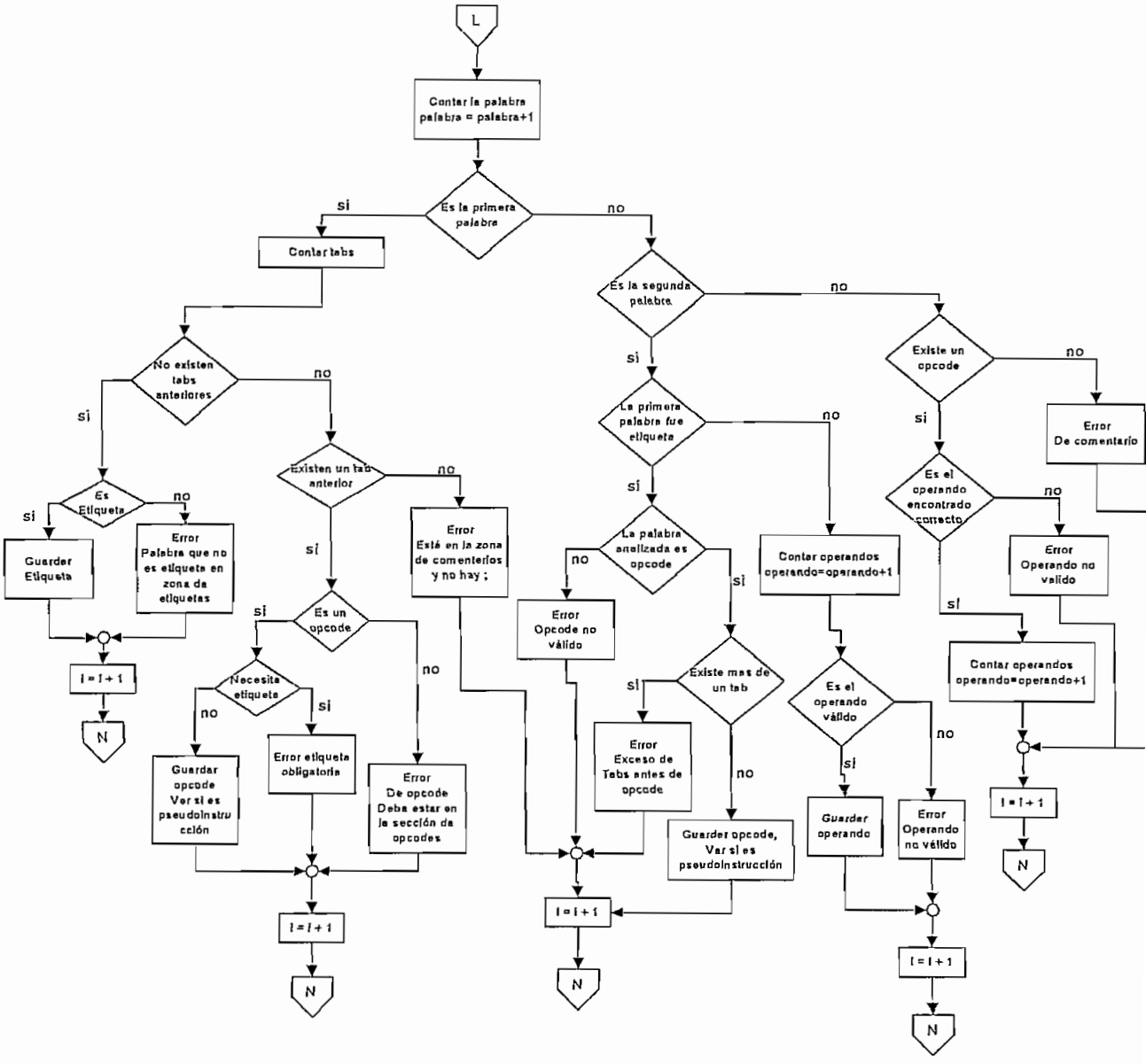


Fig. 3.7 Continuación Diagrama de flujo del procedimiento para analizar la sintaxis y colorear el programa

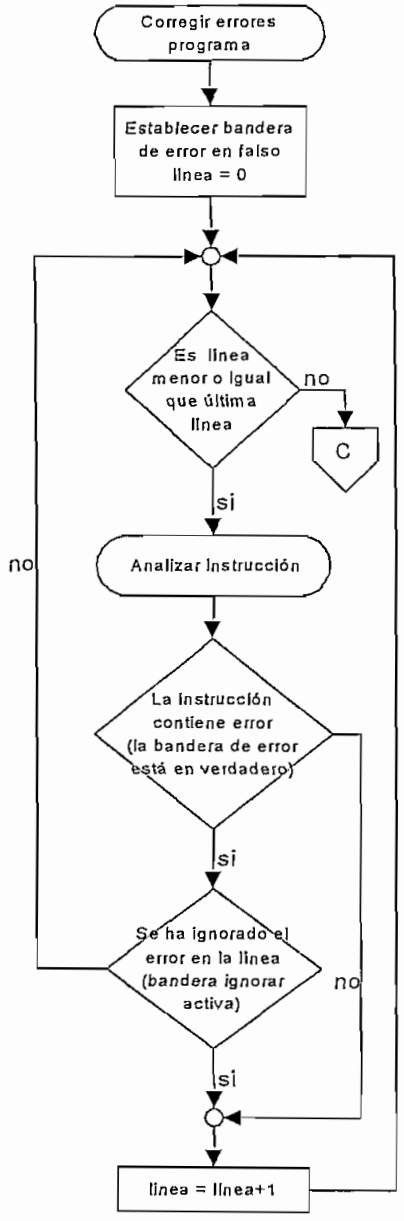


Fig. 3.8 Diagrama de flujo del procedimiento para corregir errores del programa

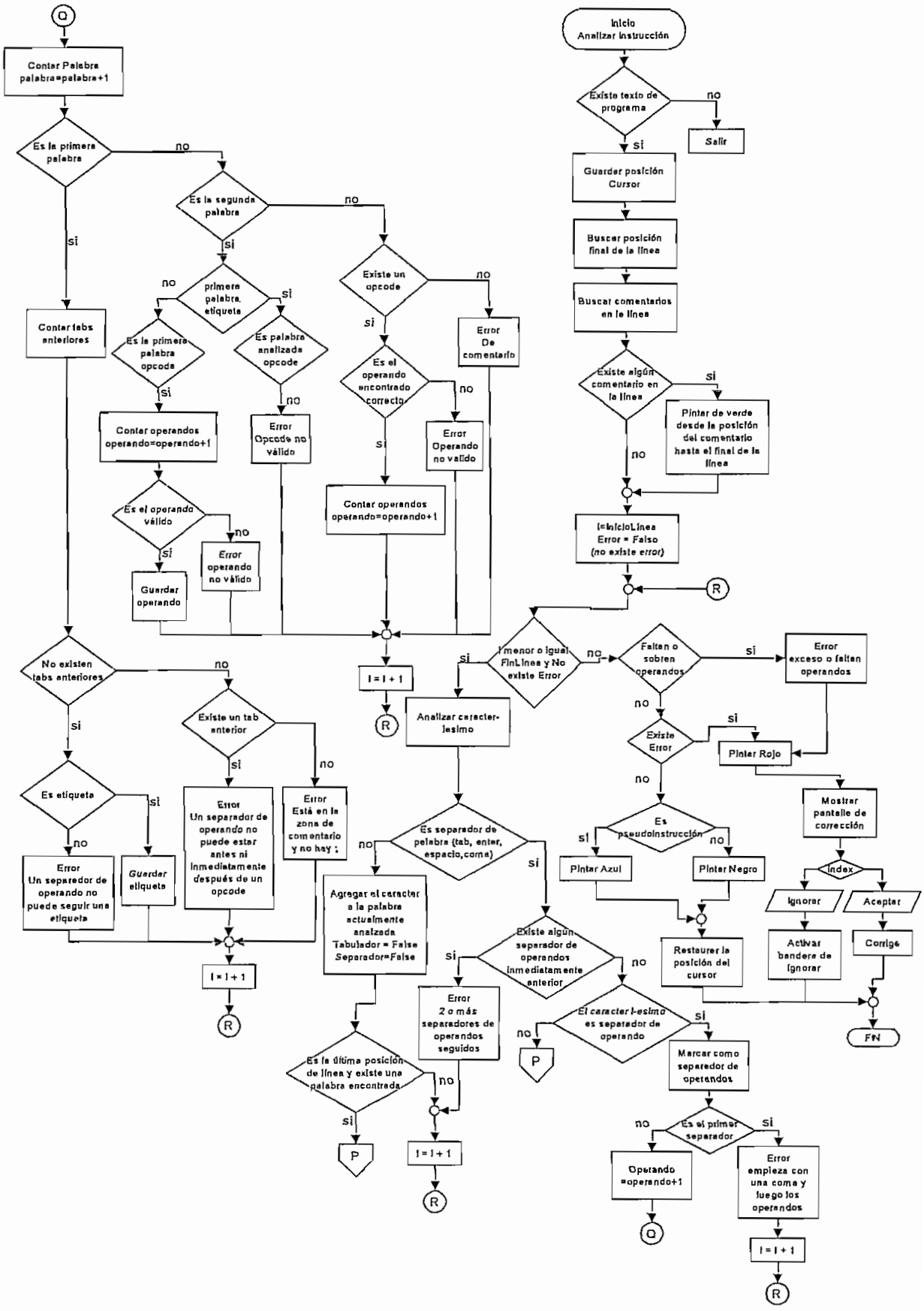


Fig. 3.9 Diagrama de flujo del procedimiento para Analizar una Instrucción

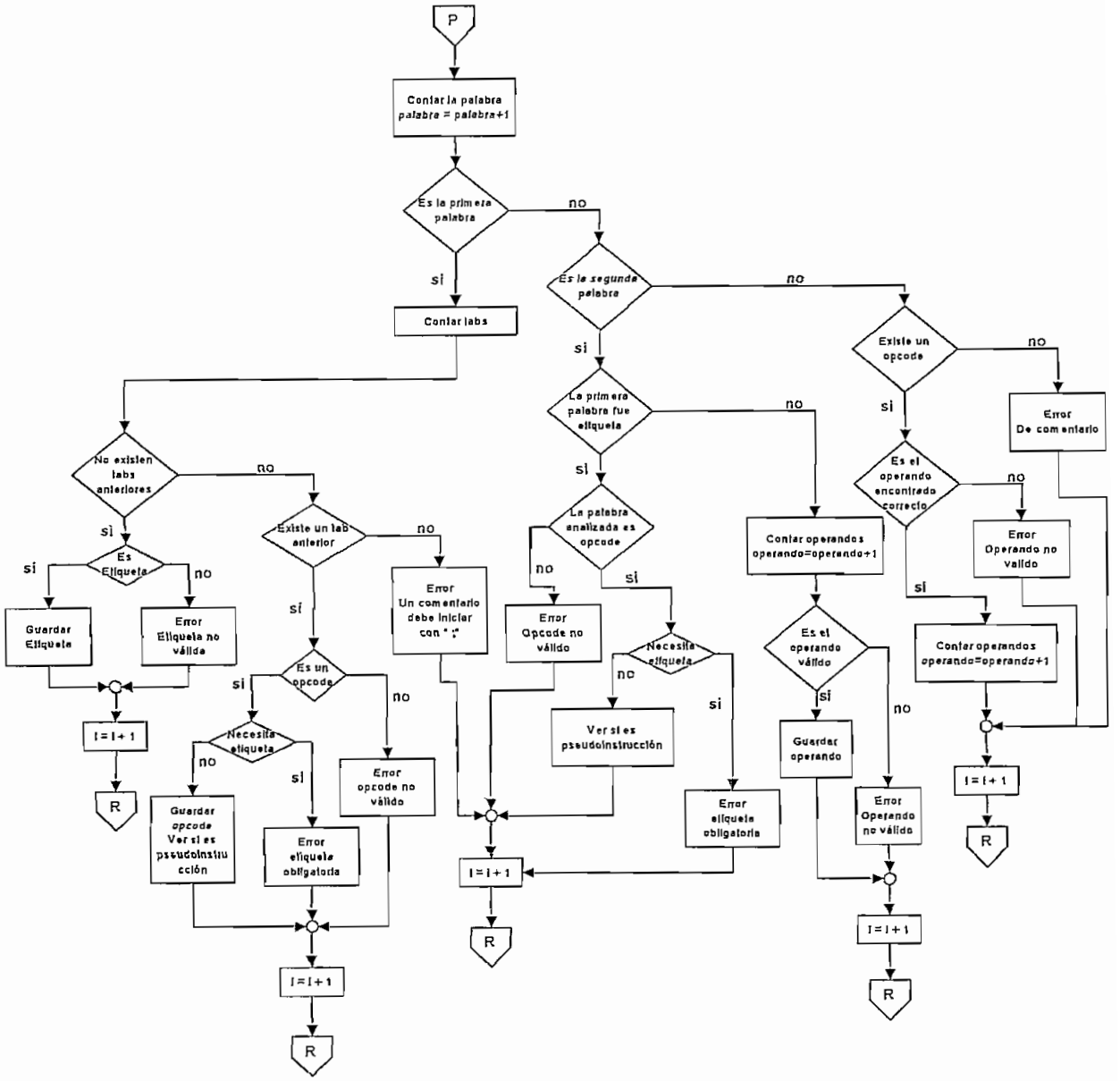


Fig. 3.10 Continuación Diagrama de flujo Procedimiento Analizar Instrucción

3.2.9 DIAGRAMA DEL MENU AYUDA

El menú Ayuda despliega el formulario Acerca de... con información breve del Smart Pic Editor. El diagrama indicado en la figura 3.11 indica los eventos que se realizan al acceder al menú Ayuda.

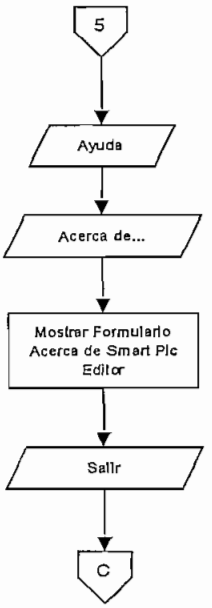


Fig. 3.11 Diagrama de flujo menú Ayuda

3.3 EL ADMINISTRADOR DE LA BASE DE DATOS

3.3.1 MOTORES DE BASES DE DATOS

Existen algunos motores o herramientas software utilizados para implementar bases de datos, se puede citar:

- ORACLE.
- SQLServer.
- ACCESS.
- FOX PRO.
- SYBASE.

ACCESS de Microsoft es un motor de base de datos que nos permite crear, manipular o borrar tablas. Como herramienta de programación permite crear consultas, reportes y macros, pero no puede crear ejecutables de la aplicación.

3.3.2 COMPONENTES DE LA BASE DE DATOS

De acuerdo a las características de la base de datos que requiere el Smart Pic Editor, y se las puede encontrar en el capitulo 2 numeral 2.4.1 se ha implementado en Microsoft Access la base de datos que considera la estructura mostrada en el diagrama de la figura 3.12 de acuerdo al modelo entidad- relación.

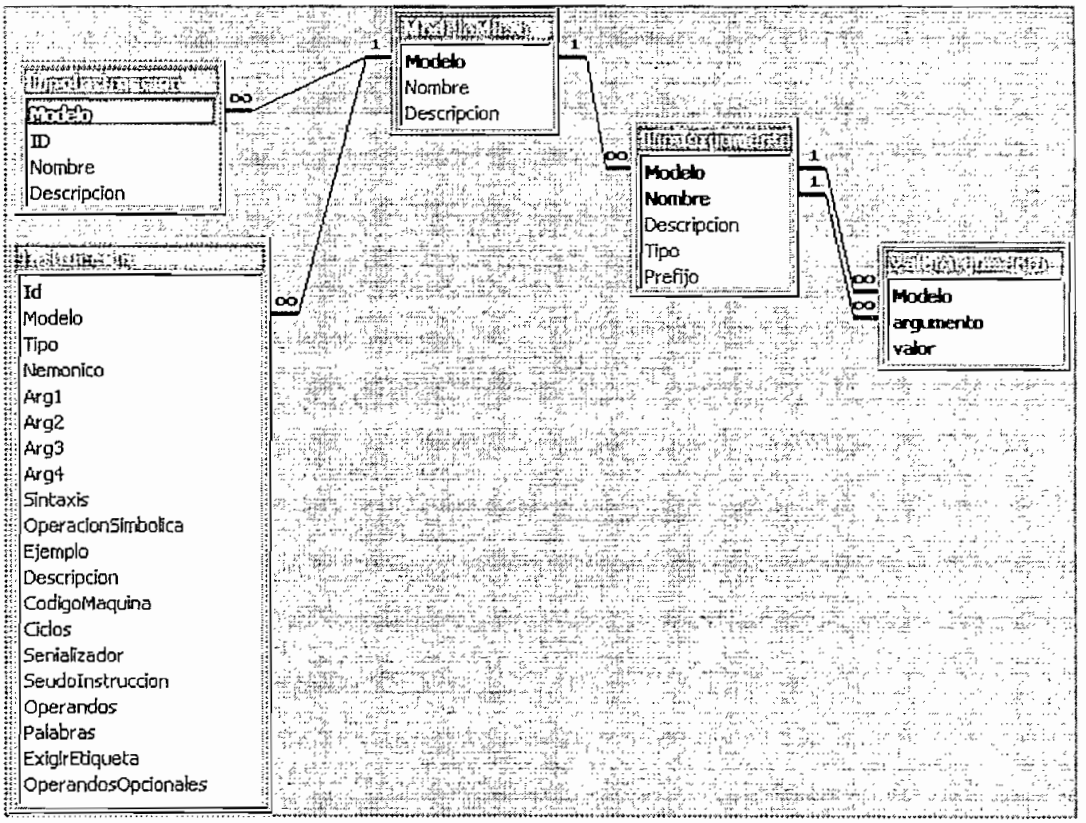


Fig.3.12 Diagrama de la Base de Datos implementada en Access

3.3.3 COMPONENTES DEL ADMINISTRADOR DE LA BASE DE DATOS "ADMIN PIC EDITOR"

3.3.3.1 Formularios

FrmPortada (frmSplash.frm)

Corresponde a la pantalla de presentación general de la Herramienta de Software "Admin Pic Editor" desarrollada para Administrar la base de datos.

Frm Modelo Pic (frmModeloPic.frm)

Constituye la pantalla principal donde el usuario puede agregar o editar un modelo de microcontrolador.

FrmTiposInstruccion (frmTiposInstruccion.frm)

Muestra la pantalla en la que se puede definir los "Tipos de instrucciones" en las que podríamos clasificar a las instrucciones de un modelo previamente establecido. Estos datos aparecerán en el formulario Instrucciones en un campo denominado Tipo de Instrucción a manera de una lista donde el usuario podrá seleccionar.

FrmTipoArg (frmTipoArg.frm)

Muestra la pantalla por medio de la cual el usuario puede agregar o editar los parámetros correspondientes a los operandos que acompañan a los opcodes o pseudo-opcodes de un modelo de microcontrolador escogido previamente.

FrmInstrucciones (frmInstrucciones.frm)

Muestra la pantalla para que el usuario pueda agregar o editar los parámetros correspondientes a las instrucciones de un modelo de microcontrolador escogido previamente.

FrmAbout (frmAbout.frm)

Muestra información breve sobre el Admin Pic Editor.

3.3.3.2 Módulos

MAdminPrincipal(mAdminPrincipal.bas)

En este módulo se incluye las variables globales utilizadas en el administrador tales como:

SBASEPIC.- Variable que almacena la ubicación y nombre de la base de datos que utiliza el administrador.

SMODELOMICRO.- Almacena el código del modelo de microcontrolador que se ha escogido en la pantalla de definición de modelos y cuyo conjunto de instrucciones se va ha editar.

SNOMBREMODELO.- Almacena el nombre del modelo de microcontrolador que se ha escogido en la pantalla de definición de modelos y cuyo conjunto de instrucciones se va ha editar.

También se incluye el procedimiento principal que da inicio a la ejecución de Admin Pic Editor, mostrando la portada del Administrador y luego presenta la pantalla para la definición del modelo de Microcontrolador.

3.4 DIAGRAMAS DEL ADMINISTRADOR

En esta sección se describen los diagramas de flujo que indican la lógica con que opera el administrador de la base de datos.

3.4.1 DIAGRAMA DE LA PANTALLA PRINCIPAL DEL "ADMIN PIC EDITOR"

A continuación se presenta un diagrama que describe el proceso que sigue el programa cuando se lo ejecuta, mostrando la pantalla de presentación del administrador y posteriormente la pantalla para la definición de modelos de microcontroladores, así como los botones de acceso a las demás pantallas del administrador.

La pantalla para la definición de modelo permite realizar las siguientes tareas:

Agregar.- Permite añadir un nuevo modelo a la base de datos.

Editar.- Habilita los campos nombre y descripción para que el usuario pueda modificarlos pudiendo luego guardar o cancelar los cambios realizados.

Eliminar.- Permite eliminar un modelo de la base de datos previa la confirmación de esta operación. Cabe anotar que para eliminar modelos deben haber sido previamente eliminados los tipos de instrucciones, tipos de operandos y las instrucciones correspondientes.

Salir.- Cierra el Administrador previa la confirmación del usuario.

Opciones de Navegación.- Adicionalmente se puede navegar entre los registros de tipos de microcontroladores existentes.

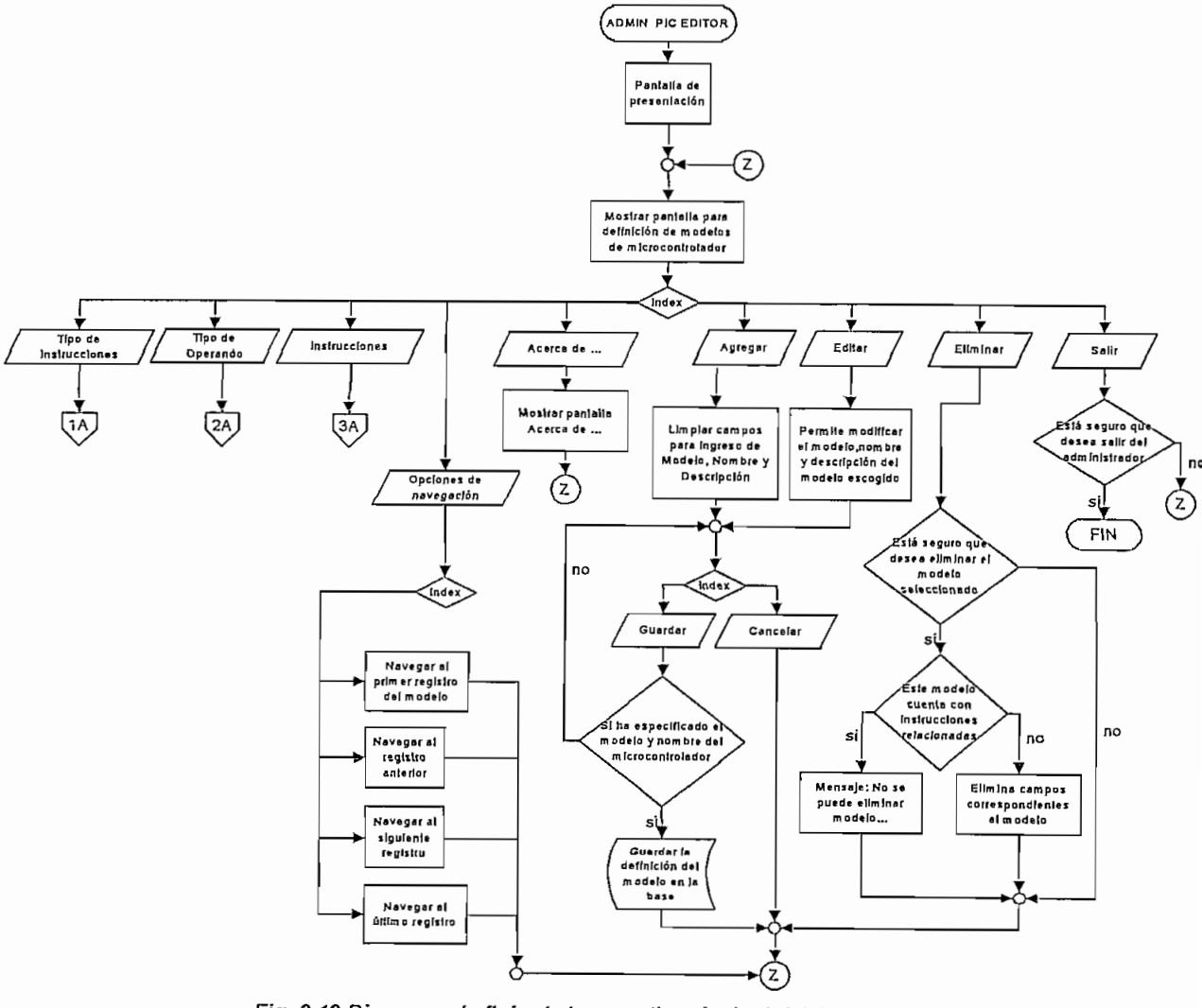


Fig. 3.13 Diagrama de flujo de la pantalla principal del Administrador

3.4.2 DIAGRAMA DE LA PANTALLA DE DEFINICION DE TIPOS DE INSTRUCCIONES

Una vez definido el modelo de microcontrolador, se puede definir los tipos de instrucciones o grupos en los que se asocian las instrucciones; para ello se debe seleccionar el botón "Tipos de Instrucción".

En la pantalla para la definición de tipos de Instrucción, se puede agregar, editar, eliminar o navegar entre los registros establecidos .

Agregar.- Limpia los campos y los pone listos para el ingreso de un nuevo tipo de instrucción. El usuario puede guardar o cancelar esta operación.

Editar.- Mediante esta opción se habilita la modificación de un tipo de instrucción; luego de realizar los cambios pertinentes, el usuario puede guardarlos o cancelarlos.

Eliminar.- Elimina un tipo de instrucción previa la confirmación del borrado. Para eliminar los tipos de instrucción deben haberse borrado previamente las instrucciones asociadas a ese tipo o bien pueden ser cambiadas a otro tipo.

Cerrar.- Cierra la pantalla y retorna a la pantalla de definición de modelos.

Opciones de Navegación.- Adicionalmente se puede navegar entre los registros de tipos de instrucción existentes.

El diagrama de figura 3.14 describe la pantalla para la definición de tipos de instrucciones.

3.4.3 DIAGRAMA DE LA PANTALLA PARA DEFINIR LOS TIPOS DE OPERANDOS DE LAS INSTRUCCIONES

Si se presiona el botón tipos de operandos se puede acceder a la pantalla para definir los tipos de operandos que pueden manejar las instrucciones.

En la pantalla para la definición de tipos de operandos, se puede agregar, editar, eliminar o navegar entre los registros establecidos. Ver la figura 3.16.

Agregar.- Limpia los campos y los pone listos para el ingreso de un nuevo tipo de operando. El usuario puede guardar o cancelar esta operación.

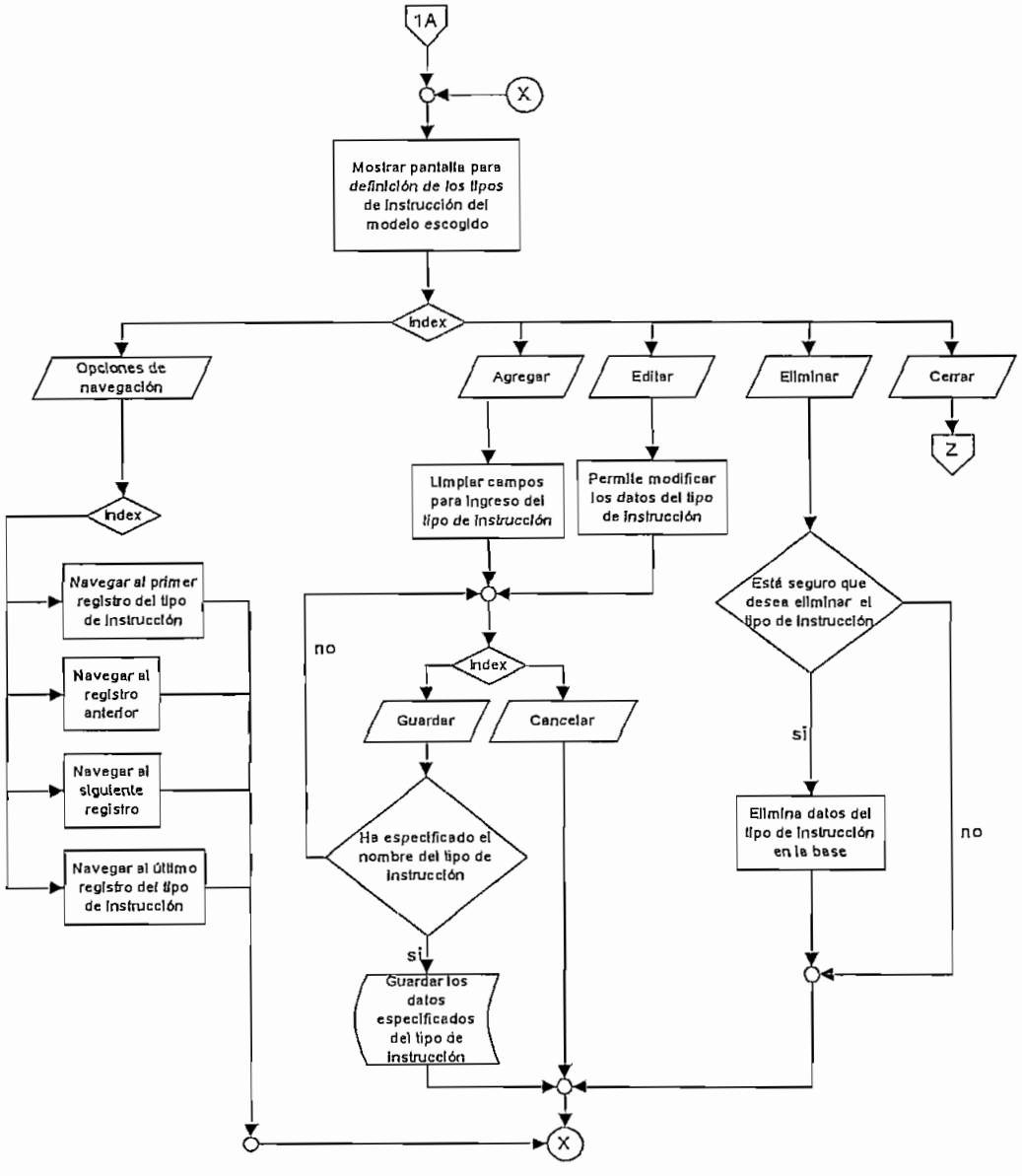


Fig. 3.14 Diagrama de flujo tipos de instrucción

Luego de haber ingresado el tipo de operando y si este corresponde a una "lista" de valores, entonces se pueden añadir o eliminar los valores permitidos.

Editar.- Mediante esta opción se habilita la modificación de un tipo de operando; luego de realizar los cambios pertinentes, el usuario puede guardarlos o cancelarlos. En caso de que el tipo de operando sea una lista de valores, entonces se pueden añadir o eliminar los valores permitidos.

Eliminar.- Elimina un tipo de operando previa la confirmación del borrado. Para eliminar los tipos de operando deben haberse borrado previamente las instrucciones que utilicen operandos de este tipo.

Cerrar.- Cierra la pantalla y retorna a la pantalla de definición de modelos.

Opciones de Navegación.- Adicionalmente se puede navegar entre los registros de tipos de operandos existentes.

3.4.4. DIAGRAMA DE LA PANTALLA DE DEFINICION DE INSTRUCCIONES

Finalmente indicamos el diagrama de flujo que muestra la pantalla donde se puede ingresar cada una de las instrucciones que están asociadas a un modelo específico de microcontrolador.

En la pantalla para la definición de instrucciones, se puede agregar, editar, eliminar o navegar entre los registros de instrucciones de un modelo específico.

Agregar.- Limpia los campos y los pone listos para el ingreso de una nueva instrucción. El usuario puede guardar o cancelar esta operación.

En caso de que la instrucción tenga al menos un operando, es necesario que se haya definido previamente el tipo de operando para luego asociarlo al momento de definir la instrucción.

Editar.- Mediante esta opción se habilita la modificación de una instrucción; luego de realizar los cambios pertinentes, el usuario puede guardarlos o cancelarlos.

Eliminar.- Elimina una instrucción previa la confirmación del borrado.

Cerrar.- Cierra la pantalla y retorna a la pantalla de definición de modelos.

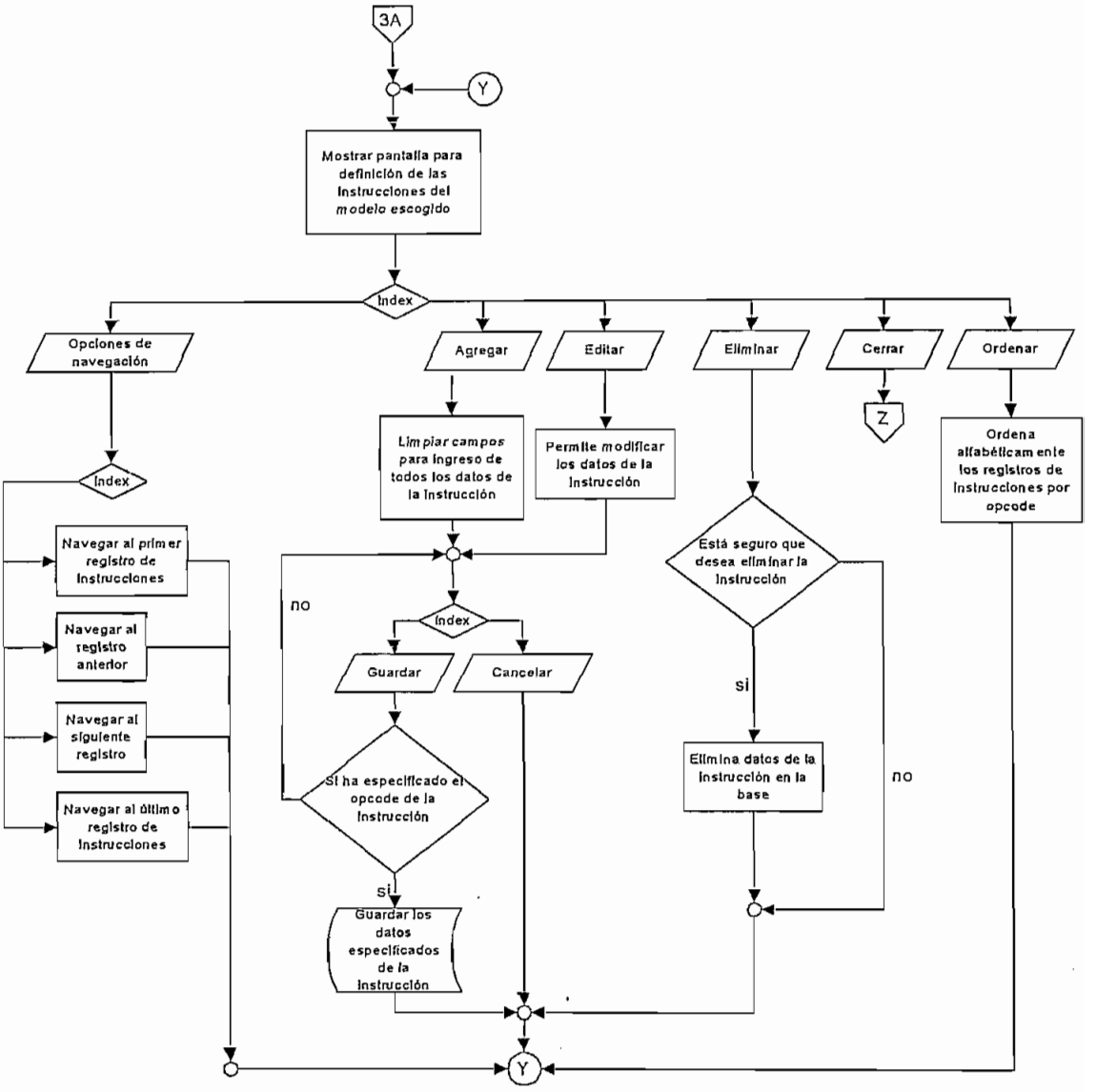


Fig. 3.15 Diagrama de flujo para la definición de las instrucciones

Opciones de Navegación.- Adicionalmente se puede navegar entre los registros de instrucciones del modelo existentes.

En el diagrama de la figura 3.15 se detalla la secuencia de pasos para la definición de instrucciones de un modelo.

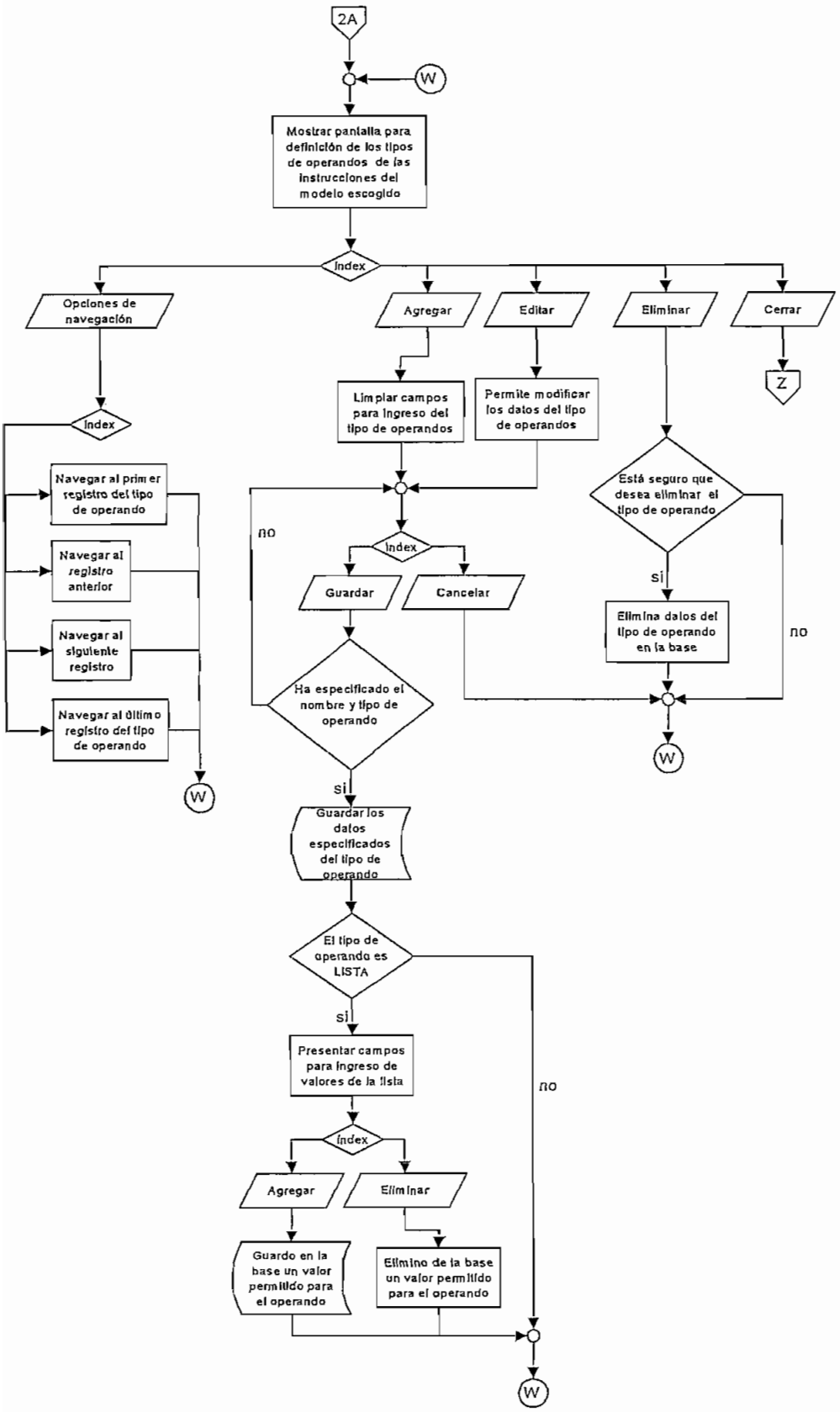


Fig. 3.16 Diagrama de flujo para la definición de los tipos de operando

CAPITULO 4. PRUEBAS

4.1 CREACION DE UNA BASE DE DATOS

La base de datos que utiliza el Smart Pic Editor y físicamente es el archivo "BasePic.mdb" puede ser modificada por la persona encargada de la administración de la base de datos usando el "Admin Pic Editor"; tanto el editor como el administrador son herramientas independientes, por lo que una vez modificada la base de datos con el administrador, esta debe ser copiada y servir de reemplazo de la base existente en el editor, que tiene el mismo nombre y formato.

Para el ingreso de un nuevo modelo de microcontrolador y su conjunto de instrucciones usando el módulo de administración "Admin Pic Editor"; los pasos generales a seguir son los siguientes:

1. Definir un modelo de microcontrolador.

2. Definir los tipos de instrucciones en que se clasificaran.

3. Definir los operandos genéricos para las instrucciones estos pueden ser de los siguientes tipos:

   - Constante.- Cuando el operando constituye una palabra definida.

   - Lista.- Cuando el operando acepta un conjunto de valores definidos.

   - Variable.- Pudiendo ser una dirección, etiqueta, es decir una valor que varíe.

   El establecimiento de estos tipos de operandos se lo ha hecho para facilitar el procesamiento de esta información por el Editor, tanto en las ayudas en pantalla como en la corrección de errores.

4. Ingresar cada una de las instrucciones del microcontrolador llenando cada uno de los registros de datos.

Para la realización de cada uno de estos pasos por parte de la persona que administrará la base de datos, de una manera sencilla, se ha establecido una pantalla para cada actividad, cada pantalla cuenta con los botones necesarios y la correspondiente activación o desactivación de los mismos, de acuerdo a las necesidades, se han definido tablas de registros navegables por el usuario.

4.1.1 DEFINICION DE UN MODELO DE MICROCONTROLADOR

A continuación y partiendo de la base de datos con sus registros vacíos se ingresa el conjunto de instrucciones del microcontrolador PIC 16X84 con 35 instrucciones.

En la figura 4.1 se puede observar la pantalla para agregar un nuevo modelo de microcontrolador.

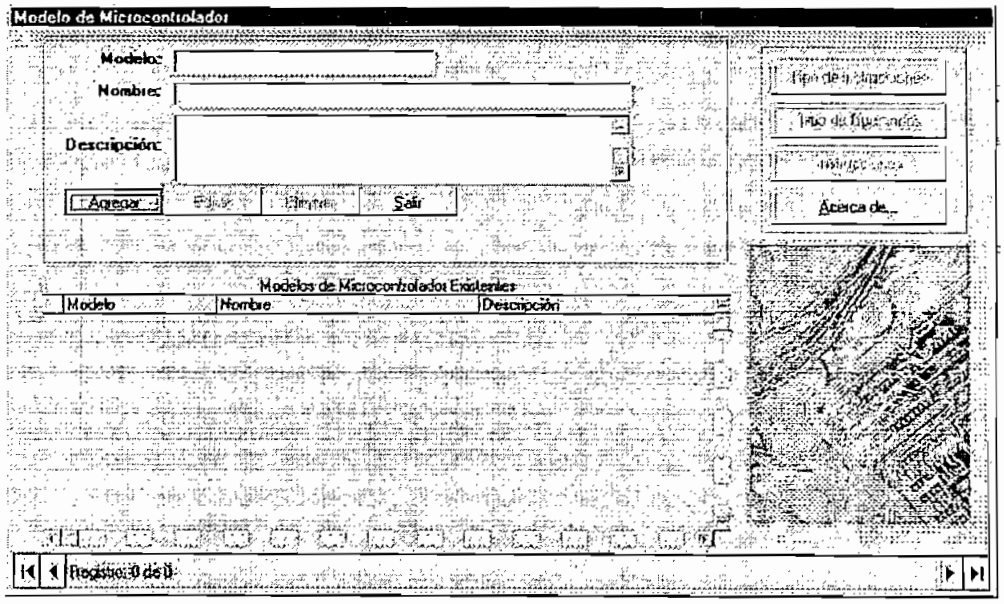


Fig. 4.1 Pantalla para agregar un nuevo modelo de microcontrolador.

En virtud de no existe ningún modelo presente los botones para Editar, Eliminar, así como los botones "Tipo de Instrucciones", "Tipo de Operandos" e "Instrucciones" no se encuentran habilitados. Así mismo se ve que la barra de navegación de registros indica cero registros existentes.

Si se presiona el botón agregar se puede agregar el nuevo modelo, ver figura 4.2.

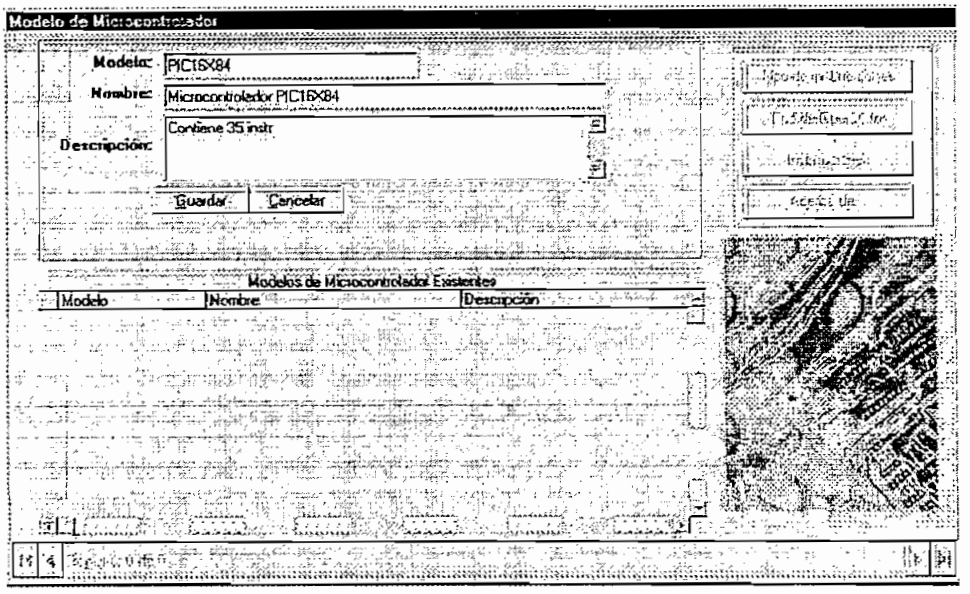


Fig. 4.2 Pantalla que muestra la agregación de un nuevo modelo de microcontrolador.

Una vez llenado los campos modelo, nombre y descripción se puede guardar o cancelar el ingreso de estos datos si se ha guardado se presenta la pantalla de la figura 4.3.

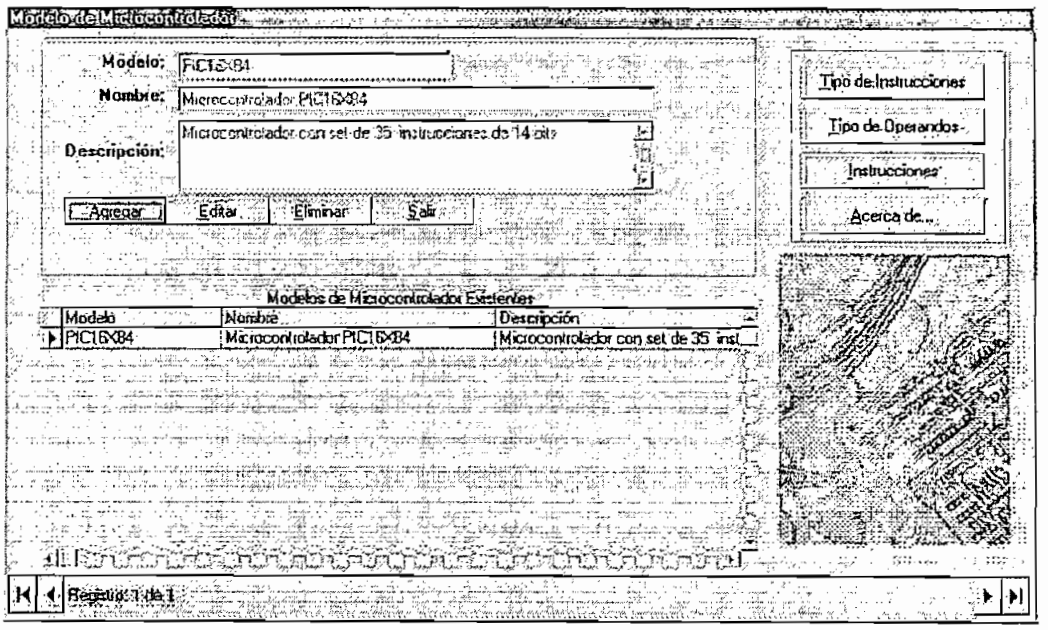


Fig. 4.3 Pantalla con modelo PIC16X84 agregado.

4.1.2 DEFINICION DE LOS TIPOS DE INSTRUCCIONES

Agregado un nuevo modelo se procede a establecer los tipos de instrucciones, para ello se hace un click en el botón Tipo de instrucciones ver la figura 4.4.

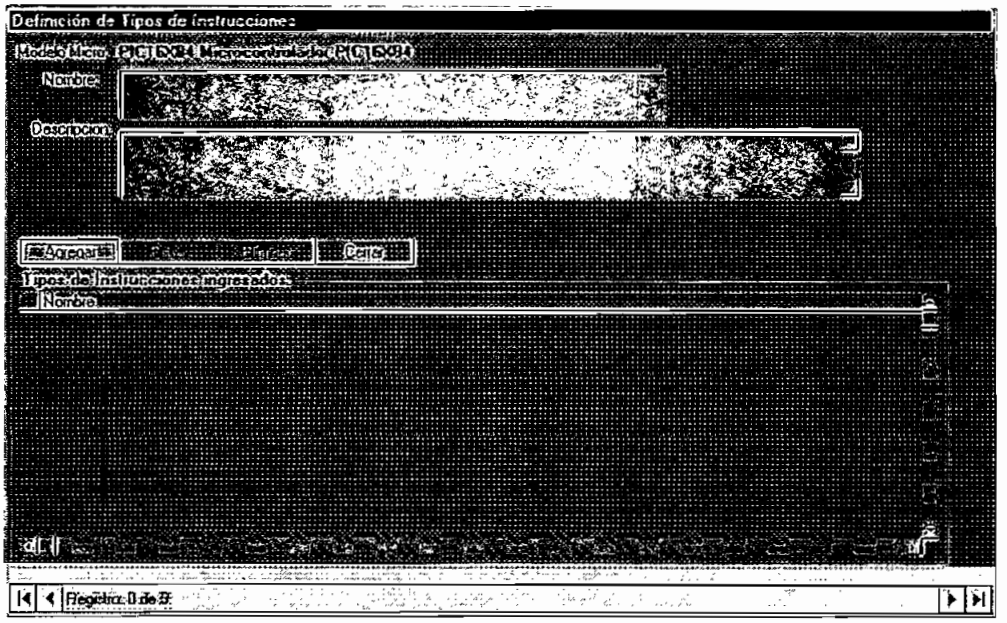


Fig. 4.4 Pantalla para agregar tipo de Instrucciones.

Se escribe un tipo de instrucciones ver la figura 4.5.

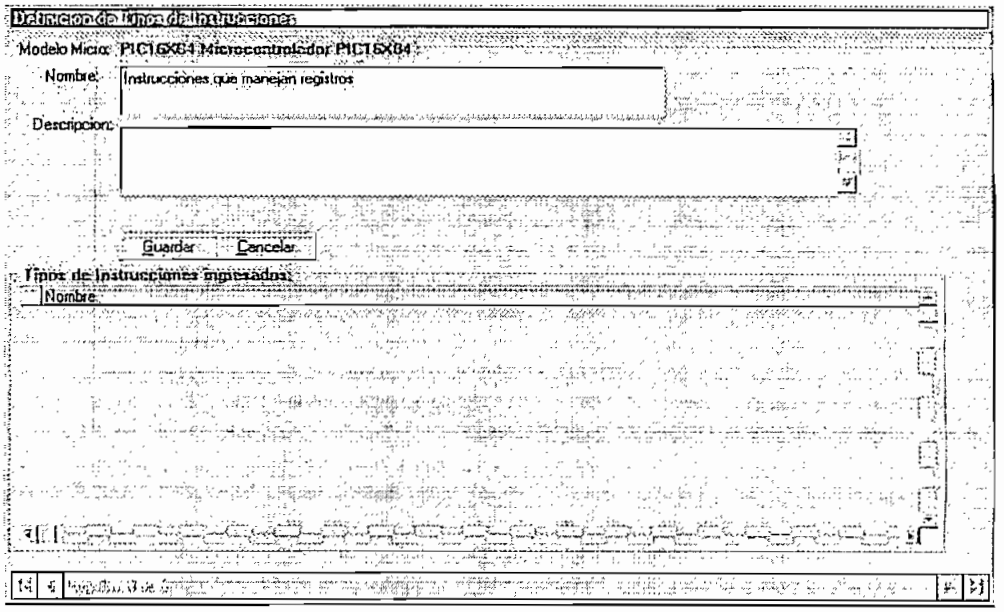


Fig. 4.5 Pantalla que muestra que se está agregando un tipo de Instrucciones.

Se almacena el tipo de instrucción añadida ver la figura 4.6.

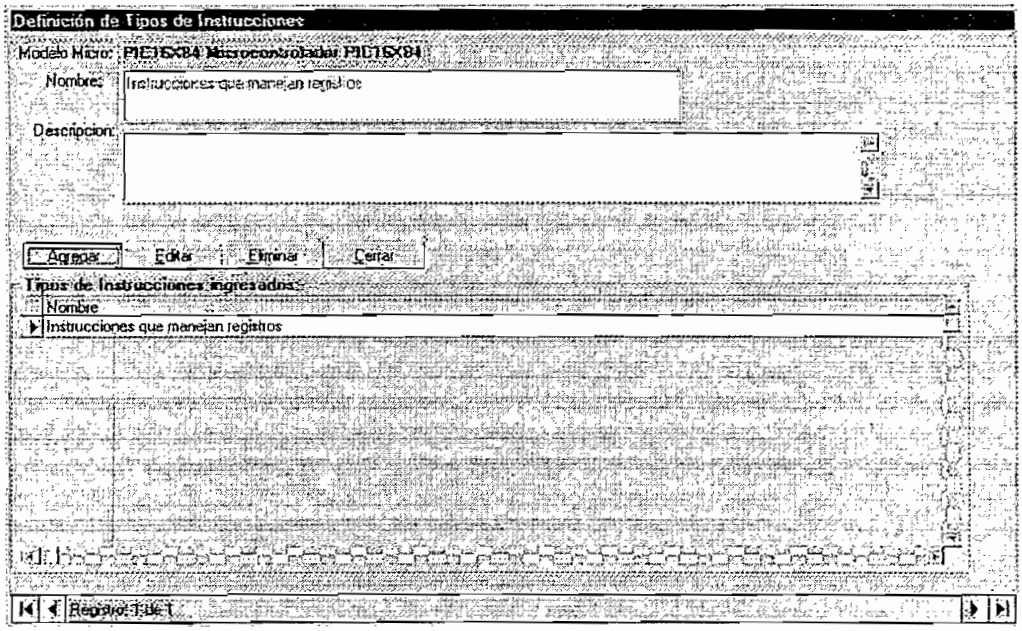


Fig. 4.6 Pantalla con un tipo de instrucción añadida

4.1.3 DEFINICION DE LOS TIPOS DE OPERANDOS

Se definen los operandos genéricos de las instrucciones a ser añadidas, para ello se hace click en el botón Tipos de operandos, ver la figura 4.7.

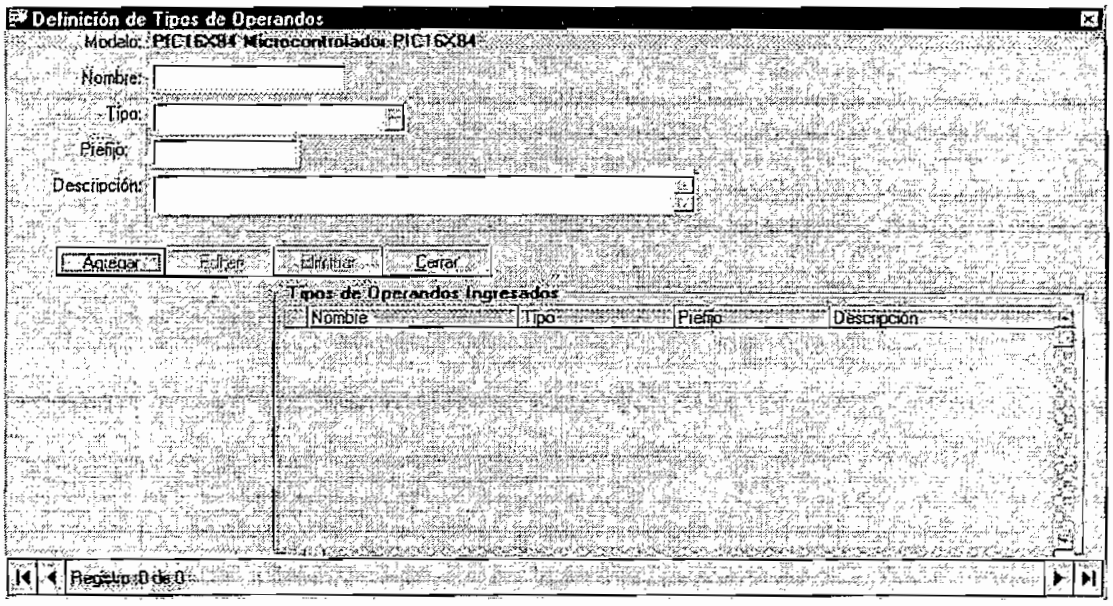


Fig. 4.7 Pantalla donde se definen los tipos de operandos

Una vez ingresado los tipos de operandos la pantalla se ve como en la figura 4.8.

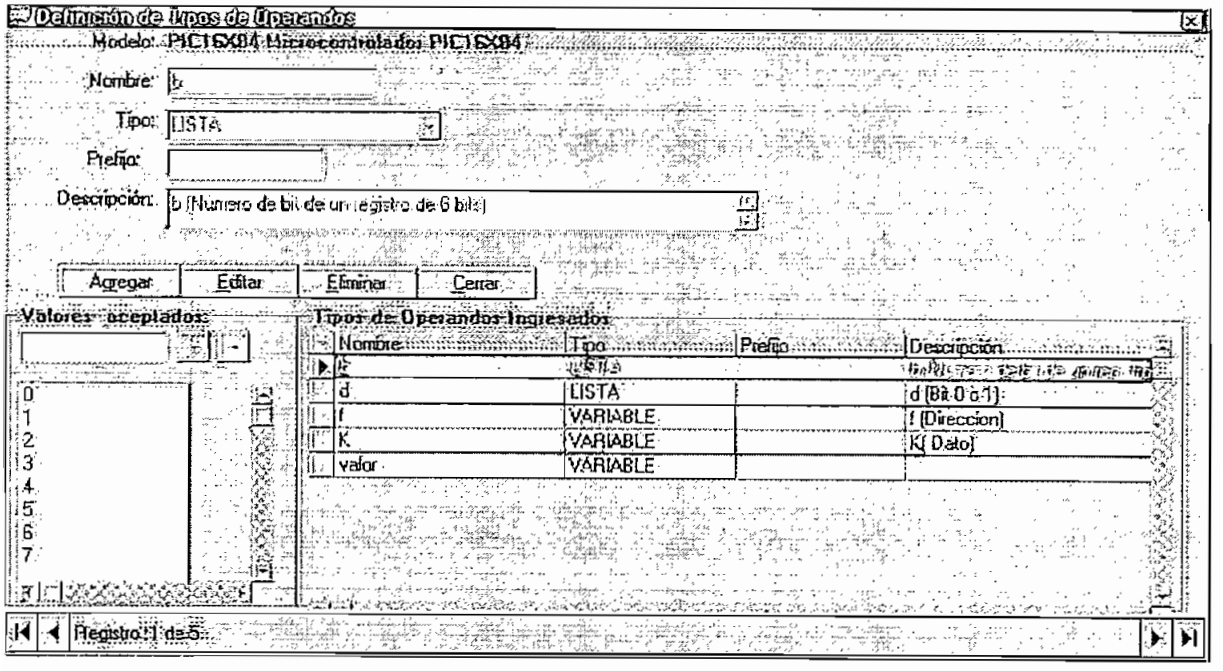


Fig. 4.8 Pantalla con tipos de operandos definidos

4.1.4 DEFINICION DE LAS INSTRUCCIONES

Para ingresar las instrucciones, se hace click en el botón "Instrucciones" y se muestra la pantalla de la figura 4.9, se hace click en el botón "Agregar" y los campos de datos son habilitados.

En esta pantalla, algunos campos se presentan como una lista de opciones, donde se pueden seleccionar valores, así se tiene: el campo "tipo" que corresoponde al tipo de Instrucciones del que formará parte la Instrucción, el campo "Seudoinstrucción?" que permite definir si se está introduciendo datos de una instrucción o de una directiva del ensamblador, el campo "Exigir Etiqueta" que permite definir si la etiqueta es opcional u obligatoria, los campos "Operandos" que permite la selección de los tipos de operandos definidos previamente el campo "Opcionales" que permite definir si son aceptados uno o más operandos, este es generalmente utilizado para definir directivas del ensamblador, ver las figuras 4.10 y 4.11.

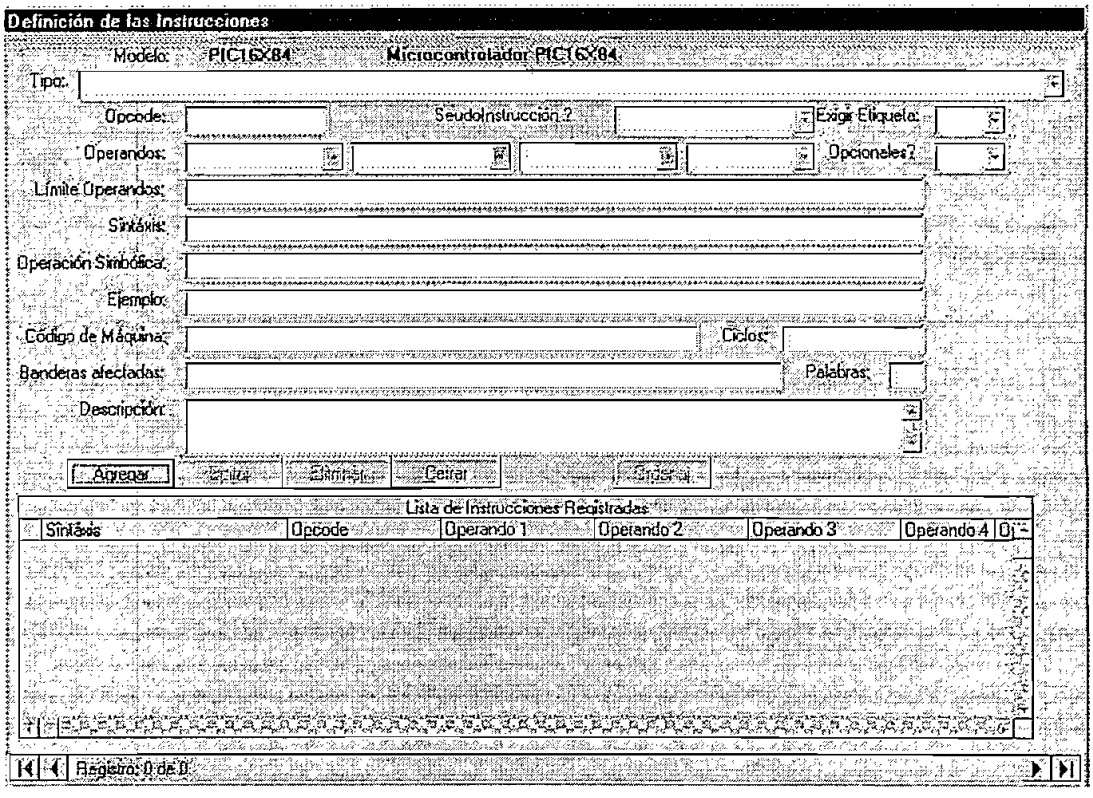


Fig. 4.9 Pantalla para agregar instrucciones

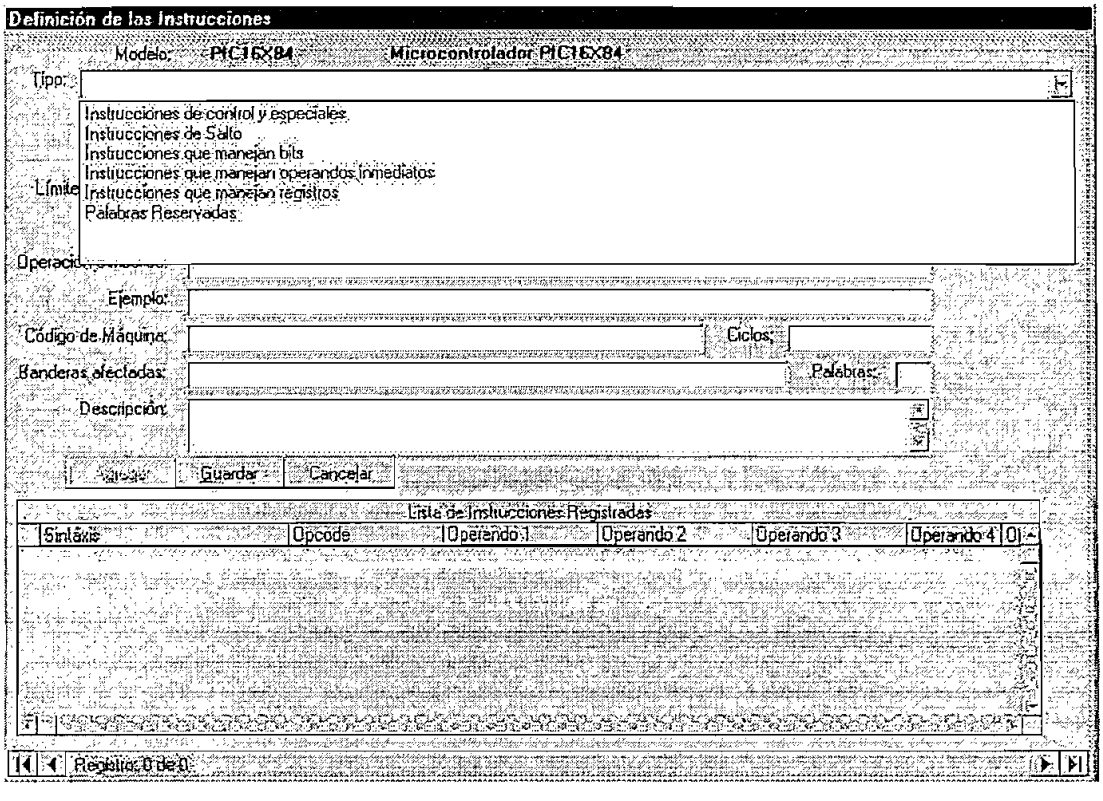


Fig. 4.10 Pantalla para agregar instrucciones con lista de tipos ingresada

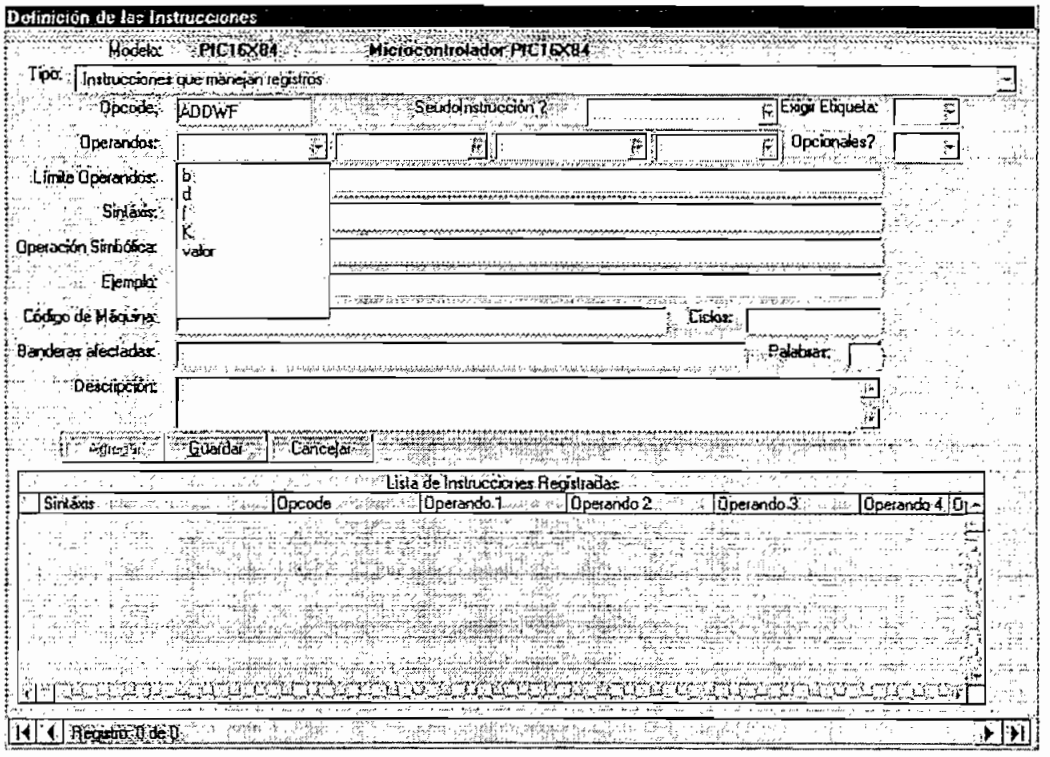


Fig. 4.11 Pantalla para agregar instrucciones con lista de operandos ingresada

Una vez ingresado y almacenados los datos de las instrucciones se tiene la pantalla de la figura 4.12.

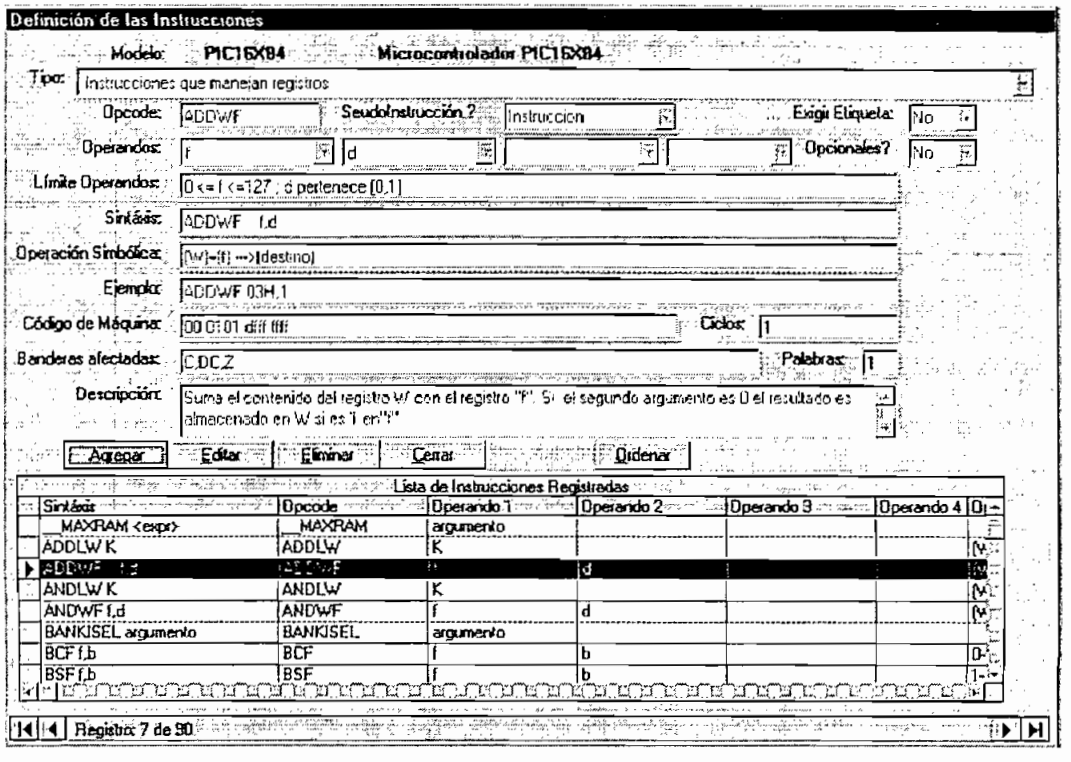


Fig. 4.12 Pantalla con las instrucciones ingresadas

4.2 MODIFICACIONES QUE SE PUEDEN REALIZAR

4.2.1 EDITAR REGISTROS EXISTENTES EN LA BASE DE DATOS.

Ingresadas las instrucciones y guardadas, cada uno de los datos que se muestran en las pantallas: modelo de microcontrolador, tipos de instrucciones, tipos de operandos e instrucciones aparecen inhabilitados y de color gris.

Las modificaciones que se pueden realizar en cada una de estas pantallas se hacen a través del botón "editar" que habilita los campos de datos de cada pantalla y el usuario puede cambiar dichos datos y guardar los cambios realizados o cancelar dicha edición.

4.2.2 ELIMINAR REGISTROS DE LA BASE DE DATOS.

Así mismo se puede eliminar:

- Registros de instrucciones
- Registros de tipos de operandos
- Registros de tipos de instrucciones
- Y registros del modelo de microcontrolador que contenía dicha información.

Debido a que la base de datos es relacional si se requiere eliminar algún tipo de operando, se debe eliminar primeramente todas las instrucciones que utilizan este tipo de operando, así mismo si se requiere eliminar un tipo de instrucción, se debe eliminar todas las instrucciones que forman parte de este tipo, o asignarlas a otro tipo que se vaya a conservar, finalmente para eliminar un modelo de microcontrolador se debe eliminar todos los datos de: instrucciones, tipos de operandos y tipos de instrucciones, el editor sin embargo indica mensajes de no poder eliminar datos que estén relacionados, como en el ejemplo al intentar borrar el tipo de operando "b" se muestra el mensaje de la figura 4.14.

Cada vez que se va ha eliminar una instrucción, tipo de argumento, tipo de instrucción o modelo se presenta un mensaje de confirmación como el de la figura 4.13.

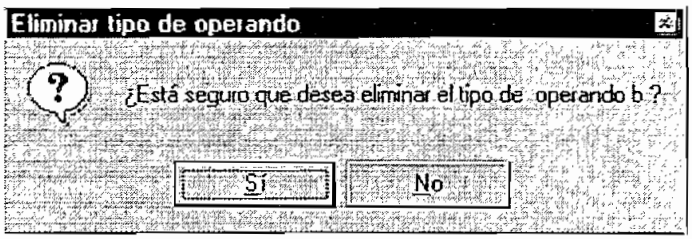


Fig. 4.13 Mensaje para eliminación de un tipo de operando

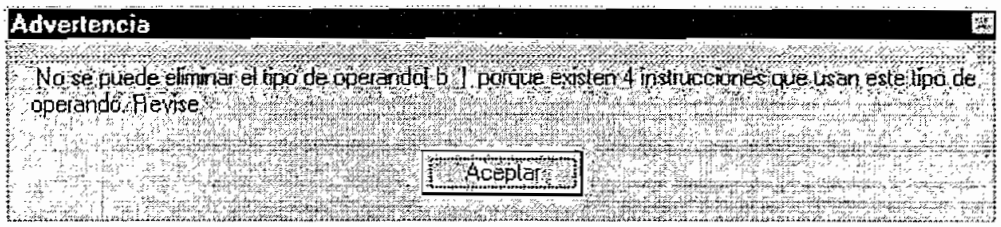


Fig. 4.14 Mensaje de que no se puede eliminar el operando

4.3 AYUDAS DE LA BASE DE DATOS EN LA EDICION

4.3.1 OPCIONES DE AYUDA PARA EL INGRESO DE INSTRUCCIONES

Una vez que se ejecuta el Editor, se solicita al usuario escoger el modelo de microcontrolador con el que va ha trabajar en la presente sesión y se abre una pantalla de edición. Ver figura 4.16.

El siguiente paso sería configurar la opción de ayuda para el ingreso de instrucciones en la edición de un programa, si no se desea trabajar con la configuración por omisión, el mostrar o no los mnemónicos durante la edición es configurable, para ello en la barra de menú, se selecciona el menú "herramientas", luego "opciones de programación" y se despliega una ventana en la que se indica si se va ha usar el teclado y/o el ratón para la selección de un mnemónico cuando se encuentre desplegada la ayuda; por omisión el Editor

ingresa con las opciones de ayuda para ingreso de instrucciones indicadas en la Figura 4.15.

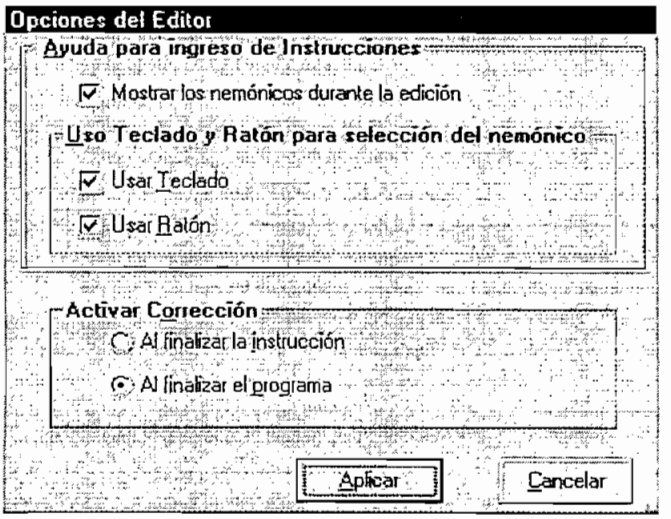


Fig. 4.15 Pantalla de opciones del Editor

Se edita a continuación un programa pequeño para ilustrar las ayudas presentes en el editor, para ello se ha seleccionado el programa indicado en la página 196 del libro MICROCONTROLADORES PIC Diseño práctico de aplicaciones de los autores José Angulo & Ignacio Angulo.

Como se puede apreciar en la secuencia de pantallas de las figuras 4.17, 4.21, 4.23, 4.24 y 4.28, si se encuentra en los campos de opcode y si se presiona la letra inicial del opcode el editor muestra una lista de todos los posibles opcodes y directivas del ensamblador (pseudo-opcodes) posibles.

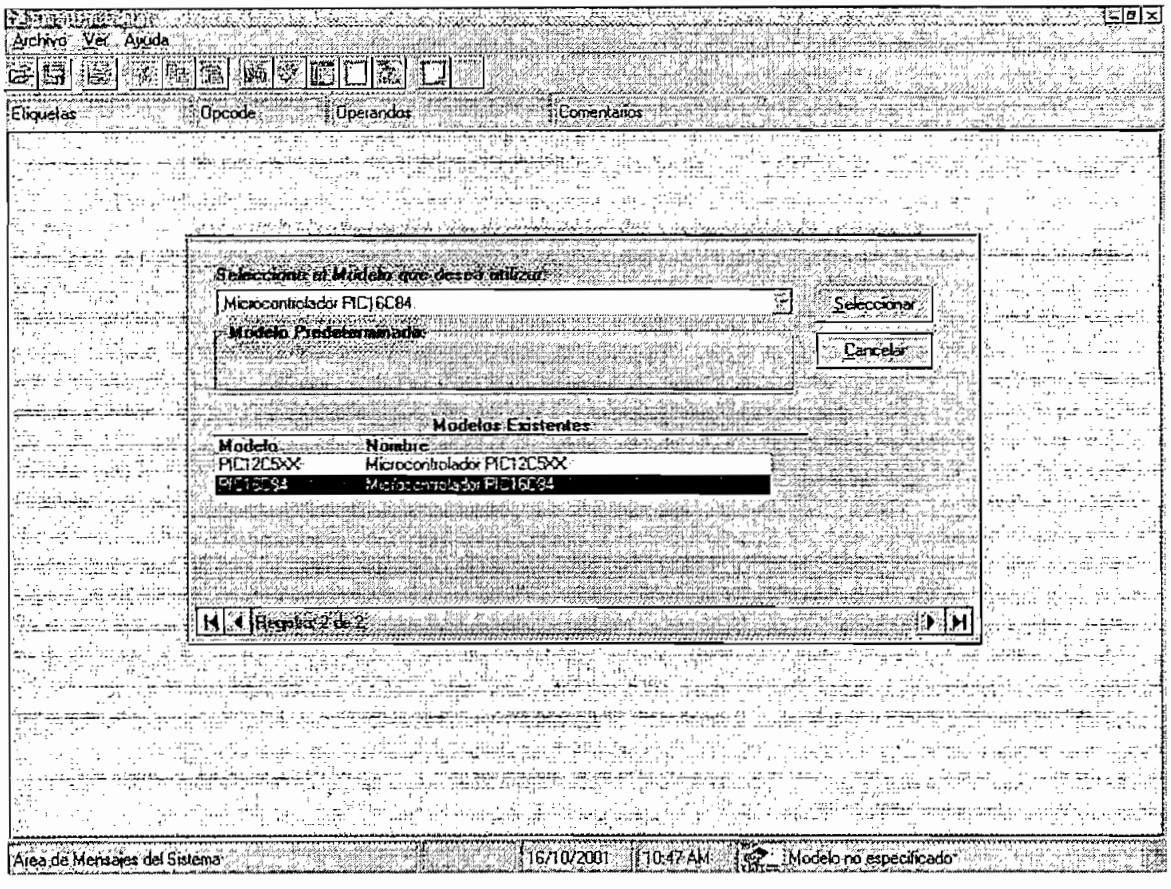


Fig. 4.16 Pantalla de selección del modelo de microcontrolador

4.3.2 AYUDA PARA LA SELECCIÓN DE UN OPCODE

Para la selección del opcode correspondiente se puede utilizar el teclado o el ratón; si se usa el ratón, al realizar un click sobre una opción de la lista desplegada, se escoge dicha opción, la misma que se resaltará, si se desea aceptar se debe hacer un doble click sobre esta opción. De la misma manera, si se utiliza el teclado, una vez presente la lista con la tecla de flecha hacia abajo o la tecla de flecha hacia la derecha se puede escoger la primera opción luego con las teclas de flechas: hacia arriba, abajo y derecha se puede navegar entre las opciones; para aceptar una opción escogida se debe presionar la flecha hacia la derecha o presionar la tecla enter.

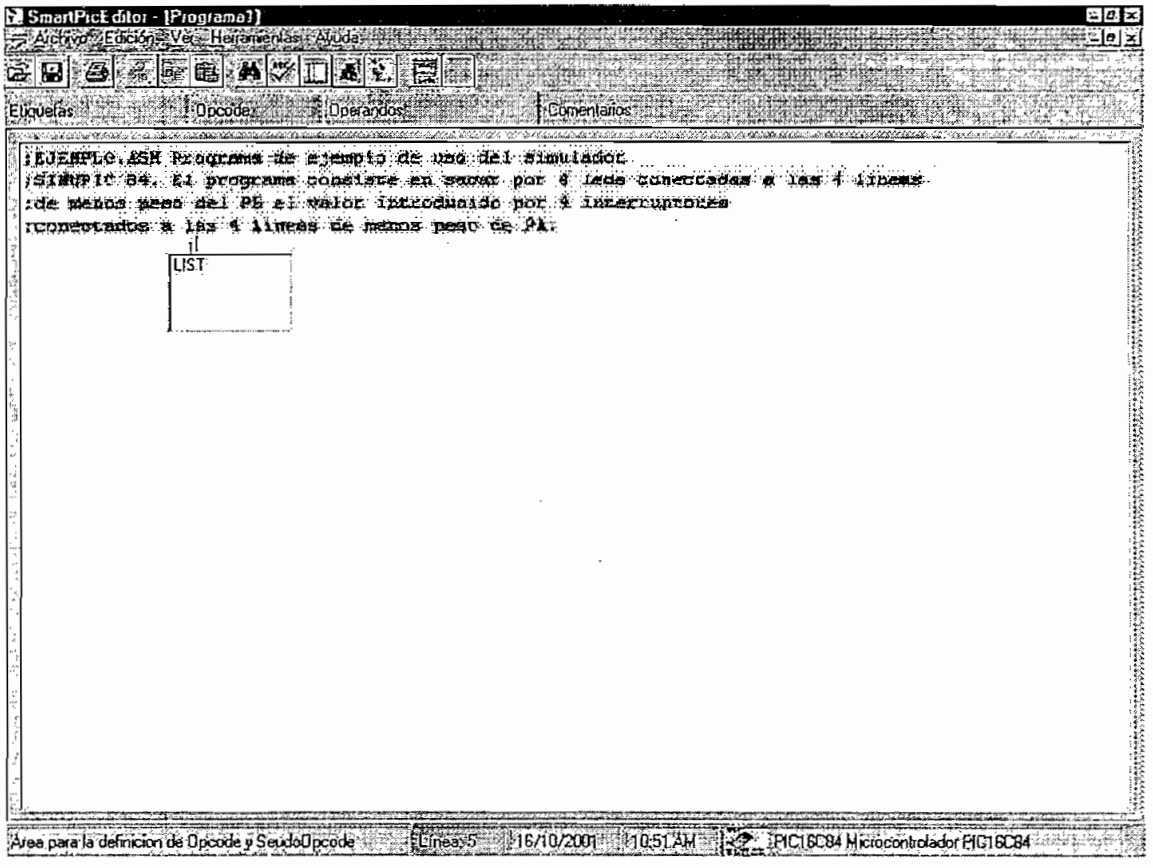


Fig. 4.17 Ingreso de una línea de directivas del programa

4.3.3 AYUDAS PARA EL INGRESO DE LOS OPERANDOS

Una vez aceptado el opcode, se despliega una lista de selección o ingreso del primer operando; si se selecciona o escribe el operando y se presiona la tecla enter y la instrucción requiere más operandos se presenta otra lista donde se puede ingresar o seleccionar el segundo operando y así hasta terminar la escritura de la instrucción. Ver la figura 4.18.

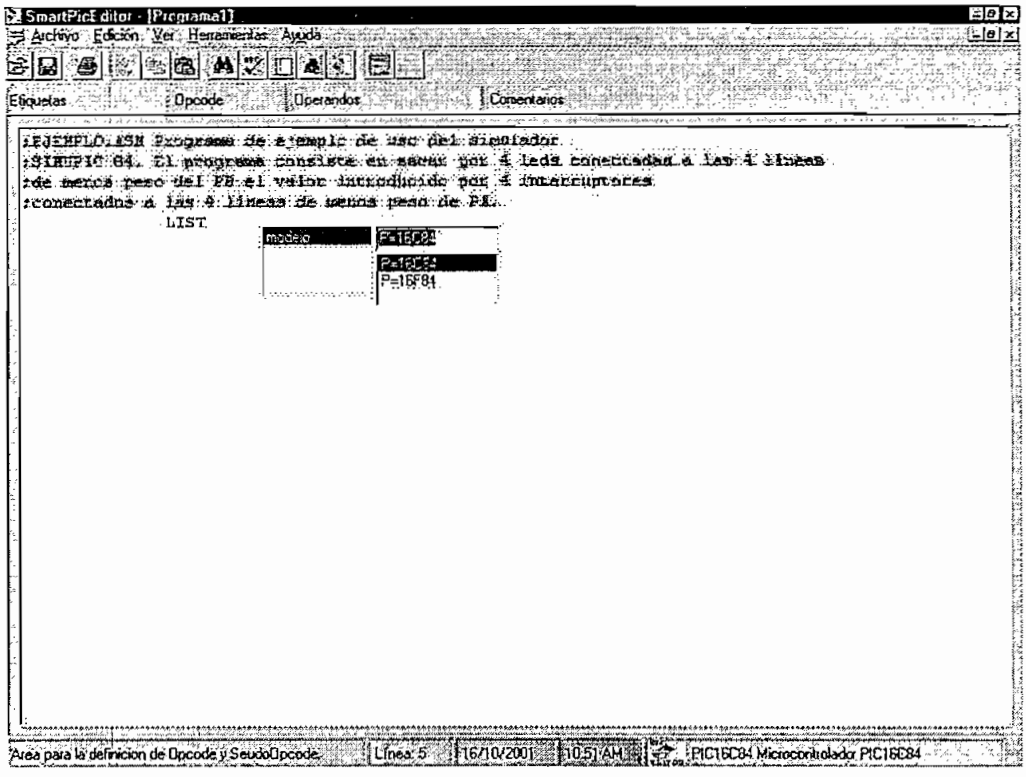


Fig. 4.18 Ingreso del pseudo-opcode de la directiva list

4.3.4 INGRESO DEL COMENTARIO

Cuando no se requieren más operados el editor automáticamente inserta un tabulador y se ubica en el campo adecuado para escribir un comentario, se deberá escribir el identificador de comentario (;) y el comentario respectivo.

En el caso de que el usuario no requiera escribir el comentario, y se presione enter, el editor borra los tabs precedentes y cambia de línea. Esto se ilustra en la figura 4.19 y 4.20.

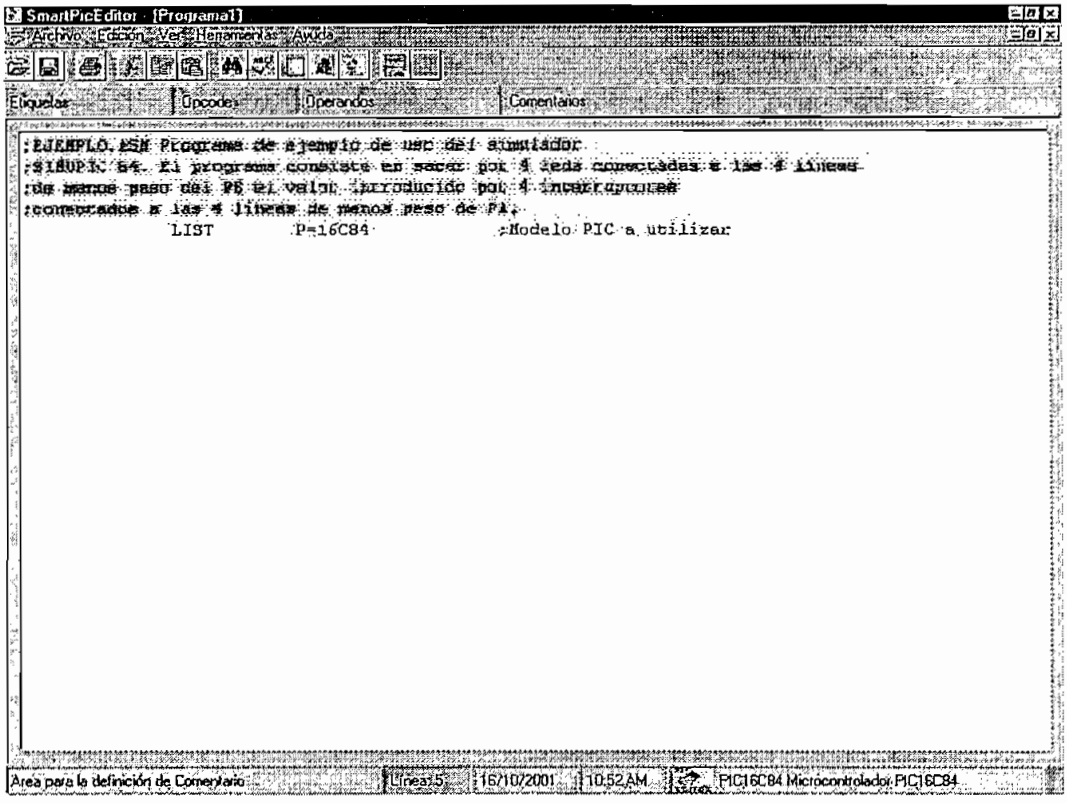


Fig. 4.19 Ingreso de un comentario

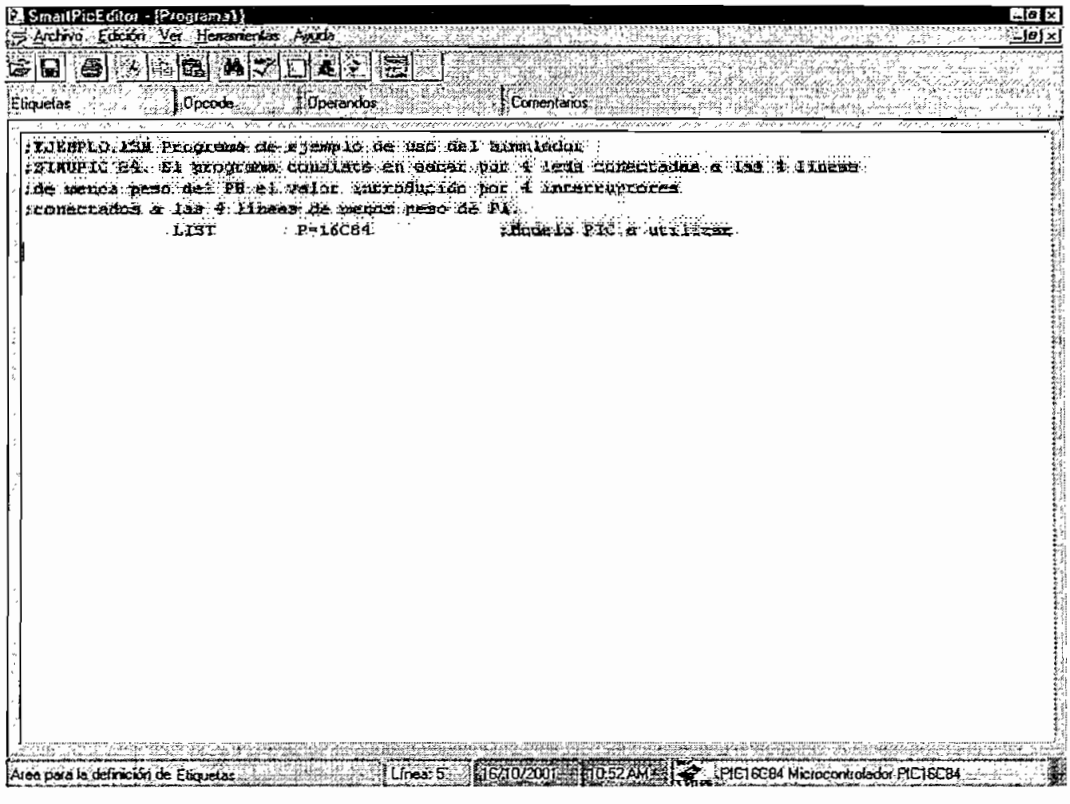


Fig. 4.20 Coloreado de la línea ingresada

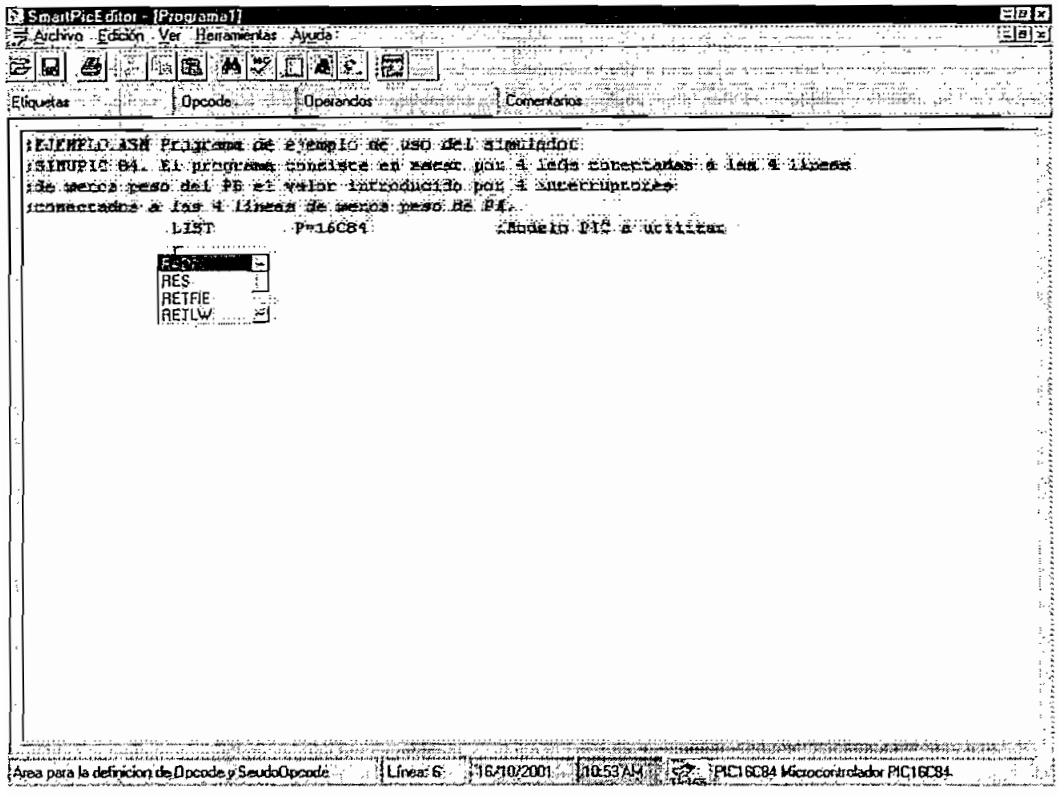


Fig. 4.21 Ingreso de la segunda línea de programa seudo-opcode

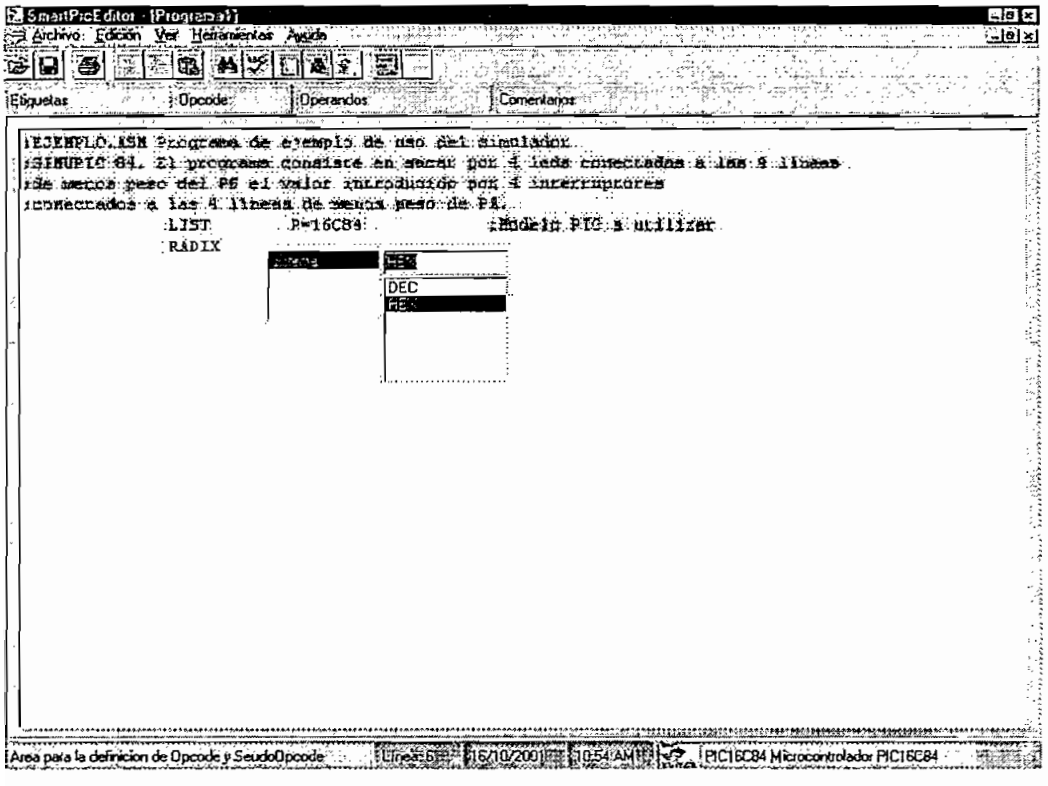


Fig. 4.22 Ingreso del seudo-operando correspondiente

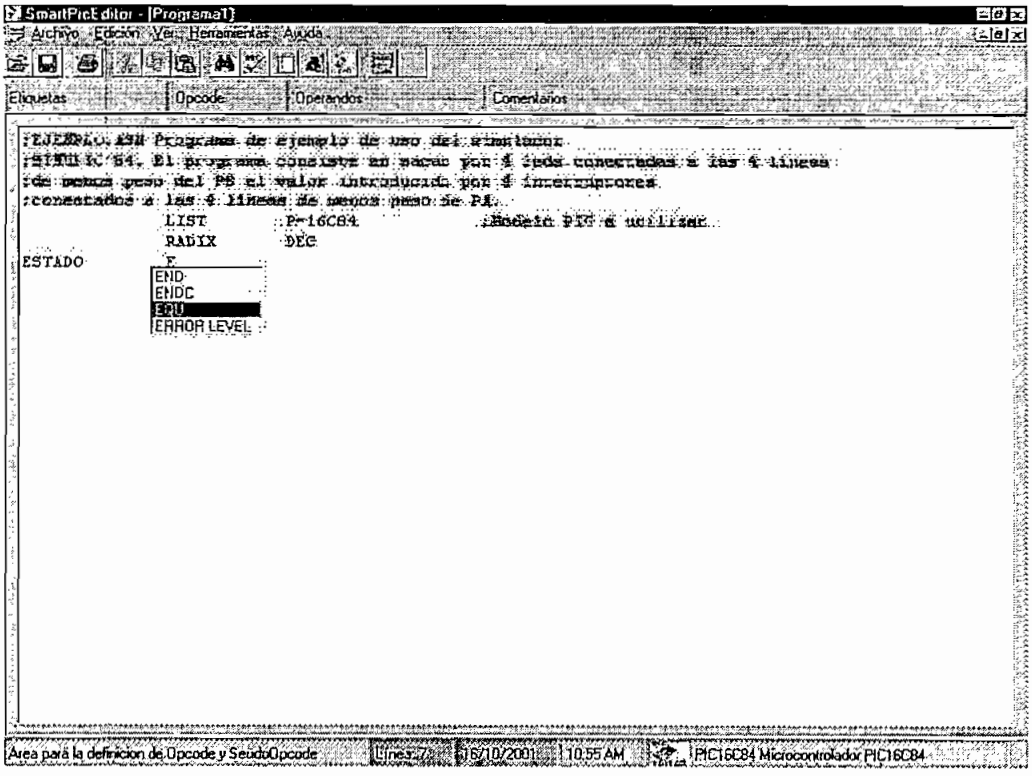


Fig. 4.23 Ingreso de la directiva EQU

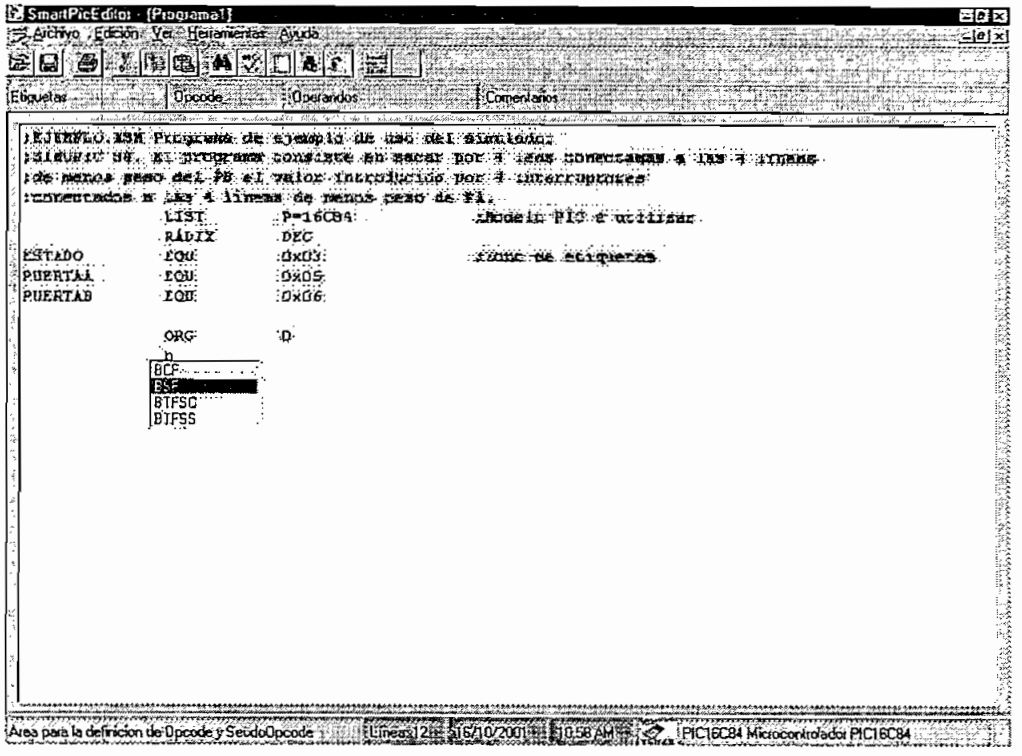


Fig. 4.24 Ingreso del opcode BSF

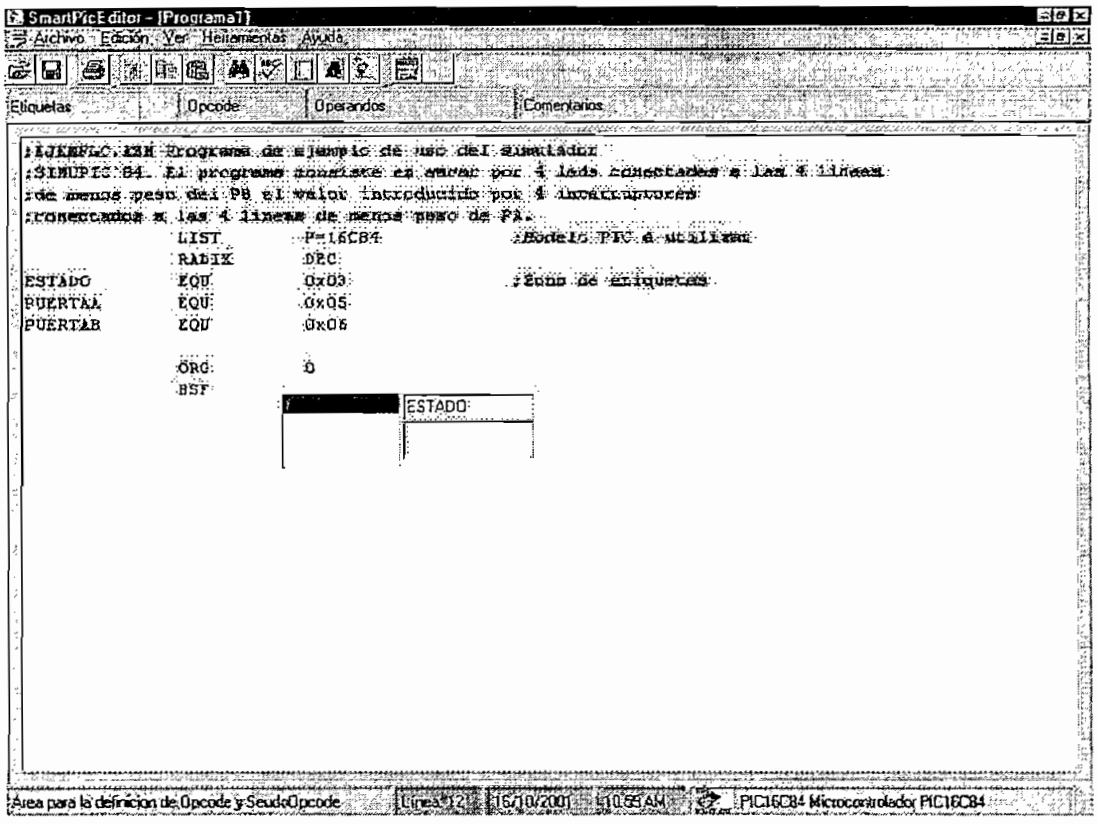


Fig. 4.25 Ingreso del primer operando de BSF

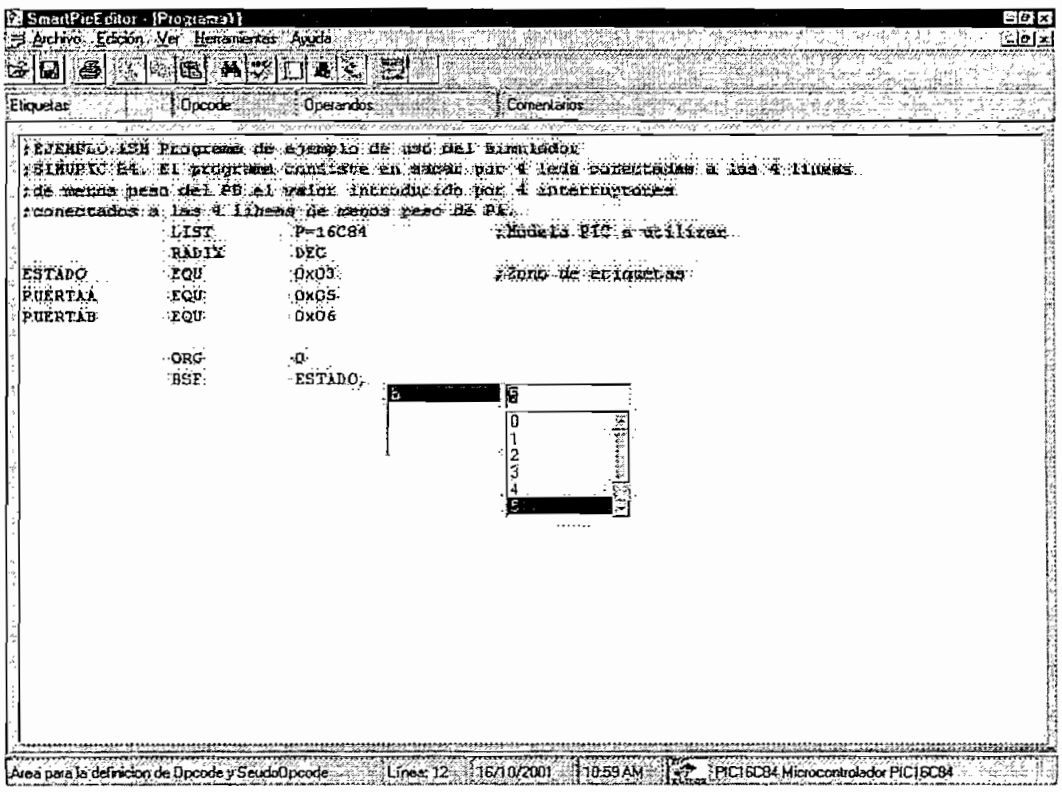


Fig. 4.26 Ingreso del segundo operando de BSF

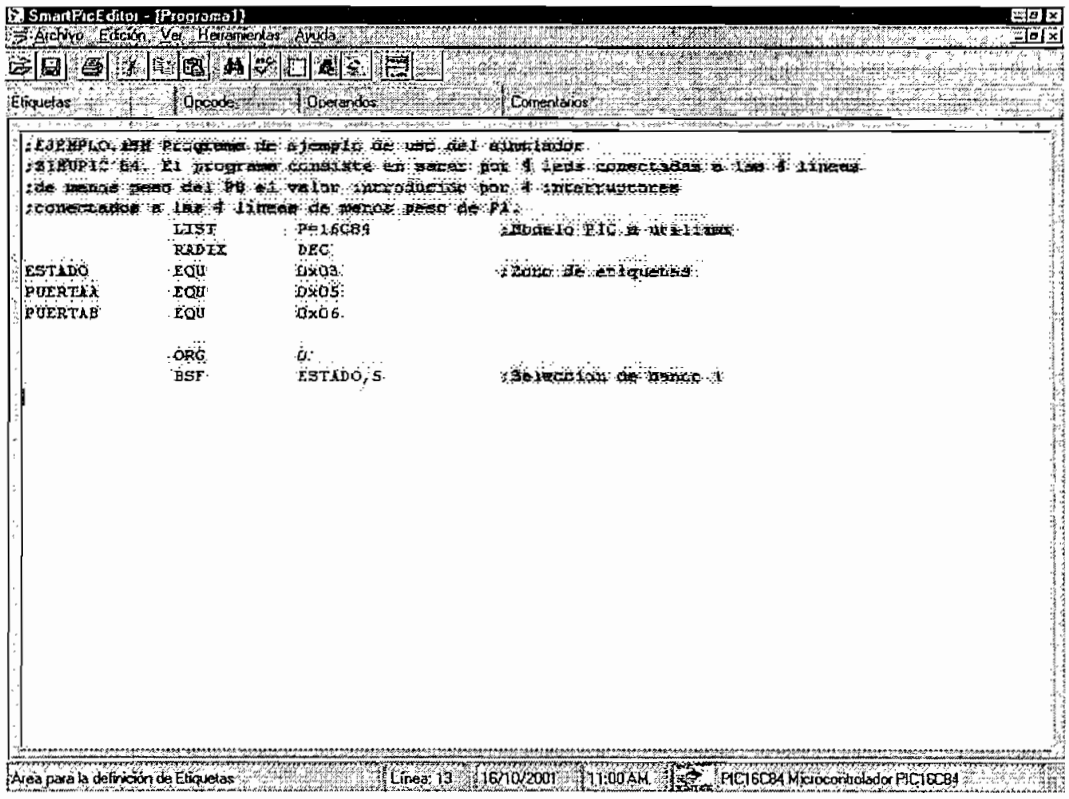


Fig. 4.27 Instrucción BSF ingresada

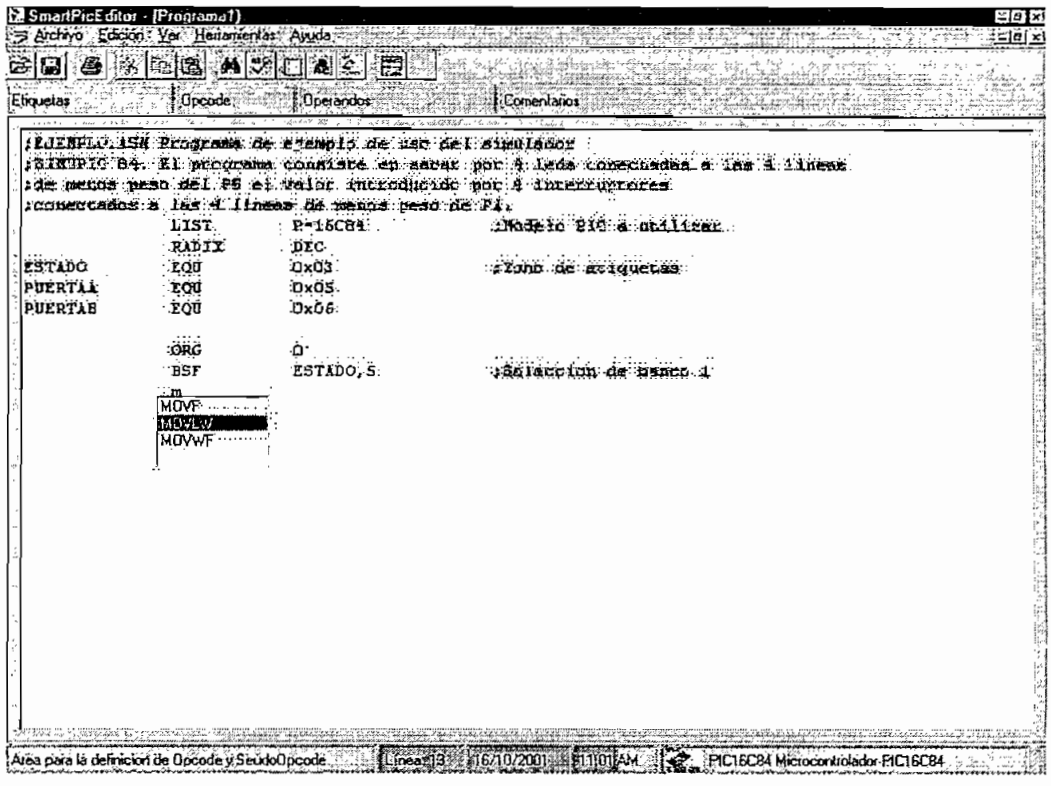


Fig. 4.28 Ingreso del opcode MOVLW

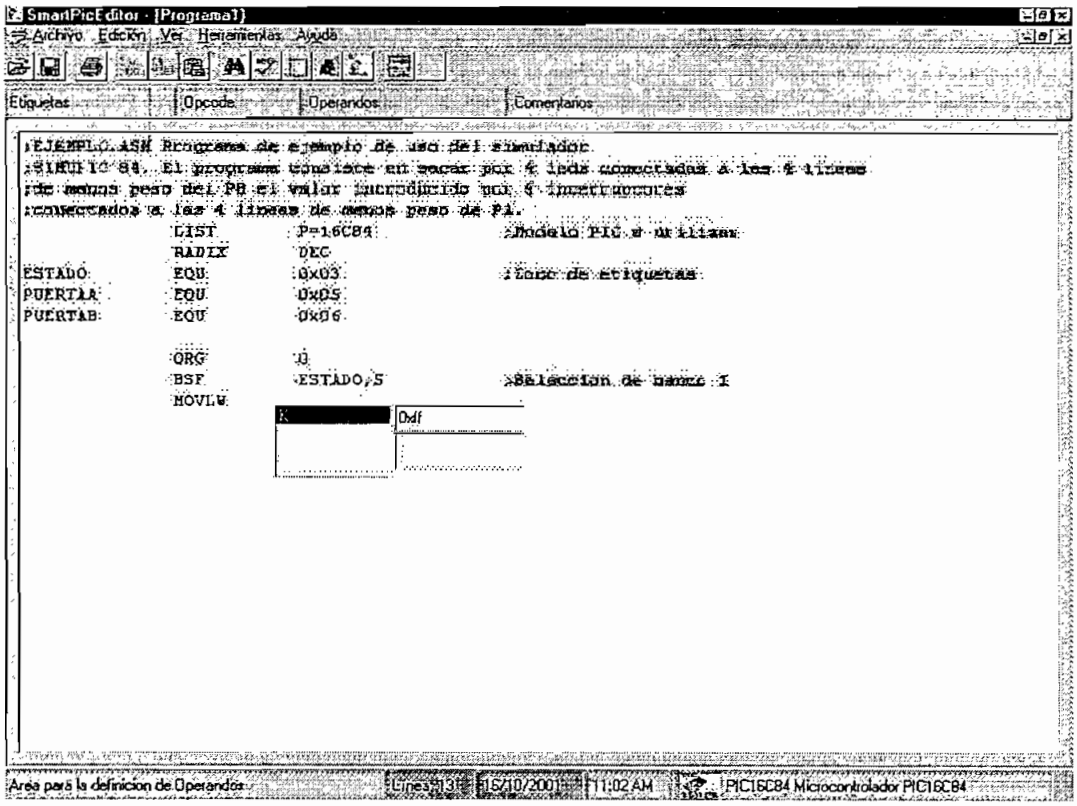


Fig. 4.29 Ingreso del único operando de MOVLW

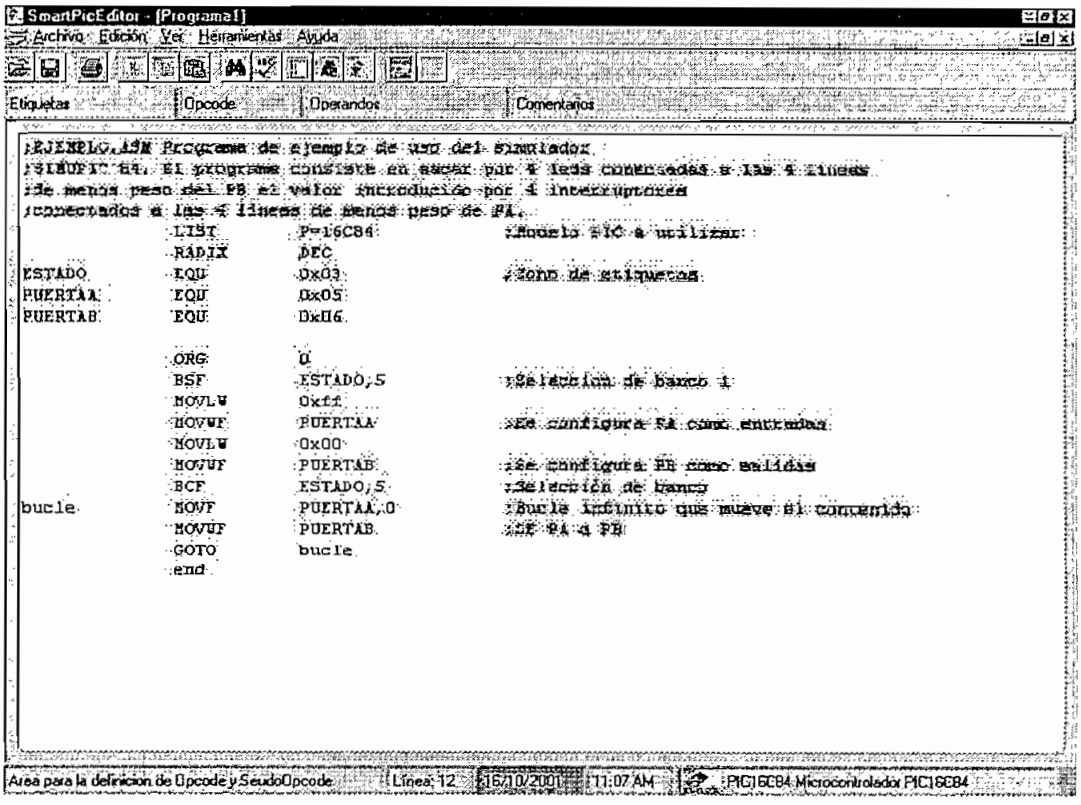


Fig. 4.30 Programa completo

4.3.5 AYUDA CON EL BOTON DERECHO DEL RATÓN

Una vez escrito una o más instrucciones, el usuario puede consultar información acerca de una instrucción, posicionando el cursor en la línea correspondiente a la instrucción y luego presionando el botón derecho del ratón, como se puede observar en los ejemplos de las figuras 4.31, 4.32 y 4.33.

Si existe una instrucción errónea por medio del botón derecho se puede acceder a la opción de corrección de dicha instrucción.

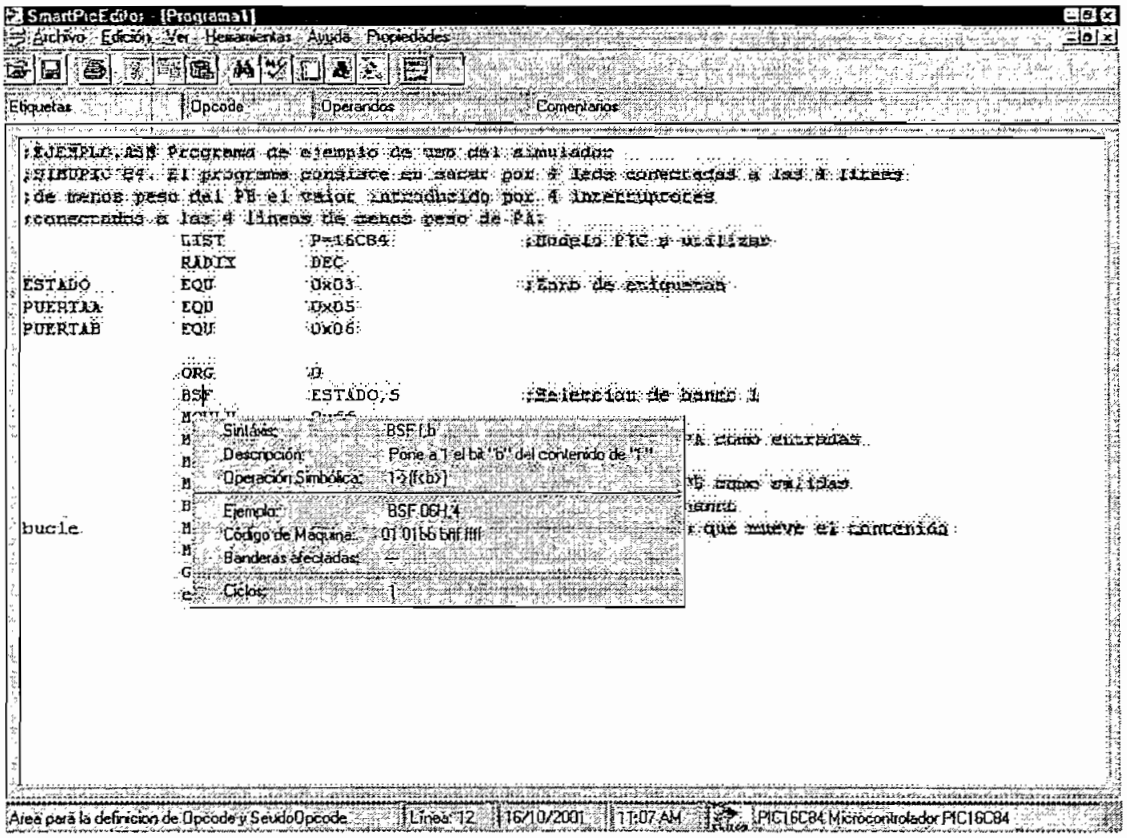


Fig. 4.31 Ayuda para la instrucción BSF f,b

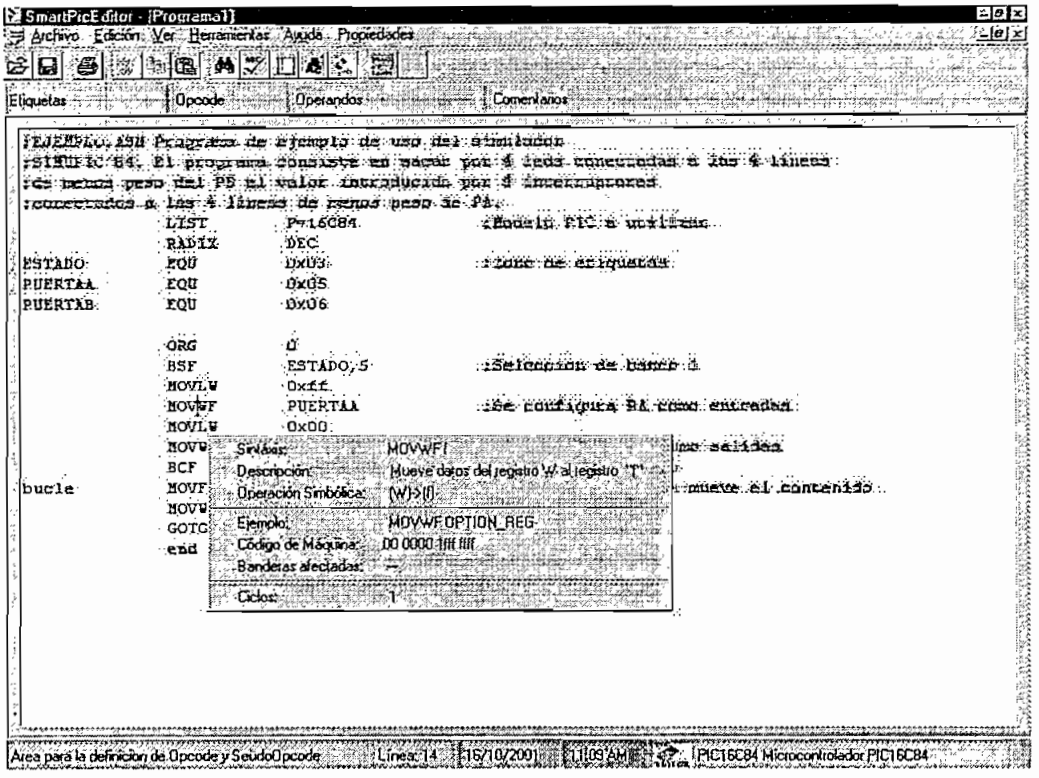


Fig. 4.32 Ayuda para la instrucción MOVWF f

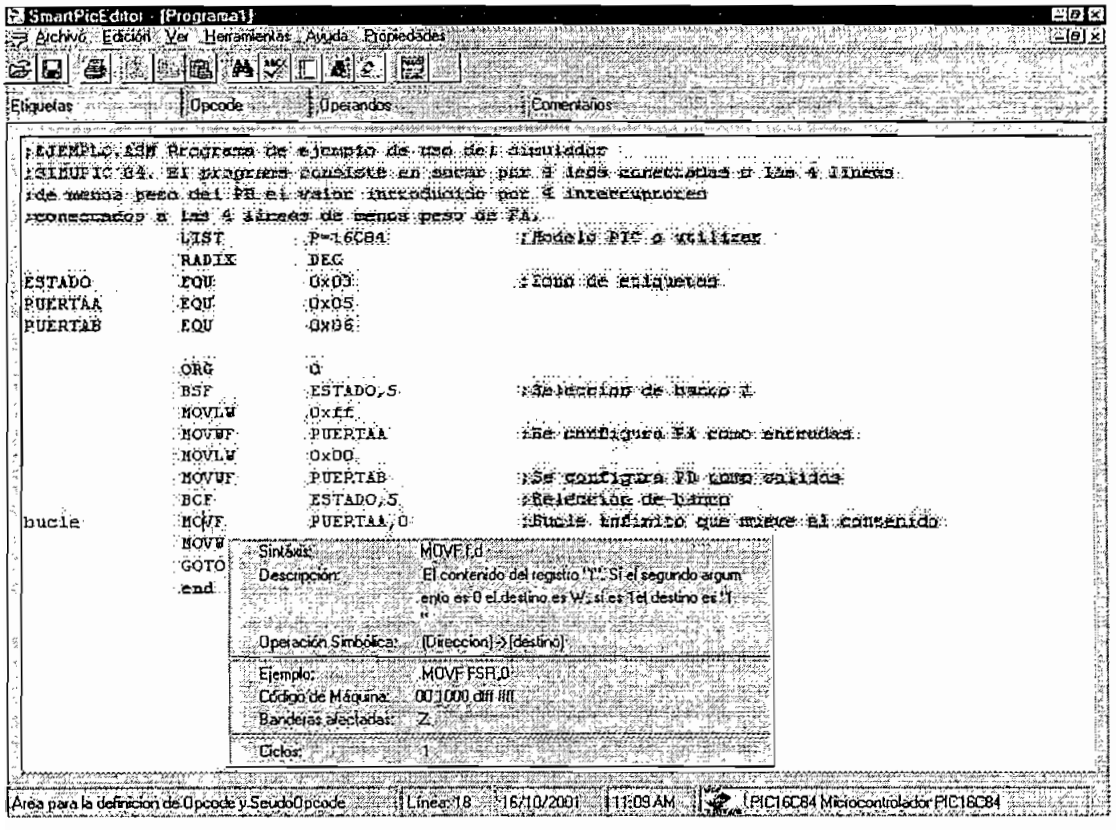


Fig. 4.33 Ayuda para la instrucción MOVF f,d

Finalmente se procede a grabar el nuevo programa editado, el Editor por omisión lo guarda con extensión asm. Ver las figuras 4.34 y 4.35.

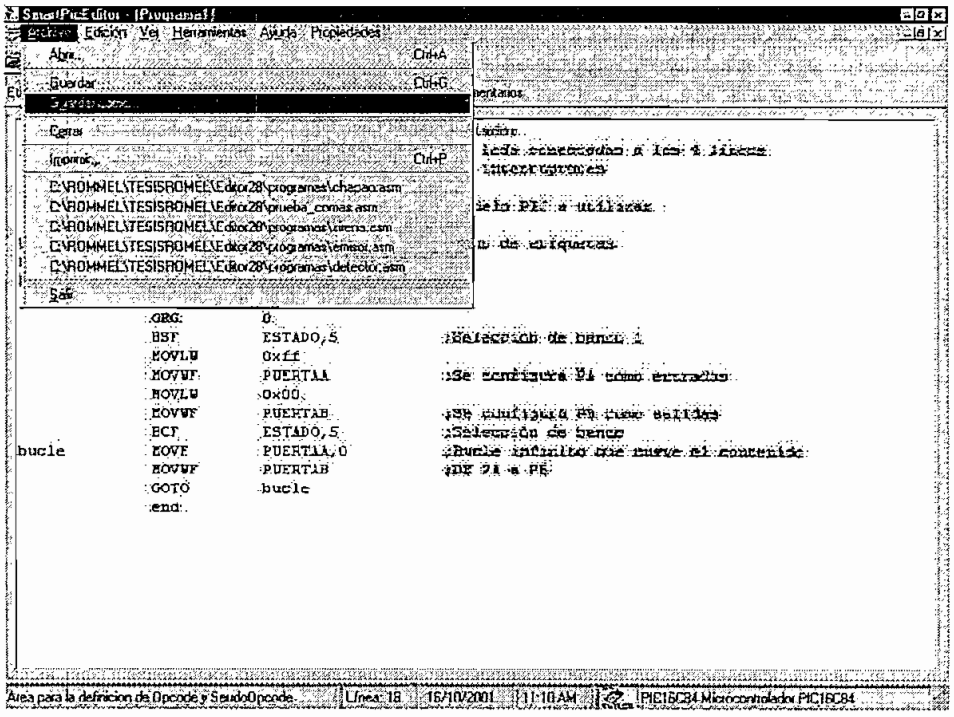


Fig. 4.34 Guardar el programa paso 1

Fig. 4.35 Guardar el programa paso 2

Una vez guardado el programa en la pantalla en la parte superior se puede apreciar el path donde se encuentra almacenado dicho programa.

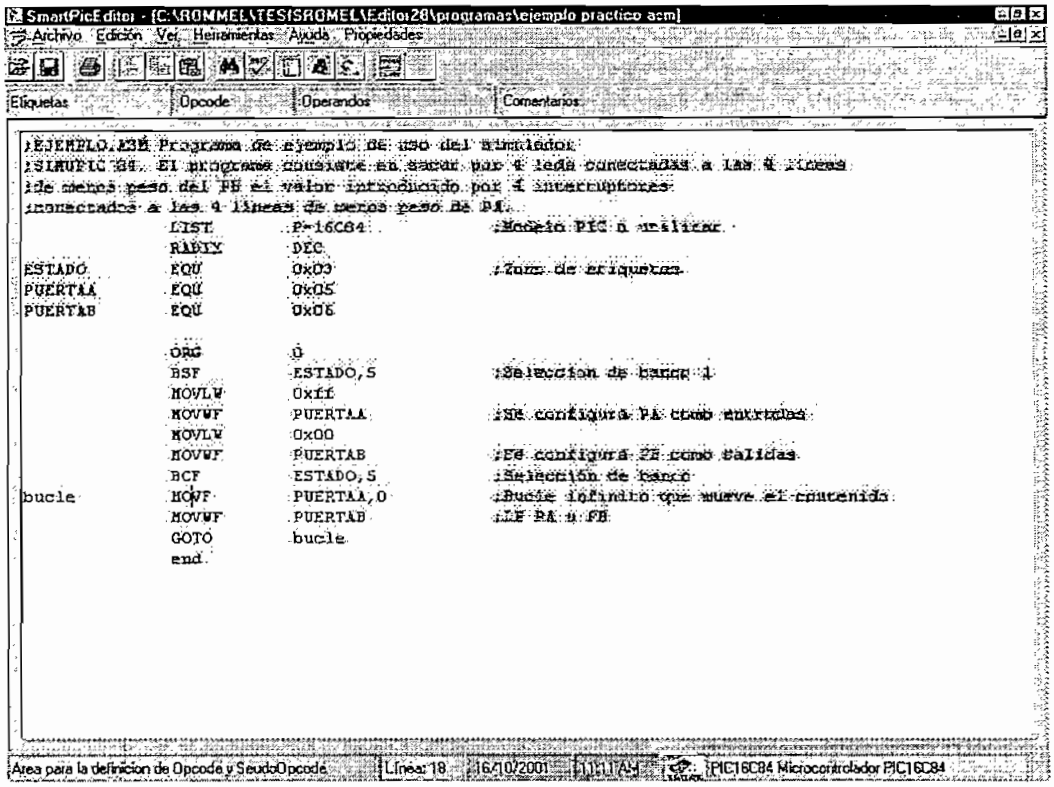


Fig. 4.36 Programa almacenado

4.3.6 OTRAS AYUDAS DEL EDITOR

El editor cuenta también con ayudas típicas de un editor como son las de buscar y reemplazar, ver la figura 4.38.

Se ha implementado adicionalmente una pantalla para la inserción de caracteres ASCII, ver la figura 4.37.

Fig. 4.37 Pantalla para inserción de caracteres ASCII

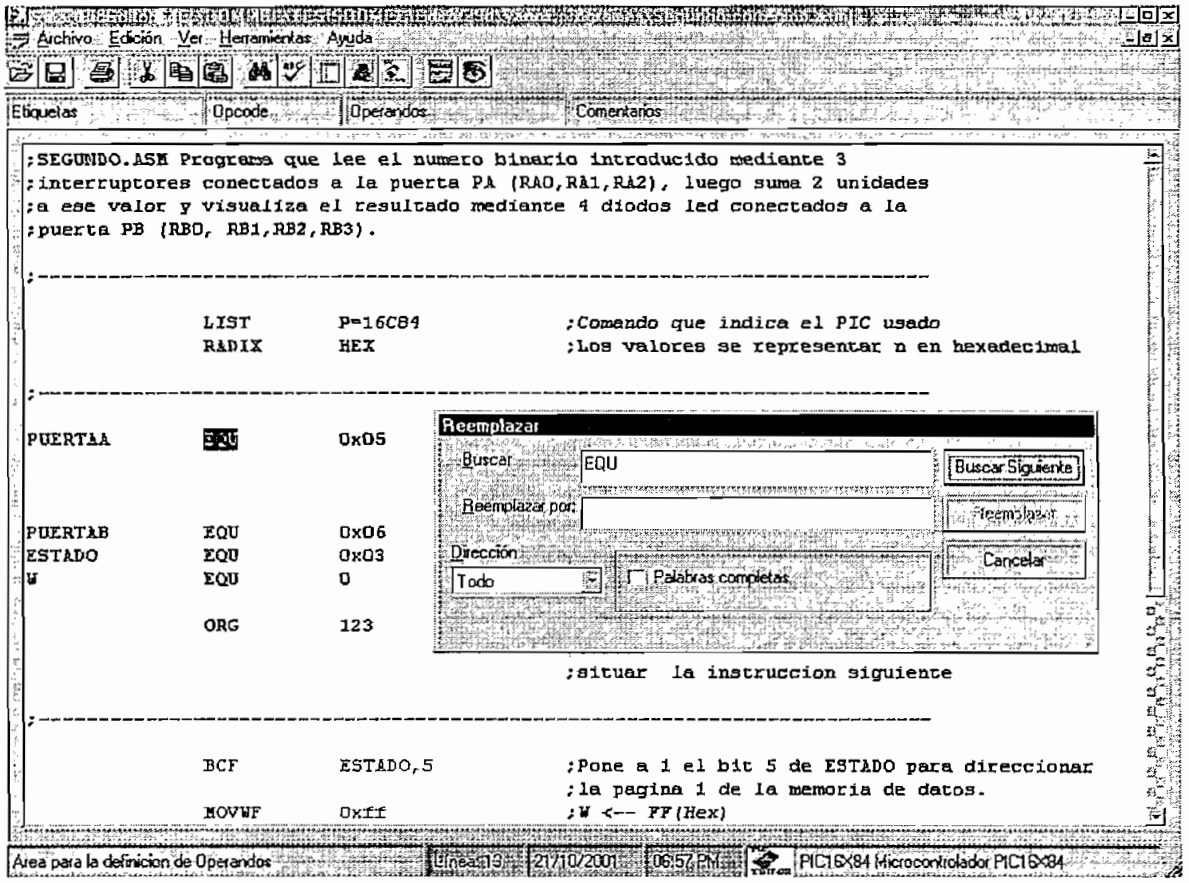


Fig. 4.38 Pantalla para reemplazar una cadena de texto buscada

4.4 CORRECCION DE UN PROGRAMA MEDIANTE EL EDITOR

4.4.1 FORMAS DE ACCEDER A LA CORRECCION DE ERRORES

Para acceder a las pantallas de corrección de errores, durante la edición de un programa se debe configurar las opciones del editor para activar la corrección de errores en la presente sesión, esto se hace en la pantalla "Opciones del Editor" que se encuentra en el menú "Herramientas" submenú "Opciones...", esto se puede observar en la Figura 4.39.

Existen dos formas de configuración del editor:

Activar la corrección al finalizar la instrucción.- Si se requiere que las pantallas para la corrección aparezcan al finalizar una instrucción es decir cuando el programador pulse la tecla de cambio de línea (enter).

Activar la corrección al finalizar el programa.- En este caso no aparecen las pantallas de corrección al finalizar una instrucción, sin embargo, si la instrucción es errónea, la línea se pinta de rojo; esta opción es la que se encuentra configurada como predeterminada al abrir una sesión de trabajo con el Smart Pic Editor.

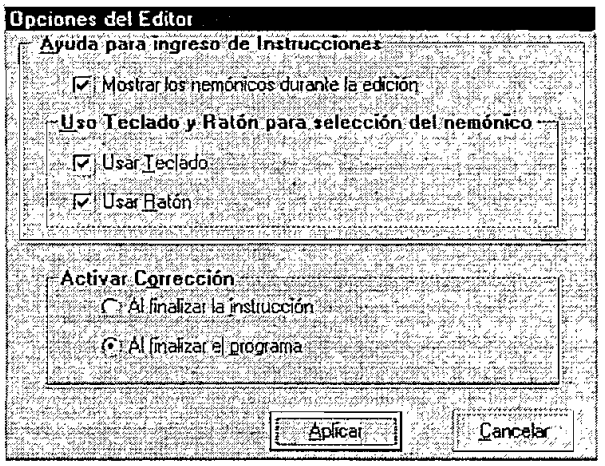


Fig. 4.39 Pantalla donde se muestran la activación de las pantallas de corrección durante la edición

Para cualquiera de estas dos formas de configuración de la activación de pantallas de corrección, existen otras alternativas para el acceso a la corrección de errores así.

Corrección de la línea donde se encuentra posicionado el cursor.- Se puede corregir la línea errónea donde se encuentra el cursor, para ello se pulsa el botón derecho del ratón, y se selecciona la opción corregir la sintaxis de la línea, esta misma opción está presente en el menú "herramientas", submenú "Corregir errores de la línea actual" o presionando la tecla de funciones F5.

Corrección todo el programa.- Otra opción es corregir todas las líneas erróneas presentes en el programa, para ello, mediante el menú "herramientas", submenú "Corregir errores de todo el programa" también presionando la tecla de funciones F7 o mediante el botón de acceso directo "Corregir programa".

En las diferentes figuras que se muestran se realiza la corrección de errores de un programa, para ilustrar los diferentes tipos de errores de sintaxis que se puede corregir mediante el Smart Pic Editor, para ello se han introducido errores en el programa "segundo.asm" presente en la página 40 y 41 del libro MICROCONTROLADORES PIC Diseño práctico de aplicaciones de los autores José Angulo & Ignacio Angulo.

En las figuras 4.40 y 4.41 se muestra el programa previamente editado y una vez abierto mediante el Smart Pic Editor.

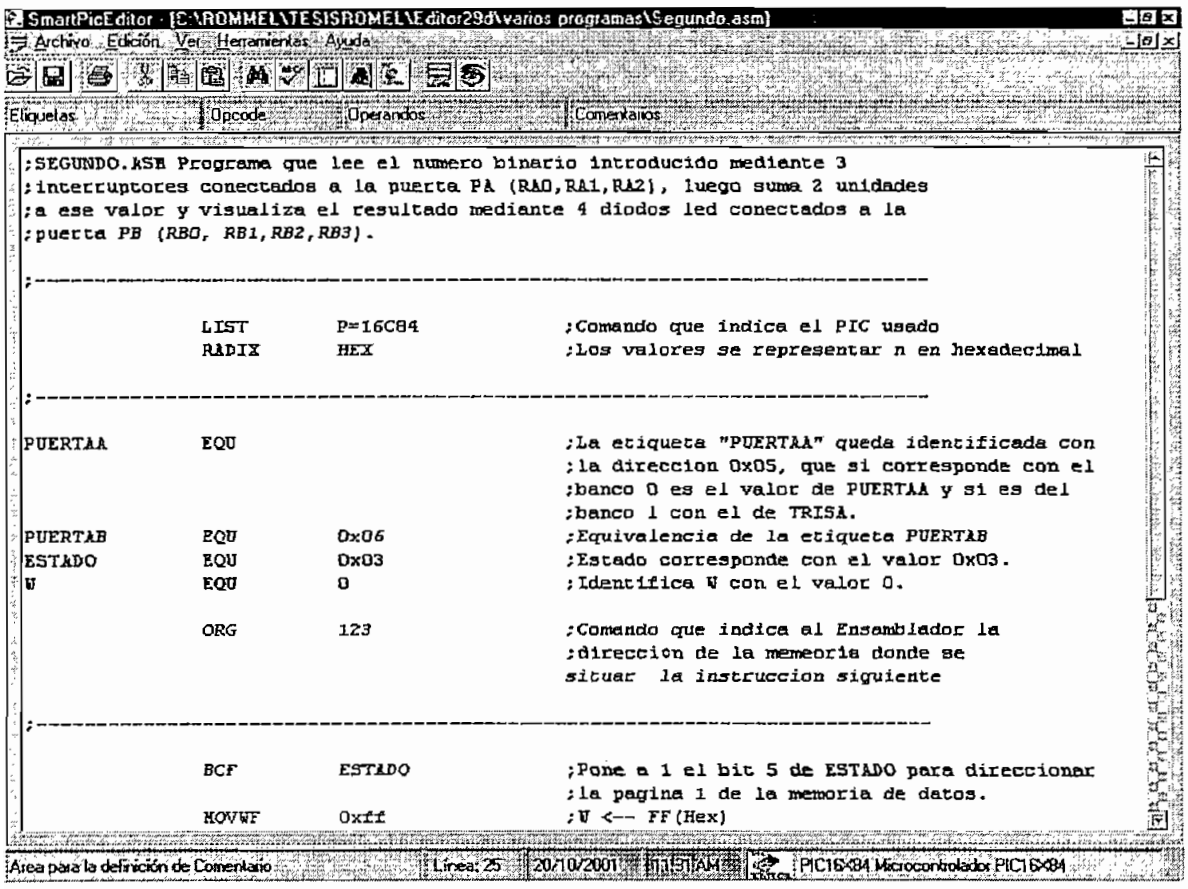


Fig. 4.40 Pantalla donde se muestra la parte inicial del programa ha ser corregido

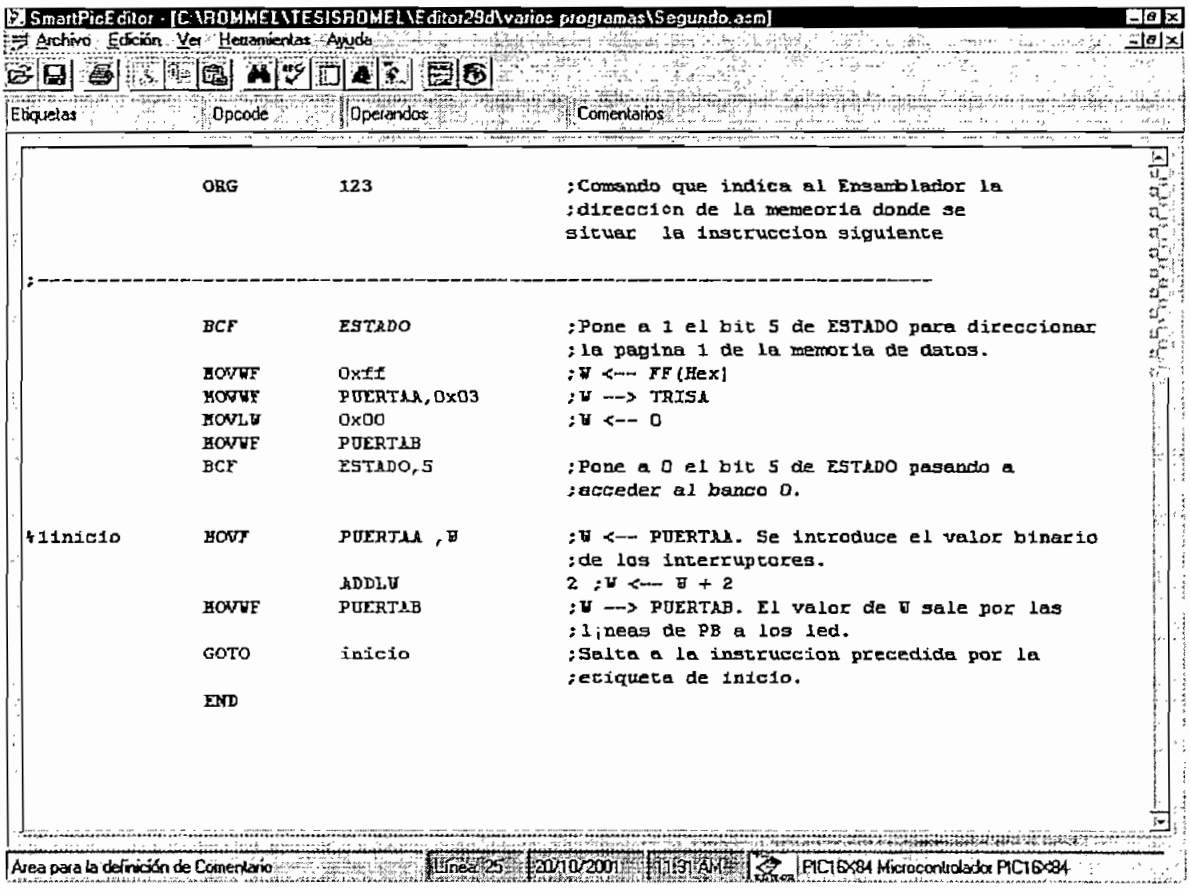


Fig. 4.41 Pantalla donde se muestra la parte final del programa ha ser corregido

4.4.2 ERROR DE ETIQUETA.

Una etiqueta acepta valores alfanuméricos A...Z, a,...z, numéricos 0,1,...9 y el signo subraya _ , además una etiqueta no debe comenzar con un número.

Una etiqueta no puede ser un opcode del conjunto de instrucciones, si una etiqueta no cumple con estos requisitos el editor lo considera como un error.

En la figura 4.48 se ilustra la pantalla usada por el editor para la corrección de una etiqueta errónea.

4.4.3 ERROR DE OPCODE o PSEUDOOPCODE

En el campo de opcode o pseudo-opcode únicamente se validan valores previamente definidos en la base de datos que corresponden *al* opcode del

conjunto de instrucciones o directivas del ensamblador, cualquier otra palabra el Editor lo considera como un error.

En las figuras 4.55, 4.56, 4.57 y 4.58 se observa la secuencia de la corrección de un opcode erróneo.

4.4.4 ERROR DE OPERANDOS.

Al analizar una instrucción, y una vez validado el opcode o pseudo-opcode se procede al análisis de los operandos.

Se analiza cada uno de los operandos para diferenciarlos, el separador de operando es la coma (,), el indicador que ha finalizado el área de operandos es el tabulador el punto y como o el enter.

Se distinguen tres tipos de operandos:

Constante.- Acepta un conjunto de caracteres único, si toma otro valor constituye un error.

Variable.- puede aceptar cualquier conjunto de caracteres.

Lista.- al igual que el tipo variable puede aceptar cualquier conjunto de caracteres, pero en la presentación de ayuda en la pantalla muestra una lista de opciones donde el usuario puede escoger.

Una vez validados los operandos, se analiza si el número de ellos presente en la instrucción corresponde al número de operandos que requiere dicha instrucción, caso contrario el editor considera un error sea este por falta o por exceso de operandos.

Si existen instrucciones que tienen el mismo opcode y diferente número de operandos en el conjunto de instrucciones como es el caso de los microprocesadores MCS 51/52, se valida el primer operando, si ninguna

instrucción cumple, el editor considera un error, de existir una o varias instrucciones cuyo primer operando es válido continua con el análisis del segundo operando y así hasta que por lo menos una instrucción sea validada, en este caso se presenta como válida la instrucción, de no existir el editor considera a la instrucción como errónea.

En las figuras 4.43 y 4.44 se muestran las pantallas que usa el Smart Pic Editor para corregir errores por falta de operandos en una instrucción.

Así mismo, en las figuras 4.45 y 4.46 se indican las pantallas que usa el Smart Pic Editor para corregir errores por exceso de operandos en una instrucción.

4.4.5 ERROR POR EXCESO DE TABULADORES.

En caso que el editor encuentre opcodes o pseudo-opcodes en los campos de operandos o comentarios el editor permite la eliminación de tabuladores o espacios anteriores, esto para una correcta tabulación y visualización del programa para el usuario. En la figura 4.49 se muestra la pantalla usada por el editor para este fin.

4.4.6 ERROR EN LA ZONA DE COMENTARIO.

Si el Editor encuentra palabras en el campo de comentarios y la primera de estas no corresponden a opcodes o pseudo-opcodes, el editor permite insertar el punto y coma (;) faltante para definir un comentario.

En la figura 4.47 se indica la pantalla que el Smart Pic Editor usa para corregir un error de falta de signo que indica un comentario.

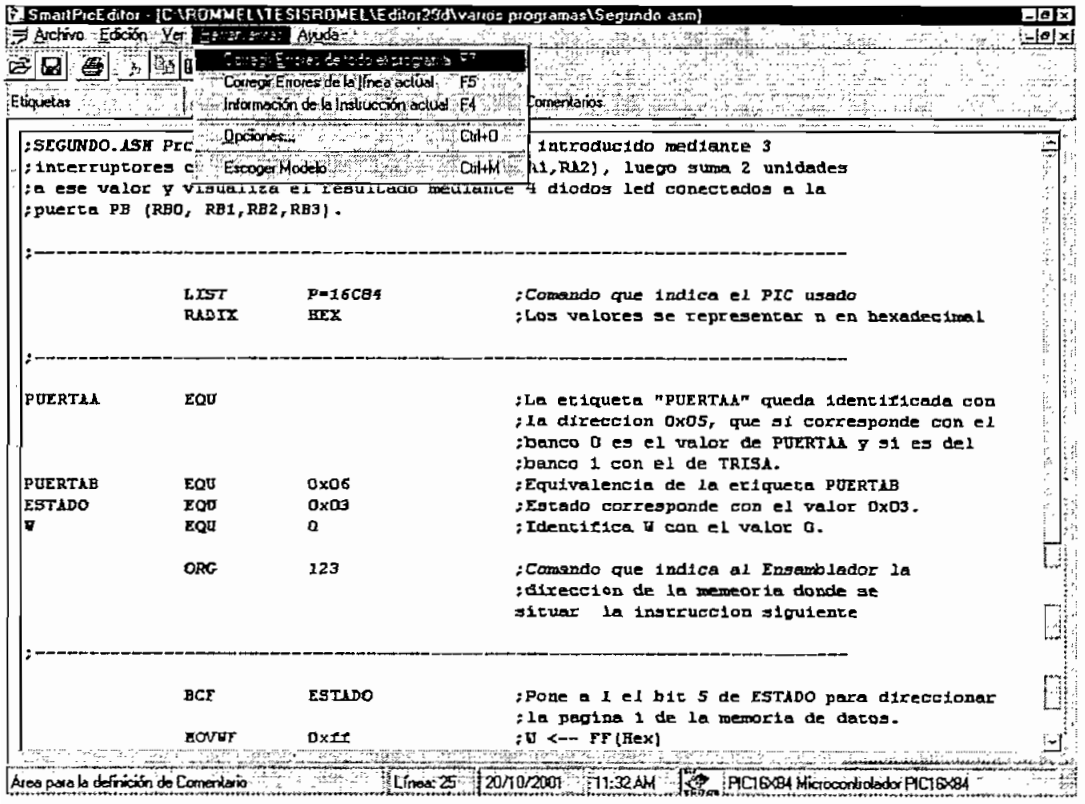


Fig. 4.42 Se escoge la opción: "Corregir errores de todo el programa"

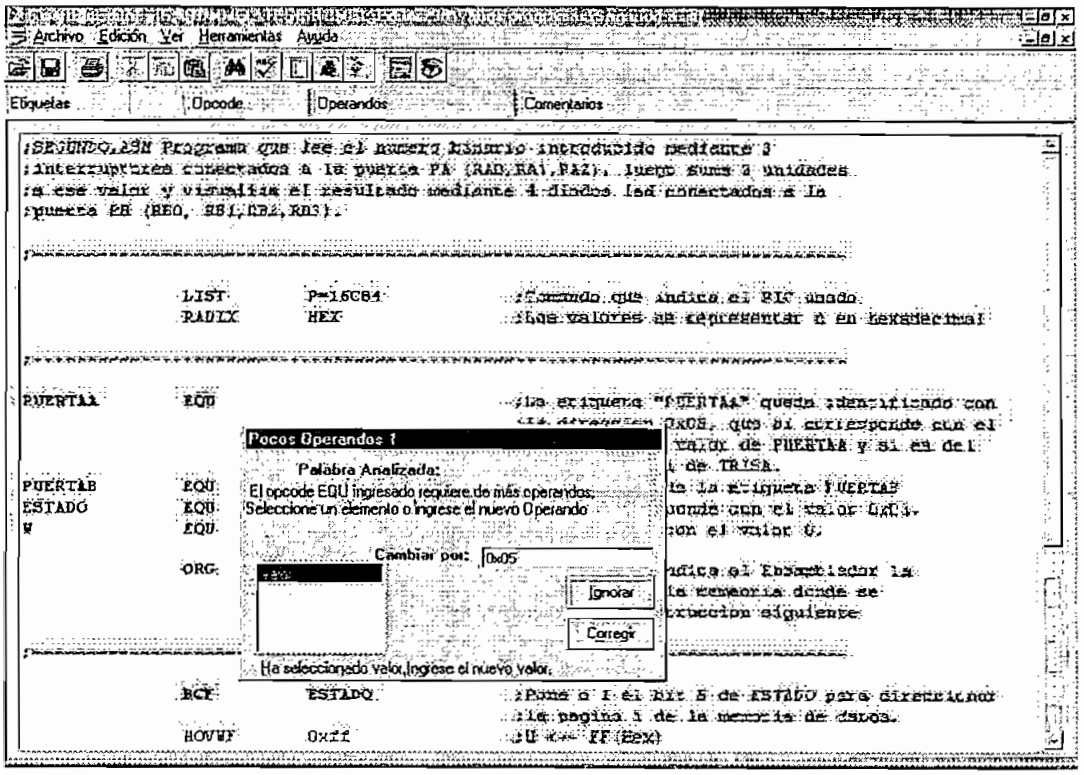


Fig. 4.43 Corrección de error por falta del primer y único operando

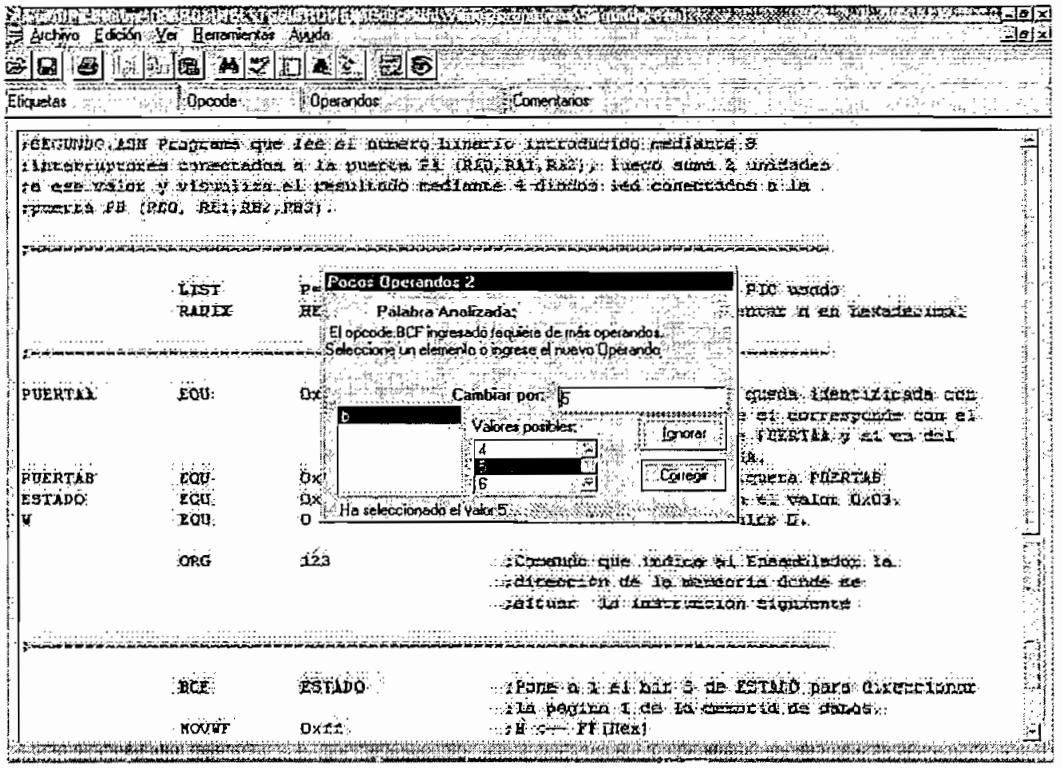


Fig. 4.44 Corrección de error por falta del segundo operando

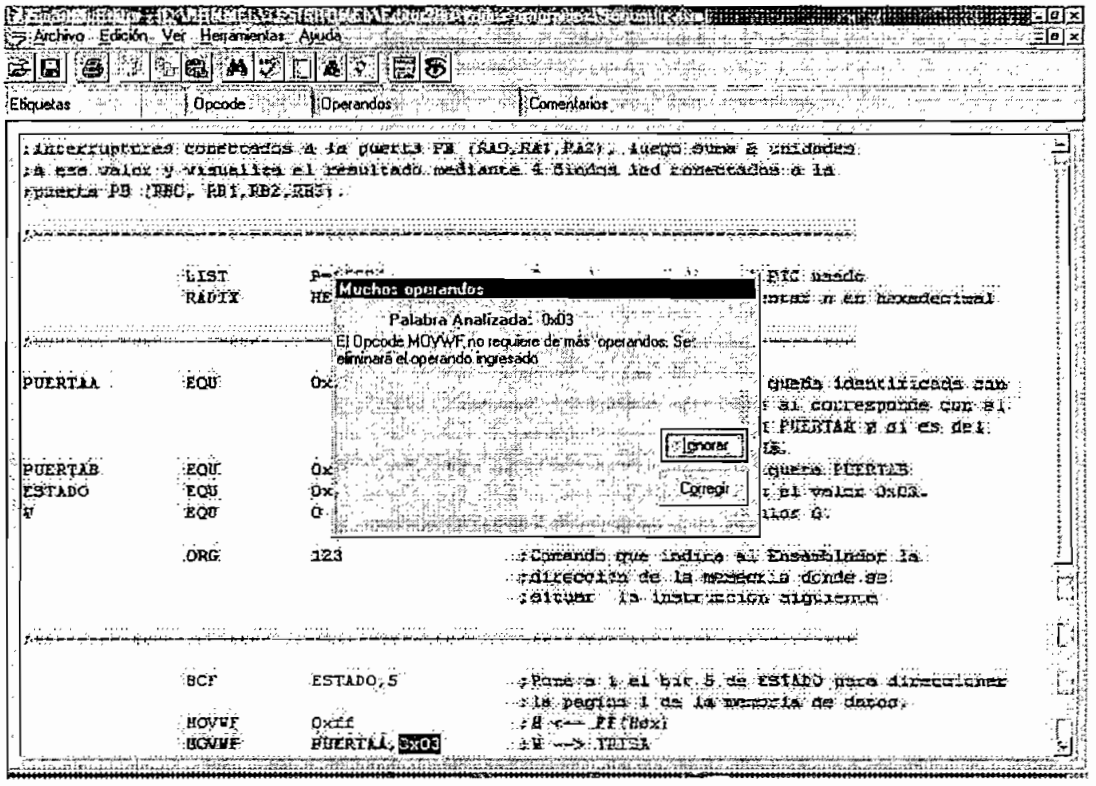


Fig. 4.45 Corrección de error por exceso de operandos

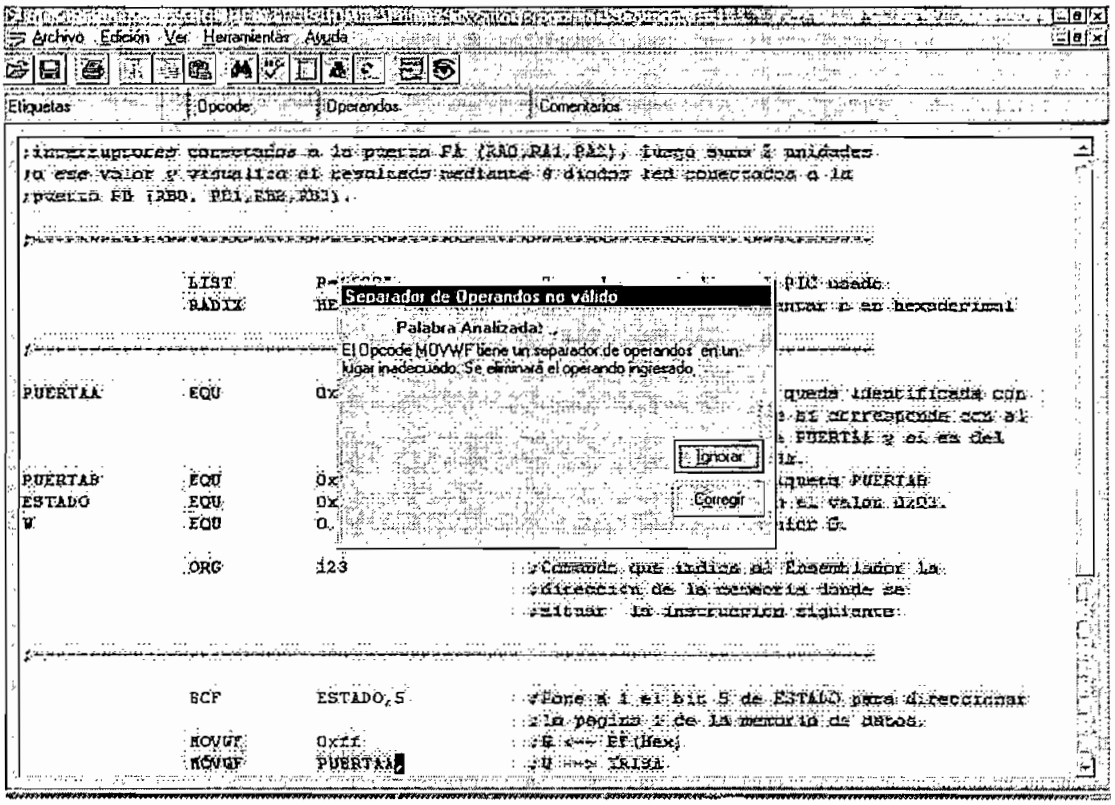


Fig. 4.46 Eliminación de una coma en lugar inadecuado.

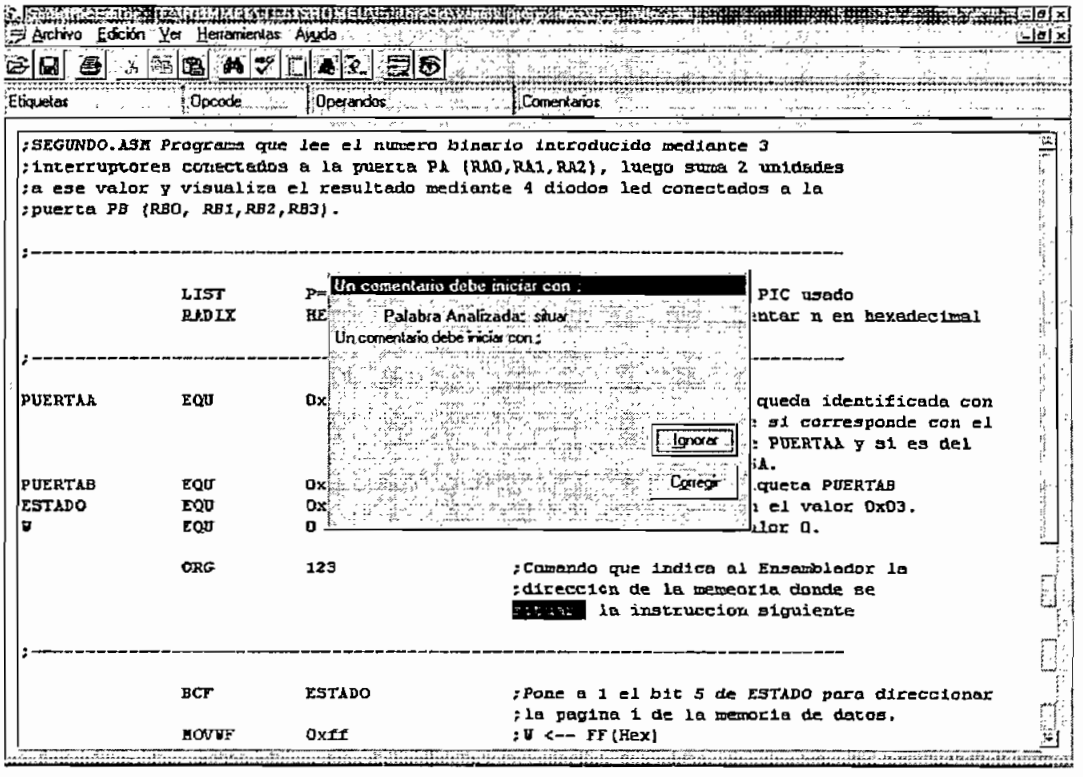


Fig. 4.47 Corrección de error por falta de signo de comentario

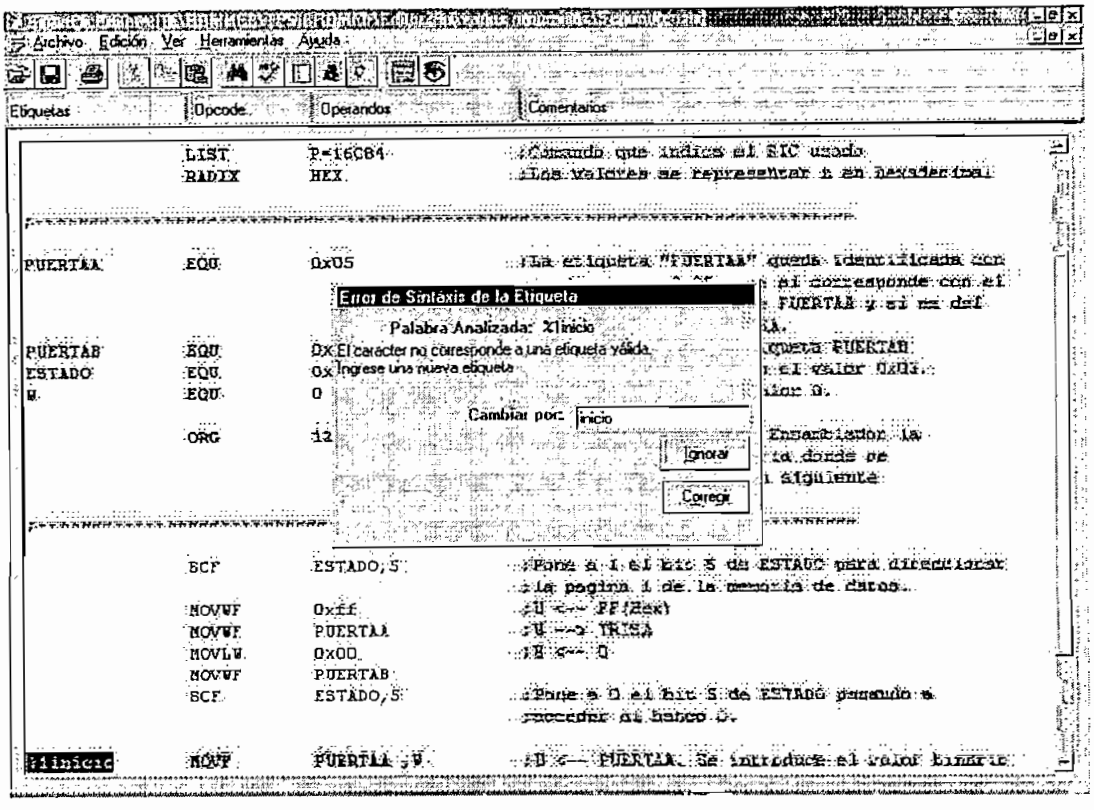


Fig. 4.48 Corrección de error de etiqueta no válida

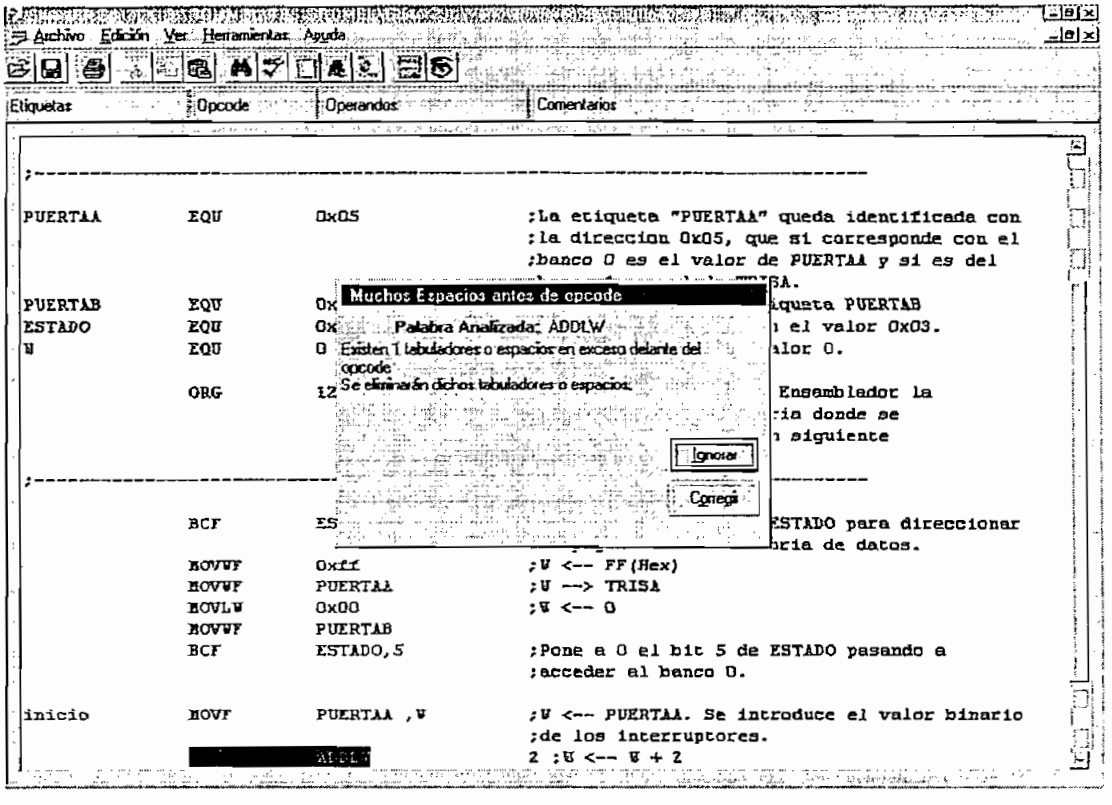


Fig. 4.49 Eliminación de exceso de tabulador

Una vez finalizada la corrección de errores, el programa se muestra correctamente coloreado en azul, negro y verde; para el caso del programa tomado como ejemplo se puede apreciar esto en las figuras 4.50 y 4.51.

Si el usuario, hace correcciones sobre este programa y comete errores de sintaxis en una línea de programa, esta línea errónea se pondrá en color rojo inmediatamente después que el usuario cambie de línea, esto facilita que se corrija prontamente este error, para ello es útil la opción "*Corrección de la línea donde se encuentra posicionado el cursor*" de acuerdo a lo indicado en el numeral 4.4.1 "Formas de acceder a la corrección de errores" en el presente capítulo.

En las figuras 4.52 y 4.56 se indican dos de las formas de acceder a la corrección de una línea de programa.

En las figuras 4.52, 4.53 y 4.54 se indica la secuencia de pantallas para la corrección de una sola línea de la pseudo-instrucción EQU.

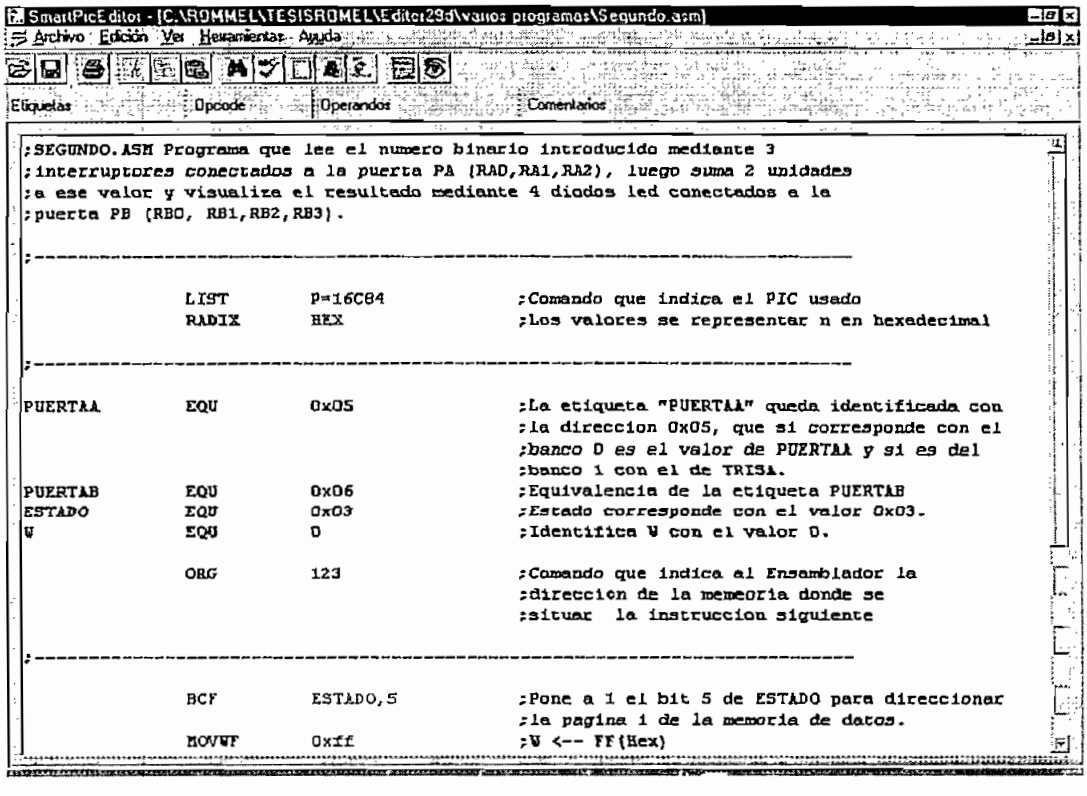


Fig. 4.50 Fin de la corrección de errores del programa parte inicial

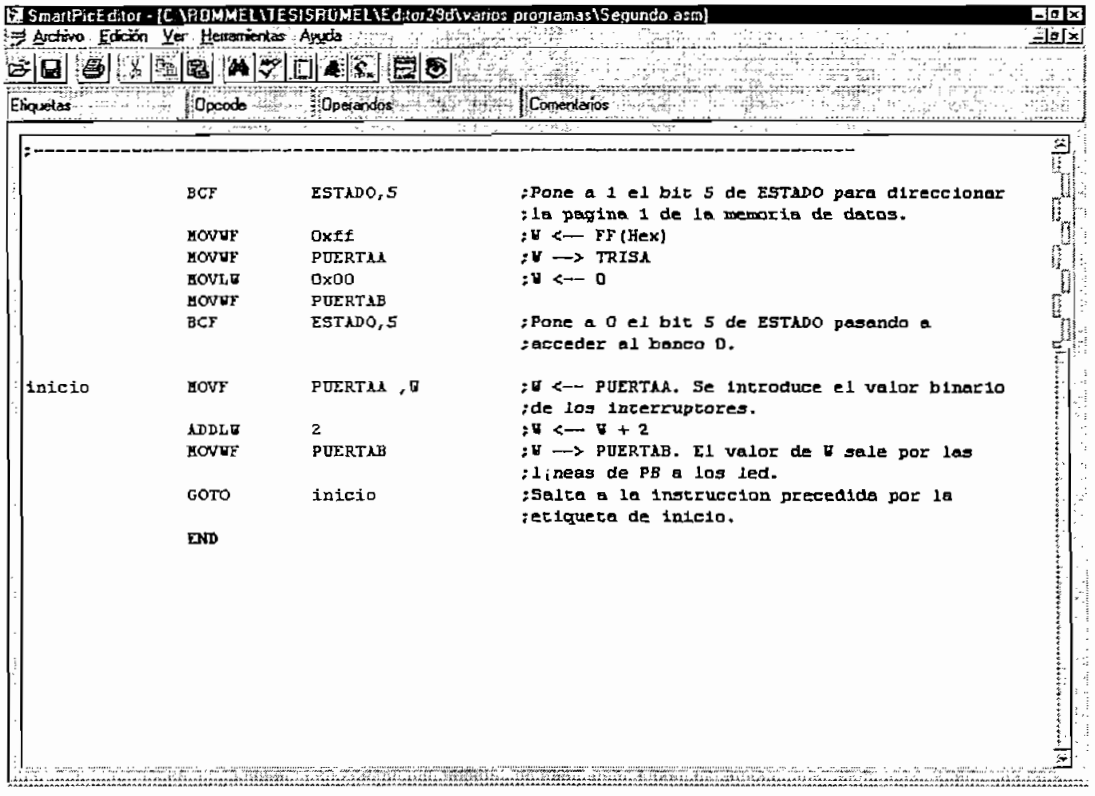


Fig. 4.51 Fin de la corrección de errores del programa parte final

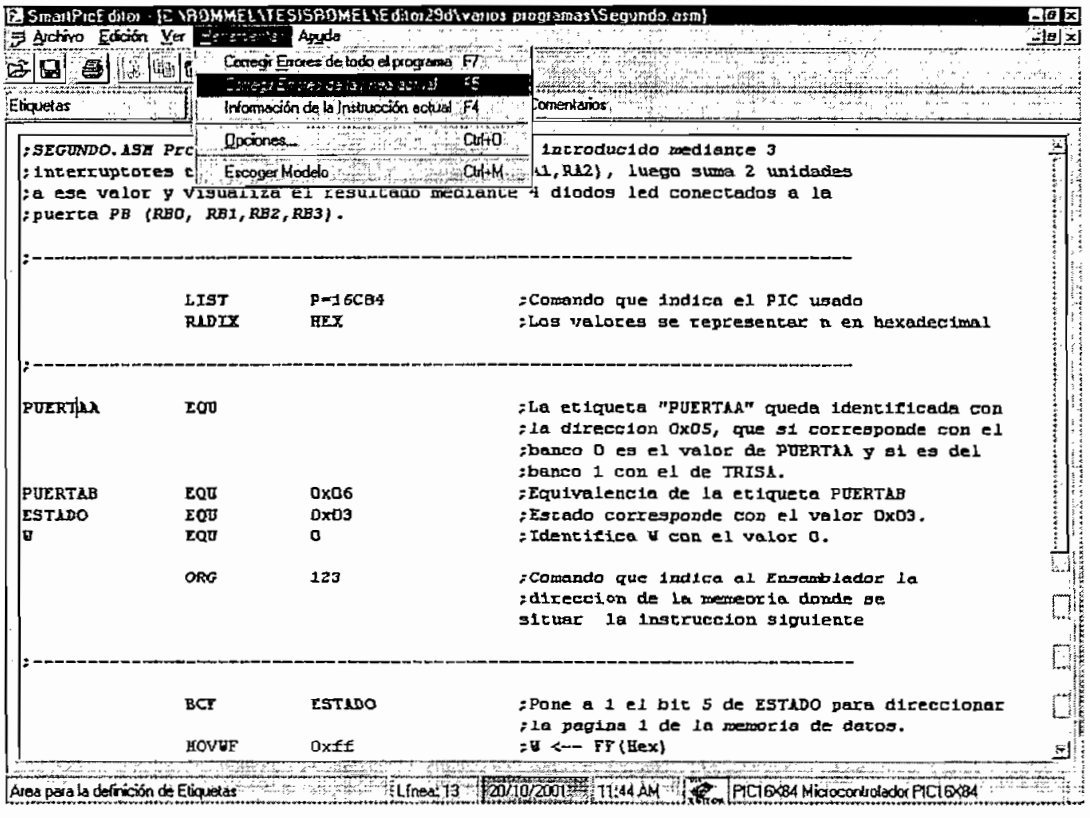


Fig. 4.52 Corrección de errores de una sola línea de programa

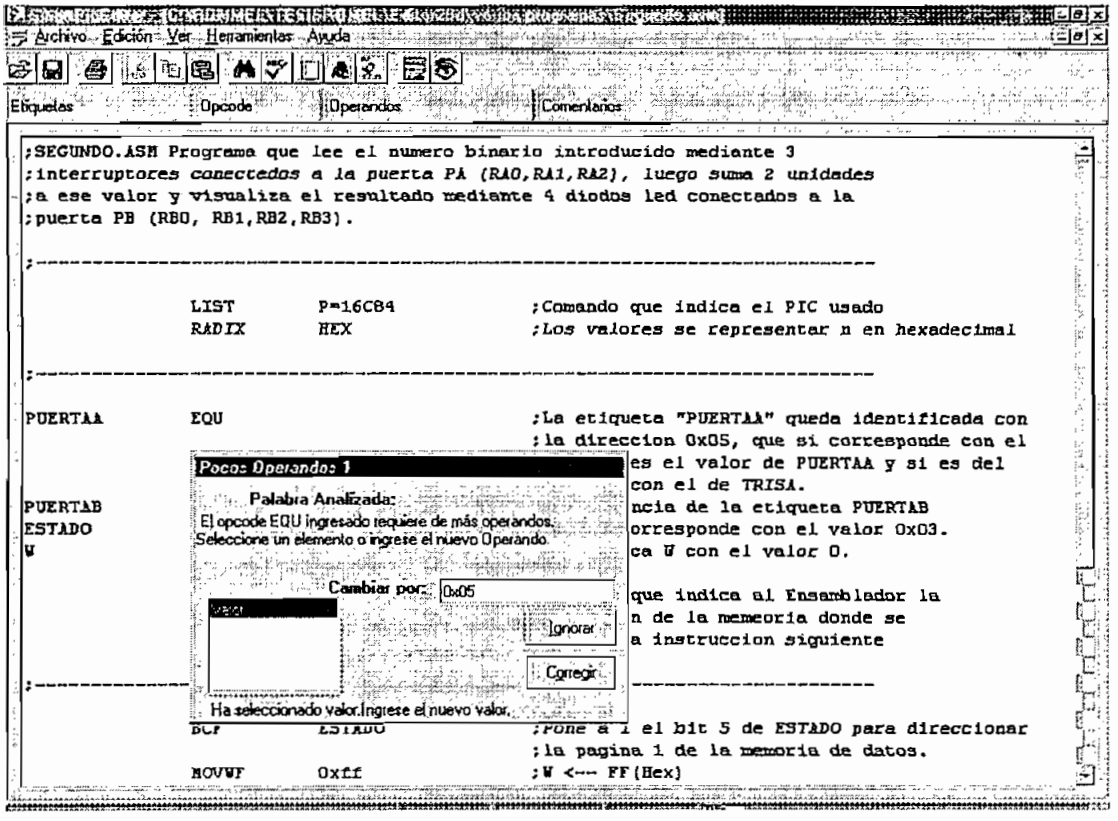


Fig. 4.53 Corrección de una sola línea de programa - directiva EQU

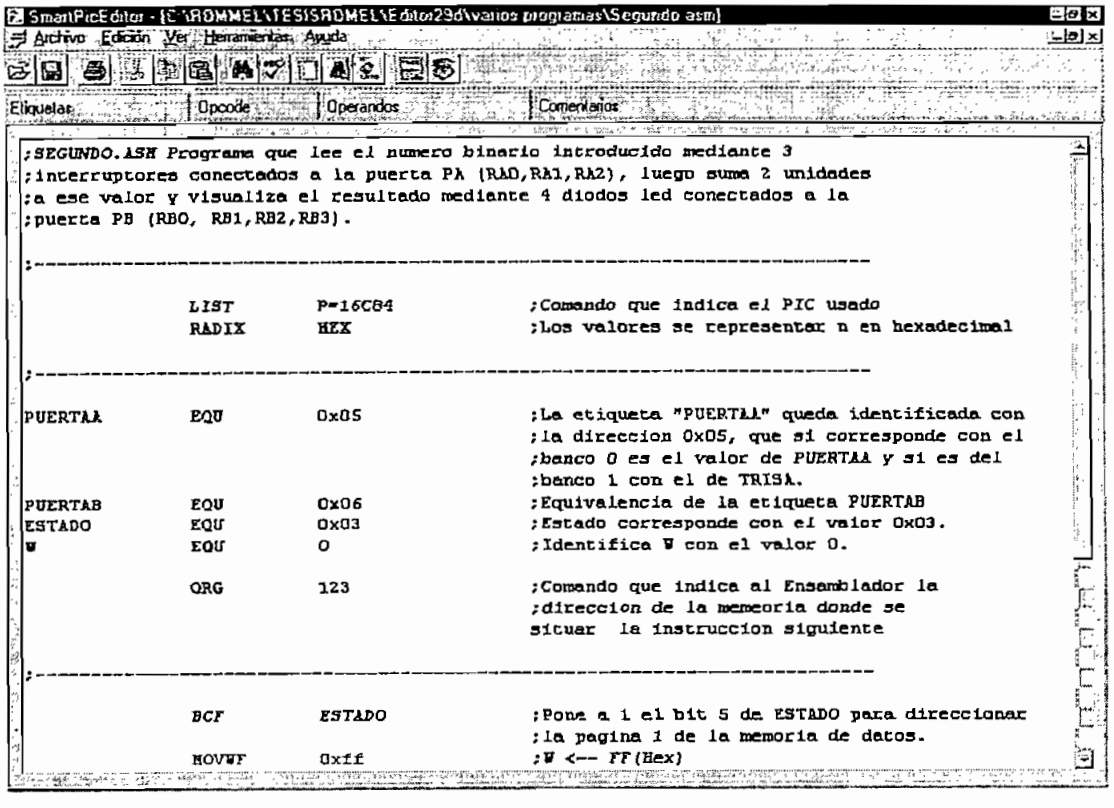


Fig. 4.54 Línea de programa de la directiva EQU corregida.

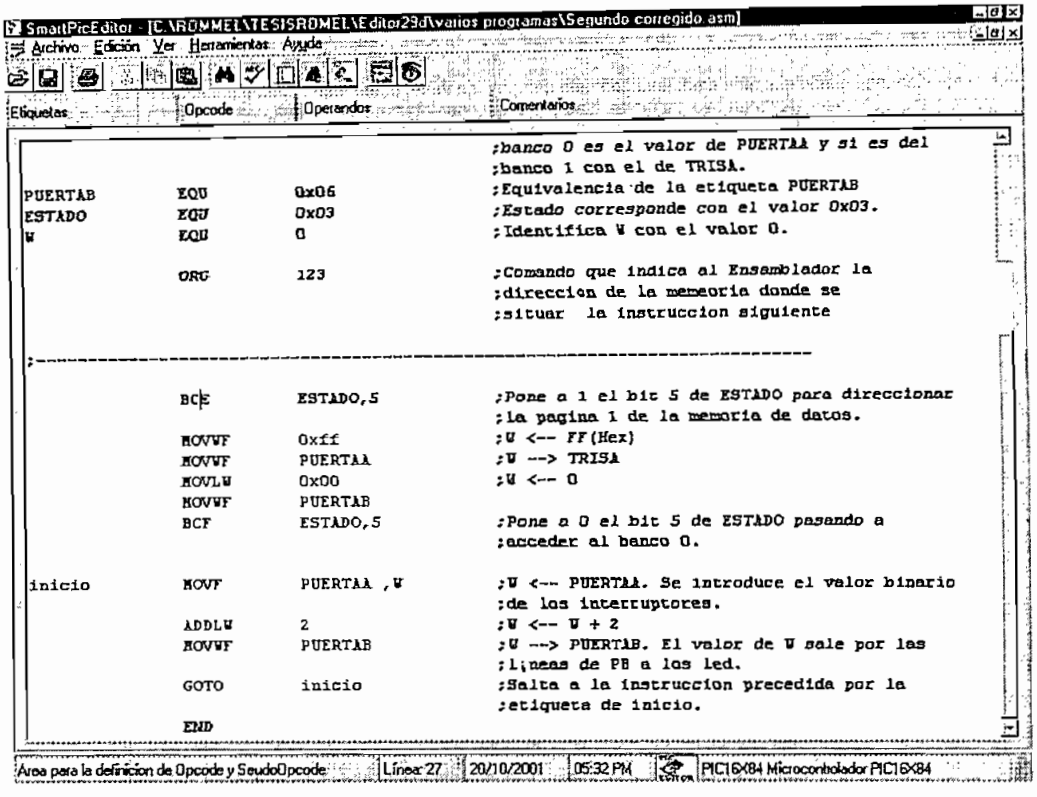


Fig. 4.55 Error de opcode en una línea de programa

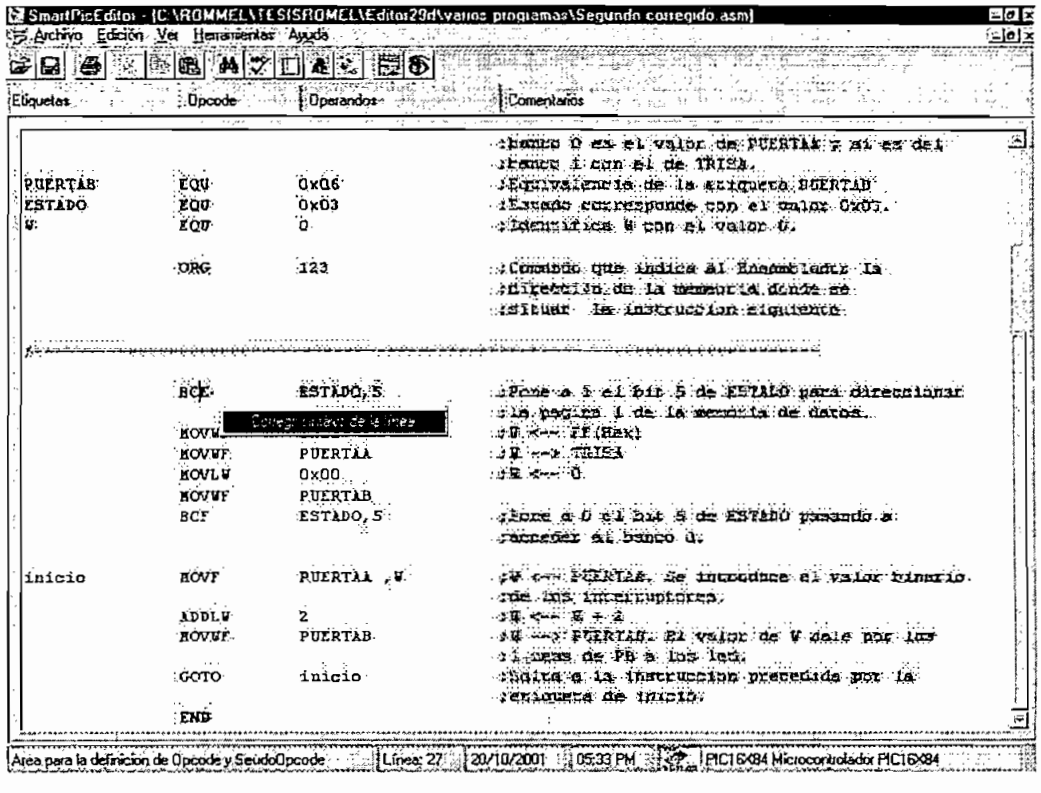


Fig. 4.56 Acceso a la corrección de errores de una línea por medio del botón derecho del ratón

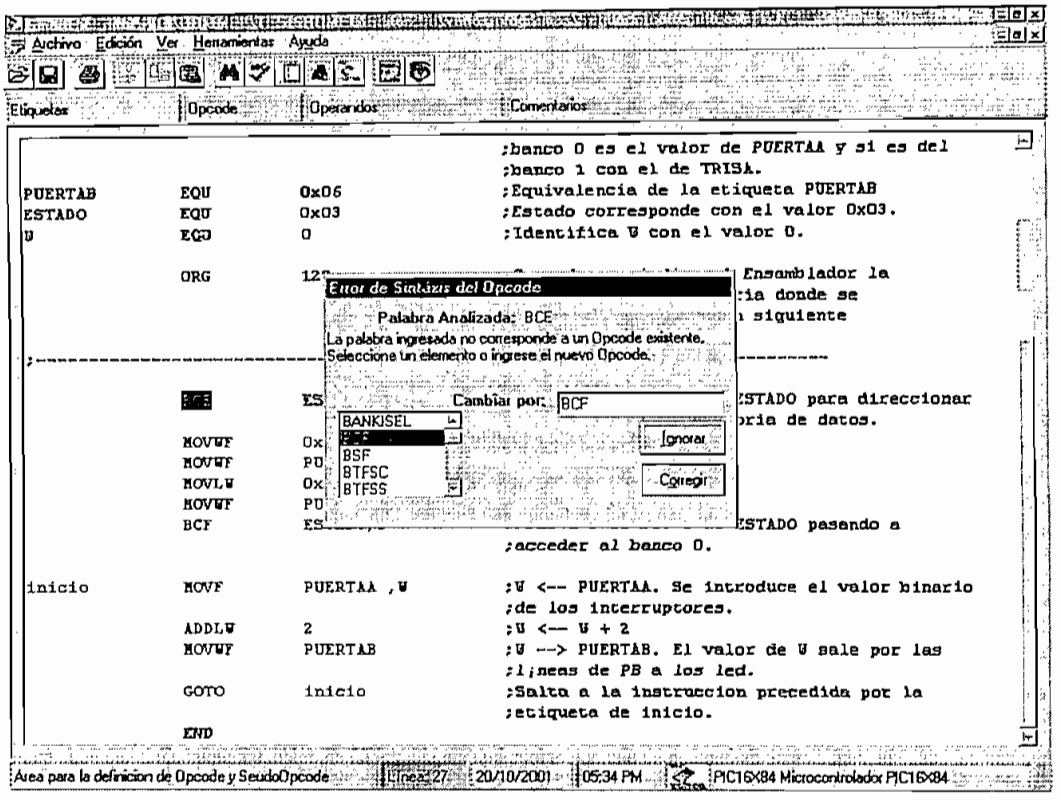


Fig. 4.57 Corrección de error de opcode de una instrucción

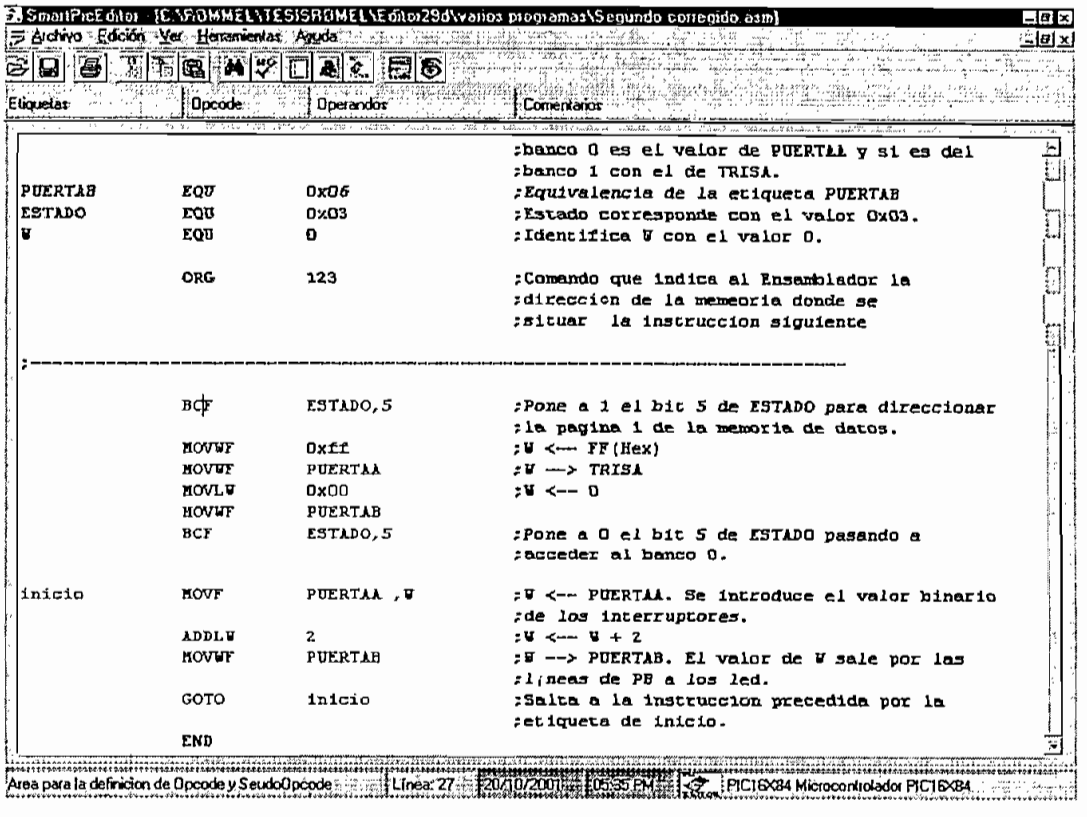


Fig. 4.58 Opcode BCF corregido.

4.4.7 RESUMEN DE PRUEBAS REALIZADAS CON EL SMART PIC EDITOR Y EL ADMIN PIC EDITOR

"Smart Pic Editor" ha sido sometido a pruebas de edición tales como:

- Edición de programas nuevos.
- Apertura y modificación de programas existentes de varias fuentes: programas didácticos de libros, programas bajados desde el internet, tesis de grado y proyectos de titulación que usan el microcontroladores PIC.
- Corrección de errores de etiqueta, opcode, operandos, comentarios intencionalmente cometidos para realizar pruebas.
- Edición de programas extensos para optimizar la velocidad del Smart Pic Editor en el análisis y corrección de errores del programa completo.

"Admin Pic Editor" ha sido sometido a pruebas tales como:

- Introducción del conjunto de instrucciones de microcontroladores de la familia 12C5XX.
- Agregación y eliminación de instrucciones en un modelo existente.
- La edición para cambiar datos de una instrucción existente.
- El vaciado completo de la base de datos.

Es importante mencionar que un aspecto importante a tomarse en cuenta es la velocidad de respuesta del Smart Pic Editor en la apertura de un programa extenso, esto se debe a que el editor analiza la sintaxis de cada una de las instrucciones de todo el programa, para colorear las líneas adecuadamente, en este proceso también se realizan consultas de las instrucciones almacenadas en variables y tomadas temporalmente de la base de datos con el fin de optimizar la

velocidad, razón por la que dependiendo del número de líneas del programa así como de las características del computador en el que se encuentre instalado el Smart Pic Editor, la apertura de un programa tarda algunos segundos.

Para realizar un análisis estadístico de la velocidad con la que el Smart Pic Editor analiza y colorea todo el programa fuente de un microcontrolador, se han escogido siete programas con diferentes números de líneas y se ha instalado el editor en computadoras personales con diferentes características, los resultados de las pruebas realizadas se indican a continuación.

4.4.7.1 Smart Pic Editor instalado en PC Pentium MMX , 233 MHz, 64 MB RAM

No.	PROGRAMA	NUMERO DE LINEAS	TIEMPO DE PROCESAMIENTO para coloreado (segundos)	Velocidad líneas/ segundo
1	lcd.asm	139	4	34.8
2	emisor.asm	158	5	31.6
3	semaforo.asm	287	13	22.1
4	*pruebalaser.asm*	314	9	34.9
5	control.asm	1032	54	19.1
6	chapai.asm	1254	76	16.5
7	principal.asm	1883	112	16.8
	PROMEDIO			**25.1**

4.4.7.2 Smart Pic Editor instalado en PC Pentium II, 333 MHz, 64MB RAM

No.	PROGRAMA	NUMERO DE LINEAS	TIEMPO DE PROCESAMIENTO Para coloreado (segundos)	Velocidad líneas/ segundo
1	lcd.asm	139	3	46.3
2	emisor.asm	158	2	79.0
3	semaforo.asm	287	5	57.4
4	*pruebalaser.asm*	314	4	78.5
5	control.asm	1032	21	49.1
6	chapai.asm	1254	30	41.8
7	principal.asm	1883	38	49.6
	PROMEDIO			**57.4**

4.4.7.2 Smart Pic Editor instalado en PC Pentium III, de 866 MHz, 128MB RAM

No.	PROGRAMA	NUMERO DE LINEAS	TIEMPO DE PROCESAMIENTO Para coloreado (segundos)	Velocidad líneas/ segundo
1	lcd.asm	139	1	139
2	emisor.asm	158	1	158
3	semaforo.asm	287	2	143.5
4	pruebalaser.asm	314	2	157
5	control.asm	1032	8	129
6	chapai.asm	1254	12	104.5
7	principal.asm	1883	16	117.7
	PROMEDIO			135.5

De todas estas pruebas realizadas tanto para el Smart Pic Editor, como para el Admin Pic Editor se concluye que las herramientas desarrolladas cumplen con los requerimientos y expectativas planteadas al iniciar este proyecto. Para un desempeño óptimo del Smart Pic Editor en cuanto a velocidad de respuesta se debe instalarlo en una PC Pentuim II o superior.

CAPITULO 5. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

En la programación del "Smart Pic Editor", se consideró una programación flexible, de tal manera que responda en función de los datos presentes en la base de datos, para ello consulta continuamente la base de datos, para responder adecuadamente con las ayudas en pantalla así como a la hora de detectar y corregir errores.

El "Admin Pic Editor" fue diseñado de manera que la persona encargada de administrar la base de datos realice esta actividad de forma sencilla y confiable pudiendo incorporar nuevos conjuntos de instrucciones de otros microcontroladores, así como el mantenimiento de las instrucciones y directivas existentes.

El editor "Smart Pic Editor" y el administrador "Admin Pic Editor" desarrollados son herramientas totalmente independientes es por ello que cada una de estas herramientas trabajan sobre base de datos distintas, que guardan el mismo formato y tienen el mismo nombre, de tal manera que una vez modificada la base de datos con el uso del Admin Pic Editor, la base de datos del mismo deberá reemplazar a la base de datos del Smart Pic Editor y de esta forma actualizar los cambios realizados con el Administrador.

El conjunto de instrucciones de los microcontroladores PIC al ser tipo RISC, es decir, al contener instrucciones pequeñas y sencillas permite una rápido adaptación del editor desarrollado Smart Pic Editor a un nuevo modelo de microcontrolador con la introducción de las instrucciones por medio del Admin Pic Editor, es claro que el formato de las instrucciones debe ser similar a los de la familia 16X84 que se tomó como referencia para la programación.

Acogiendo las recomendaciones de la tesis "Editor de programas para los microcontroladores INTEL MCS-51/52 que incluye verificación de sintaxis de los

mnemónicos de las instrucciones" desarrollado para microprocesadores MCS 51/52, se creó una interfaz de usuario de similares características, así como se incorporó información de la instrucción iterativa mediante el uso del botón derecho, botón de acceso directo o tecla de funciones F4.

De las pruebas realizadas se concluye que tanto el editor como el administrador son programas estables y cumplen con las expectativas planteadas al inicio del presente proyecto, sin embargo uno de los factores a tener presente es la velocidad de respuesta del editor ante programas extensos; la secuencia de los algoritmos de análisis que utiliza el editor fueron optimizadas adecuadamente tratando que la respuesta del editor sea rápida, sin embargo el tiempo de respuesta es directamente proporcional al número de tareas que Smart Pic Editor debe realizar en el momento de la apertura de un programa así como en la corrección de errores. El desempeño del programa también depende de las características del PC sobre el que está instalada la aplicación, por ello para un desempeño óptimo se recomienda el uso de PC con procesadores Pentium II o superior.

4.2 RECOMENDACIONES

Se recomienda la difusión y uso del Smart Pic Editor, para fines de enseñanza en la programación con microcontroladores PIC.

El Admin Pic Editor debe ser restringido a personal que ha de administrar correctamente la base de datos, con el fin de garantizar la confiabilidad de la información que el editor utiliza y constituirse en ayuda eficaz en la edición de programas.

La estructura interna de la base de datos (número de tablas, registros, campos, longitud de los campos así como los valores que genera y acepta) es fija, el Administrador y el editor fueron programados para responder a un base de datos con una estructura definida es por ello que a la hora de administrar la base de datos, los archivos "BasePic.mdb" generados, deben ser correctamente

gestionados, y de esta manera se pueda utilizar adecuadamente tanto el Smart Pic Editor como el Admin Pic Editor.

REFERENCIAS BIBLIOGRAFICAS

ANGULO, Jose
ANGULO,Ignacio
"MICROCONTROLADORES PIC Diseño práctico de aplicaciones". Editorial McGraw-Hill/ INTERAMERICANA DE ESPAÑA S.A.U. 1997

BASANTES, Wilson
"Editor de programas para los microcontroladores INTEL MCS-51/52 que incluye verificación de sintaxis de los mnemónicos de las instrucciones". Tesis de grado, E.P.N. 2000.

BENSON David
"PIC ´n up the PACE PIC 16/17 MICROCONTROLLER APPLICATIONS GUIDE From Square 1". Version 1.0. Publisher Square 1 Electronics. 1997.

HEYMAN Mark Steven
"La Esencia de Visual Basic 4". SAMS Publishing & Prentice Hall. 1997.

KORTH,Henry
SILBERSCHATZ, Abraham
"FUNDAMENTOS DE BASE DE DATOS". Editorial McGraw-Hill/ INTERAMERICANA DE ESPAÑA S.A.U. 1998.

MICROCHIP Company
"Embedded Control Handbook Volumen 1"Microchip Technology Incorporated. USA.1997.

MICROCHIP Company
"Technical Library CD ROM". July 1999.

MICROCHIP Company WWW.MICROCHIP.COM

MICROSOFT,Corporation "Microsoft Visual Basic. Manual del Programador" 1999.

TANZILLI , Sergio WWW.TANZILLI.COM. Area SX S.r./Informática & Microelectrónica Via Luigi Robecchi Brichetti 13-00154 Roma.

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

ANEXO A

SET DE INSTRUCCIONES DE LOS MICROCONTROLADORES PICs
PIC16C84
PIC12C5XX

9.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-bit word divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The PIC16CXX instruction set summary in Table 9-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 9-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

TABLE 9-1 OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer/Counter
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
dest	Destination either the W register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true or the program counter is changed as a result of an instruction, the *instruction execution time is 2 µs.*

Table 9-2 lists the instructions recognized by the MPASM assembler.

Figure 9-1 shows the general formats that the instructions can have.

Note: To maintain upward compatibility with future PIC16CXX products, do not use the OPTION and TRIS instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

FIGURE 9-1: GENERAL FORMAT FOR INSTRUCTIONS

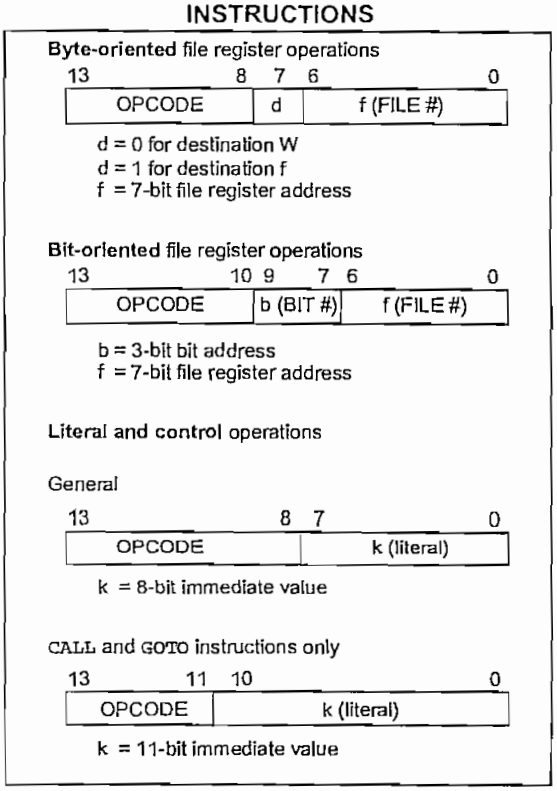


TABLE 9-2 PIC16CXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

9.1 Instruction Descriptions

ADDLW — **Add Literal and W**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (W) + k → (W)

Status Affected: C, DC, Z

Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example:

```
ADDLW   0x15
```

Before Instruction
W = 0x10
After Instruction
W = 0x25

ANDLW — **AND Literal with W**

Syntax: [*label*] ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. (k) → (W)

Status Affected: Z

Encoding:

11	1001	kkkk	kkkk

Description: The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal "k"	Process data	Write to W

Example

```
ANDLW   0x5F
```

Before Instruction
W = 0xA3
After Instruction
W = 0x03

ADDWF — **Add W and f**

Syntax: [*label*] ADDWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) + (f) → (destination)

Status Affected: C, DC, Z

Encoding:

00	0111	dfff	ffff

Description: Add the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
ADDWF   FSR, 0
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0xD9
FSR = 0xC2

ANDWF — **AND W with f**

Syntax: [*label*] ANDWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) .AND. (f) → (destination)

Status Affected: Z

Encoding:

00	0101	dfff	ffff

Description: AND the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
ANDWF   FSR, 1
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0x17
FSR = 0x02

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $0 \rightarrow (f<b>)$

Status Affected: None

Encoding:

01	00bb	bfff	ffff

Description: Bit 'b' in register 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example

```
BCF     FLAG_REG, 7
```

Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BSF **Bit Set f**

Syntax: [*label*] BSF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $1 \rightarrow (f<b>)$

Status Affected: None

Encoding:

01	01bb	bfff	ffff

Description: Bit 'b' in register 'f' is set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example

```
BSF     FLAG_REG,   7
```

Before Instruction
FLAG_REG = 0x0A
After Instruction
FLAG_REG = 0x8A

BTFSC **Bit Test, Skip if Clear**

Syntax: [*label*] BTFSC f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: skip if $(f<b>) = 0$

Status Affected: None

Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '1' then the next instruction is executed.
If bit 'b', in register 'f', is '0' then the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No-Operation

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address TRUE
if FLAG<1>=1,
PC = address FALSE

BTFSS **Bit Test f, Skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: $0 \le f \le 127$
$0 \le b < 7$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

01	11bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0' then the next instruction is executed.
If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read *register 'f'*	Process data	No-Operation

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address FALSE
if FLAG<1> = 1,
PC = address TRUE

CALL **Call Subroutine**

Syntax: [*label*] CALL k

Operands: $0 \le k \le 2047$

Operation: (PC)+ 1→ TOS,
k → PC<10:0>,
(PCLATH<4:3>) → PC<12:11>

Status Affected: None

Encoding:

10	0kkk	kkkk	kkkk

Description: Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k', Push PC to Stack	Process data	Write to PC
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE    CALL    THERE
```

Before Instruction
PC = Address HERE
After Instruction
PC = Address THERE
TOS = Address HERE+1

CLRF **Clear f**

Syntax: [*label*] CLRF f

Operands: $0 \le f \le 127$

Operation: 00h → (f)
1 → Z

Status Affected: Z

Encoding:

00	0001	lfff	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example

```
CLRF    FLAG_REG
```

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00
Z = 1

CLRW **Clear W**

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → (W)
1 → Z

Status Affected: Z

Encoding:

00	0001	0xxx	xxxx

Description: W register is cleared. Zero bit (Z) is set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	*No-Operation*	*Process data*	*Write to W*

Example

```
CLRW
```

Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
0 → WDT prescaler,
1 → $\overline{TO}$
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

00	0000	0110	0100

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	Process data	Clear WDT Counter

Example

```
CLRWDT
```

Before Instruction
WDT counter = ?
After Instruction
WDT counter = 0x00
WDT prescaler= 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **Complement f**

Syntax: [*label*] COMF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ (destination)

Status Affected: Z

Encoding:

00	1001	dfff	ffff

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
COMF    REG1,0
```

Before Instruction
REG1 = 0x13
After Instruction
REG1 = 0x13
W = 0xEC

DECF **Decrement f**

Syntax: [*label*] DECF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 → (destination)

Status Affected: Z

Encoding:

00	0011	dfff	ffff

Description: Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
DECF    CNT, 1
```

Before Instruction
CNT = 0x01
Z = 0
After Instruction
CNT = 0x00
Z = 1

DECFSZ **Decrement f, Skip if 0**

Syntax: [*label*] DECFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 → (destination); skip if result = 0

Status Affected: None

Encoding:

00	1011	dfff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 1, the next instruction, is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE    DECFSZ  CNT, 1
        GOTO    LOOP
CONTINUE •
         •
         •
```

Before Instruction
PC = address HERE
After Instruction
CNT = CNT - 1
if CNT = 0,
PC = address CONTINUE
if CNT ≠ 0,
PC = address HERE+1

GOTO **Unconditional Branch**

Syntax: [*label*] GOTO k

Operands: $0 \le k \le 2047$

Operation: k → PC<10:0>
PCLATH<4:3> → PC<12:11>

Status Affected: None

Encoding:

10	1kkk	kkkk	kkkk

Description: GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k'	Process data	Write to PC
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
GOTO THERE
```

After Instruction
PC = Address THERE

INCF **Increment f**

Syntax: [*label*] INCF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (destination)

Status Affected: Z

Encoding:

00	1010	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
INCF    CNT, 1
```

Before Instruction
CNT = 0xFF
Z = 0
After Instruction
CNT = 0x00
Z = 1

INCFSZ **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (destination), skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead making it a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE     INCFSZ    CNT, 1
         GOTO      LOOP
CONTINUE •
         •
         •
```

Before Instruction
PC = address HERE
After Instruction
CNT = CNT + 1
if CNT= 0,
PC = address CONTINUE
if CNT≠ 0,
PC = address HERE +1

IORLW **Inclusive OR Literal with W**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → (W)

Status Affected: Z

Encoding:

11	1000	kkkk	kkkk

Description: The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example

```
IORLW   0x35
```

Before Instruction
W = 0x9A
After Instruction
W = 0xBF
Z = 1

IORWF **Inclusive OR W with f**

Syntax: [*label*] IORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) .OR. (f) → (destination)

Status Affected: Z

Encoding:

00	0100	dfff	ffff

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example
```
IORWF   RESULT, 0
```
Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93
Z = 1

MOVF **Move f**

Syntax: [*label*] MOVF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) → (destination)

Status Affected: Z

Encoding:

00	1000	dfff	ffff

Description: The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example
```
MOVF    FSR, 0
```
After Instruction
W = value in FSR register
Z = 1

MOVLW **Move Literal to W**

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: k → (W)

Status Affected: None

Encoding:

11	00xx	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example
```
MOVLW   0x5A
```
After Instruction
W = 0x5A

MOVWF **Move W to f**

Syntax: [*label*] MOVWF f

Operands: $0 \le f \le 127$

Operation: (W) → (f)

Status Affected: None

Encoding:

00	0000	1fff	ffff

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example
```
MOVWF   OPTION_REG
```
Before Instruction
OPTION = 0xFF
W = 0x4F
After Instruction
OPTION = 0x4F
W = 0x4F

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	00 | 0000 | 0xx0 | 0000
Description:	No operation.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	No-Operation	No-Operation

Example

```
NOP
```

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	00 | 0000 | 0000 | 1001
Description:	Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Operation	Set the GIE bit	Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
RETFIE
```

After Interrupt
PC = TOS
GIE = 1

OPTION	Load Option Register
Syntax:	[*label*] OPTION
Operands:	None
Operation:	(W) → OPTION
Status Affected:	None
Encoding:	00 | 0000 | 0110 | 0010
Description:	The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.
Words:	1
Cycles:	1
Example	To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

RETLW **Return with Literal in W**

Syntax: [*label*] RETLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow (W)$;
TOS → PC

Status Affected: None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k'	No-Operation	Write to W, Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
CALL TABLE  ;W contains table
            ;offset value
            ;W now has table value
TABLE ADDWF PC   ;W = offset
      RETLW k1   ;Begin table
      RETLW k2   ;
      RETLW kn   ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Operation	No-Operation	Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
RETURN
```

After Interrupt
PC = TOS

RLF **Rotate Left f through Carry**

Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

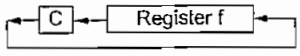
Operation: *See description below*

Status Affected: C

Encoding:

00	1101	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
RLF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

RRF **Rotate Right f through Carry**

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

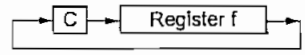
Operation: *See description below*

Status Affected: C

Encoding:

00	1100	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
RRF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 0111 0011
C = 0

SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

00	0000	0110	0011

Description: The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See *Section 14.8 for more details.*

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	No-Operation	Go to Sleep

Example: `SLEEP`

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: *The W register is subtracted* (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example 1: `SUBLW 0x02`

Before Instruction
W = 1
C = ?
Z = ?

After Instruction
W = 1
C = 1; result is positive
Z = 0

Example 2: Before Instruction
W = 2
C = ?
Z = ?

After Instruction
W = 0
C = 1; result is zero
Z = 1

Example 3: Before Instruction
W = 3
C = ?
Z = ?

After Instruction
W = 0xFF
C = 0; result is negative
Z = 0

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1:
```
SUBWF    REG1,1
```
Before Instruction
```
REG1 = 3
W    = 2
C    = ?
Z    = ?
```
After Instruction
```
REG1 = 1
W    = 2
C    = 1; result is positive
Z    = 0
```
Example 2: Before Instruction
```
REG1 = 2
W    = 2
C    = ?
Z    = ?
```
After Instruction
```
REG1 = 0
W    = 2
C    = 1; result is zero
Z    = 1
```
Example 3: Before Instruction
```
REG1 = 1
W    = 2
C    = ?
Z    = ?
```
After Instruction
```
REG1 = 0xFF
W    = 2
C    = 0; result is negative
Z    = 0
```

SWAPF **Swap Nibbles in f**

Syntax: [*label*] SWAPF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<3:0>) → (destination<7:4>),
(f<7:4>) → (destination<3:0>)

Status Affected: None

Encoding:

00	1110	dfff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example
```
SWAPF REG,  0
```
Before Instruction
```
REG1 = 0xA5
```
After Instruction
```
REG1 = 0xA5
W    = 0x5A
```

TRIS **Load TRIS Register**

Syntax: [*label*] TRIS f

Operands: $5 \le f \le 7$

Operation: (W) → TRIS register f;

Status Affected: None

Encoding:

00	0000	0110	0fff

Description: The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.

Words: 1

Cycles: 1

Example

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

XORLW **Exclusive OR Literal with W**

Syntax: *[label]* XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → (W)

Status Affected: Z

Encoding:

11	1010	kkkk	kkkk

Description: *The contents of the W register are* XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example:

```
XORLW   0xAF
```

Before Instruction

W = 0xB5

After Instruction

W = 0x1A

XORWF **Exclusive OR W with f**

Syntax: *[label]* XORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) .XOR. (f) → (destination)

Status Affected: Z

Encoding:

00	0110	dfff	ffff

Description: Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read *register* 'f'	Process data	Write to *destination*

Example

```
XORWF   REG  1
```

Before Instruction

REG = 0xAF
W = 0xB5

After Instruction

REG = 0x1A
W = 0xB5

9.0 INSTRUCTION SET SUMMARY

Each PIC12C5XX instruction is a 12-bit word divided into an OPCODE, which specifies the instruction type, and one or more operands which further specify the operation of the instruction. The PIC12C5XX instruction set summary in Table 9-2 groups the instructions into byte-oriented, bit-oriented, and literal and control operations. Table 9-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator is used to specify which one of the 32 file registers is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control** operations, 'k' represents an 8 or 9-bit constant or literal value.

TABLE 9-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0 (store result in W) d = 1 (store result in file register 'f') Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-Out bit
$\overline{PD}$	Power-Down bit
dest	Destination, either the W register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

All instructions are executed within a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles. One *instruction cycle consists of four oscillator periods.* Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs.

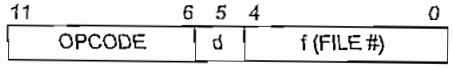
Figure 9-1 shows the three general formats that the instructions can have. All examples in the figure use the following format to represent a hexadecimal number:

0xhhh

where 'h' signifies a hexadecimal digit.

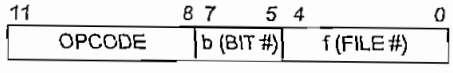
FIGURE 9-1: GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations



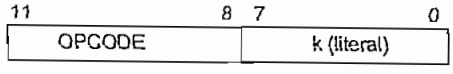
d = 0 for destination W
d = 1 for destination f
f = 5-bit file register address

Bit-oriented file register operations



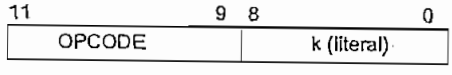
b = 3-bit bit address
f = 5-bit file register address

Literal and control operations (except GOTO)



k = 8-bit immediate value

Literal and control operations - GOTO instruction



k = 9-bit immediate value

TABLE 9-2: INSTRUCTION SET SUMMARY

Mnemonic, Operands		Description	Cycles	12-Bit Opcode MSb		LSb	Status Affected	Notes
ADDWF	f,d	Add W and f	1	0001	11df	ffff	C,DC,Z	1,2,4
ANDWF	f,d	AND W with f	1	0001	01df	ffff	Z	2,4
CLRF	f	Clear f	1	0000	011f	ffff	Z	4
CLRW	–	Clear W	1	0000	0100	0000	Z	
COMF	f, d	Complement f	1	0010	01df	ffff	Z	
DECF	f, d	Decrement f	1	0000	11df	ffff	Z	2,4
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	0010	11df	ffff	None	2,4
INCF	f, d	Increment f	1	0010	10df	ffff	Z	2,4
INCFSZ	f, d	Increment f, Skip if 0	1(2)	0011	11df	ffff	None	2,4
IORWF	f, d	Inclusive OR W with f	1	0001	00df	ffff	Z	2,4
MOVF	f, d	Move f	1	0010	00df	ffff	Z	2,4
MOVWF	f	Move W to f	1	0000	001f	ffff	None	1,4
NOP	–	No Operation	1	0000	0000	0000	None	
RLF	f, d	Rotate left f through Carry	1	0011	01df	ffff	C	2,4
RRF	f, d	Rotate right f through Carry	1	0011	00df	ffff	C	2,4
SUBWF	f, d	Subtract W from f	1	0000	10df	ffff	C,DC,Z	1,2,4
SWAPF	f, d	Swap f	1	0011	10df	ffff	None	2,4
XORWF	f, d	Exclusive OR W with f	1	0001	10df	ffff	Z	2,4
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit Clear f	1	0100	bbbf	ffff	None	2,4
BSF	f, b	Bit Set f	1	0101	bbbf	ffff	None	2,4
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	0110	bbbf	ffff	None	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	0111	bbbf	ffff	None	
LITERAL AND CONTROL OPERATIONS								
ANDLW	k	AND literal with W	1	1110	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	1001	kkkk	kkkk	None	1
CLRWDT	k	Clear Watchdog Timer	1	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
GOTO	k	Unconditional branch	2	101k	kkkk	kkkk	None	
IORLW	k	Inclusive OR Literal with W	1	1101	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	1100	kkkk	kkkk	None	
OPTION	–	Load OPTION register	1	0000	0000	0010	None	
RETLW	k	Return, place Literal in W	2	1000	kkkk	kkkk	None	
SLEEP	–	Go into standby mode	1	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register	1	0000	0000	0fff	None	3
XORLW	k	Exclusive OR Literal to W	1	1111	kkkk	kkkk	Z	

Note 1: The 9th bit of the program counter will be forced to a '0' by any instruction that writes to the PC except for GOTO. (Section 4.6)

2: When an I/O register is modified as a function of itself (e.g. MOVF GPIO, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

3: The instruction TRIS f, where f = 6 causes the contents of the W register to be written to the tristate latches of GPIO. A '1' forces the pin to a hi-impedance state and disables the output buffers.

4: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared (if assigned to TMR0).

ADDWF **Add W and f**

Syntax: [*label*] ADDWF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (W) + (f) → (dest)

Status Affected: C, DC, Z

Encoding:

0001	11df	ffff

Description: Add the contents of the W register and register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is '1' the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example:
```
ADDWF   FSR, 0
```
Before Instruction
W = 0x17
FSR = 0xC2

After Instruction
W = 0xD9
FSR = 0xC2

ANDLW **And literal with W**

Syntax: [*label*] ANDLW k

Operands: $0 \le k \le 255$

Operation: (W).AND. (k) → (W)

Status Affected: Z

Encoding:

1110	kkkk	kkkk

Description: The contents of the W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example:
```
ANDLW   0x5F
```
Before Instruction
W = 0xA3

After Instruction
W = 0x03

ANDWF **AND W with f**

Syntax: [*label*] ANDWF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (W) .AND. (f) → (dest)

Status Affected: Z

Encoding:

0001	01df	ffff

Description: The contents of the W register are AND'ed with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is '1' the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example:
```
ANDWF   FSR,    1
```
Before Instruction
W = 0x17
FSR = 0xC2

After Instruction
W = 0x17
FSR = 0x02

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b

Operands: $0 \le f \le 31$
$0 \le b \le 7$

Operation: 0 → (f<b>)

Status Affected: None

Encoding:

0100	bbbf	ffff

Description: Bit 'b' in register 'f' is cleared.

Words: 1

Cycles: 1

Example:
```
BCF     FLAG_REG,  7
```
Before Instruction
FLAG_REG = 0xC7

After Instruction
FLAG_REG = 0x47

BSF **Bit Set f**

Syntax: [*label*] BSF f,b

Operands: $0 \le f \le 31$
$0 \le b \le 7$

Operation: $1 \rightarrow (f<b>)$

Status Affected: None

Encoding:

0101	bbbf	ffff

Description: Bit 'b' in register 'f' is set.

Words: 1

Cycles: 1

Example:
```
BSF     FLAG_REG,   7
```
Before Instruction
FLAG_REG = 0x0A

After Instruction
FLAG_REG = 0x8A

BTFSC **Bit Test f, Skip if Clear**

Syntax: [*label*] BTFSC f,b

Operands: $0 \le f \le 31$
$0 \le b \le 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

0110	bbbf	ffff

Description: If bit 'b' in register 'f' is 0 then the next instruction is skipped.
If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and an NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example:
```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0,
PC = address (TRUE);
If FLAG<1> = 1,
PC = address (FALSE)

BTFSS **Bit Test f, Skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: $0 \le f \le 31$
$0 \le b < 7$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

0111	bbbf	ffff

Description: If bit 'b' in register 'f' is '1' then the next instruction is skipped.
If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example:
```
HERE    BTFSS   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0,
PC = address (FALSE);
If FLAG<1> = 1,
PC = address (TRUE)

CALL **Subroutine Call**

Syntax: [*label*] CALL k

Operands: 0 ≤ k ≤ 255

Operation: (PC) + 1→ Top of Stack;
k → PC<7:0>;
(STATUS<6:5>) → PC<10:9>;
0 → PC<8>

Status Affected: None

Encoding:

1001	kkkk	kkkk

Description: Subroutine call. First, return address (PC+1) is pushed onto the stack. The eight bit immediate address is loaded into PC bits <7:0>. The upper bits PC<10:9> are loaded from STATUS<6:5>, PC<8> is cleared. CALL is a two cycle instruction.

Words: 1

Cycles: 2

Example:

```
HERE    CALL    THERE
```

Before Instruction
PC = address (HERE)

After Instruction
PC = address (THERE)
TOS = address (HERE + 1)

CLRF **Clear f**

Syntax: [*label*] CLRF f

Operands: 0 ≤ f ≤ 31

Operation: 00h → (f);
1 → Z

Status Affected: Z

Encoding:

0000	011f	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Example:

```
CLRF    FLAG_REG
```

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00
Z = 1

CLRW **Clear W**

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → (W);
1 → Z

Status Affected: Z

Encoding:

0000	0100	0000

Description: The W register is cleared. Zero bit (Z) is set.

Words: 1

Cycles: 1

Example:

```
CLRW
```

Before Instruction
W = 0x5A

After Instruction
W = 0x00
Z = 1

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT;
0 → WDT prescaler (if assigned);
1 → $\overline{TO}$;
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0100

Description: The CLRWDT instruction resets the WDT. It also resets the prescaler, if the prescaler is assigned to the WDT and not Timer0. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Example:

```
CLRWDT
```

Before Instruction
WDT counter = ?

After Instruction
WDT counter = 0x00
WDT prescale = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **Complement f**

Syntax: [*label*] COMF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: $(\bar{f}) \rightarrow (dest)$

Status Affected: Z

Encoding:

0010	01df	ffff

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example:
```
COMF    REG1,0
```
Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
W = 0xEC

DECF **Decrement f**

Syntax: [*label*] DECF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f) – 1 → (dest)

Status Affected: Z

Encoding:

0000	11df	ffff

Description: Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example:
```
DECF    CNT,    1
```
Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

DECFSZ **Decrement f, Skip if 0**

Syntax: [*label*] DECFSZ f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f) – 1 → d; *skip if result = 0*

Status Affected: None

Encoding:

0010	11df	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

If the result is 0, the next instruction, which is already fetched, is discarded and an NOP is executed instead making it a two cycle instruction.

Words: 1

Cycles: 1(2)

Example:
```
HERE    DECFSZ  CNT, 1
        GOTO    LOOP
CONTINUE •
         •
         •
```
Before Instruction
PC = address (HERE)

After Instruction
CNT = CNT - 1;
if CNT = 0,
PC = address (CONTINUE);
if CNT ≠ 0,
PC = address (HERE+1)

GOTO **Unconditional Branch**

Syntax: [*label*] GOTO k

Operands: $0 \le k \le 511$

Operation: k → PC<8:0>;
STATUS<6:5> → PC<10:9>

Status Affected: None

Encoding:

101k	kkkk	kkkk

Description: GOTO is an unconditional branch. The 9-bit immediate value is loaded into PC bits <8:0>. The upper bits of PC are loaded from STATUS<6:5>. GOTO is a two cycle instruction.

Words: 1

Cycles: 2

Example:
```
GOTO THERE
```
After Instruction
PC = address (THERE)

INCF **Increment f**

Syntax: [*label*] INCF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f) + 1 → (dest)

Status Affected: Z

Encoding:

0010	10df	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example:
```
INCF    CNT,    1
```
Before Instruction
CNT = 0xFF
Z = 0

After Instruction
CNT = 0x00
Z = 1

INCFSZ **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f) + 1 → (dest), skip if result = 0

Status Affected: None

Encoding:

0011	11df	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

If the result is 0, then the next instruction, which is already fetched, is discarded and an NOP is executed *instead making it a two cycle instruction.*

Words: 1

Cycles: 1(2)

Example:
```
HERE    INCFSZ    CNT,    1
        GOTO      LOOP
CONTINUE •
         •
         •
```
Before Instruction
PC = address (HERE)

After Instruction
CNT = CNT + 1;
if CNT = 0,
PC = address (CONTINUE);
if CNT ≠ 0,
PC = address (HERE +1)

IORLW **Inclusive OR literal with W**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. (k) → (W)

Status Affected: Z

Encoding:

1101	kkkk	kkkk

Description: The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example:
```
IORLW   0x35
```
Before Instruction
W = 0x9A

After Instruction
W = 0xBF
Z = 0

IORWF **Inclusive OR W with f**

Syntax: [*label*] IORWF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (W).OR. (f) → (dest)

Status Affected: Z

Encoding:

0001	00df	ffff

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example:
```
IORWF           RESULT, 0
```
Before Instruction
RESULT = 0x13
W = 0x91

After Instruction
RESULT = 0x13
W = 0x93
Z = 0

MOVF **Move f**

Syntax: [*label*] MOVF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f) → (dest)

Status Affected: Z

Encoding:

0010	00df	ffff

Description: The contents of register 'f' is moved to destination 'd'. If 'd' is 0, destination is the W register. If 'd' is 1, the destination is file register 'f'. 'd' is 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Example:
```
MOVF    FSR,    0
```
After Instruction
W = value in FSR register

MOVLW **Move Literal to W**

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: k → (W)

Status Affected: None

Encoding:

1100	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into the W register. The don't cares will assemble as 0s.

Words: 1

Cycles: 1

Example:
```
MOVLW   0x5A
```
After Instruction
W = 0x5A

MOVWF **Move W to f**

Syntax: [*label*] MOVWF f

Operands: $0 \le f \le 31$

Operation: (W) → (f)

Status Affected: None

Encoding:

0000	001f	ffff

Description: Move data from the W register to register 'f'.

Words: 1

Cycles: 1

Example:
```
MOVWF   TEMP_REG
```
Before Instruction
TEMP_REG = 0xFF
W = 0x4F

After Instruction
TEMP_REG = 0x4F
W = 0x4F

NOP **No Operation**

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

0000	0000	0000

Description: No operation.

Words: 1

Cycles: 1

Example:
```
NOP
```

OPTION **Load OPTION Register**

Syntax: [*label*] OPTION

Operands: None

Operation: (W) → *OPTION*

Status Affected: None

Encoding:

0000	0000	0010

Description: The content of the W register is loaded into the OPTION register.

Words: 1

Cycles: 1

Example `OPTION`

Before Instruction
W = 0x07

After Instruction
OPTION = 0x07

RETLW **Return with Literal in W**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W);
TOS → PC

Status Affected: None

Encoding:

1000	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Example:

```
CALL TABLE ;W contains
           ;table offset
           ;value.
•          ;W now has table
•          ;value.
•
TABLE      ADDWF PC   ;W = offset
           RETLW k1   ;Begin table
           RETLW k2   ;
           •
           •
           •
           RETLW kn   ; End of table
```

Before Instruction
W = *0x07*

After Instruction
W = *value of k8*

RLF **Rotate Left f through Carry**

Syntax: [*label*] RLF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

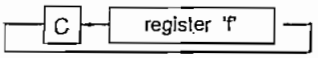
Operation: See description below

Status Affected: C

Encoding:

0011	01df	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1

Cycles: 1

Example:

```
RLF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0

After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

RRF **Rotate Right f through Carry**

Syntax: [*label*] RRF f,d

Operands: 0 ≤ f ≤ 31
d ∈ [0,1]

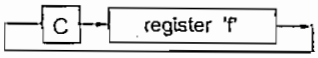
Operation: See description below

Status Affected: C

Encoding:

0011	00df	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Example:

```
RRF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0

After Instruction
REG1 = 1110 0110
W = 0111 0011
C = 0

SLEEP — **Enter SLEEP Mode**

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT;
0 → WDT prescaler;
1 → $\overline{TO}$;
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$, GPWUF

Encoding:

0000	0000	0011

Description: *Time-out status bit ($\overline{TO}$) is set. The* power down status bit ($\overline{PD}$) is cleared.

GPWUF is unaffected.

The WDT and its prescaler are cleared.

The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP for more details.

Words: 1

Cycles: 1

Example: `SLEEP`

SUBWF — **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f) – (W) → (dest)

Status Affected: C, DC, Z

Encoding:

0000	10df	ffff

Description: Subtract (2's complement method) the W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example 1: `SUBWF REG1, 1`

Before Instruction
REG1 = 3
W = 2
C = ?

After Instruction
REG1 = 1
W = 2
C = 1 ; result is positive

Example 2:

Before Instruction
REG1 = 2
W = 2
C = ?

After Instruction
REG1 = 0
W = 2
C = 1 ; result is zero

Example 3:

Before Instruction
REG1 = 1
W = 2
C = ?

After Instruction
REG1 = FF
W = 2
C = 0 ; result is negative

SWAPF **Swap Nibbles in f**

Syntax: [*label*] SWAPF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (f<3:0>) → (dest<7:4>);
(f<7:4>) → (dest<3:0>)

Status Affected: None

Encoding:

0011	10df	ffff

Description: *The upper and lower nibbles of register* 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.

Words: 1

Cycles: 1

Example `SWAPF REG1, 0`

Before Instruction
REG1 = 0xA5

After Instruction
REG1 = 0xA5
W = 0X5A

TRIS **Load TRIS Register**

Syntax: [*label*] TRIS f

Operands: f = 6

Operation: (W) → TRIS register f

Status Affected: None

Encoding:

0000	0000	0fff

Description: TRIS register 'f' (f = 6) is loaded with the contents of the W register

Words: 1

Cycles: 1

Example `TRIS GPIO`

Before Instruction
W = 0XA5

After Instruction
TRIS = 0XA5

Note: f = 6 for PIC12C5XX only.

XORLW **Exclusive OR literal with W**

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → (W)

Status Affected: Z

Encoding:

1111	kkkk	kkkk

Description: The contents of the W register are XOR'ed with the eight bit literal 'k'. The *result is placed in the* W register.

Words: 1

Cycles: 1

Example: `XORLW 0xAF`

Before Instruction
W = 0xB5

After Instruction
W = 0x1A

XORWF **Exclusive OR W with f**

Syntax: [*label*] XORWF f,d

Operands: $0 \le f \le 31$
$d \in [0,1]$

Operation: (W) .XOR. (f) → (dest)

Status Affected: Z

Encoding:

0001	10df	ffff

Description: Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example `XORWF REG,1`

Before Instruction
REG = 0xAF
W = 0xB5

After Instruction
REG = 0x1A
W = 0xB5

ANEXO B

MANUAL DEL USUARIO

Manual del Usuario

del

Editor Inteligente para Microcontroladores PIC

"Smart PIC Editor "

Manual del Usuario del Editor Inteligente para Microcontroladores PIC. "Smart Pic Editor"

1. Ambito

Este manual tiene como objetivo servir de guía para que el usuario pueda conocer y utilizar el Editor Inteligente para Microcontroladores Pic, al cual se ha denominado "Smart Pic Editor".

A lo largo de este documento se proporciona el material necesario para asistir al usuario en el aprendizaje de todos los aspectos que conllevan el uso de este editor.

> *Este manual está orientado a personas involucradas en el desarrollo de programas para microcontroladores PIC, sean estudiantes, profesores, ingenieros, etc.*

2. Estructura de este Manual

Con el fin de lograr la mayor claridad posible en la explicación de los diferentes tópicos correspondientes al uso del Editor se ha organizado el presente manual de la siguiente manera:

Acerca de Smart PIC Editor	Proporciona una descripción general del editor
Términos utilizados	Se explican brevemente los términos utilizados tanto en el manual como en el programa editor
Estructura del Editor	En esta sección se describe la estructura general de la pantalla de edición y sus componentes.
Opciones de Configuración	En esta sección se describen las opciones de configuración del editor y selección de modelos.
Edición de Programas	Aquí se explican en detalle los pasos para utilizar las opciones de edición de Smart PIC Editor.
Corrección de Errores	Se explica las formas como se puede acceder a la corrección de errores, los diferentes tipos de errores de sintaxis que puede corregir Smart Pic Editor.

3. Acerca de Smart PIC Editor

Bienvenido a Smart PIC Editor, este editor ha sido creado con el fin de brindar facilidades durante la edición de un programa y el análisis de la sintaxis de las instrucciones de acuerdo a los conjuntos de instrucciones de diferentes modelos de microcontroladores PIC.

4. Términos Utilizados

En esta sección se describen algunos de los términos utilizados tanto a lo largo del manual, como en el editor.

Smart Pic Editor	Editor Inteligente de programas para microcontroladores PIC
Instrucción	Constituye un conjunto de campos claramente definidos, opcionales u obligatorios que al ser traducidos a código de máquina tienen una longitud definida en bits. Considera los siguientes campos: [Etiqueta] [Opcode [Operandos]]
Etiqueta	Constituye un conjunto de caracteres letra, números, subraya que cumplan que el primer caracter no sea un número.
Opcode	Corresponde a un conjunto de caracteres propios para cada tipo o familia de microcontroladores (mnemónicos) que indican el tipo de instrucción.
Operando	Conjunto de caracteres que con el opcode definen la operación de la instrucción, cada operando va separado por comas, y para una instrucción es claramente definido el número de operandos que acompañan al opcode así como su posición.
Comentario	Corresponde a un conjunto de caracteres que el programador puede escribir como contenido aclaratorio del programa que se encuentra realizando. Este viene precedido de un indicador o signo que advierte al programador pero sobre todo al ensamblador que lo que está escrito a continuación es un comentario. Para el PIC este signo es el punto y coma ";".
Pseudo-instrucción o directiva del ensamblador	Presenta una estructura similar al de las instrucciones variando el contenido del opcode y operandos a nemónicos propios de la pseudo instrucción.

5. Estructura del Editor

A continuación se presentan los componentes principales que conforman el Smart Pic Editor.

5.1 Pantalla de Presentación

La primera pantalla mostrada por el Smart Pic Editor es la pantalla de presentación, esta pantalla se muestra por unos pocos segundos

FIGURA B.1 Pantalla de presentación del Smart Pic Editor

5.2 Pantalla Principal

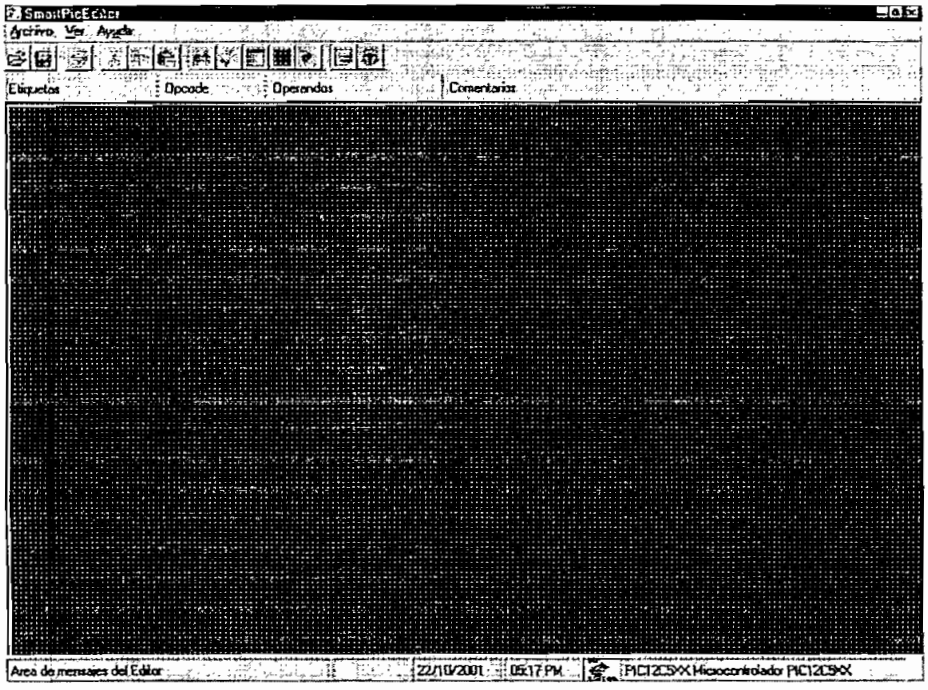


FIGURA B.2 Pantalla principal del Smart Pic Editor

5.3 Pantalla de Edición

La pantalla de edición es la pantalla por medio de la cual el usuario puede acceder a las distintas herramientas de edición y corrección de errores esta compuesta por:

BARRA DE MENU

Comprende una barra de menú típica de Windows con la incorporación de herramientas propias para el Smart Pic Editor. Está compuesta por los siguientes Menús: Archivo, edición, ver, herramientas y Ayuda. Ver figura B.3

FIGURA B.3 Barra de Menú de la pantalla de edición

Menú Archivo.- En este menú se encuentran opciones para trabajar con archivos, tiene los siguientes submenús:

Abrir.- Permite abrir un archivo, verificando que no se encuentre abierto, la primera vez que se abre un archivo en una sesión con el Smart Pic Editor, se solicita al usuario escoger un modelo de microcontrolador a utilizar durante dicha sesión.

Guardar.- El editor guarda los cambios realizados al archivo de programa que se encuentre abierto.

Por omisión el Editor guarda el archivo con extensión "asm".

Guardar como.- Permite guardar el archivo de programa que se encuentra abierto, para ello presenta un cuadro de diálogo en el que se puede seleccionar la ubicación y nombre con que se grabará el archivo.

Cerrar.- Cierra el archivo que se encuentra abierto, en el caso de que el archivo haya sido modificado, se pregunta al usuario si desea guardar los cambios antes de cerrarlo.

Imprimir.- Permite imprimir el archivo abierto, para ello se presenta el cuadro de diálogo para imprimir.

Historial de los últimos archivos abiertos.- Muestra la dirección de los cinco últimos archivos recientemente editados, se puede hacer un clic sobre esta dirección y abrirlos directamente.

Salir.- Permite salir del editor previa confirmación del usuario.

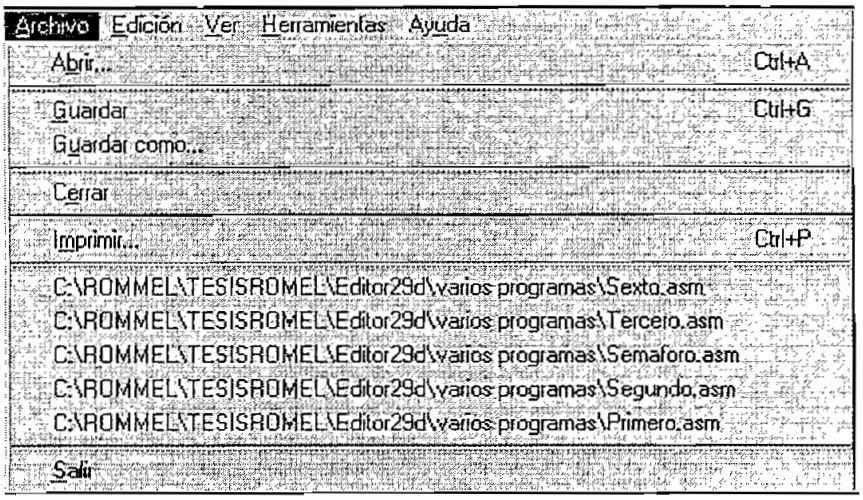


FIGURA B.4 Menú Archivo de la pantalla de edición

Menú Edición.- Agrupa opciones para la manipulación de texto durante la edición de un programa. Contiene los siguientes submenús.

Cortar.- Mueve el texto seleccionado desde el archivo al portapapeles.

Copiar.- Coloca una copia del texto seleccionado en el portapapeles.

Pegar.- Coloca el contenido de texto del portapapeles en la posición actual del cursor o reemplaza un texto que se encuentra seleccionado.

Borrar.- Elimina el texto seleccionado.

Seleccionar todo.- Selecciona todo el texto del programa que se encuentra abierto.

Buscar.- Presenta un formulario en el que el usuario puede ingresar el texto a buscar.

Buscar siguiente.- Busca y resalta la siguiente cadena que coincida con el patrón de búsqueda especificado por el usuario.

Reemplazar.- Presenta el formulario para reemplazar, en este formulario, el usuario puede especificar el patrón de búsqueda, así como la cadena de texto con que se sustituirá en el caso de que se encuentre.

Insertar caracteres.- Presenta un formulario que contiene los caracteres ASCII que el usuario puede insertar en la posición actual del cursor dentro del programa.

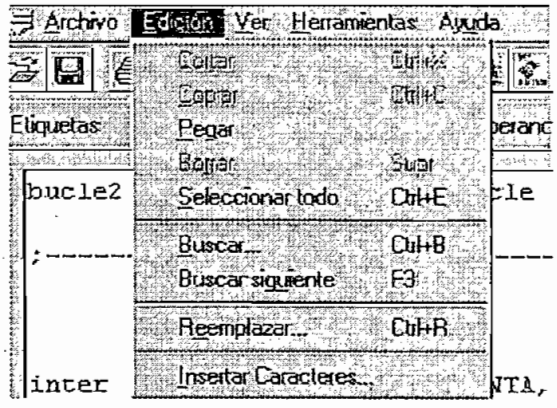


FIGURA B.5 Menú Edición de la pantalla de edición

Menú Ver.- Presenta u oculta las barras: de herramientas, de estado y de campo del editor.

FIGURA B.6 Menú Ver de la pantalla de edición

Menú Herramientas.- Contiene un conjunto de herramientas para la corrección de errores, establece también las opciones del editor y permite además cambiar el modelo de microcontrolador con el que se está trabajando. Contiene los siguientes submenús:

Corregir errores de todo el programa.- Realiza el análisis, coloreado y corrección de las instrucciones del programa, en caso de presentar errores, muestra un formulario que permite al usuario corregir o ignorar dichos errores.

Corregir errores de la línea actual.- Realiza el análisis, coloreado y en caso de contener errores también la corrección de errores de la instrucción donde el cursor se encuentra posicionado.

Información de la Instrucción actual.- Despliega información de la instrucción donde el cursor se encuentra posicionado.

Opciones.- Presenta un formulario que permite establecer las opciones del editor

Escoger Modelo.- Muestra la pantalla para cambiar de modelo de microcontrolador con el que se trabajará en la presente sesión.

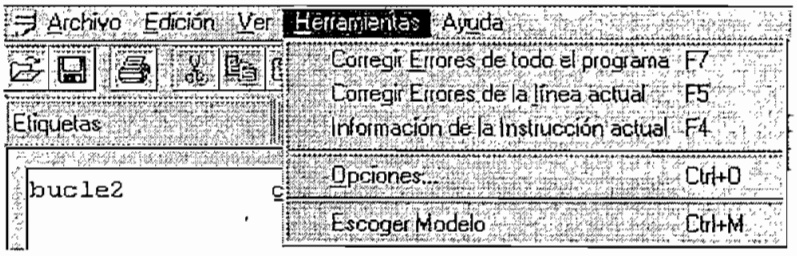


FIGURA B.7 Menú Herramientas de la pantalla de edición

Menú Ayuda.- Muestra el formulario informativo Acerca del Editor.

FIGURA B.8 Menú Ayuda de la pantalla de edición

BARRA DE HERRAMIENTAS

Contiene los botones de acceso directo del editor. En el orden de izquierda a derecha tenemos los botones:

- Abrir.
- Guardar
- Imprimir
- Cortar
- Copiar
- Pegar
- Buscar
- Corregir errores de todo el programa
- Opciones de programación
- Insertar carácter
- Escoger modelo de microcontrolador
- Acerca de...
- Información de la instrucción actual y/o corregir línea.

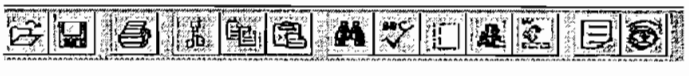


FIGURA B.9 Barra de herramientas de la pantalla de edición

BARRA DE CAMPOS

Es una barra que permite identificar los campos que forman una línea de programa.

FIGURA B.10 Barra de campos de la pantalla de edición

BARRA DE ESTADO

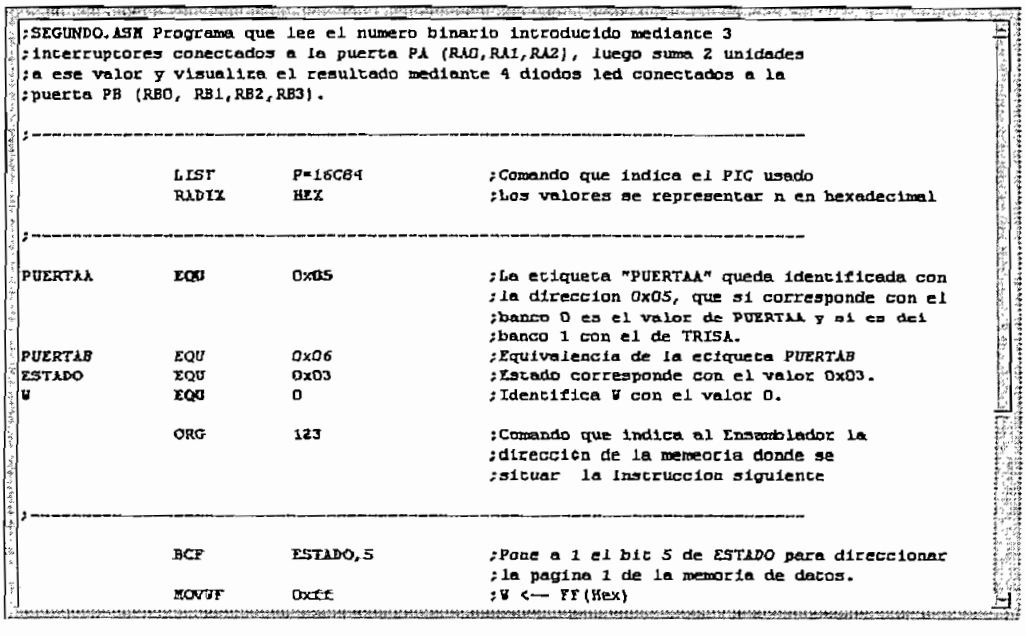
Corresponde una barra con cinco paneles de descripción de eventos durante la edición y corrección de un programa usando el Smart Pic Editor, se compone de los siguientes paneles:

- Panel de mensajes del editor
- Panel de línea actual
- Panel de fecha del PC
- Panel de hora actual
- Panel informativo del modelo de microcontrolador con el que se trabaja en dicha sesión.

FIGURA B.11 Barra de estado de la pantalla de edición

AREA DE EDICION

Comprende el espacio destinado a la edición del programa.



```
;SEGUNDO.ASM Programa que lee el numero binario introducido mediante 3
;interruptores conectados a la puerta PA (RA0,RA1,RA2), luego suma 2 unidades
;a ese valor y visualiza el resultado mediante 4 diodos led conectados a la
;puerta PB (RB0, RB1,RB2,RB3).

;-------------------------------------------------------------------------

                LIST        P=16C84             ;Comando que indica el PIC usado
                RADIX       HEX                 ;Los valores se representar n en hexadecimal

;-------------------------------------------------------------------------

PUERTAA         EQU         0x05                ;La etiqueta "PUERTAA" queda identificada con
                                                ;la direccion 0x05, que si corresponde con el
                                                ;banco 0 es el valor de PUERTAA y si es del
                                                ;banco 1 con el de TRISA.
PUERTAB         EQU         0x06                ;Equivalencia de la eciqueta PUERTAB
ESTADO          EQU         0x03                ;Estado corresponde con el valor 0x03.
W               EQU         0                   ;Identifica W con el valor 0.

                ORG         123                 ;Comando que indica al Ensamblador la
                                                ;direccion de la memeoria donde se
                                                ;situar  la instruccion siguiente

;-------------------------------------------------------------------------

                BCF         ESTADO,5            ;Pone a 1 el bit 5 de ESTADO para direccionar
                                                ;la pagina 1 de la memoria de datos.
                MOVWF       0xff                ;W <-- FF(Hex)
```

FIGURA B.12 Area de edición de la pantalla de edición

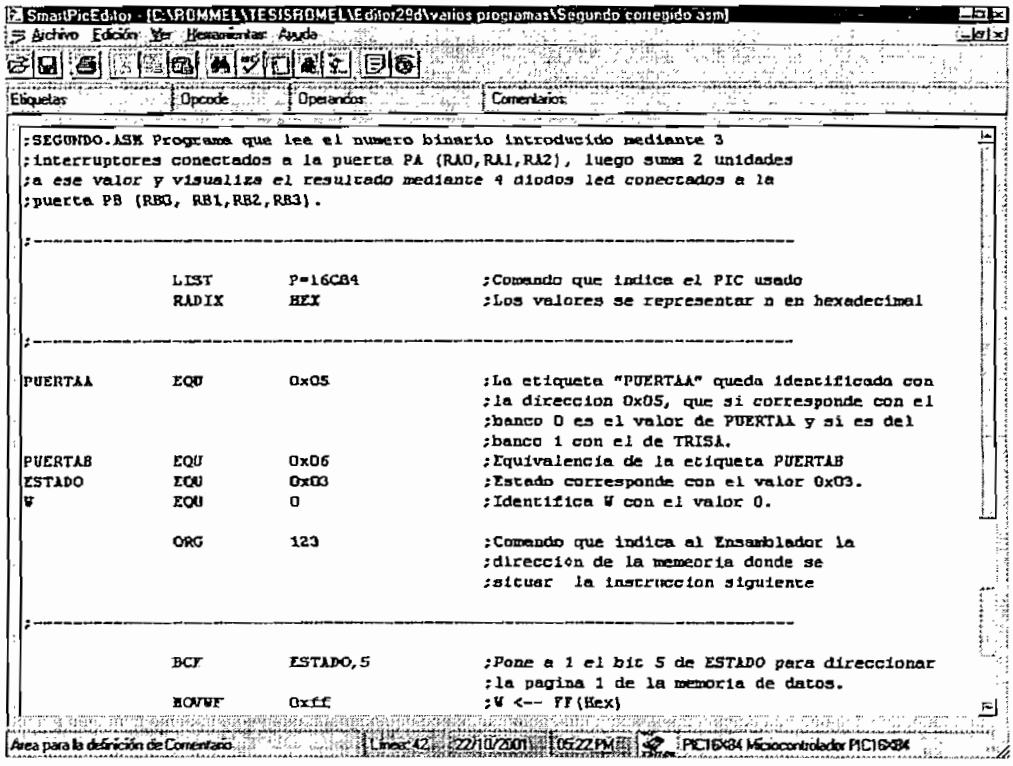


FIGURA B.13 Pantalla de Edición del Smart Pic Editor con todos sus componentes

6. Opciones de Configuración

El editor dispone de las siguientes opciones de configuración

6.1 Pantalla de Selección de Modelos

Por medio de esta pantalla el usuario puede escoger un modelo de microcontrolador con el que trabajará en la presente sesión, sin embargo podrá acceder a esta pantalla mientras el usuario se encuentre editando un programa mediante el menú "Herramientas" dentro del que se encuentra un submenú denominado "Escoger modelo" o también con el botón de acceso directo "Escoger Modelo". Ver la figura B.14.

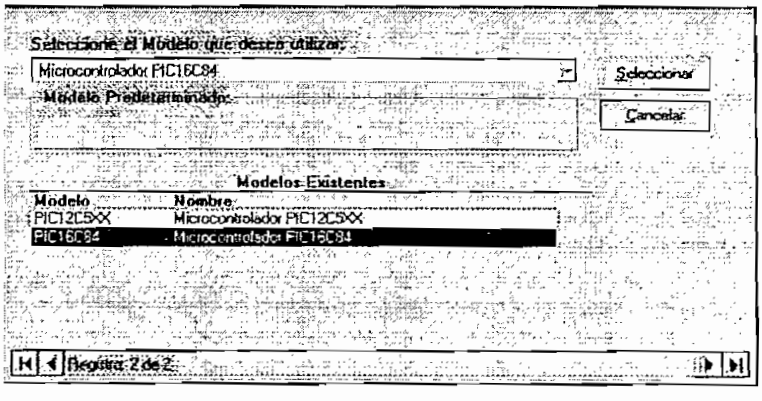


FIGURA B.14 Pantalla de selección del modelo de microcontrolador

6.2 Pantalla de Opciones del Editor

El Smart Pic Editor cuenta con una pantalla para la configuración tanto de la ayuda que se va ha presentar en pantalla durante la edición, como de la activación de las pantallas para la corrección de errores. Ver figura B.15.

Si se selecciona la opción "mostrar los nemónicos durante la edición", se activan las opciones de uso de teclado y ratón para la selección del nemónico y el usuario podrá escoger alguna de las dos opciones o ambas formas de selección, ver figura B.16; si no se escoge ninguna opción, únicamente con la tecla enter se aceptará el elemento escogido de la lista de ayuda que se presenta.

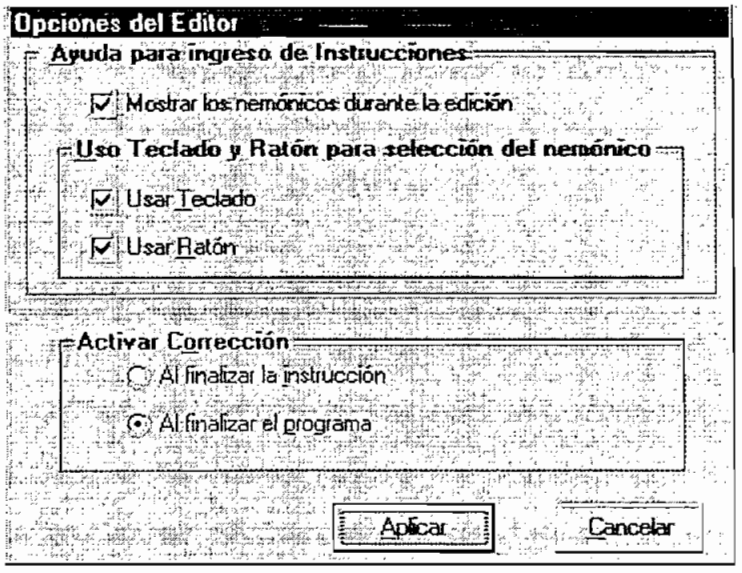


FIGURA B.15 Pantalla de configuración de las Opciones de Ayuda y Corrección del Editor - Configuración por omisión.

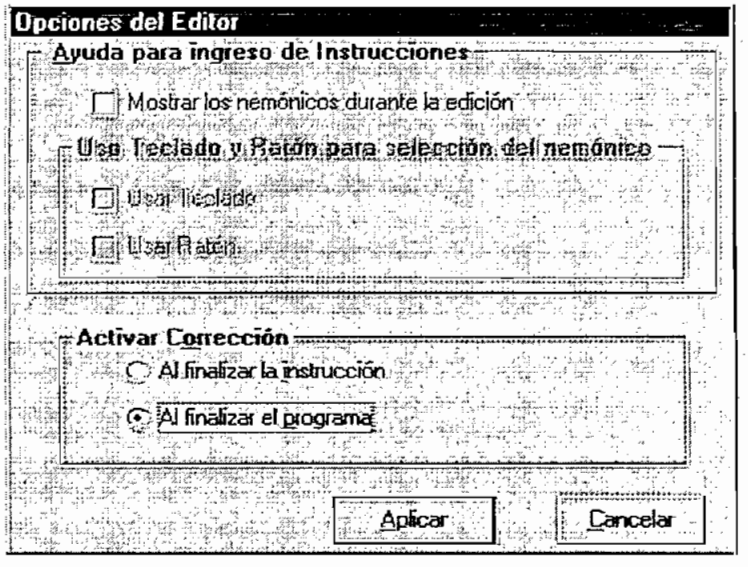


FIGURA B.16 Configuración para no mostrar ayuda de listas de nemónicos durante la edición

7. Edición de Programas

A continuación se explica en detalle el proceso de edición de un programa.

Para la edición de un programa se debe hacer clic sobre el icono Smart Pic Editor presente en el Menú Programas del escritorio de Windows. Ver Figura B.17.

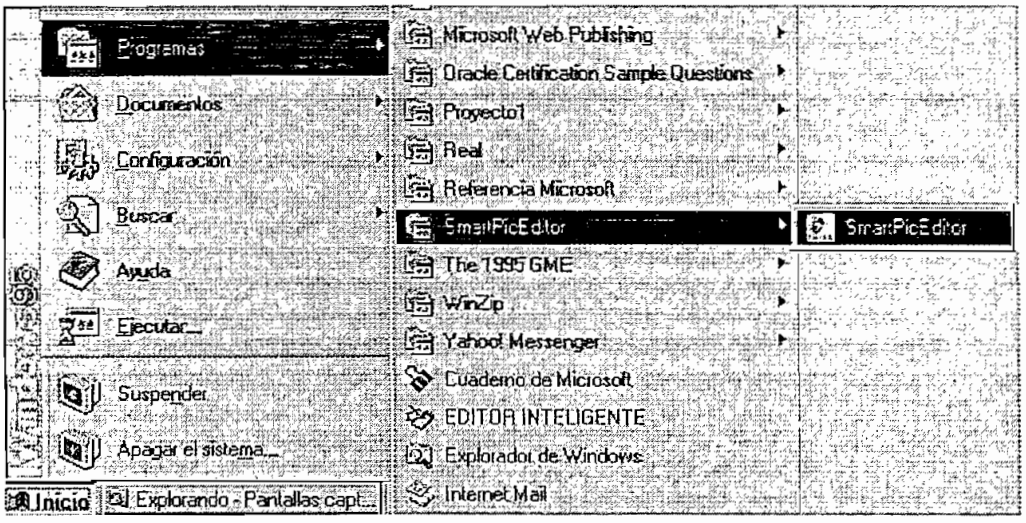


FIGURA B.17 Pantalla de presentación del Smart Pic Editor

A continuación se muestra por unos segundos la pantalla de presentación del Smart Pic Editor que se puede apreciar en la Figura B.1.

El editor pide al usuario seleccionar un modelo de microcontrolador con el que se trabajará en la presente sesión para ello se muestra la pantalla de selección del modelo de microcontrolador mostrada en la figura B.14.; a continuación el Smart Pic Editor muestra la pantalla de Edición con un programa vacío denominado "Programa1" para que el usuario inicie con la escritura del nuevo programa; la configuración por omisión de la ayuda que se va ha presentar en pantalla durante la edición, como de la activación de las pantallas para la corrección de errores es la mostrada en la figura B.15, sin embargo el usuario podrá modificarlas de acuerdo a lo indicado en el numeral 6.2 del presente manual.

En este punto el usuario podrá hacer uso de las siguientes ayudas en la Edición:

Listas de opcodes y directivas del modelo escogido.

Listas de operandos y pseudo-operandos.

Información de la instrucción o pseudo-instrucción.

7.1 Ayuda para la selección de un opcode o de una directiva del ensamblador

Como se puede apreciar en la figura B.18, si el cursor se encuentra en los campos de opcode y si se presiona la letra inicial del opcode el editor muestra una lista de todos los posibles opcodes y directivas del ensamblador (pseudo-opcodes) posibles.

Para la selección del opcode correspondiente se puede utilizar el teclado o el ratón (mouse); si se usa el ratón, al realizar un clic sobre una opción de la lista desplegada se escoge dicha opción, la misma que se resaltará, si se desea aceptar se debe hacer un doble click sobre esta opción. De la misma manera, si se utiliza el teclado, una vez presente la lista con la tecla de flecha hacia abajo o la tecla de flecha hacia la derecha se puede escoger la primera opción luego con las teclas de flechas: hacia arriba, abajo y derecha se puede navegar entre las opciones; para aceptar una opción escogida se debe presionar la flecha hacia la derecha o presionar la tecla enter.

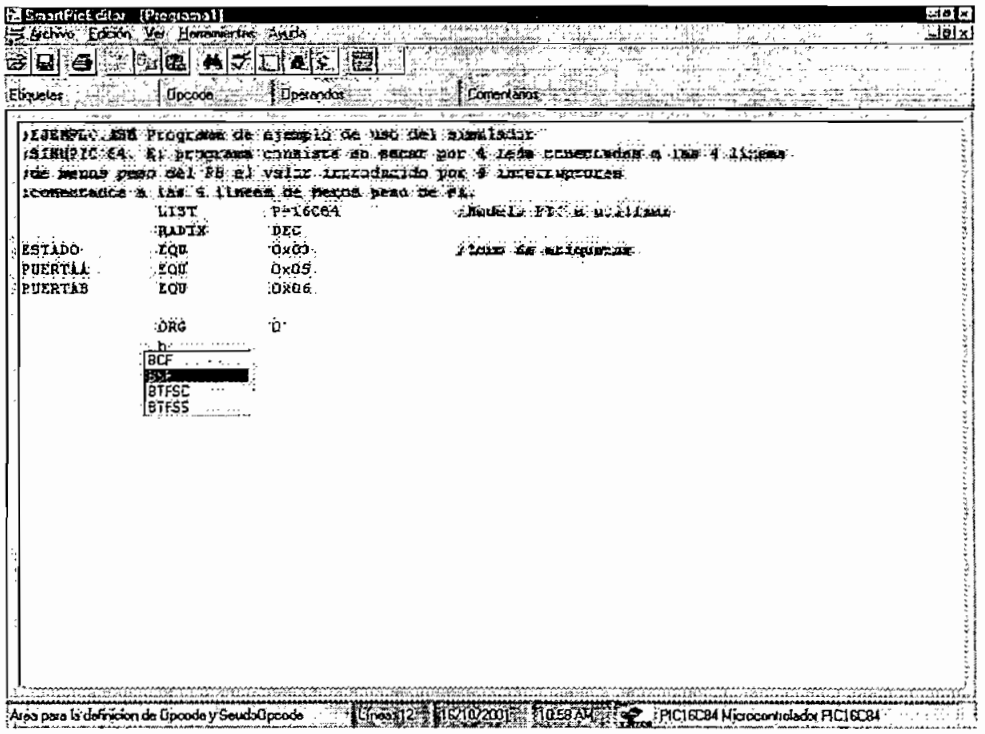


Fig. B.18 Ingreso del opcode BSF

7.2 Ayudas para el ingreso de los operandos

Una vez aceptado el opcode, se despliega una lista de selección o ingreso del primer operando; si se selecciona o escribe el operando y se presiona la tecla enter y la instrucción requiere más operandos se presenta otra lista donde se puede ingresar o seleccionar el segundo operando y así hasta terminar la escritura de la instrucción. Ver las figuras B.19 y B.20.

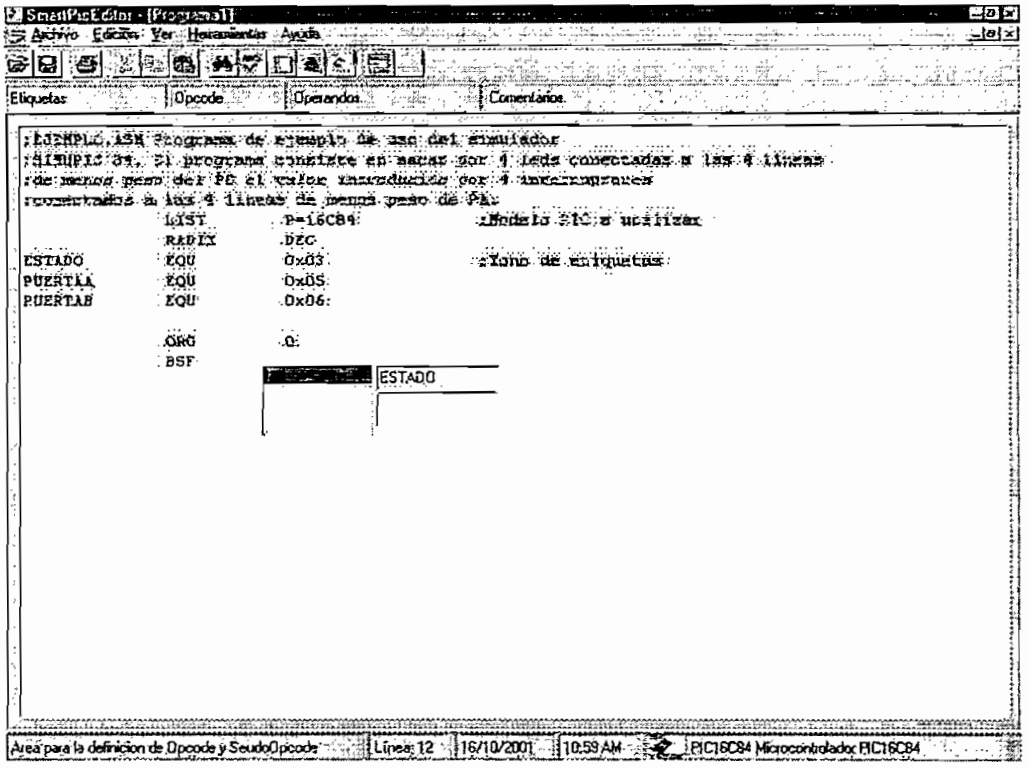


Fig. B.19 Ingreso del primer operando de BSF

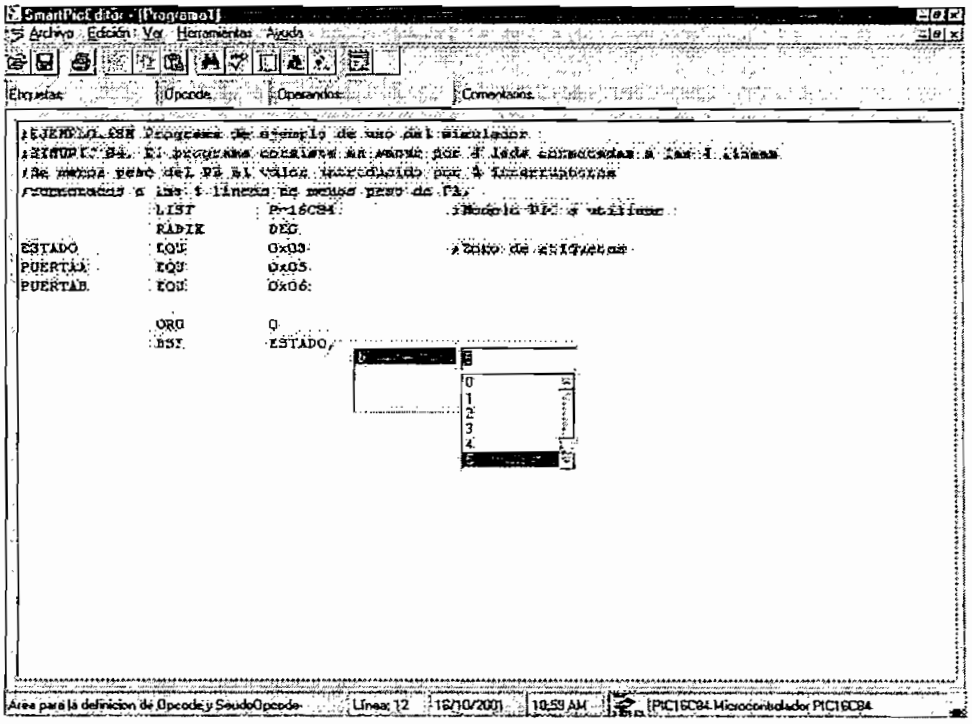


Fig. B.20 Ingreso del segundo operando de BSF

7.3 Ingreso del comentario

Cuando no se requieren más operados el editor automáticamente inserta un tabulador y se ubica en el campo adecuado para escribir un comentario, se deberá escribir el identificador de comentario (;) y el comentario respectivo.

En el caso de que el usuario no requiera escribir el comentario, y se presione enter, el editor borra los tabuladores precedentes y cambia de línea.

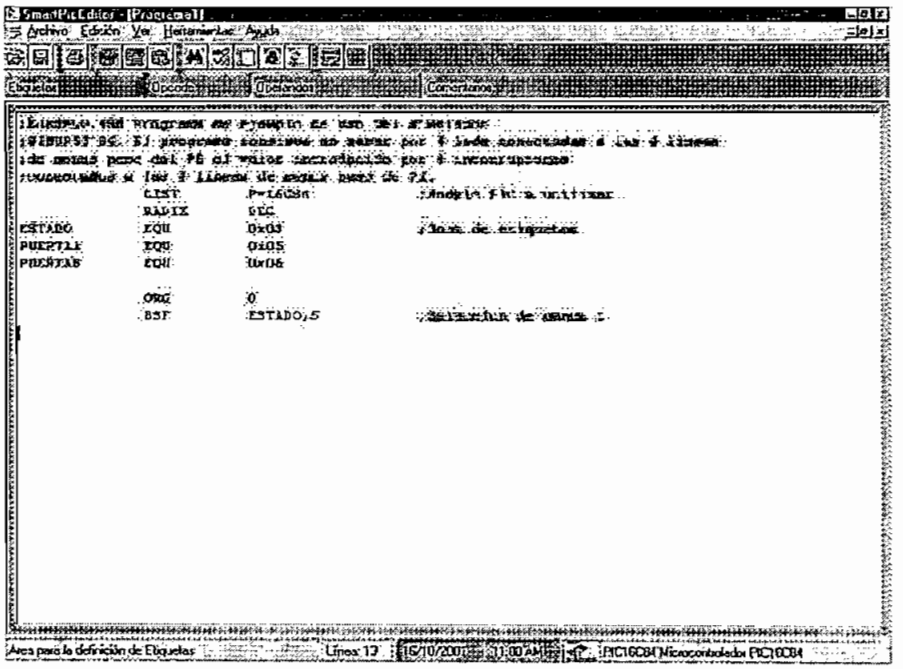


Fig. B.21 Instrucción BSF ingresada.

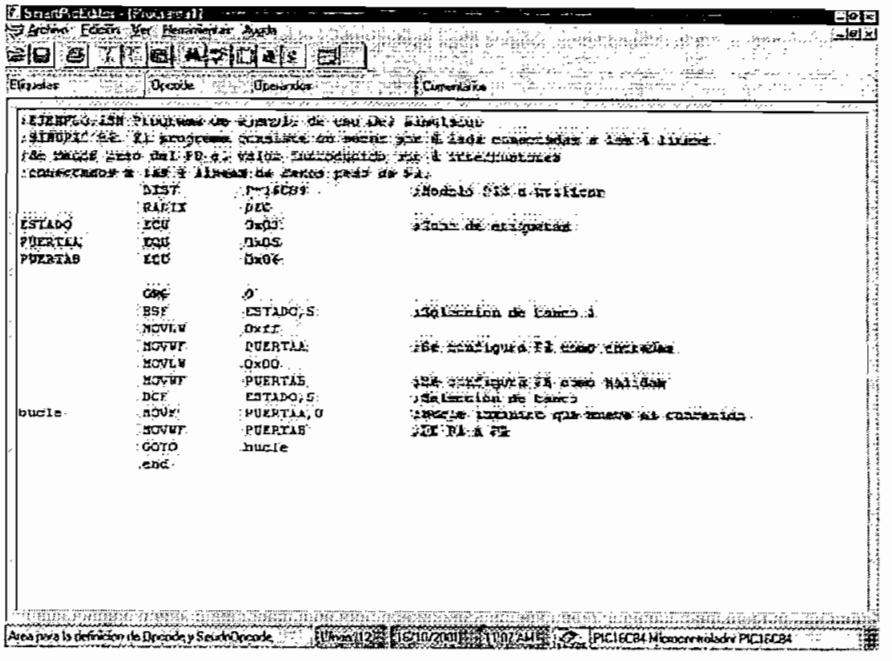


Fig. 4.22 Programa completo

7.4 Ayuda con el botón derecho del mouse

Una vez escrita una o más instrucciones, el usuario puede consultar información acerca de una instrucción, posicionando el cursor en la línea correspondiente a la instrucción y luego presionando el botón derecho del mouse, como se puede observar en el ejemplo de la figura B.23.

Si existe una instrucción errónea por medio del botón derecho se puede acceder a la opción de corrección de dicha instrucción.

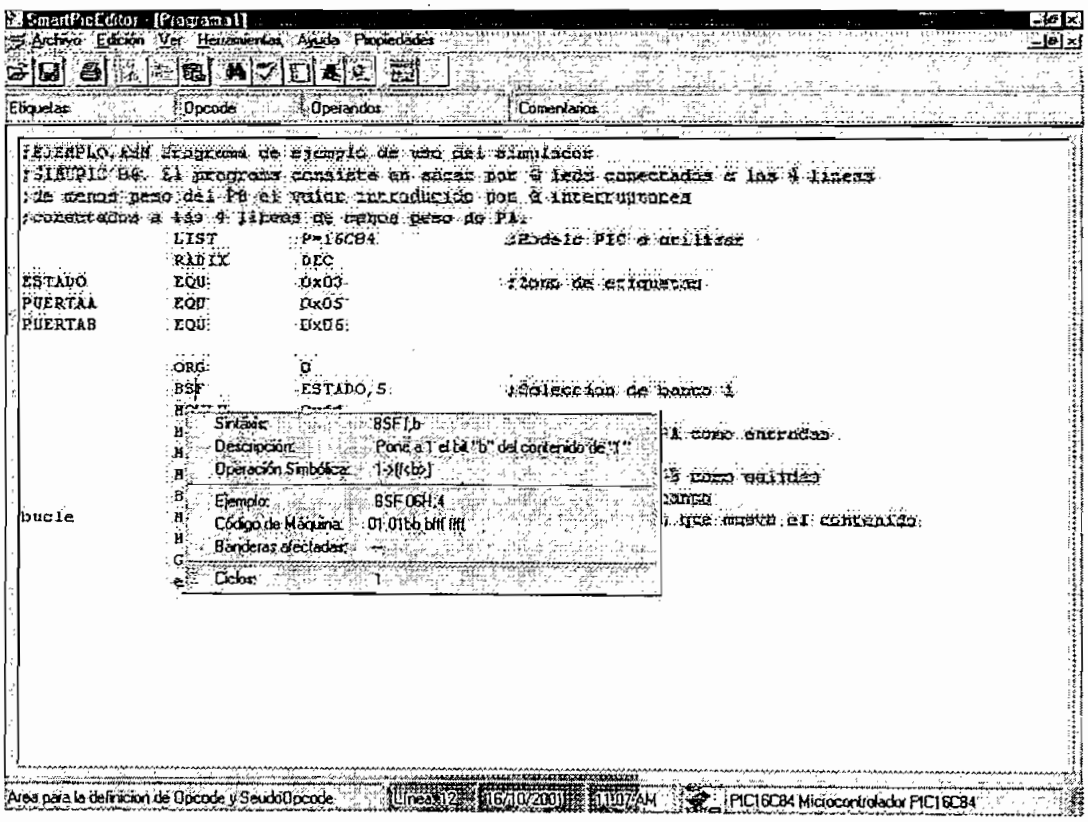


Fig. B.23 Ayuda para la instrucción BSF f,b

Finalmente se procede a grabar el nuevo programa editado, el Editor por omisión lo guarda con extensión asm. Ver las figuras B.24 B.25 y B.26.

Fig. B.24 Guardar el programa paso 1

Fig. B.25 Guardar el programa paso 2

Una vez guardado el programa en la pantalla en la parte superior se puede apreciar el path donde se encuentra almacenado dicho programa, ver figura *B.26.*

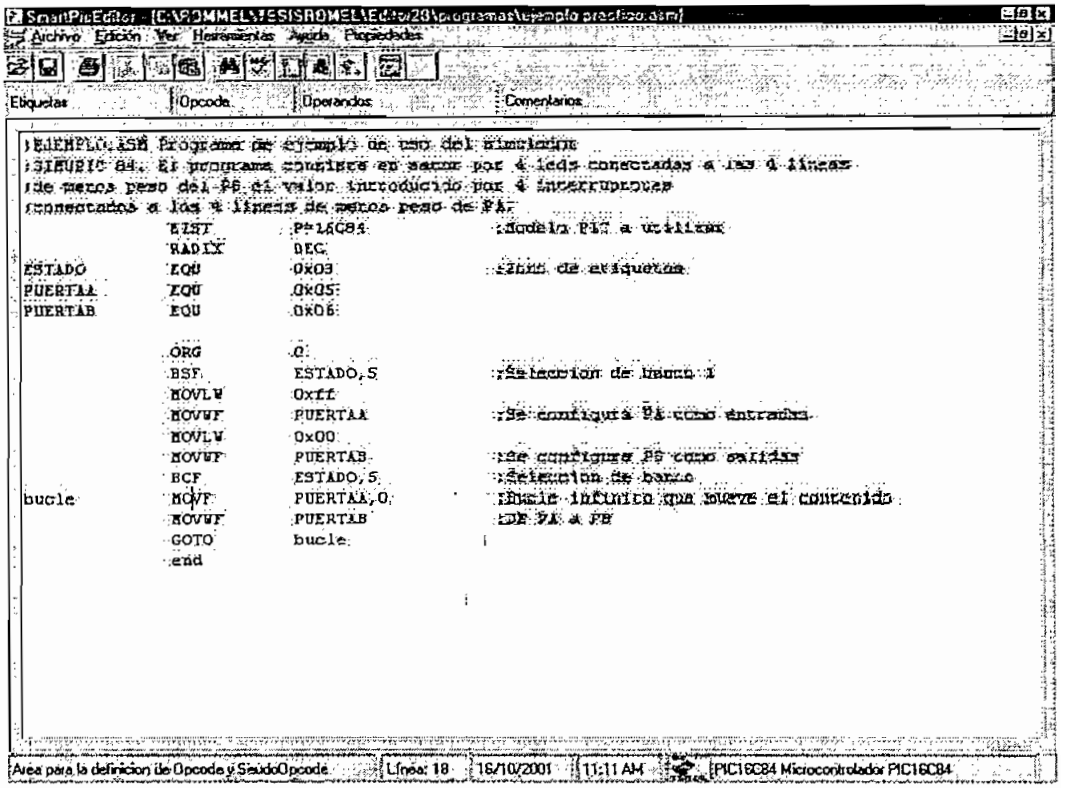


Fig. B.26 Programa almacenado

7.5 Otras ayudas del editor

Se ha implementado adicionalmente una pantalla para la inserción de caracteres ASCII, ver la figura B.27.

El editor cuenta también con ayudas típicas de un editor como son las de buscar y reemplazar, ver la figura B.28.

Fig. B.27 Pantalla para inserción de caracteres ASCII

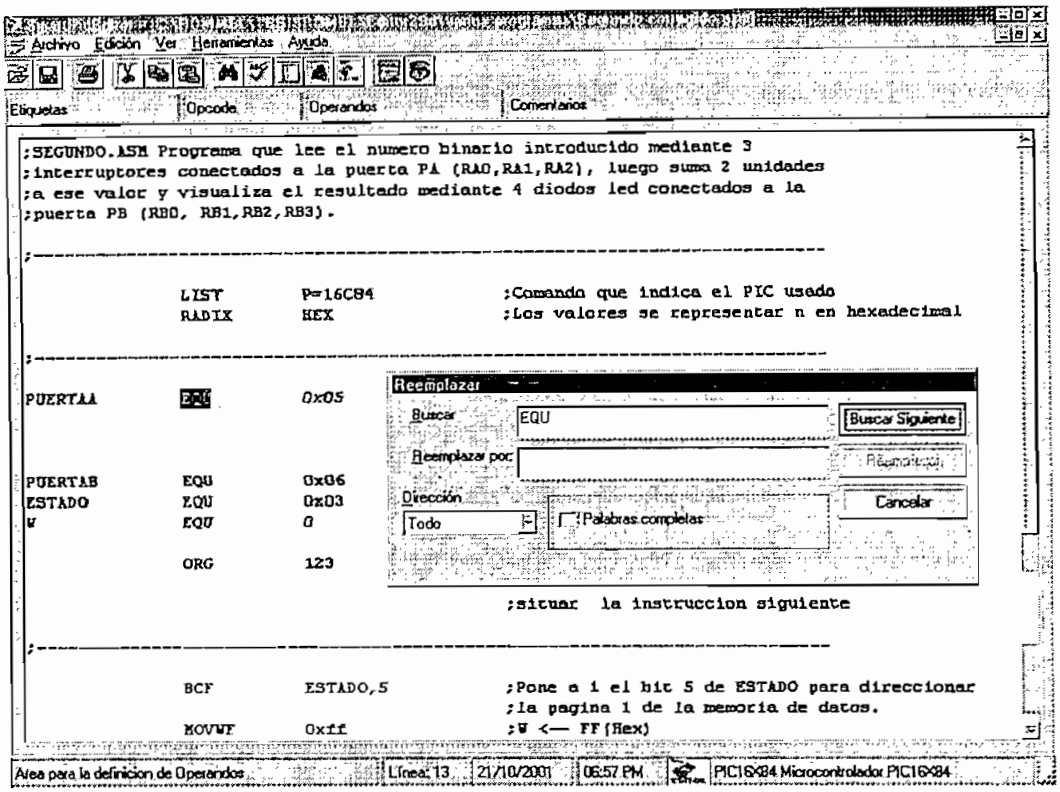


Fig. B.28 Pantalla para reemplazar una cadena de texto buscada

8. Corrección de Errores

La corrección de errores con el uso del Smart Pic Editor puede realizarse de dos maneras:

Corrección de errores al editar una línea de programa.

Corrección de errores de todas las líneas que forman un programa.

8.1 Formas de acceder a la corrección de errores

Para acceder a las pantallas de corrección de errores, durante la edición de un programa se debe configurar las opciones del editor para activar la corrección de errores en la presente sesión, esto se hace en la pantalla "Opciones del Editor" que se encuentra en el menú "Herramientas" submenú "Opciones...", esto se puede observar en la Figura B.15.

Existen dos formas de configuración del editor:

Activar la corrección al finalizar la instrucción.- Si se requiere que las pantallas para la corrección aparezcan al finalizar una instrucción es decir cuando el programador pulse la tecla de cambio de línea (enter).

Activar la corrección al finalizar el programa.- En este caso no aparecen las pantallas de corrección al finalizar una instrucción, sin embargo, si la instrucción es

errónea, la línea se pinta de rojo; esta opción es la que se encuentra configurada como predeterminada al abrir una sesión de trabajo con el Smart Pic Editor.

Para cualquiera de estas dos formas de configuración de la activación de pantallas de corrección, existen otras alternativas para el acceso a la corrección de errores así.

Corrección de la línea donde se encuentra posicionado el cursor.- Se puede corregir la línea errónea donde se encuentra el cursor, para ello se pulsa el botón derecho del mouse, y se selecciona la opción corregir la sintaxis de la línea, esta misma opción está presente en el menú "herramientas", submenú "Corregir errores de la línea actual" o presionando la tecla de funciones F5.

Corrección todo el programa.- Otra opción es corregir todas las líneas erróneas presentes en el programa, para ello, mediante el menú "herramientas", submenú "Corregir errores de todo el programa" también presionando la tecla de funciones F7 o mediante el botón de acceso directo "Corregir programa".

En las diferentes figuras que se muestran se realiza la corrección de errores de un programa, para ilustrar los diferentes tipos de errores de sintaxis que se puede corregir mediante el Smart Pic Editor.

En las figuras B.29 y B.30 se muestra el programa previamente editado y una vez abierto mediante el Smart Pic Editor.

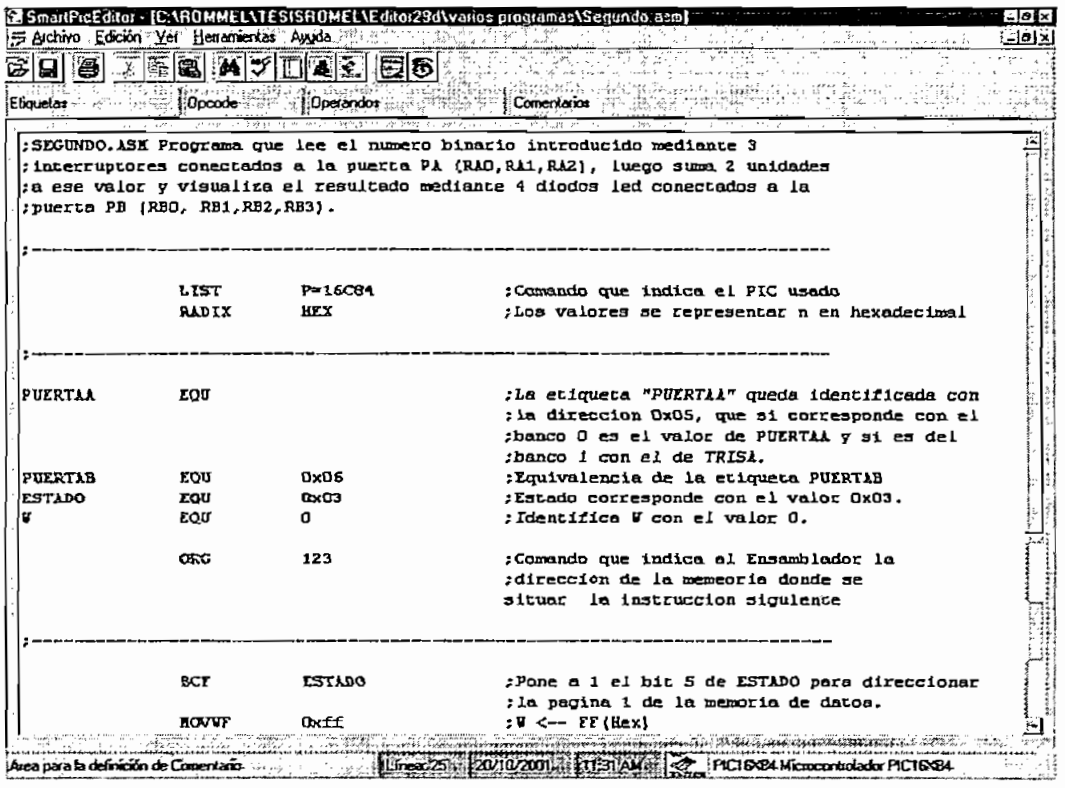


Fig. B.29 Pantalla donde se muestra la parte inicial del programa ha ser corregido

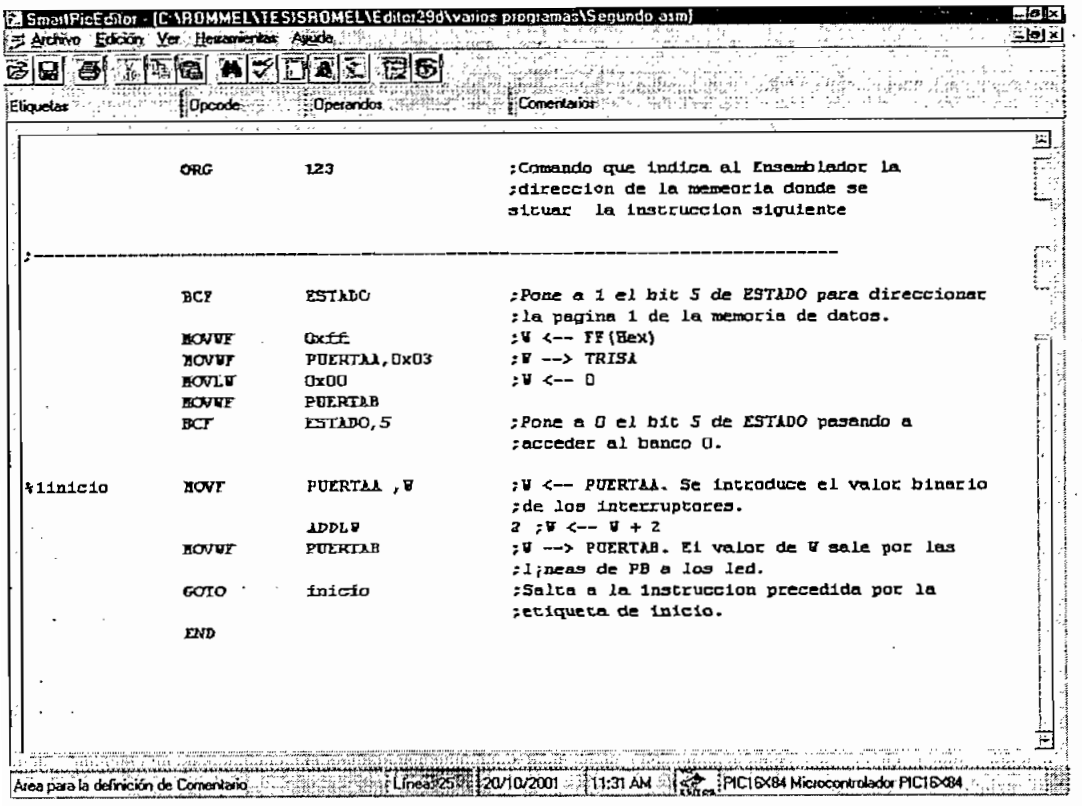


Fig. B.30 Pantalla donde se muestra la parte final del programa ha ser corregido

8.2 Error de etiqueta

Una etiqueta acepta valores alfanuméricos A...Z, a,...,z numéricos 0,1,...9 y el signo subraya _ , además una etiqueta no debe comenzar con un número.

Una etiqueta no puede ser un opcode del set de instrucciones, si una etiqueta no cumple con estos requisitos el editor lo considera como un error.

En la figura B.37 se ilustra la pantalla usada por el editor para la corrección de una etiqueta errónea.

8.3 Error de opcode y pseudo-opcode

Si estamos en el campo de opcode o pseudo-opcode únicamente se validan valores previamente definidos en la base de datos que corresponden al opcode del set de instrucciones o directivas del ensamblador, cualquier otra palabra el Editor lo considera como un error.

En las figuras B.41, B.42, B.43 y B.44 se observan la secuencia de la corrección de un opcode erróneo.

8.4 Error de operandos.

Al analizar una instrucción, y una vez validado el opcode o pseudo-opcode se procede al análisis de los operandos.

Se analiza cada uno de los operandos para diferenciarlos, el separador de operandos es la coma (,), el indicador que ha finalizado el área de operandos es el tabulador el punto y coma o el enter.

Se distinguen tres tipos de operandos:

Constante.- Acepta un conjunto de caracteres único, si toma otro valor constituye un error.

Variable.- puede aceptar cualquier conjunto de caracteres.

Lista.- al igual que el tipo variable puede aceptar cualquier conjunto de caracteres, pero en la presentación de ayuda en la pantalla muestra una lista de opciones donde el usuario puede escoger.

Una vez validados los operandos, se analiza si el número corresponde al número de operandos que requiere la instrucción, caso contrario el editor considera un error sea este por falta o por exceso de operandos.

Si existen instrucciones que tienen el mismo opcode y diferente número de operandos, se valida el primer operando, de esas instrucciones si ninguno cumple el editor considera un error, de existir algunas o una instrucción cuyo primer operando es válido continua con el análisis del segundo operando y así hasta que por lo menos una instrucción sea validada, en este caso se presenta como válida la instrucción, de no existir el editor considera a la instrucción como errónea.

En las figuras B.32 y B.33 se muestran las pantallas que usa el Smart Pic Editor, para corregir errores por falta de operandos en una instrucción.

Así mismo, en las figuras B.34 y B.35 se indican las pantallas que usa el Smart Pic Editor para corregir errores por exceso de operandos en una instrucción.

8.5 Error por exceso de tabuladores.

En caso que el editor encuentre opcodes o pseudo-opcodes en los campos de operandos o comentarios el editor permite la eliminación de tabuladores o espacios anteriores. En la figura B.38 se muestra la pantalla usada por el editor para este fin.

8.6 Error en la zona de comentarios.

Si el Editor encuentra palabras en el campo de comentarios y estas no corresponden a opcodes o pseudo-opcodes, el editor permite insertar el punto y coma (;) faltante para definir un comentario.

En la figura B.36 se indica la pantalla que el Smart Pic Editor usa para corregir un error de falta de signo que indica un comentario.

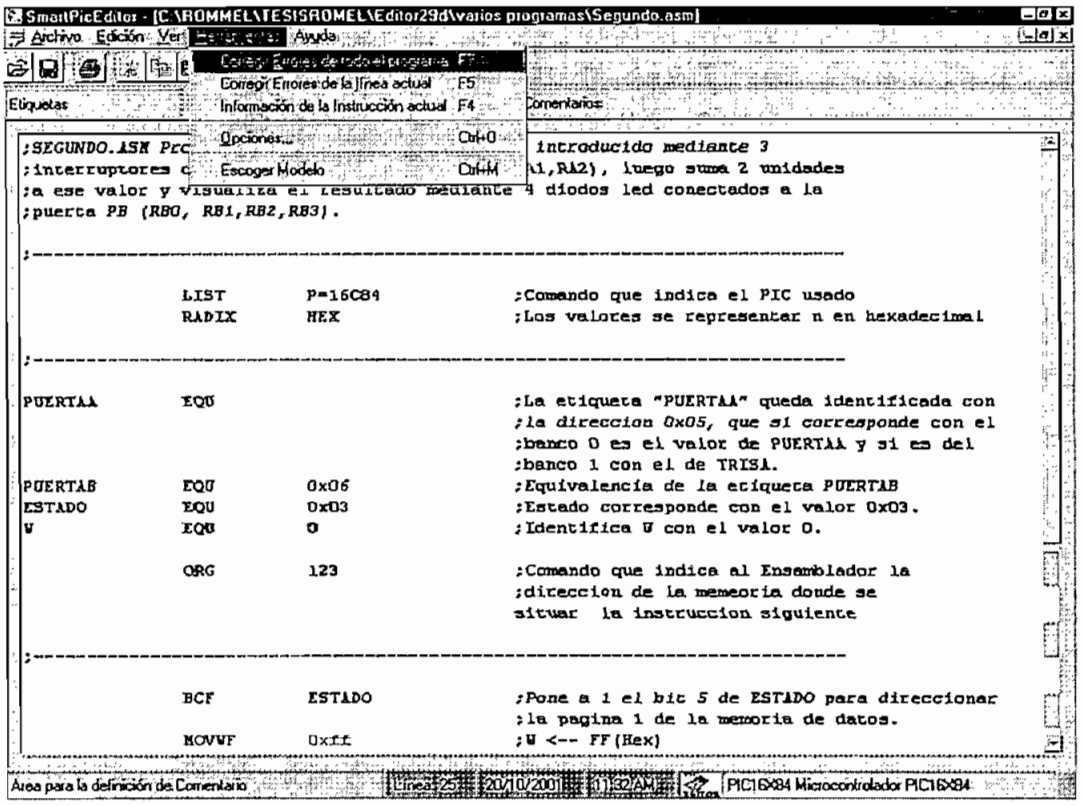


Fig. B.31 Se escoge la opción: "Corregir errores de todo el programa"

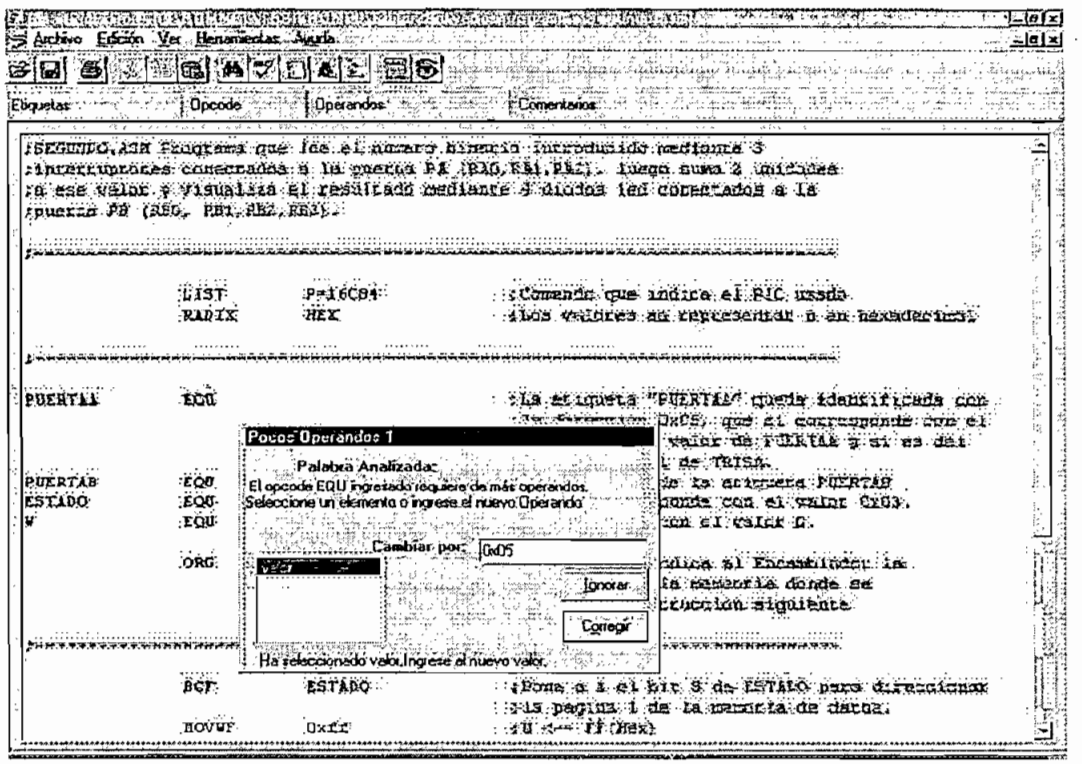


Fig. B.32 Corrección de error por falta del primer y único operando

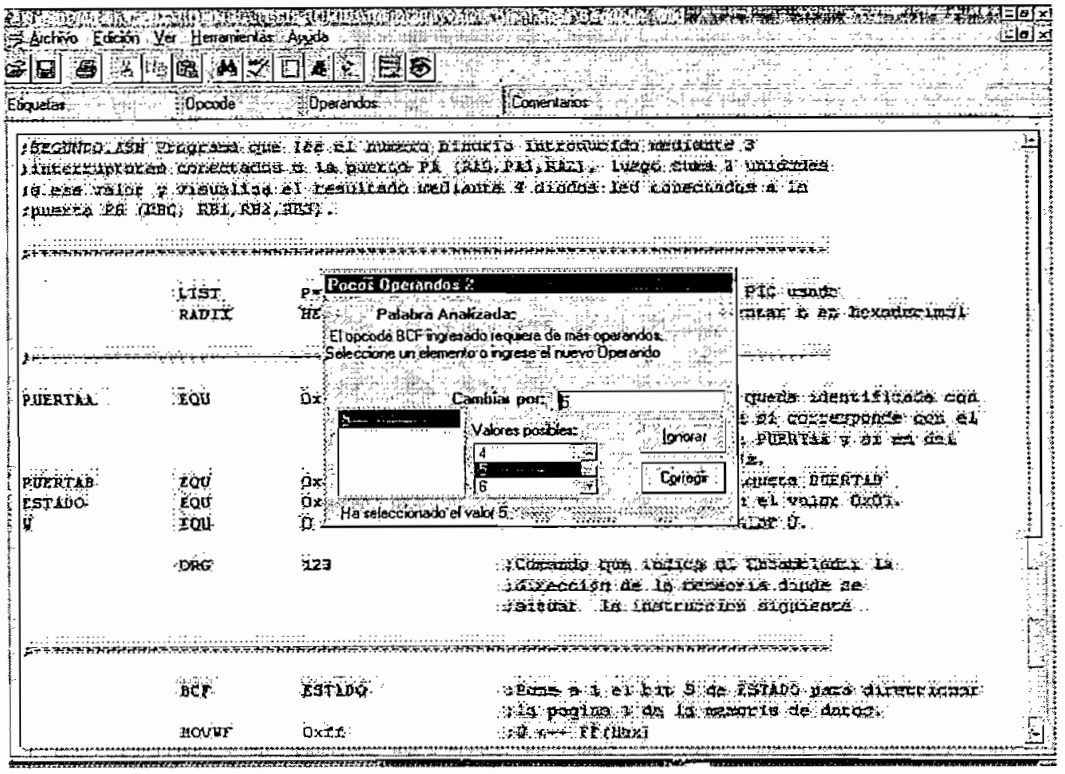


Fig. B.33 Corrección de error por falta del segundo operando

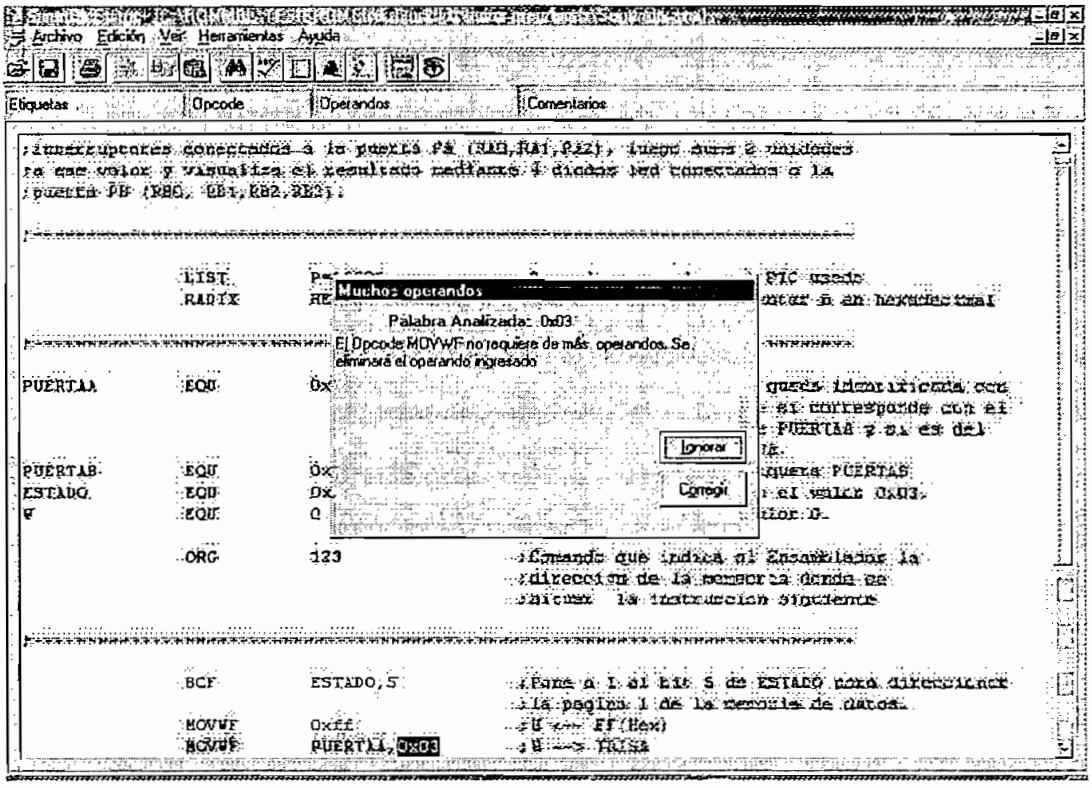


Fig. B.34 Corrección de error por exceso de operandos

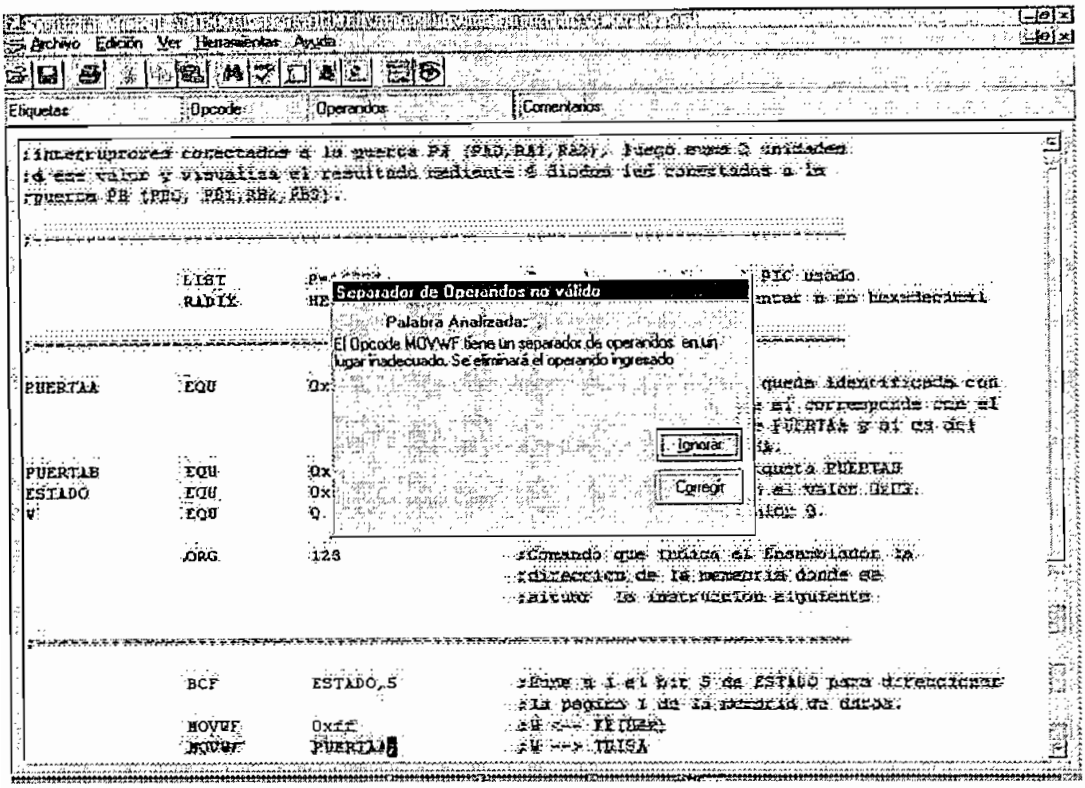


Fig. B.35 Eliminación de una coma en lugar inadecuado.

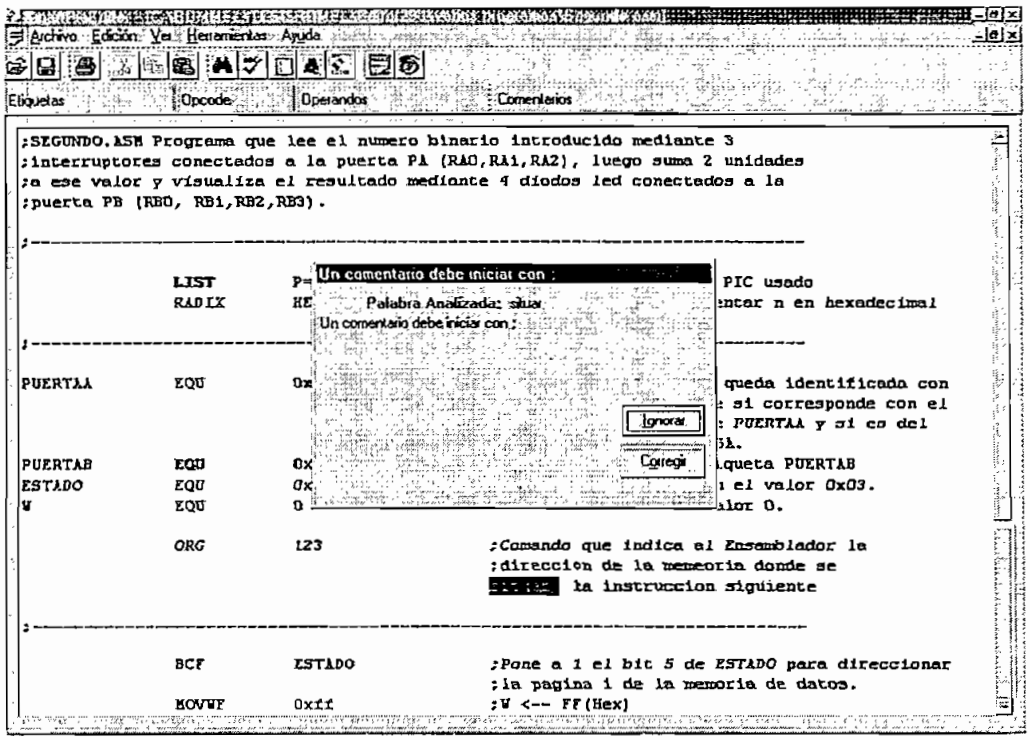


Fig. B.36 Corrección de error por falta de signo de comentario

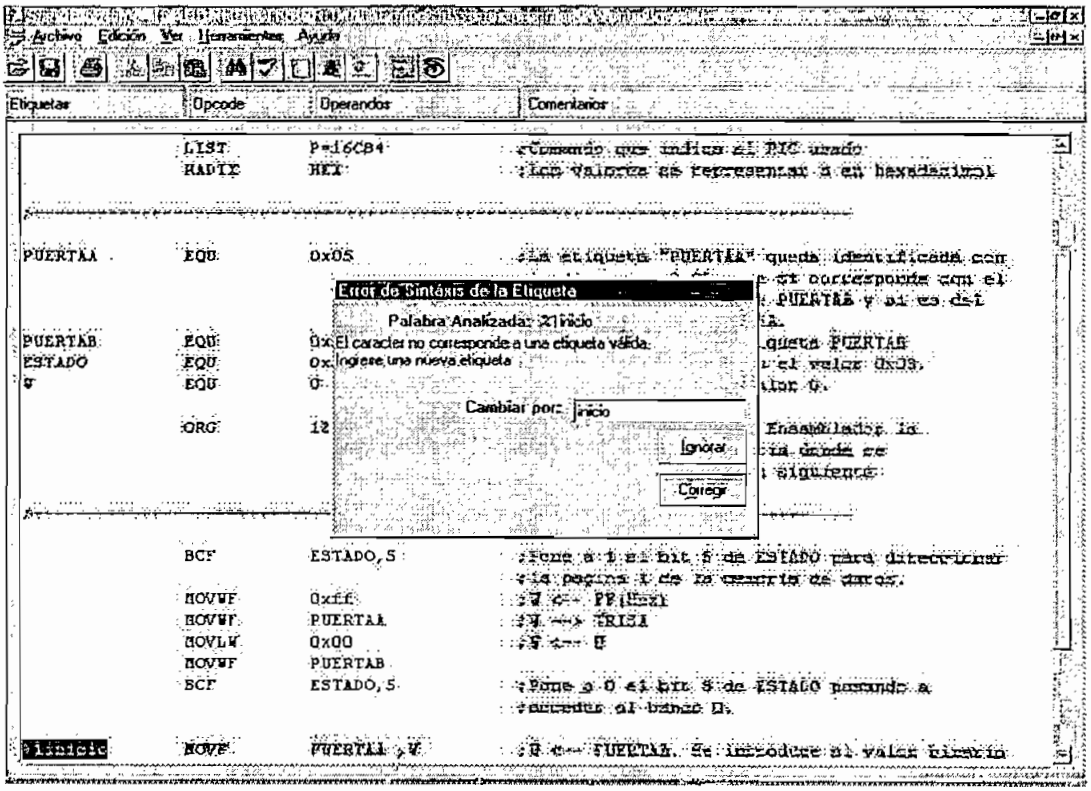


Fig. B.37 Corrección de error de etiqueta no válida

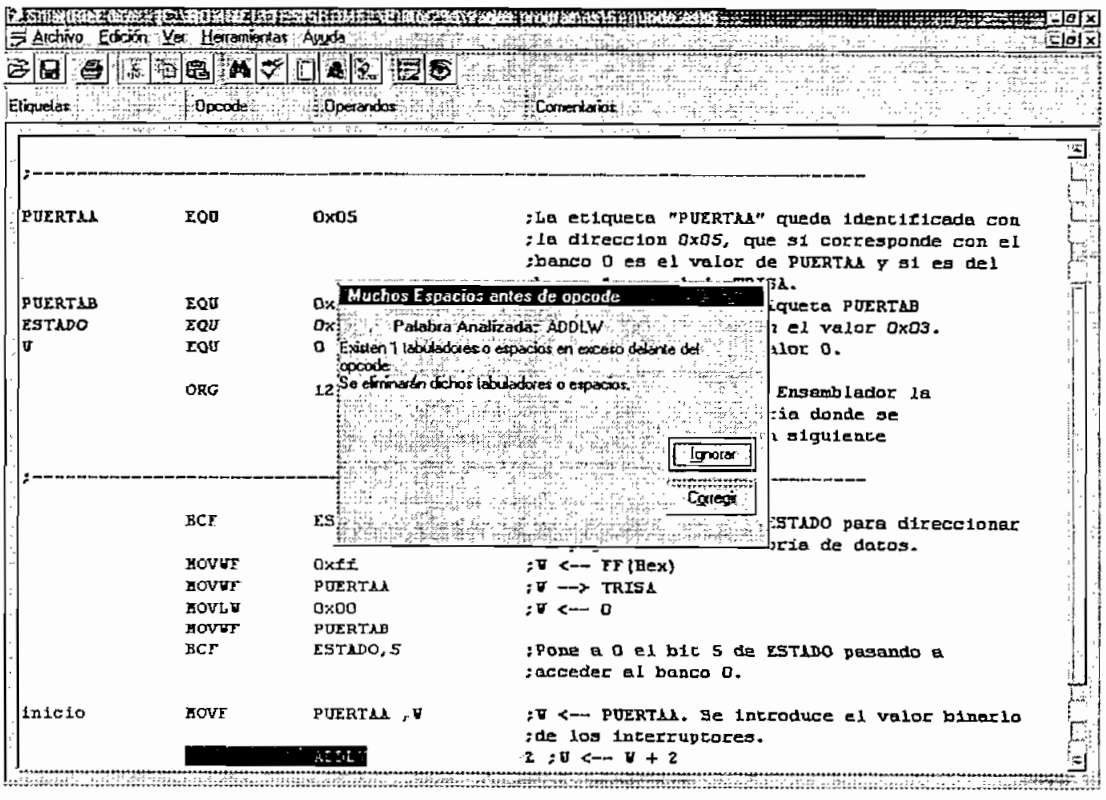


Fig. B.38 Eliminación de exceso de tabulador

Una vez finalizada la corrección de errores, el programa se muestra correctamente coloreado en azul, negro y verde; este hecho para el caso del programa tomado como ejemplo se puede apreciar en las figuras B.39 y B.40.

Si el usuario, hace correcciones sobre este programa y comete errores de sintaxis, dicha línea se pondrá en color rojo, y si únicamente se desea corregir dicha línea se procederá de acuerdo a lo indicado en el numeral 4.4.1 "Formas de acceder a la corrección de errores".

En las figuras B.41, B.42, B.43 y B.44 se indica la secuencia de pantallas para la corrección de una sola línea de la instrucción BCF.

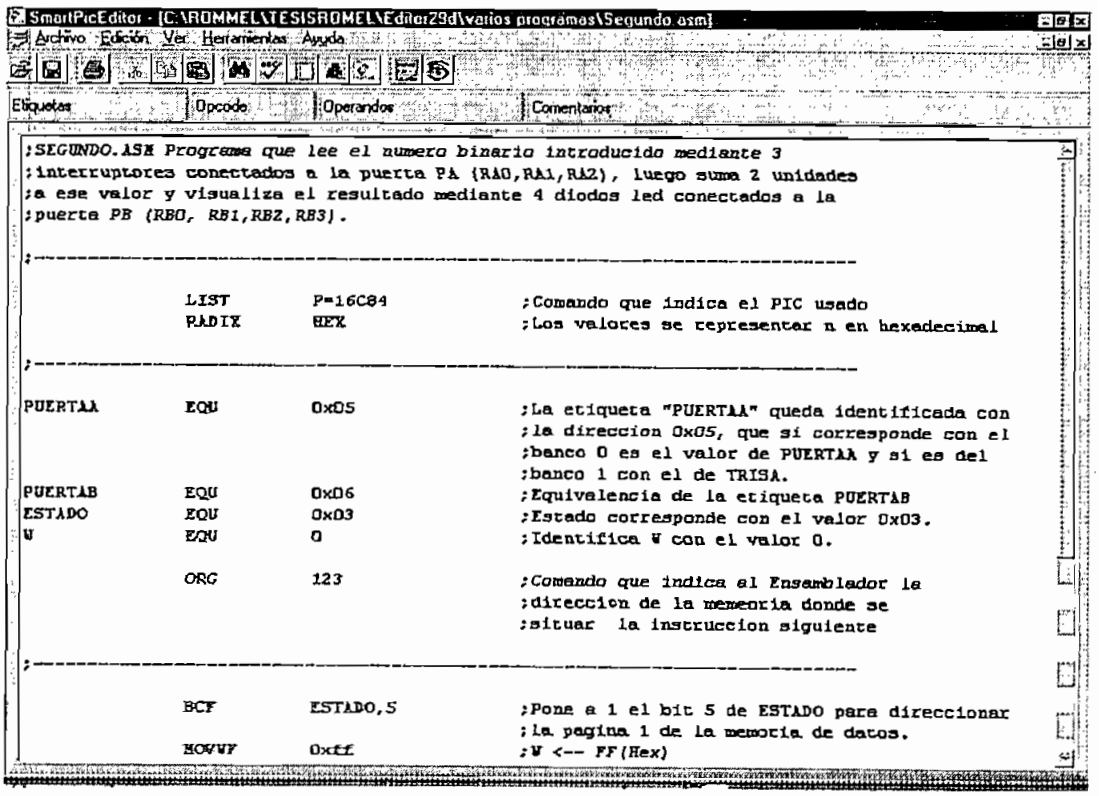


Fig. B.39 Fin de la corrección de errores del programa parte inicial

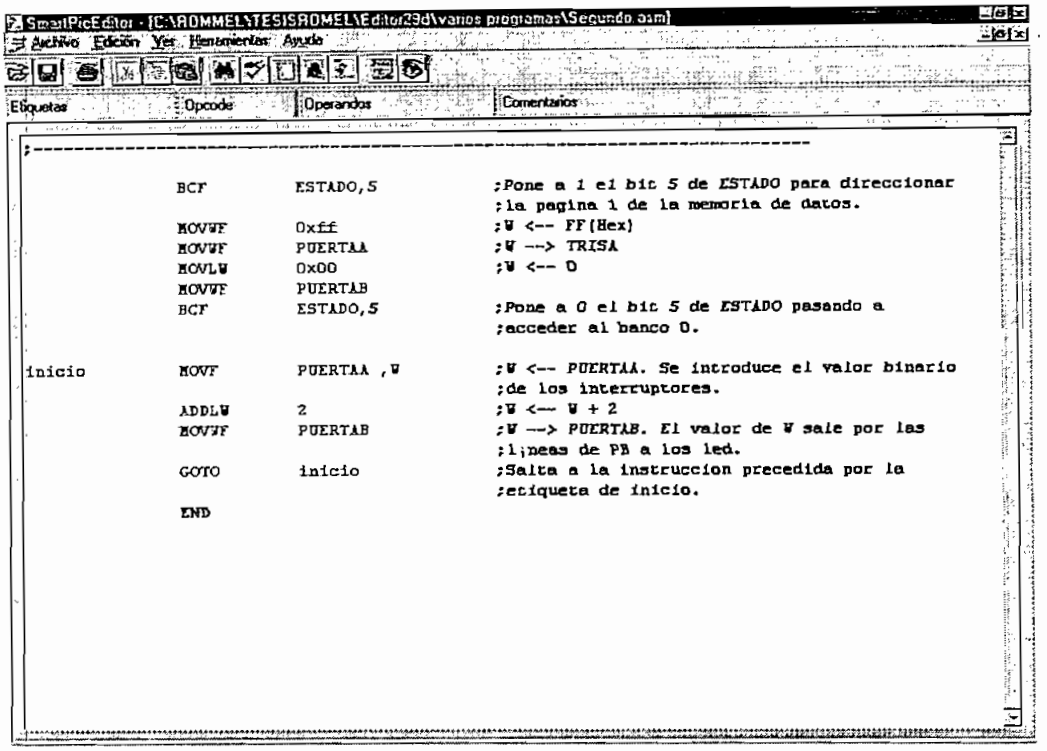


Fig. B.40 Fin de la corrección de errores del programa parte final

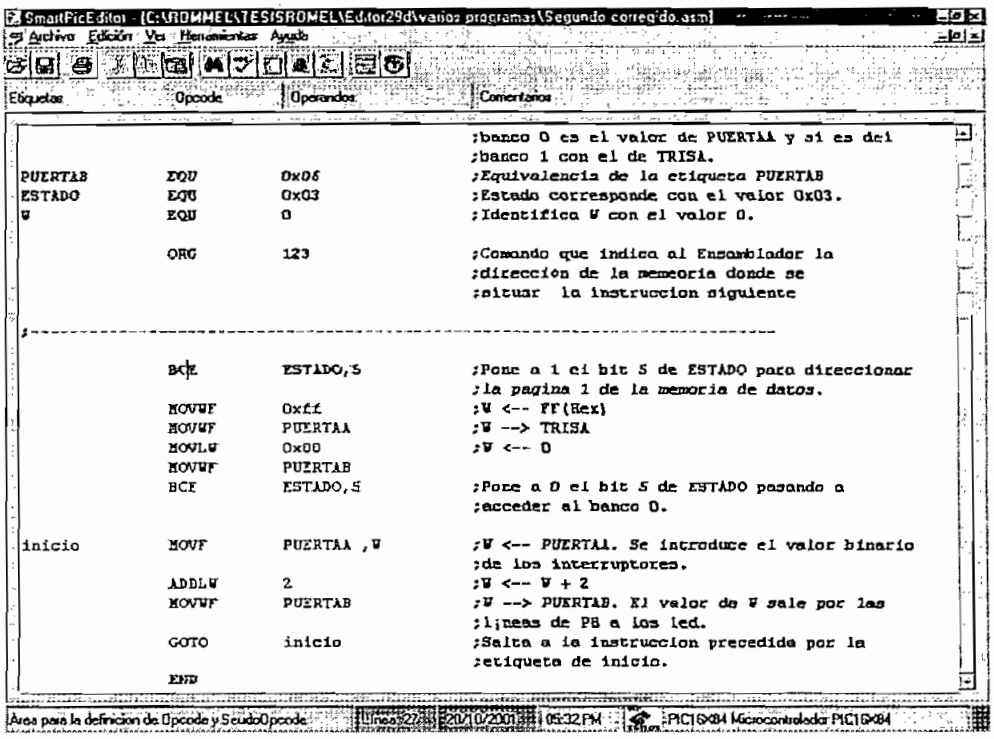


Fig. B.41 Error de opcode en una línea de programa

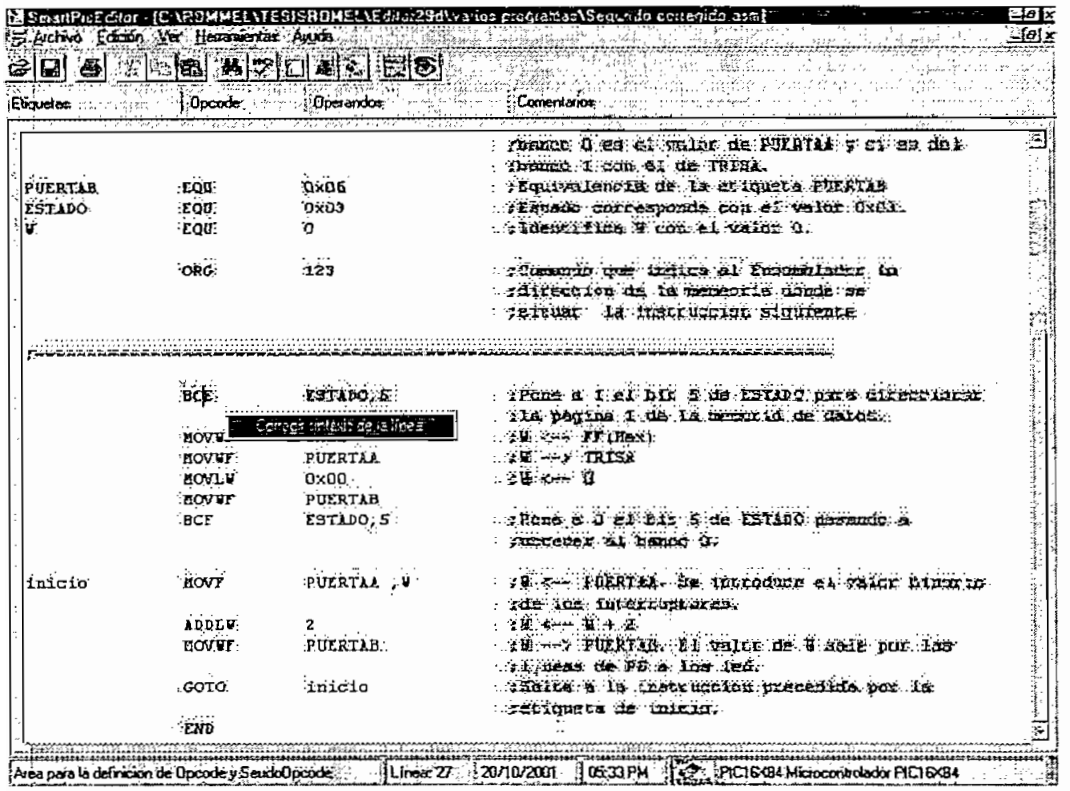


Fig. B.42 Acceso a la corrección de errores de una línea por medio del botón derecho del ratón

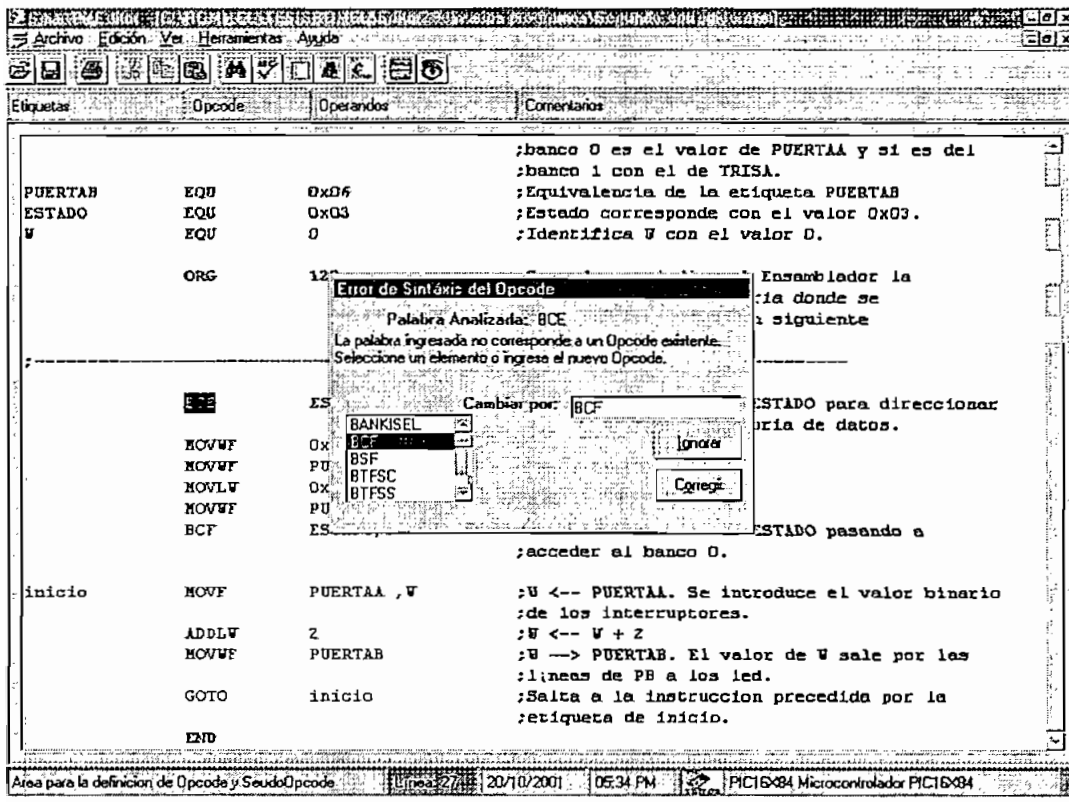


Fig. B.43 Corrección de error de opcode de una instrucción

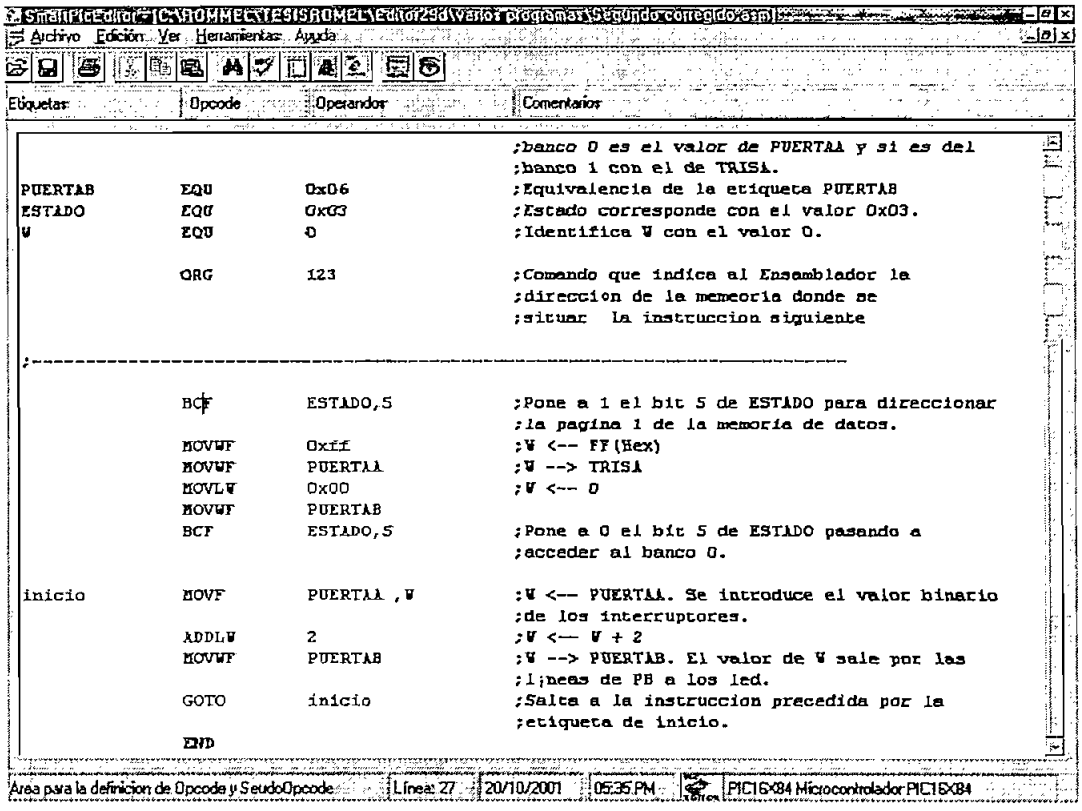
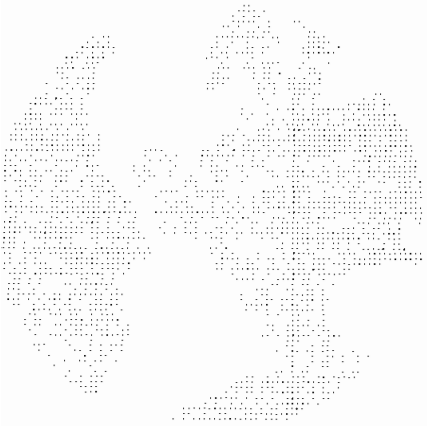


Fig. B.44 Opcode BCF corregido

Manual del Administrador



Módulo de Administración

"Admin PIC Editor"

Versión 1.0

Manual del Administrador del Módulo de Administración "Admin Pic Editor"

1. Ambito

Este manual tiene como objetivo servir de guía para que el usuario pueda conocer y utilizar el Módulo de administración de la base de datos, al que se ha denominado "Admin Pic Editor".

A lo largo de este documento se proporciona el material necesario para asistir al usuario en el aprendizaje de todos los aspectos que conllevan el uso de este módulo Administrador.

> *Este manual está orientado a personas involucradas en el desarrollo de programas para microcontroladores PIC, sean estudiantes, profesores, ingenieros, etc.*

2. Estructura de este Manual

Con el fin de lograr la mayor claridad posible en la explicación de los diferentes tópicos correspondientes al uso del Módulo de administración se ha organizado el presente manual de la siguiente manera:

Acerca del Admin PIC Editor	*Proporciona una descripción general del editor*
Términos utilizados	Se explican brevemente los términos utilizados tanto en el *manual como en el programa editor*
Estructura del Admin Pic Editor	*En esta sección se describe la estructura general del módulo administrador y sus componentes.*
Ingreso del conjunto de instrucciones de un modelo de microcontrolador	Aquí se explica en detalle los pasos para ingresar el conjunto de instrucciones de un modelo de microcontrolador.
Modificaciones que se pueden realizar	*Se indica como editar o eliminar registros de la base de datos.*

3. Acerca del Admin PIC Editor

Bienvenido a Admin PIC Editor, este módulo de administración ha sido creado con el fin de brindar facilidades durante la administración de la base de datos que es usada por el editor Smart Pic Editor.

4. Términos Utilizados

En esta sección se describen algunos de los términos utilizados tanto a lo largo del manual, como en el Administrador.

Admin Pic Editor	Módulo de administración de la base de datos del Editor Smart Pic Editor.
Instrucción	Constituye un conjunto de campos claramente definidos, opcionales u obligatorios que al ser traducidos a código de máquina tienen una longitud definida en bits. Considera los siguientes campos: [Etiqueta] [Opcode [Operandos]]
Etiqueta	Constituye un conjunto de caracteres letra, números, subraya que cumplan que el primer caracter no sea un número.
Opcode	Corresponde a un conjunto de caracteres propios para cada tipo o familia de microcontroladores (nemónicos) que indican el tipo de instrucción.
Operando	Conjunto de caracteres que con el opcode definen la operación de la instrucción, cada operando va separado por comas, y para una instrucción es claramente definido el número de operandos que acompañan al opcode así como su posición.
Comentario	Corresponde a un conjunto de caracteres que el programador puede escribir como contenido aclaratorio del programa que se encuentra realizando. Este viene precedido de un indicador o signo que advierte al programador pero sobre todo al ensamblador que lo que está escrito a continuación es un comentario. Para el PIC este signo es el punto y coma ";".

Pseudoinstrucción o directiva del ensamblador	Presenta una estructura similar al de las instrucciones variando el contenido del opcode y operandos a nemónicos propios de la pseudo instrucción.

5. Estructura del Admin Pic Editor

A continuación se presentan los componentes principales que conforman el Admin Pic Editor

5.1 Pantalla de Presentación

La primera pantalla mostrada por el Admin Pic Editor es la pantalla de presentación, esta pantalla se muestra por unos pocos segundos antes de dar paso a la pantalla principal.

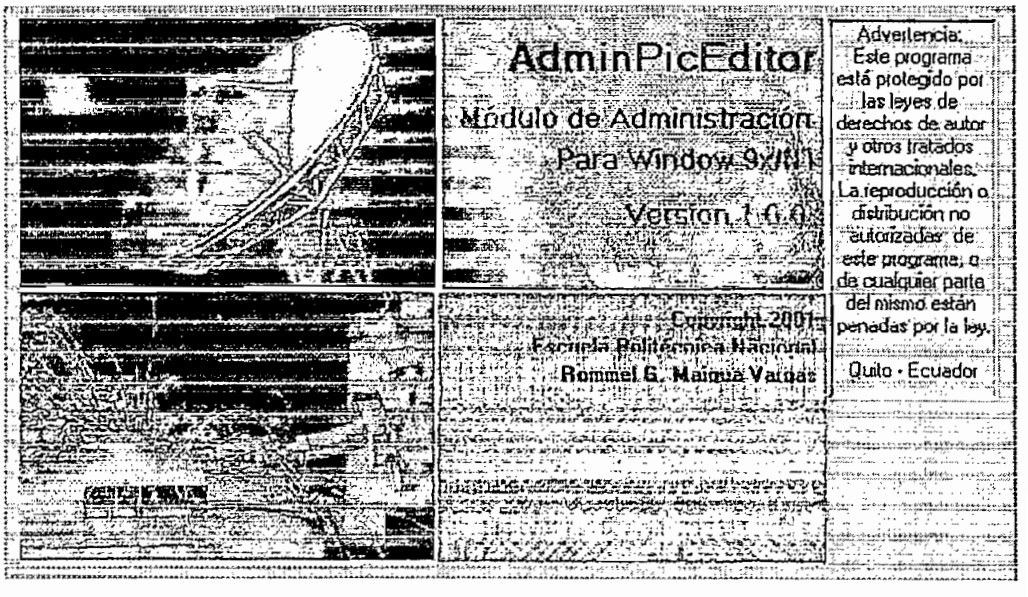


FIGURA B.2.1 Pantalla de presentación del Admin Pic Editor Versión 1.0.0

5.2 Pantalla Principal

La pantalla principal permite la definición o selección del modelo de microcontrolador así como se constituye en la pantalla central desde donde la persona que realiza la administración de la base de datos podrá navegar hacia el resto de pantallas de datos de las instrucciones y directivas del microcontrolador seleccionado y finalmente por esta pantalla se puede salir del Admin Pic Editor.

La pantalla principal esta constituida de los siguientes componentes:

- Botones de Edición.
- Tabla de Navegación de modelos de microcontroladores existentes.
- Botones de acceso a otras pantallas.
- Barra de Navegación.

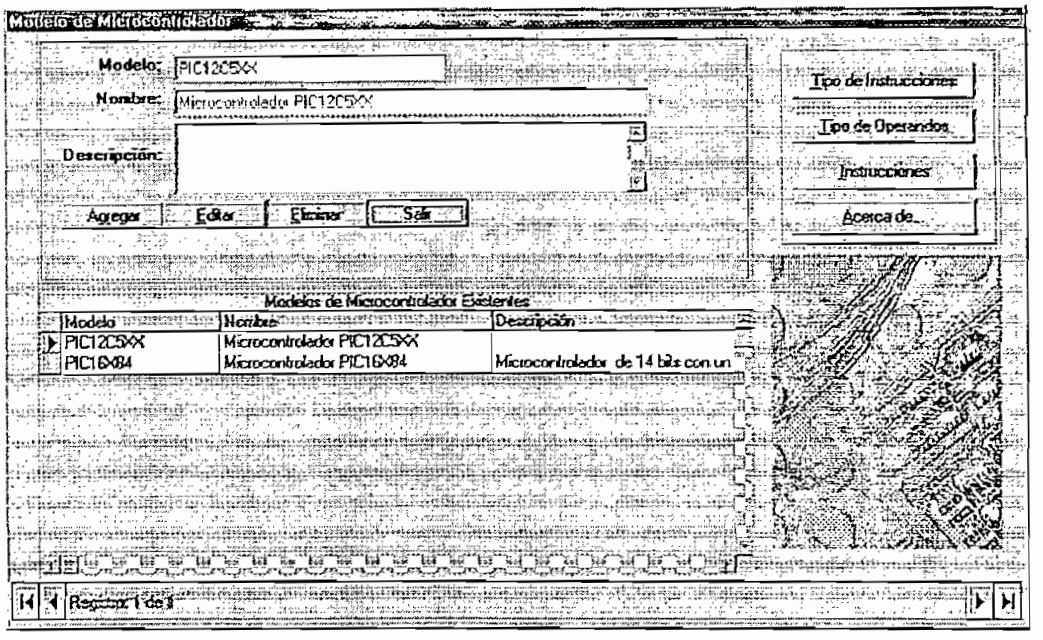


FIGURA B.2.2 Pantalla principal del Admin Pic Editor

5.2.1 BOTONES DE EDICION

Comprende un conjunto de botones que permiten editor los datos de instrucciones y directivas de modelo existentes, comprende los botones: Agregar, editar, eliminar y salir.

Botón Agregar.- Permite añadir un nuevo modelo de microcontrolador a la base de datos.

Botón Editar.- Habilita los campos: nombre y descripción para que el usuario pueda modificarlos , pudiendo luego guardar o cancelar los cambios realizados.

Botón Eliminar.- Permite eliminar un modelo de la base de datos previa la confirmación de esta operación. Cabe anotar que para eliminar modelos deben haber sido previamente eliminados los tipos de instrucciones, tipos de operandos y las instrucciones correspondientes.

Botón Salir.- Cierra el administrador previa la confirmación del usuario.

5.2.2 TABLA DE NAVEGACION DE MODELOS EXISTENTES

Corresponde a una tabla cuyas filas indican los modelos de microcontroladores existentes.

5.2.3 BOTONES DE ACCESO A LAS OTRAS PANTALLAS

En el extremo superior derecho se tienen cuatro botones para acceso al resto de pantallas del administrador, estos botones son:

Botón Tipo de Instrucciones

Botón Tipo de Operandos

Botón Instrucciones

Botón Acerca de.

5.2.4 BARRA DE NAVEGACION

Corresponde una barra que permite la navegación entre los registros de modelos de microcontroladores existentes.

6. Ingreso del conjunto de instrucciones de un nuevo modelo de microcontrolador

Para ejecutar el Módulo de Administración de la base de datos, se debe hacer clic sobre el icono Admin Pic Editor presente en el Menú Programas del escritorio de Windows, Ver la figura B.2.3.; a continuación se muestra por unos segundos la pantalla de presentación del Admin Pic Editor que se puede apreciar en la figura B.2.1.

Finalmente el Módulo de administración, muestra la pantalla de principal "Modelo de Microcontrolador" para que el usuario inicie con la sesión de administración.

A continuación se describe la secuencia de pasos a realizarse para la introducción del set de instrucciones de un nuevo modelo de microcontrolador.

6.1 Definición de un modelo de microcontrolador

Partiendo de la base de datos con sus registros vacíos se ingresa el set de instrucciones del microcontrolador PIC 16X84.

En la figura B.2.4 se puede observar la pantalla para agregar un nuevo modelo de microcontrolador.

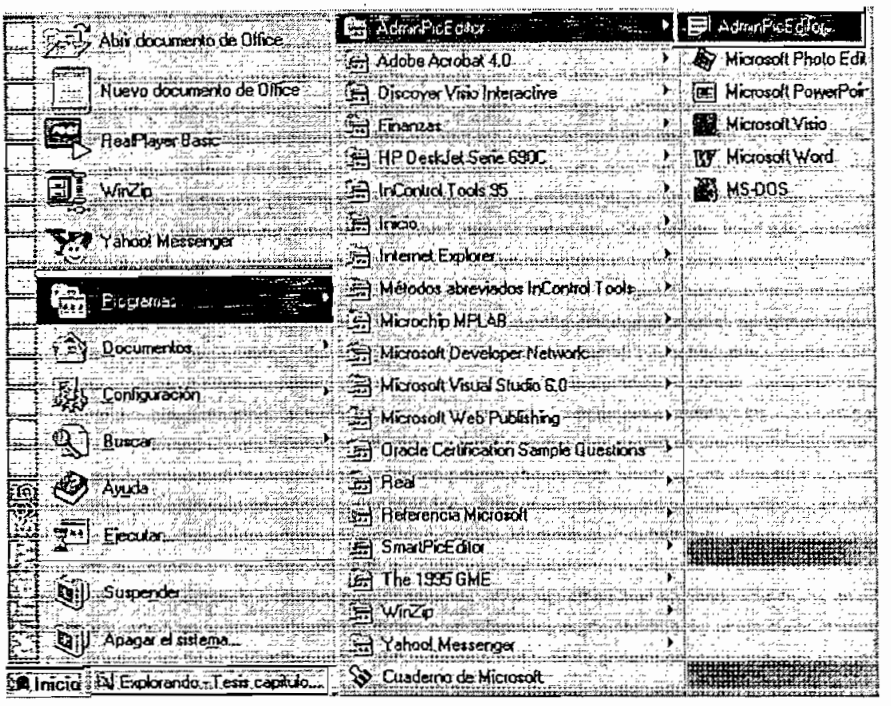


FIGURA B.2.3 Pantalla de presentación del Admin Pic Editor

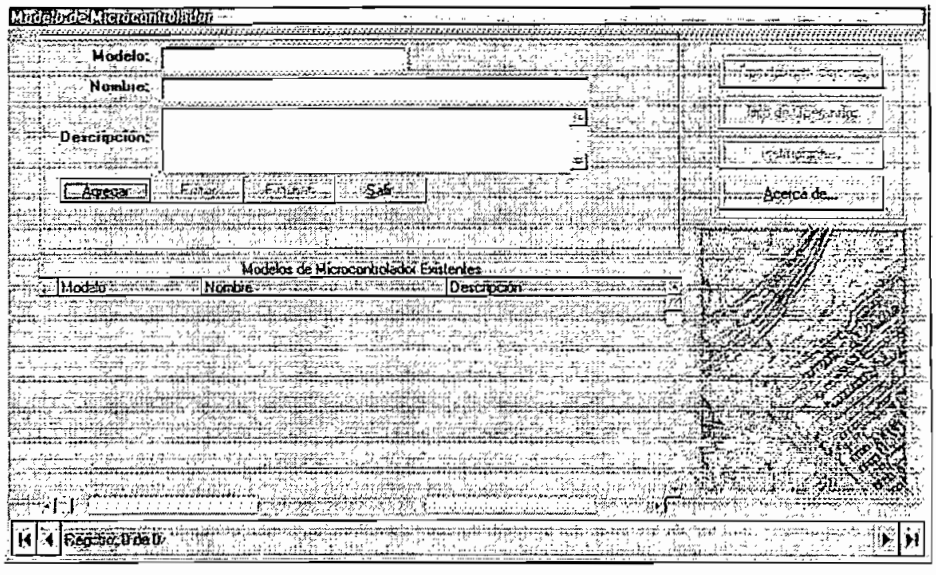


Fig. B.2.4 Pantalla para agregar un nuevo modelo de microcontrolador (base de datos vacía).

En virtud de que no existe ningún modelo presente los botones de edición: "Editar" y "Eliminar", así como los botones "Tipo de Instrucciones", "Tipo de operandos" e "Instrucciones" no se encuentran habilitados. Así mismo se ve que la barra de navegación de registros indica cero registros existentes.

Si se presiona el botón agregar se puede agregar el nuevo modelo, ver figura B.2.5.

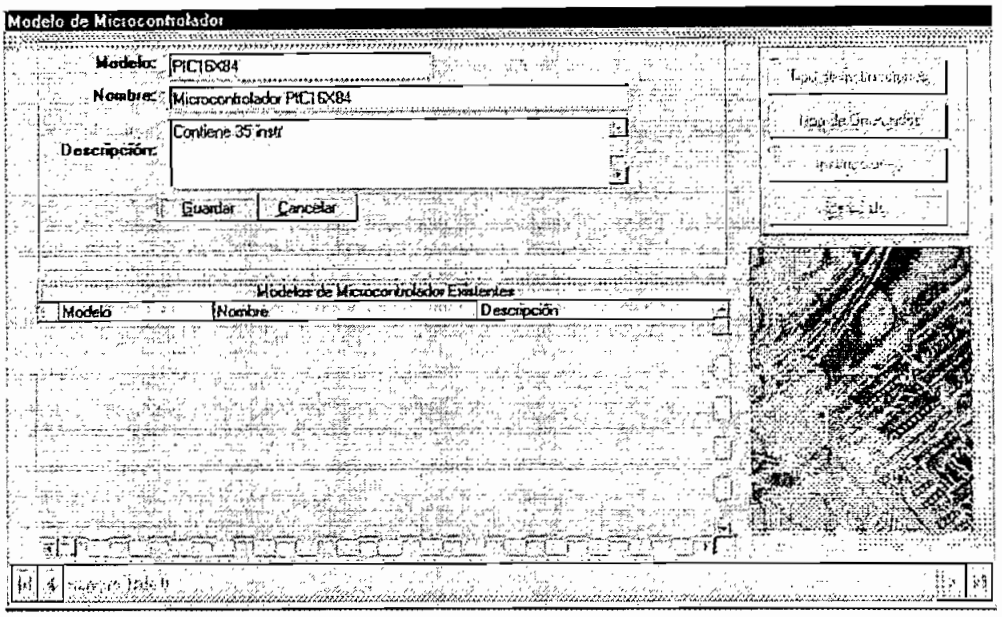


Fig. B.2.5 Pantalla que muestra la agregación de un nuevo modelo de microcontrolador.

Campo Modelo.- Corresponde al código o identificador del modelo de microcontrolador, conjunto de caracteres limitado.

Campo Nombre.- Puede ser un nombre largo del modelo de microcontrolador.

Campo Descripción.- Corresponde a la descripción general del modelo de microcontrolador o anotaciones que el usuario desee realizar.

Los campos modelos y nombre son obligatorios, es por ello que si no se escribe la información correspondiente el Admin Pic Editor solicita llenar dichos campos, el llenado del campo descripción es opcional.

Una vez llenado los campos: modelo, nombre y descripción se puede guardar o cancelar el ingreso de estos datos si se ha guardado se presenta la pantalla de la figura B.2.6, en donde se puede ver que se ha creado un registro cuyos datos son los campos descritos anteriormente.

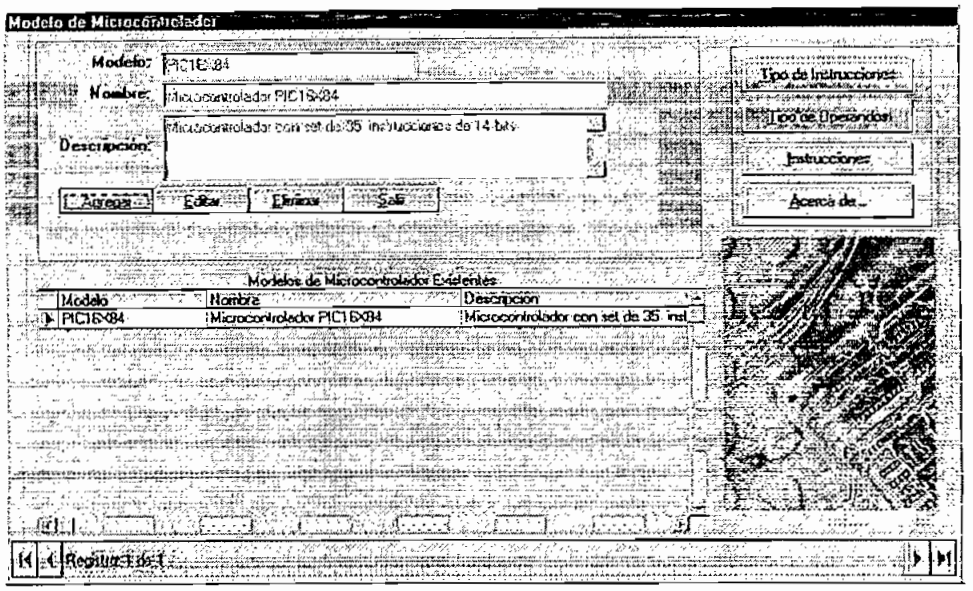


Fig. B.2.6 Pantalla con modelo PIC16X84 agregado.

6.2 Definición de los tipos de instrucciones

Agregado un nuevo modelo se procede a establecer los tipos de instrucciones en los que se podrían clasificar las instrucciones, para ello se hace un clic en el botón "Tipo de instrucciones", ver la figura B.2.7, la definición o no de estos tipos no es obligatoria, pero es aconsejable por motivos de administración.

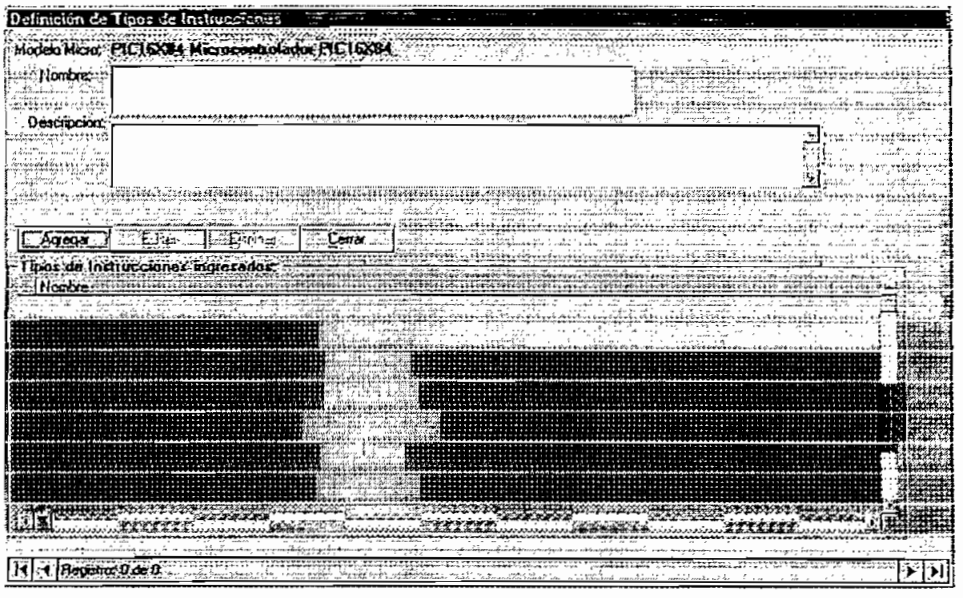


Fig. B.2.7 Pantalla para agregar tipo de Instrucciones.

Los campos presentes para ello son:

Campo Nombre.- Corresponde al nombre del tipo de instrucción por ejemplo: "Instrucciones de salto", si se ha presionado el botón Agregar, este campo es obligatorio para proceder a guardar.

Campo Descripción.- Corresponde a una descripción del tipo de instrucción, este campo es opcional cuando se introduce un tipo de instrucción.

A manera de ejemplo se introduce un tipo de instrucciones ver la figura B.2.8.

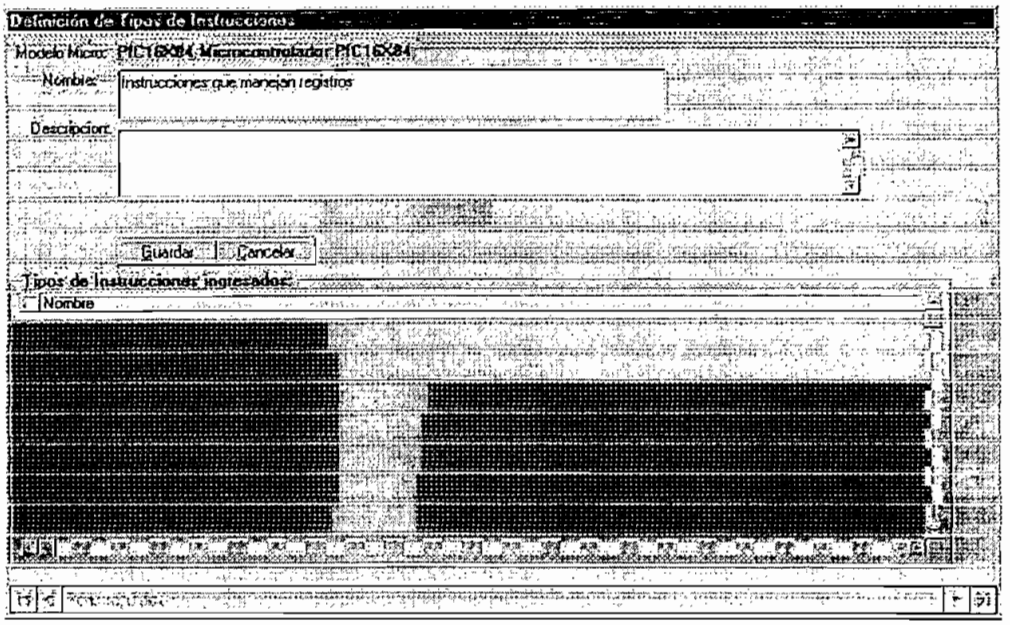


Fig. B.2.8 Pantalla que muestra que se está agregando un tipo de Instrucciones.

Una vez introducida se presiona el botón Guardar para almacenar el tipo de instrucción añadida y la pantalla de tipos de instrucción se puede ver la figura B.2.9.

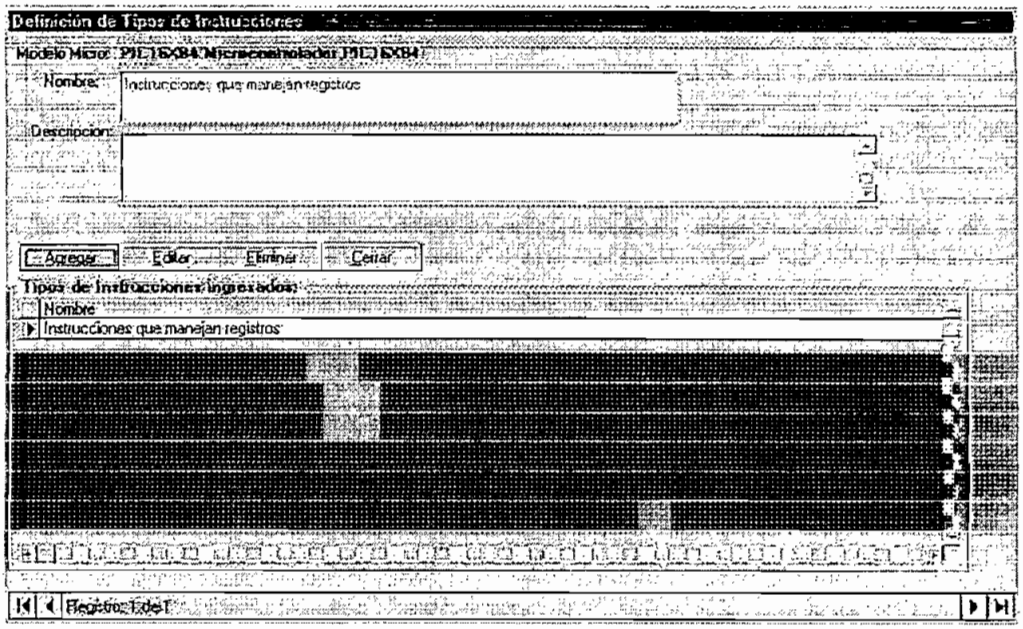


Fig. B.2.9 Pantalla con un tipo de instrucción añadida

6.3 Definición de los tipos de operandos

Antes de definir las instrucciones, se definen los operandos genéricos de las instrucciones a ser añadidas, para ello se hace clic en el botón Tipos de operandos, ver la figura B.2.10.

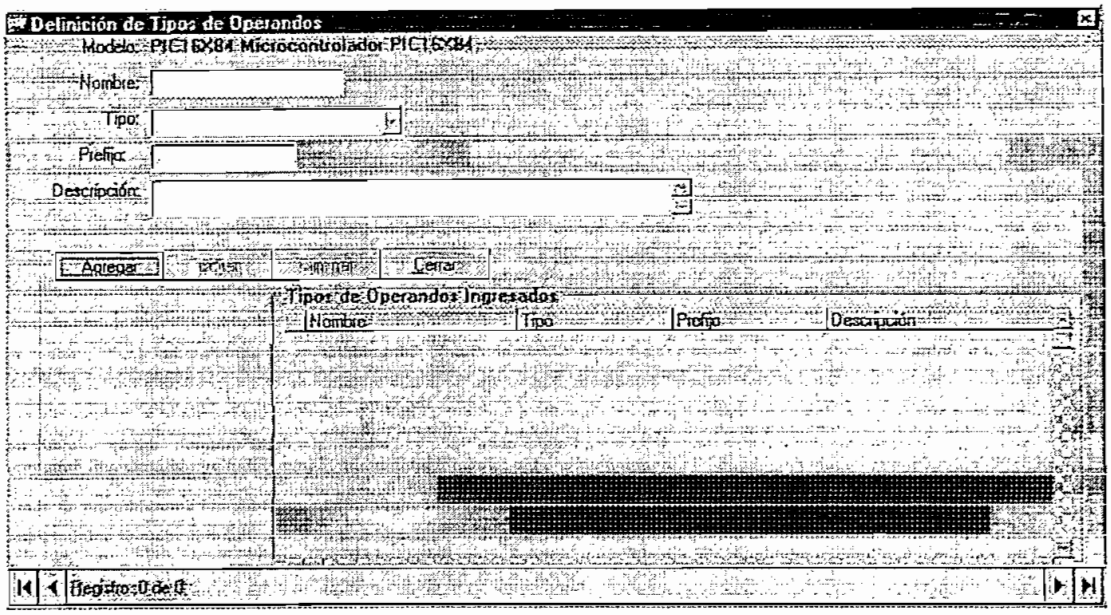


Fig. B.2.10 Pantalla donde se definen los de tipos de operandos

Los campos presentes en esta pantalla son:

Campo Nombre.- Corresponde al nombre del tipo de operando por ejemplo : "f", "d", "b", "argumento", "etiqueta", etc.

Campo Tipo.- Corresponde a un campo, donde se puede seleccionar de entre tres tipos:

- **Constante.-** Corresponde a un conjunto de caracteres definido y que el editor aceptará como único argumento, por ejemplo A (acumulador del set de instrucciones del MCS 51/52).
- **Variable.-** Corresponde a un operando que puede tomar un conjunto de caracteres variable, por ejemplo valores de direcciones, etiquetas, etc, en este caso el editor despliega un cuadro donde el usuario podrá ingresar este operando.
- **Lista.-** Corresponde a un operando similar al tipo variable con la diferencia que el administrador puede sugerir una lista de valores posibles, para ello el usuario debe primero guardar el nuevo operando y luego añadir los valores sugeridos, para ello en la parte inferir derecha se encuentra un

campo donde se puede escribir estos valores y con el botón "+" añadir o con el botón "-" eliminar.

Campo Prefijo.- Corresponde a un campo opcional que permite ingresar un prefijo que debe ser puesto al inicio del operando por ejemplo "#" (en algunos operandos del set de instrucciones del MCS 51/52).

Campo Descripción.- Corresponde a un campo opcional que permite introducir una descripción general del operando.

Una vez ingresados los tipos de operandos la pantalla se ve como en la figura B.2.11.

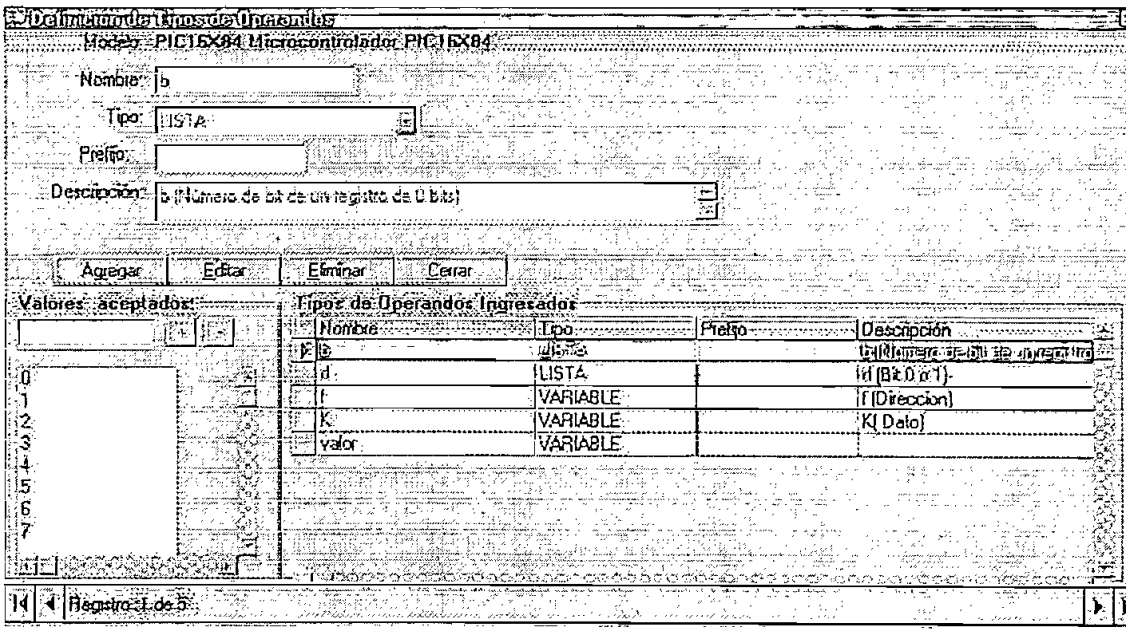


Fig. B.2.11 Pantalla con tipos de operandos definidos

6.4 Definición de las instrucciones

Una vez definidos los tipos de operandos, se pueden ingresar las instrucciones, se hace clic en el botón Instrucciones de la pantalla principal y se accede a la pantalla de la figura B.2.12.

En esta pantalla, los siguientes campos se muestran como listas de opciones, donde se puede seleccionar los valores existentes:

Campo "Tipo ".- Lista de tipos de instrucciones previamente definida.

Campo Opcode.- Constituye el nemónico de la instrucción

Campo "Seudoinstrucción?".- Este campo es un indicador para el Editor, si los datos que se están introduciendo son de una instrucción o pseudo-intrucción, para que el editor pinte azul en caso de pseudo-instrucción, o negro en el caso de una instrucción. Si el usuario no escoge opción, el editor asumirá que se trata de una instrucción.

Campo "Exigir Etiqueta".- Este campo corresponde a un indicador si la instrucción o pseudo-instrucción requiere o no de etiqueta obligatoria, se presenta una lista con dos opciones: "Si" o "No".

Los demás campos deben ser introducidos por el usuario, estos son:

Campo de "Operandos".- Se presenta para cada uno de los cuatro campos una lista de los operandos previamente definidos en la pantalla de definición de tipos de operandos.

Campo "Operandos opcionales".- Se presenta una lista con dos opciones: "Si" o "No", se debe escoger "Si" principalmente para pseudo-instrucciones, que admiten como opcional ninguno, uno o varios opcodes.

Campo Límite de Operandos.- Información del rango de valores que pueden tomar los operandos.

Campo Sintáxis.- Corresponde a la forma de escritura de la instrucción.

Campo Operación Simbólica.- Información de la operación que realiza la instrucción utilizando símbolos.

Campo Ejemplo.- Corresponde a un ejemplo de uso de la instrucción.

Campo Código de Máquina.- Es el código correspondiente a la instrucción después de ensamblarla.

Campo Ciclos.- Indica el número de ciclos de máquina que demora en ejecutarse la instrucción.

Campo Banderas afectadas.- Indican las banderas activadas o desactivadas al ejecutarse la instrucción.

Campo Palabras.- Indica el número de palabras que ocupa la instrucción.

Campo Descripción.- Descripción general de la instrucción

Los campos obligatorios son: Opcode, Exigir etiqueta y operandos opcionales.

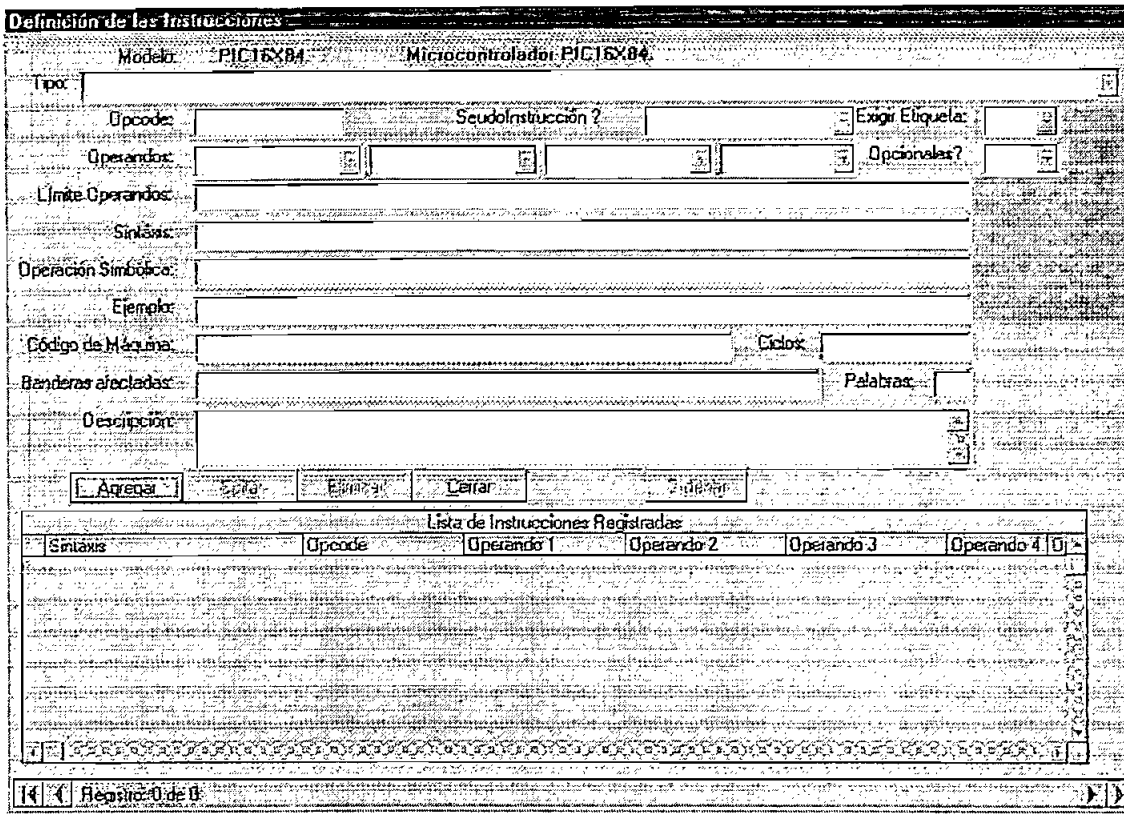


Fig. B.2.12 Pantalla para agregar instrucciones

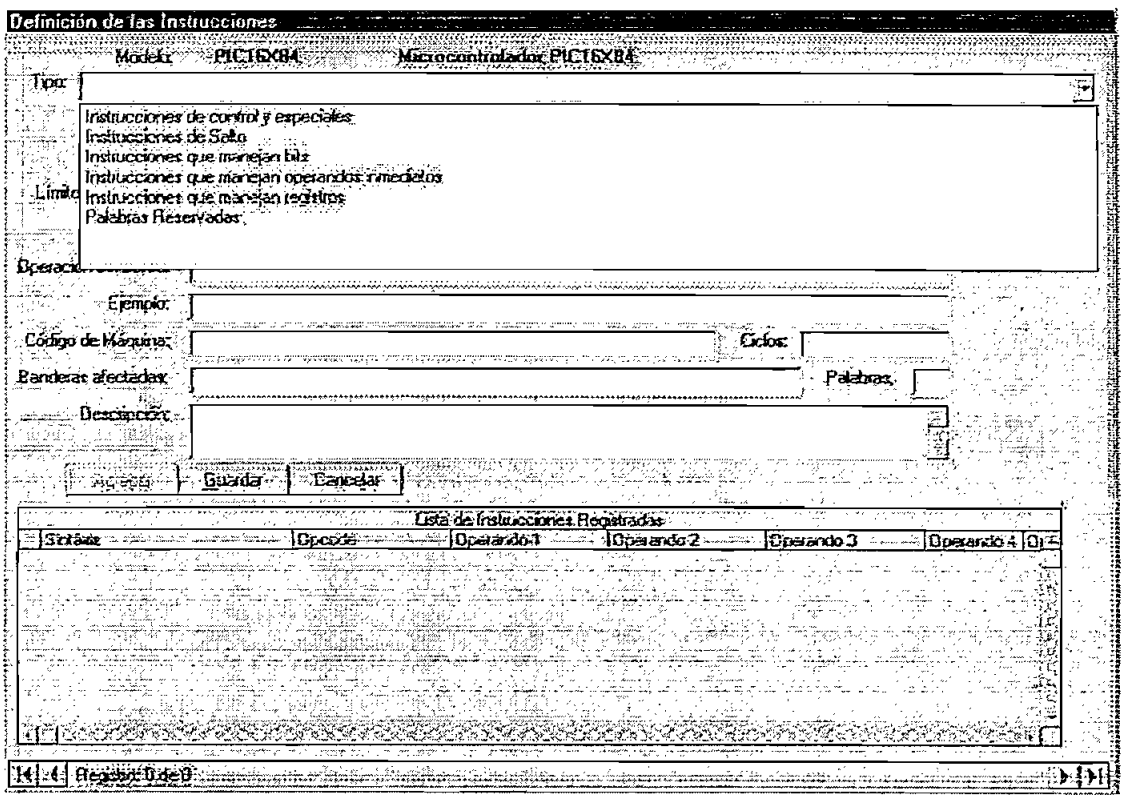


Fig. B.2.13 Pantalla para agregar instrucciones con lista de tipos ingresada

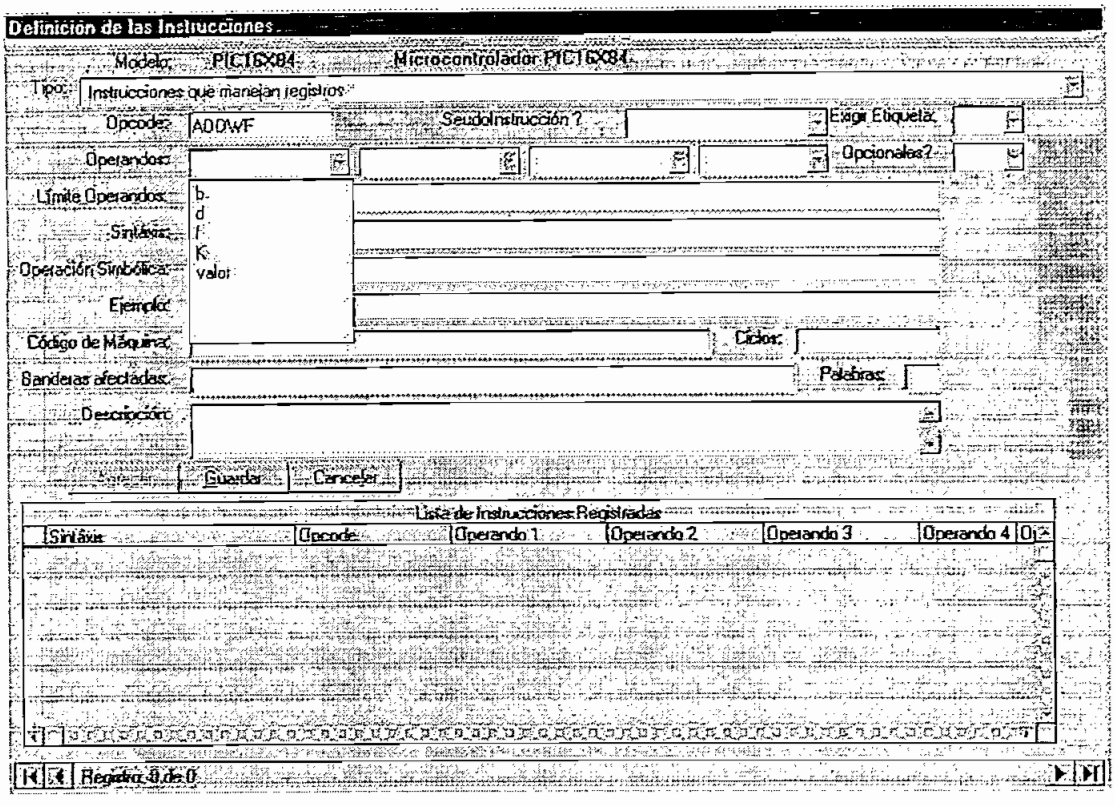


Fig. B.2.14 Pantalla para agregar instrucciones con lista de operandos ingresada

Una vez ingresado los datos de las instrucciones se tiene la pantalla de la figura B.2.15.

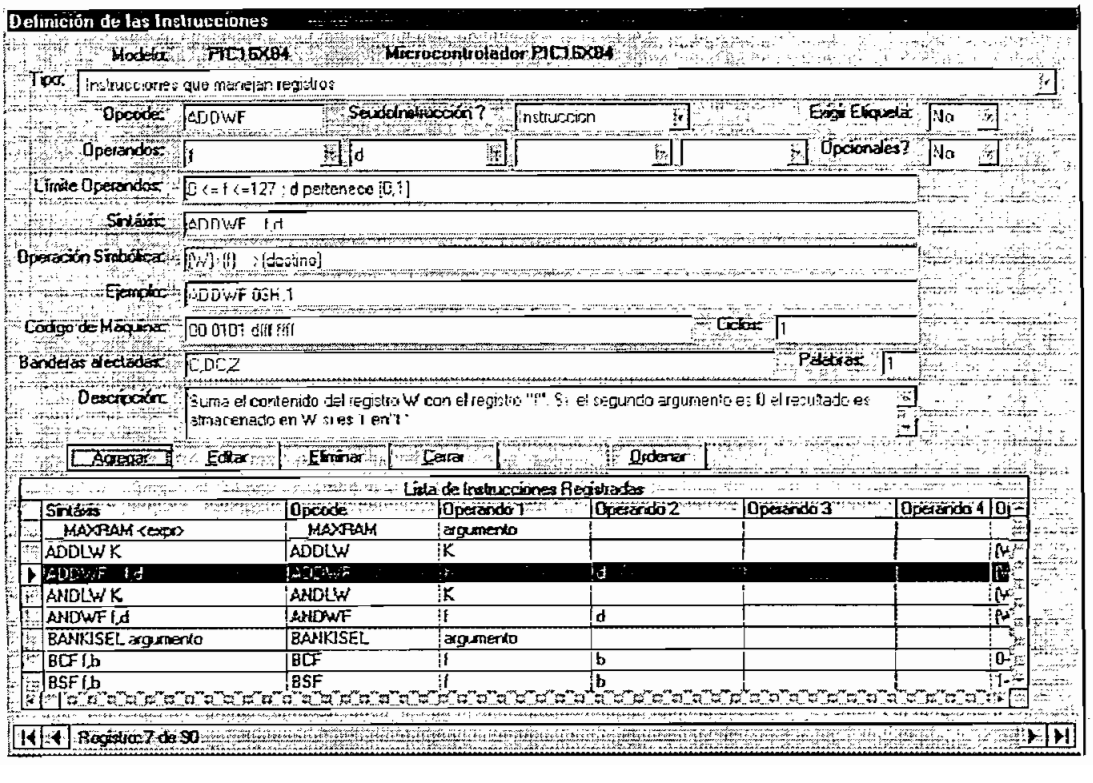


Fig. B.2.15 Pantalla con las instrucciones ingresadas

7. Modificaciones que se pueden realizar

7.1 Edición de registros existentes en la base de datos.

Ingresadas las instrucciones y almacenadas, cada uno de los datos del modelo de microcontrolador, tipos de instrucciones, tipos de operandos, instrucciones aparecen inhabilitados y de color gris.

Las modificaciones que se pueden realizar en cada una de las pantallas se las hace a través del botón "editar" que habilita los campos de datos de cada pantalla y el usuario puede cambiar dichos datos y guardar los cambios realizados.

7.2 Eliminación de registros de la base de datos.

Así mismo se puede eliminar:

- las instrucciones
- los tipos de operandos
- los tipos de instrucciones
- Y el modelo de microcontrolador que contenía dicha información.

Debido a que la base de datos es relacional si se requiere eliminar algún tipo de operando, se debe eliminar primeramente todas las instrucciones que utilizan este tipo de operando, así mismo si se requiere eliminar un tipo de instrucción, se debe eliminar todas las instrucciones que forman parte de este tipo, o asignarlas a otro tipo que se vaya a conservar, finalmente para eliminar un modelo de microcontrolador se debe eliminar todos los datos de: instrucciones, tipos de operandos y tipos de instrucciones, el editor sin embargo indica mensajes de no poder eliminar datos que estén relacionados, como en el ejemplo al intentar borrar el tipo de operando "b" se muestra el mensaje de la figura B.2.16.

Cada vez que se va ha eliminar una instrucción, tipo de argumento, tipo de instrucción o modelo se presenta un mensaje de confirmación como el de la figura B.2.17

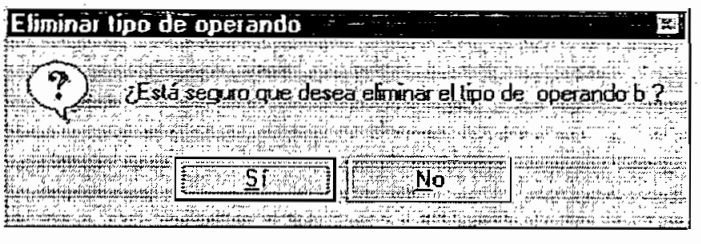


Fig. B.2.16 Mensaje para eliminación de un tipo de operando

Fig. B.2.17 Mensaje de que no se puede eliminar el operando

ANEXO C

LISTADO DEL PROGRAMA

ANEXO C. LISTADO DEL PROGRAMA

La herramienta de software desarrollada consta de los siguientes elementos que se listan a continuación.

CODIGO FUENTE DEL "SMART PIC EDITOR"

- Módulo mVerVariablesyConstantes.
- Módulo mPrincipalEditor.
- Módulo mArchivos.
- Módulo mEdicion.
- Módulo mHerramientas.
- Módulo mInteligente.
- Módulo mVer.
- Módulo mAyudaEditor.
- Formulario frmAcercade.
- Formulario frmBuscar.
- Formulario frmCorreccion.
- Formulario frmModeloMicro.
- Formulario frmOpciones.
- Formulario frmPortada.
- Formulario frmPrincipal.
- Formulario frmPrograma.
- Formulario frmReemplazar.
- Formulario frmSimbolos.

CODIGO FUENTE DEL "ADMIN PIC EDITOR"

- Módulo mAdminPrincipal.
- Formulario frmPortada.
- Formulario frmModeloPic.
- Formulario frmTipoArg.
- Formulario frmTiposInstruccion.
- Formulario frmInstrucciones.
- Formulario frmAbout.

ANEXO D

BASE DE DATOS DEL EDITOR INTELIGENTE

ANEXO D. BASE DE DATOS DEL EDITOR INTELIGENTE

En la presente sección se presenta las diferentes tablas con datos de registros y campos para microcontroladores: PIC 16X84, PIC 12C5XX y 16F87X.

1. TABLA MODELO MICRO

Modelo	Nombre	Descripcion
PIC16X84	Microcontrolador PIC16X84	Microcontrolador con un set de 35 instrrucciones, cada instrucción corresponde a una palabra de 14 bits.
PIC12C5XX	Microcontrolador PIC12C5XX	Microcontrolador con un set de 33 instrucciones, cada instrucción corresponde a una palabra de 12 bits.
PIC16F87X	Microcontrolador PIC16F87X	Microcontrolador con un set de 35 instrrucciones, cada instrucción corresponde a una palabra de 14 bits.

2. TABLA TIPOS DE INSTRUCCIONES

Modelo	ID	Nombre	Descripcion
PIC16X84	1	Instrucciones que manejan registros	
PIC16X84	2	Instrucciones que manejan bits	
PIC16X84	3	Instrucciones de Salto	
PIC16X84	4	Instrucciones que manejan operandos inmediatos	
PIC16X84	5	Instrucciones de control y especiales	
PIC16X84	6	Palabras Reservadas	Palabras Reservadas de PIC que sirven como directivas del compilador.
PIC16F87X	190	Instrucciones de operación de registros orientados al byte	
PIC16F87X	191	Instrucciones de operación de registros orientados al bit.	
PIC16F87X	192	Instrucciones de operación literal y de control	
PIC16F87X	193	Directiva del MPASM (PseudoInstrucción)	

3. TABLA TIPOS DE OPERANDOS

Modelo	Nombre	Descripcion	Tipo	Prefijo
PIC12C5XX	f		VARIABLE	
PIC12C5XX	d		LISTA	
PIC12C5XX	b		LISTA	
PIC12C5XX	k		VARIABLE	
PIC12C5XX	valor		VARIABLE	
PIC12C5XX	argumento		VARIABLE	
PIC16F87X	f		VARIABLE	
PIC16F87X	d		LISTA	
PIC16F87X	b		LISTA	
PIC16F87X	k		VARIABLE	
PIC16F87X	argumento		VARIABLE	
PIC16F87X	valor		VARIABLE	
PIC16F87X	list_option		LISTA	
PIC16F87X	micro		LISTA	
PIC16F87X	sistema		LISTA	
PIC16X84	d	d (Bit 0 o 1)	LISTA	
PIC16X84	f	f (Dirección)	VARIABLE	
PIC16X84	K	K(Dato)	VARIABLE	
PIC16X84	b	b (Número de bit de un registro de 8 bits)	LISTA	
PIC16X84	valor		VARIABLE	
PIC16X84	list_option		LISTA	
PIC16X84	sistema		LISTA	
PIC16X84	micro		LISTA	
PIC16X84	argumento		VARIABLE	

4. TABLA VALOR OPERANDOS

Modelo	argumento	valor
PIC12C5XX	b	0
PIC12C5XX	b	1
PIC12C5XX	b	2
PIC12C5XX	b	3
PIC12C5XX	b	4
PIC12C5XX	b	5
PIC12C5XX	b	6
PIC12C5XX	b	7
PIC12C5XX	d	0
PIC12C5XX	d	1
PIC16F87X	d	0
PIC16F87X	d	1
PIC16F87X	b	0

PIC16F87X	b	1
PIC16F87X	b	2
PIC16F87X	b	3
PIC16F87X	b	4
PIC16F87X	b	5
PIC16F87X	b	6
PIC16F87X	b	7
PIC16F87X	sistema	HEX
PIC16F87X	sistema	DEC
PIC16F87X	sistema	OCT
PIC16F87X	list_option	P=16F874
PIC16F87X	list_option	P=16F873
PIC16F87X	micro	16F873
PIC16F87X	list_option	P=16F877
PIC16F87X	list_option	P=16F876
PIC16F87X	micro	16F874
PIC16F87X	micro	16F876
PIC16F87X	micro	16F877
PIC16X84	d	0
PIC16X84	d	1
PIC16X84	b	0
PIC16X84	b	1
PIC16X84	b	2
PIC16X84	b	3
PIC16X84	b	4
PIC16X84	list_option	P=16C84
PIC16X84	list_option	P=16F84
PIC16X84	sistema	HEX
PIC16X84	b	5
PIC16X84	b	6
PIC16X84	b	7
PIC16X84	sistema	DEC
PIC16X84	micro	16C84
PIC16X84	micro	16F84
PIC16X84	sistema	OCT

5. TABLA DE INSTRUCCIONES

Id	Modelo	Tipo	Mnemonico	Arg1	Arg2	Arg3	Arg4	Sintaxis	Operacion/simbolica	Ejemplo	Descripcion	Codigo maquina	Ciclos	Señalizador	Seudoinstruccion	Operandos	Palabras	Existe Etiqueta	Operandos Opcionales
644	PIC12CSXX		#DEFINE	argumento				#DEFINE <name> [<string>]			Define una etiqueta para sustitución de texto				Pseudoinstruccion			No	SI
674	PIC12CSXX		#UNDEFINE	argumento				#UNDEFINE <label>			Borra una sustitución de etiqueta				Pseudoinstruccion			No	No
636	PIC12CSXX		__BANDRAM	argumento				__BANDRAM <expr>[-<expr>] [, <expr>[-<expr>]]			Specify invalid RAM locations				Pseudoinstruccion			No	SI
633	PIC12CSXX		__CONFIG	valor				__CONFIG <expr> OR __CONFIG <addr>, <expr>		__CONFIG 3FF1H	Configuración de bits específica				Pseudoinstruccion			No	SI
655	PIC12CSXX		__IDLOCS	argumento				IDLOCS <EXPR>			Specify ID locations				Pseudoinstruccion			No	SI
662	PIC12CSXX		__MAXRAM	argumento				__MAXRAM <expr>			Especifica máxima dirección de RAM				Pseudoinstruccion			No	No
350	PIC12CSXX		ADDWF	f	d			ADDWF f,d	(W)+(f)->(destino)	ADDWF 03H,1	Suma el contenido del registro W con el registro 'f'. Si el segundo argumento es 0 el resultado es almacenado en W si es 1 en'f'	0001 00df ffff	1	C,DC,Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
372	PIC12CSXX		ANDLW	K				ANDLW K	(W). AND. (K)->(W)	ANDLW 05H	El contenido del registro W se hace un AND con los 8 bits de K	1110 kkkk kkkk	1	Z	Instruccion	0<=K<=255	1	No	No
351	PIC12CSXX		ANDWF	f	d			ANDWF f,d	(W).AND.(f)->(destino)	ANDWF 05H,0	AND entre el contenido del registro W con el contenido de 'f' . Si 'd' es 0 se almacena en W si d es 1 en 'f'	0001 01df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
637	PIC12CSXX		BANKISEL	argumento				BANKISEL <label>			Genera codigo para selección de banco de RAM para direccionamiento indirecto				Pseudoinstruccion			No	No
661	PIC12CSXX		BANKSEL	argumento				BANKSEL <label>			Genera código de selección de banco de RAM				Pseudoinstruccion			No	No
366	PIC12CSXX		BCF	f	b			BCF f,b	0->(f<b>)	BCF 03H,3	Borra bit 'b' del contenido de 'f'	0100 bbbf ffff	1	...	Instruccion	0<=f<=31; 0<=b<=7	1	No	No
367	PIC12CSXX		BSF	f	b			BSF f,b	1->(f<b>)	BSF 06H,4	Pone a 1 el bit 'b' del contenido de 'f'	0101 bbbf ffff	1	...	Instruccion	0<=f<=31; 0<=b<=7	1	No	No
368	PIC12CSXX		BTFSC	f	b			BTFSC f,b	Brinca si (f<b>)=0	BTFSC FLAG,1	Explora un bit de 'f' y brinca si vale 0	0110 bbbf ffff	1(2)	...	Instruccion	0<=f<=31; 0<=b<=7	1	No	No
369	PIC12CSXX		BTFSS	f	b			BTFSS f,b	Brinca si (Direccion<b>)=1	BTFSS FLAG,1	Explora un bit de 'f' y brinca si vale 1	0111 bbbf ffff	1(2)	...	Instruccion	0<=f<=31; 0<=b<=7	1	No	No
337	PIC12CSXX		CALL	K				CALL K	(PC)+1->TOS, k->PC<10:0>, (PCLATH<4:3>)->PC<12:11>	CALL CALCULAR	Llamada a subrutina	1001 kkkk kkkk	2		Instruccion	0<=K<=255	1	No	No
389	PIC12CSXX		CBLOCK					CBLOCK			Reservación de variables de memoria				Pseudoinstruccion			No	No
311	PIC12CSXX		CLRF	f				CLRF f	00h -> (Direccion) ; 1 -> Z	CLRF FLAG_REG	Borra 'f' (Pone los bits a 0)	0000 011f ffff	1	Z	Instruccion		1	No	No
312	PIC12CSXX		CLRW					CLRW	00h -> (W) ; 1 -> Z	CLRW	Borra W	0000 0100 0000	1	Z	Instruccion		1	No	No
379	PIC12CSXX		CLRWDT					CLRWDT	00h ->WDT ; 0 ->WDT prescaler ; 1 -> TO ; 1-PD	CLRWDT	Borra o refresca el Wath Dog Timer (WDT). También resetea el prescaler del WDT. Los bits de estado TO y PD son seteados.	0000 0000 0100	1	/TO, /TP	Instruccion		1	No	No
638	PIC12CSXX		CODE					[<label>] CODE [<address>]			Inicio de sección de código ejecutable				Pseudoinstruccion			No	SI
313	PIC12CSXX		COMF	f	d			COMF f,d	Complemento (f) -> (destino)	COMF REG1,0	El contenido del registro 'f' es complementado. Si d es 0 el resultado es almacenado en W. Si d es 1 el resultado es almacenado en el registro 'Direccion'	0010 01df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
639	PIC12CSXX		CONSTANT	argumento				CONSTANT <label>=<expr> [,...<label>=<expr>]		constant BufLength=512	Declara constantes tipo simbolo				Pseudoinstruccion			No	SI
640	PIC12CSXX		DA	argumento				[<label>] DA <expr> [, <expr2>, ..., <expr>]		da "sbcad"	Almacena "strings" en memoria de programa				Pseudoinstruccion			No	SI
641	PIC12CSXX		DATA	argumento				[<label>] DATA <expr>,[,<expr>,...,<expr>] or [<label>] DATA "<text_string>"[,"<text_string>",...]		DATA 1,2,ext_label ; constants, externals db 't', 0x0f, 'e', 0x0f, 's', 0x0f, 't', 0x0f	Crea datos texto y numérico				Pseudoinstruccion			No	SI
642	PIC12CSXX		DB	argumento				[<label>] DB <expr>[,<expr>,...,<expr>]			Declara datos de un byte				Pseudoinstruccion			No	SI
643	PIC12CSXX		DE	argumento				[<label>] DE <expr> [, <expr>, ..., <expr>]			Define EEPROM Data				Pseudoinstruccion			No	SI
314	PIC12CSXX		DECF	f	d			DECF f,d	(f) - 1 ->(destino)	DECT CNT,1	Decrementa el registro 'f', Si 'd' es cero el resultado va a W. Si 'd' es 1 es almacenado en 'f'	0000 11df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
329	PIC12CSXX		DECFSZ	f	d			DECFSZ f,d	(Direccion) -1 -> (destino) salta si result=0	DECFSZ CNT,1	El contenido del registro 'f' es decrementado; si 'd' es 0 el resultado va a W; si 'd' es 1 el resultado va a 'f'.Si el resultado es 0 salta una instrucción. ☐	0010 11df ffff	1(2)	...	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
645	PIC12CSXX		DT	argumento				[<label>] DT <expr> [, <expr>, ... <expr>]		dt 'A Message',0	Genera una serie de instrucciones RETLW. Una por cada <expr>. Cada <expr> puede ser valores de 8 bits				Pseudoinstruccion			No	SI
646	PIC12CSXX		DW	argumento				[<label>] DW <expr>[,<expr>,...,<expr>]		DW 39, 'diagnostic 39', (d_list*2+d_offset)	Declara dato de una palabra				Pseudoinstruccion			No	SI
647	PIC12CSXX		ELSE					ELSE			Inicio de alternativa				Pseudoinstruccion			No	No
629	PIC12CSXX		END					END			Directiva de fin de programa				Pseudoinstruccion			No	No
390	PIC12CSXX		ENDC					ENDC		ENDC	Fin de un bloque				Pseudoinstruccion			No	No
648	PIC12CSXX		ENDIF					ENDIF			Fin de bloque #ember condicional				Pseudoinstruccion			No	No
400	PIC12CSXX		ENDM					ENDM		ENDM	Fin de definición de macro				Pseudoinstruccion			No	No
649	PIC12CSXX		ENDW					ENDW			Fin de lazo While				Pseudoinstruccion			No	No
349	PIC12CSXX		EQU	valor				Etiqueta EQU valor	Etiqueta <- valor	DATO1 EQU 05H	Operador e igualdad				Pseudoinstruccion			SI	No
632	PIC12CSXX		ERRORLEVEL	argumento				ERRORLEVEL{0|1|2}+<msgnum>|-<msgnum> [, ...]		ERRORLEVEL 1, -202	Set de tipos de error mostrados en los archivos de error				Pseudoinstruccion			No	SI
650	PIC12CSXX		EXITM					EXITM			Exit de MACRO				Pseudoinstruccion			No	No
651	PIC12CSXX		EXPAND					EXPAND			Expand Macro Listing				Pseudoinstruccion			No	No
652	PIC12CSXX		EXTERN	argumento				EXTERN <label> [,<label>...]			Declara etiquetas externas				Pseudoinstruccion			No	SI
653	PIC12CSXX		FILL	argumento				[<label>] FILL<expr>,<count>		FILL 0x1009, 5 ; fill with a constant	Genera <count> ocurrencias de palabras o bytes de programa. Si se coloca entre paréntesis <expr> puede ser una instrucción #emb!er				Pseudoinstruccion			No	SI
654	PIC12CSXX		GLOBAL	argumento				GLOBAL <label> [,<label>]			Exporta una etiqueta definida				Pseudoinstruccion			No	SI
380	PIC12CSXX		GOTO	K				GOTO K	k->PC <10:0>, PCLATH<4:3> -> PC<12:11>	GOTO THERE	Salto incondicional	101k kkkk kkkk	2	...	Instruccion	0<=K<=255	1	No	No
677	PIC12CSXX		IDATA					[<label>] IDATA [<RAM address>]			Inicia la sección de datos inicializados.				Pseudoinstruccion			No	SI
656	PIC12CSXX		IF	argumento				IF <expr>			Inicio de bloque de código ensamblado				Pseudoinstruccion			No	SI
657	PIC12CSXX		IFDEF	argumento				IFDEF <label>			Ejecuter si el simbolo ha sido definido				Pseudoinstruccion			No	No
356	PIC12CSXX		INCF	f	d			INCF f,d	(f) +1 -> (destino)	INCF CNT,1	Incremente el contenido del registro 'f', Si 'd' es 0 el resultado va a W. Si 'd' es 1 es almacenado en 'f'	0010 10df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No

Id	Modelo	Tipo	Nemonico	Arg1	Arg2	Arg3	Arg4	Sintaxis	OperacionSimbolica	Ejemplo	Descripcion	CodigoMaquina	Ciclos	Semaforo	PseudoInstruccion	Operandos	Palabras	Exige Etiqueta	Operandos Opcionales
330	PIC12CSXX		INCFSZ	f	d			INCFSZ f,d	(f) +1 -> (destino) salta si result=0	INCFSZ CNT,1	El contenido del registro 'f' es incrementado; si 'd' es 0 el resultado va a W, si 'd' es 1 el resultado va a 'f'.Si el resultado es 0 salta una instrucción. El	0011 11df ffff	1(2)	--	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
631	PIC12CSXX		INCLUDE	valor				INCLUDE <<include_file>> OR ICLUDE "<include_file>"		include "c:\sys\sysdefs.inc"	Incluye un archivo fuente adicional				PseudoInstruccion			No	No
373	PIC12CSXX		IORLW	K				IORLW K	(W) OR. K ->W	IORLW 0x35	El contenido del registro W se hace un OR con los 8 bits de K , el resultado es puesto en W	1101 kkkk kkkk	1	Z	Instruccion	0<=K<=255	1	No	No
357	PIC12CSXX		IORWF	f	d			IORWF f,d	(W).OR.(f) -> (destino)	IORWF RESULT,0	OR inclusive del registro W con el registro 'f'. Si el segundo argumento es 0 el resultado es almacenado en W si es 1 en'f'	0001 00df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
386	PIC12CSXX		LIST	valor				LIST valor		LIST P=16C84	Directiva				PseudoInstruccion			No	No
635	PIC12CSXX		LOCAL	argumento				LOCAL <label>[,<label>...]		LOCAL ten, label	Declaración de variable local de MACRO				PseudoInstruccion			No	SI
658	PIC12CSXX		MACRO					<label> MACRO [<arg>,...<arg>]		Read MACRO device, buffer, count	Declara definición de Macro				PseudoInstruccion			SI	SI
663	PIC12CSXX		MESSG	argumento				MESSG "<message_text>"		messg "mssg_macro-001 invoked without argument"	Crea mensajes definidos por el usuario				PseudoInstruccion			No	No
317	PIC12CSXX		MOVF	f	d			MOVF f,d	(Direccion) ->(destino)	MOVF FSR,0	El contenido del registro 'f'. Si el segundo argumento es 0 el destino es W, si es 1el destino es 'f'	0010 00df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
374	PIC12CSXX		MOVLW	K				MOVLW K	K ->(W)	MOVLW 0x5A	Mueve a W los 8 bits de K	1100 kkkk kkkk	1	---	Instruccion	0<=K<=255	1	No	No
318	PIC12CSXX		MOVWF	f				MOVWF f	(W)->(f)	MOVWF OPTION_REG	Mueve datos del registro W al registro 'f'	0000 001f ffff	1	---	Instruccion		1	No	No
664	PIC12CSXX		NOEXPAND					NOEXPAND			Terminar Macro Expansion				PseudoInstruccion			No	No
665	PIC12CSXX		NOLIST					NOLIST			Terminar Listine Output				PseudoInstruccion			No	No
360	PIC12CSXX		NOP					NOP	Ninguna operación	NOP	No opera	0000 0000 0000	1	---	Instruccion		1	No	No
334	PIC12CSXX		OPTION					OPTION		OPTION	Carga el registro OPTION	0000 0000 0010	1	---	Instruccion		1	No	No
342	PIC12CSXX		ORG	valor				ORG valor		ORG 06H	Directiva de Inicio de Programa				PseudoInstruccion			No	No
666	PIC12CSXX		PAGE					PAGE			Insert Listine Page Eject				PseudoInstruccion			No	No
667	PIC12CSXX		PAGESEL	argumento				PAGESEL <label>			Genera codigo de selección de pag. ROM				PseudoInstruccion			No	No
387	PIC12CSXX		RADIX	valor				RADIX valor		RADIX HEX	Formato de Numeración empleado en el programa				PseudoInstruccion			No	No
634	PIC12CSXX		RES	argumento				{<label>} RES <mem_units>		RES 2	Memoria de reserva				PseudoInstruccion			No	No
340	PIC12CSXX		RETLW	K				RETLW K	K->(W)	RETLW VALOR2	Retorno subrutina y carga W = k	1000 kkkk kkkk	2	---	Instruccion	0<=K<=255	1	No	No
320	PIC12CSXX		RLF	f	d			RLF f,d	<- C<- Registro f<- C	RLF REG1,0	El contenido del registro 'f' es rotado un bit a la izquierda a través del acarreo. Si 'd' es 0 el resultado es puesto en el registro W. Si 'd' es 1 el resultado va a 'f'.	0011 01df ffff	1	C	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
362	PIC12CSXX		RRF	f	d			RRF f,d	C-> Registro direccion -> C	RRF REG1,0	Rota f a la dcha. a través del acarreo.	0011 00df ffff	1	C	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
668	PIC12CSXX		SET	argumento				<label> SET <exp>			A la etiqueta <label> es asignada la expresión valida MPASM dada en <exp>. Funcionalmente es similar a la directiva EQU excepto que el valor asignado puede ser alterado por una subsiguiente directiva SET				PseudoInstruccion			SI	No
382	PIC12CSXX		SLEEP					SLEEP	00h -> WDT ; 0-> WDT prescaler;	SLEEP	Pasa al modo de reposo	0000 0000 0011	1	/TO,/PD	Instruccion		1	No	No
669	PIC12CSXX		SPACE	argumento				SPACE <expr>			Inserta un número de líneas en blanco dado por <expr> en el 'listing File' o archivo de texto que indica el código de máquina generado para cada instrucción, directiva MPASM o macro encontrado en el archivo fuente.				PseudoInstruccion			No	No
670	PIC12CSXX		SUBTITLE	argumento				SUBTITLE "<sub_text>"		SUBTITLE "diagnostic section"	Especifica subtitulo de programa. <sub_text> constituye strings ASCII dentro de comillas, de longitud máxima 60 caracteres				PseudoInstruccion			No	SI
363	PIC12CSXX		SUBWF	f	d			SUBWF f,d	(f) - (W) -> (destino)	SUBWF REG1,1	Resta W a f	0000 10df ffff	1	C,DC,Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
323	PIC12CSXX		SWAPF	f	d			SWAPF f,d	(f<3:0>) -> (destino <7:4>) ; (f<7:4>) -> (destino <3:0>)	SWAPF REG,0	Intercambia nibbles	0011 10df ffff	1	---	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
671	PIC12CSXX		TITLE	argumento				TITLE "<title_text>"		TITLE"operational code, rev 5.0"	Especifica título de programa que aparecerá en la parte superior del archivo de código de máquina (listing file). Su longitud máxima es de 60 caracteres y los caracteres ASCII deben ir entre comillas.				PseudoInstruccion			No	SI
376	PIC12CSXX		TRIS	f				TRIS f		TRIS DIRECCION	Carga el registro TRIS	0000 0000 0fff	1	---	Instruccion		1	No	No
676	PIC12CSXX		UDATA					[<label>] UDATA [<RAM address>]			Inicio de sección de datos no inicializados. Los datos no inicializados son datos que son definidos sin un valor inicial.				PseudoInstruccion			No	SI
675	PIC12CSXX		UDATA_ACS					[<label>] UDATA_ACS [<RAM address>]			Inicia el acceso a la sección de datos no inicializados de un archivo objeto				PseudoInstruccion			No	SI
672	PIC12CSXX		UDATA_OVR					[<label>] UDATA_OVR [<address>]			Beginins overlayed unitialized data section				PseudoInstruccion			No	SI
673	PIC12CSXX		UDATA_SHR					[<label>] UDATA_SHR [<address>]			Beginins shared unitialized data section				PseudoInstruccion			No	SI
659	PIC12CSXX		VARIABLE	argumento				VARIABLE <label>[= <expr>,...<label> [=<expr>]]			Declara variable simbólica				PseudoInstruccion			No	SI
660	PIC12CSXX		WHILE	argumento				WHILE <expr>			Realiza un lazo mientras la condición sea verdadera				PseudoInstruccion			No	No
377	PIC12CSXX		XORLW	K				XORLW K	(W).XOR.K ->(W)	XORLW 0xAF	OR Exclusiva de los ocho bits de K con el contenido de W	1111 kkkk kkkk	1	---	Instruccion	0<=K<=255	1	No	No
365	PIC12CSXX		XORWF	f	d			XORWF f,d	(W).XOR.(f) -> (destino)	XORWF REG,1	XOR del contenido de W con f. Si 'd' es 0 es almacenado en el registro W y si 'd' es 1 el resultado es almacenado en el registro 'f'	0001 10df ffff	1	Z	Instruccion	0<=f<=31; d pertenece [0,1]	1	No	No
735	PIC16F87X	193	#DEFINE	argumento				#DEFINE <name> [<string>]			Define una etiqueta para sustitución de texto				PseudoInstruccion			No	SI
761	PIC16F87X	193	#UNDEFINE	argumento				#UNDEFINE <label>			Borra una sustitución de etiqueta				PseudoInstruccion			No	No

Id	Modelo	Tipo	Mnemonico	Arg1	Arg2	Arg3	Arg4	Sintaxis	Operacion Simbolica	Ejemplo	Descripcion	Codigo Maquina	Ciclos	Senalizadores	Es Instruccion	Operandos	Palabras	Exige Etiqueta	Operandos Opcionales
727	PIC16F87X	193	__BANDRAM	argumento				__BANDRAM <expr>[-<expr>] [, <expr>[-<expr>]]			Specify invalid RAM locations				Pseudoinstruccion			No	SI
723	PIC16F87X	193	__CONFIG	valor				__CONFIG <expr> OR __CONFIG <addr>, <expr>		__CONFIG 3FF1H	Configuración de bits específica				Pseudoinstruccion			No	SI
746	PIC16F87X	193	__IDLOCS	argumento				IDLOCS <EXPR>			Specify ID locations				Pseudoinstruccion			No	SI
749	PIC16F87X	193	__MAXRAM	argumento				__MAXRAM <expr>			Especifica máxima dirección de RAM				Pseudoinstruccion			No	No
699	PIC16F87X	192	ADDLW	K				[label] ADDLW K	(W)+k --->(W)	ADDLW 0x15	El contenido del registro W es sumado a los 8 bits de "K" y el resultado es puesto en el registro W.	11 111x kkkk kkkk	1	C,DC,Z	Instruccion			No	No
678	PIC16F87X	190	ADDWF	f	d			[label] ADDWF f,d	(W)+(f) --->(destino)	ADDWF fuente,1	Suma el contenido del registro W con el registro "f". SI el segundo argumento es 0 el resultado es almacenado en W si es 1 en"f"	00 0111 dfff ffff	1	C, DC, Z	Instrucción		1	No	No
700	PIC16F87X	192	ANDLW	K				[label] ANDLW K	(W). AND. (K) --->(W)	ANDLW 0x5F	El contenido del registro W se hace un AND con los 8 bits de K	11 1001 kkkk kkkk	1	Z	Instruccion			No	No
679	PIC16F87X	190	ANDWF	f	d			[label] ANDWF f,d	(W).AND.(f)--->(destino)	ANDWF 05H,0	AND entre el contenido del registro W con el contenido de "f". SI "d" es 0 se almacena en W si d es 1 en "f"	00 0101 dfff ffff	1	Z	Instruccion		1	No	No
728	PIC16F87X	193	BANKISEL	argumento				BANKISEL <label>			Genera codigo para selección de banco de RAM para direccionamiento indirecto				Pseudoinstruccion			No	No
693	PIC16F87X	191	BCF	f	b			[label] BCF f,b	0 --->(f<b>)	BCF FLAG_REG,3	Borra (pone en cero) el bit el "b" del contenido de "f"	01 00bb bfff ffff	1	---	Instruccion			No	No
694	PIC16F87X	191	BSF	f	b			[label] BSF f,b	1--->(f<b>)	BSF FLAG_REG,4	Pone a 1 el bit "b" del contenido de "f"	01 01bb bfff ffff	1	---	Instruccion		1	No	No
695	PIC16F87X	191	BTFSC	f	b			[label] BTFSC f,b	Brinca si (f<b>)=0	BTFSC FLAG,1	Explora un bit de "f", si el bit "b" en el registro "f" es 1 entonces se ejecuta la siguiente instrucción, si el bit b en el registro "f" es 0 la siguiente instrucción es descartada, y un NOP es ejecutado instantaneamente marcando 2 ciclos	01 10bb bfff ffff	1(2)	---	Instruccion		1	No	No
696	PIC16F87X	191	BTFSS	f	b			[label] BTFSS f,b	Brinca si (Direccion<b>)=1	BTFSS FLAG,1	Explora un bit de "f", si el bit "b" en el registro "f" es 0 entonces se ejecuta la siguiente instrucción, si el bit b en el registro "f" es 1 entonces la siguiente instrucción es descartada, y un NOP es ejecutado instantaneamente marcando 2 ciclos	01 11bb bfff ffff	1(2)	---	Instruccion		1	No	No
705	PIC16F87X	192	CALL	K				[label] CALL K	(PC)+1->TOS, k->PC<10:0>, (PCLATH<4:3>)->PC<12:11>	CALL CALCULAR	Llamada a subrutina. Primero, retorna la dirección (PC)+1 es puesta en el stack. El onceavo bit inmediato de la dirección es cargado en los bits del PC <10:0>. Los bits altos del PC son cargados del PCLATH.	10 0kkk kkkk kkkk	2	---	Instruccion		1	No	No
719	PIC16F87X	193	CBLOCK					CBLOCK [<expr>]		CBLOK	Define un bloque de constantes				Pseudoinstruccion			No	SI
680	PIC16F87X	190	CLRF	f				[label] CLRF f	00h ---> (f) ; 1 ---> Z	CLRF FLAG_REG	El contenido del registro "f" es borrado (Pone los bits a 0) y el bit Z es puesto en 1	00 0001 1fff ffff	1	Z	Instruccion		1	No	No
726	PIC16F87X	190	CLRW					[label] CLRW	00h ---> (W) ; 1 ---> Z	CLRW	Borra el registro W. Zero bit (Z) es puesto en 1.	00 0001 0xxx xxxx	1	Z	Instruccion		1	No	No
706	PIC16F87X	192	CLRWDT					[label] CLRWDT	00h --->WDT ; 0 --->WDT prescaler ; 1 ---> /TO ; 1--->/PD	CLRWDT	Borra o refresca el Wath Dog Timer (WDT). También resetea el prescaler del WDT. Los bits de estado /TO y /PD son seteados (puestos en 1).	00 0000 0110 0100	1	/TO, /PD	Instruccion		1	No	No
729	PIC16F87X	193	CODE					[<label>] CODE [<address>]			Inicio de sección de código ejecutable				Pseudoinstruccion			No	SI
681	PIC16F87X	190	COMF	f	d			[label] COMF f,d	Complemento (f) -> (destino)	COMF REG1,0	El contenido del registro "f" es complementado. Si d es 0 el resultado es almacenado en W. Si d es 1 el resultado es almacenado en el registro "f".	00 1001 dfff ffff	1	Z	Instruccion		1	No	No
730	PIC16F87X	193	CONSTANT	argumento				CONSTANT <label>=<expr> [,<label>=<expr>]		constant BufLength=512	Declara constantes tipo simbolo				Pseudoinstruccion			No	SI
731	PIC16F87X	193	DA	argumento				[<label>] DA <expr> [, <expr2>, ..., <expr n>]		de 'abcdef'	Almacena "strings" en memoria de programa				Pseudoinstruccion			No	SI
732	PIC16F87X	193	DATA	argumento				[<label>] DATA <expr>,[,<expr>,...,<expr>] or [<label>] DATA "<text string>"[,"<text string>", ...]		DATA 1,2,ext_label ; constants, externals	Crea datos texto y numérico				Pseudoinstruccion			No	SI
733	PIC16F87X	193	DB	argumento				[<label>] DB <expr>[,<expr>,...,<expr>]		db 'f', 0x0f, 'e', 0x0f, 'e', 0x0f, 'r', '\n'	Declara datos de un byte				Pseudoinstruccion			No	SI
734	PIC16F87X	193	DE	argumento				[<label>] DE <expr> [, <expr>, ..., <expr>]			Define EEPROM Data				Pseudoinstruccion			No	SI
682	PIC16F87X	190	DECF	f	d			[label] DECF f,d	(f) - 1 ->(destino)	DECT CNT,1	Decrementa el registro "f". SI "d" es cero el resultado va a W. SI "d" es 1 el resultado es almacenado en "f"	00 0011 dfff ffff	1	Z	Instruccion		1	No	No
697	PIC16F87X	190	DECFSZ	f	d			[label] DECFSZ f,d	(Direccion) -1 -> (destino) salta si result=0	DECFSZ CNT,1	El contenido del registro "f" es decrementado; si "d" es 0 el resultado va a W; si "d" es 1 el resultado va a "f".SI el resultado es 0 salta una instrucción. SI es 1 se ejecuta la siguiente instrucción.	00 1011 dfff ffff	1(2)	---	Instruccion		1	No	No
736	PIC16F87X	193	DT	argumento				[<label>] DT <expr> [, <expr>, ..., <expr>]		dt '% Message',0	Genera una serie de instrucciones RETLW. Una por cada <expr>. Cada <expr> puede ser valores de 8 bits				Pseudoinstruccion			No	SI
737	PIC16F87X	193	DW	argumento				[<label>] DW <expr>[,<expr>,...<expr>]		DW 39, 'diagnostic 39', (rd_list*2+d_offset)	Declara dato de una palabra				Pseudoinstruccion			No	SI
738	PIC16F87X	193	ELSE					ELSE			Inicio de alternativa				Pseudoinstruccion			No	No
714	PIC16F87X	193	END					END		END	Directiva de Fin de Programa				Pseudoinstruccion			No	No
719	PIC16F87X	193	ENDC					ENDC		ENDC	Fin de un bloque				Pseudoinstruccion			No	No
739	PIC16F87X	193	ENDIF					ENDIF			Fin de bloque asembler condicional				Pseudoinstruccion			No	No
725	PIC16F87X	193	ENDM					ENDM		ENDM	Fin de definición de MACRO				Pseudoinstruccion			No	No
740	PIC16F87X	193	ENDW					ENDW			Fin de lazo While				Pseudoinstruccion			No	No
713	PIC16F87X	193	EQU	valor				<label> EQU <valor>	label <--- valor	DATO1 EQU 05H	El valor indicado en <valor> es asignado a la etiqueta <label>				Pseudoinstruccion			SI	No

Id	Modelo	Tipo	Nemonico	Arg1	Arg2	Arg3	Arg4	Sintaxis	OperacionSimbolica	Ejemplo	Descripcion	CodigoMaquina	Ciclos	Sensitizador	SeudoInstruccion	Operandos	Palabras	Exigir Etiqueta	Operandos Opcionales
722	PIC16F87X	193	ERRORLEVEL	argumento				ERRORLEVEL[0|1|2|+<msgnum>|-<msgnum>] [, ...]		ERRORLEVEL 1, -202	Set de tipos de errgr mostrados en los archivos de error				Pseudolnstruccion			No	SI
741	PIC16F87X	193	EXITM					EXITM			Exit de MACRO				Pseudolnstruccion			No	No
742	PIC16F87X	193	EXPAND					EXPAND			Expand Macro Listing				Pseudolnstruccion			No	No
743	PIC16F87X	193	EXTERN	argumento				EXTERN <label> [,<label> ...]			Declara etiquetas externas				Pseudolnstruccion			No	SI
744	PIC16F87X	193	FILL	argumento				[<label>] FILL(<expr>,<count>		FILL 0x1009, 5 ; fill with 4 constant	Genere <count> ocurrencias de palabras o bytes de programa. Si se coloca entre paréntesis <expr> puede ser una instrucción ejemplar				Pseudolnstruccion			No	SI
745	PIC16F87X	193	GLOBAL	argumento				GLOBAL <label> [,<label>]			Exporta una etiqueta definida				Pseudolnstruccion			No	SI
707	PIC16F87X	192	GOTO	K				[label] GOTO K	k--->PC <10:0>, PCLATH<4:3> ---> PC<12:11>	GOTO THERE	Salto incondicional	10 1kkk kkkk kkkk	2	---	Instruccion			No	No
747	PIC16F87X	193	IF	argumento				IF <expr>			Inicio de bloque de código ensamblado				Pseudolnstruccion			No	SI
748	PIC16F87X	193	IFDEF	argumento				IFDEF <label>			Ejecuter si el símbolo ha sido definido				Pseudolnstruccion			No	No
683	PIC16F87X	190	INCF	f	d			[label] INCF f,d	(f) +1 ---> (destino)	INCF CNT,1	Incrementa el contenido del registro "f". Si "d" es 0 el resultado va a W. Si "d" es 1 es almacenado en "f"	00 1010 dfff ffff	1	Z	Instruccion		1	No	No
696	PIC16F87X	190	INCFSZ	f	d			[label] INCFSZ f,d	(f) +1 ---> (destino) salta si result=0	INCFSZ CNT,1	El contenido del registro "f" es incrementado; si "d" es 0 el resultado va a W; si "d" es 1 el resultado va a "f".Si el resultado es 0 salta una instrucción 0	00 1111 dfff ffff	1(2)	---	Instruccion		1	No	No
717	PIC16F87X	193	INCLUDE	valor				INCLUDE <<include_file>> OR ICLUDE "<include_file>"		include "c:\asm\sysdefs.inc"	Incluye un archivo fuente adicional				Pseudolnstruccion			No	No
701	PIC16F87X	192	IORLW	K				[label] IORLW K	(W) OR. K --->W	IORLW 0x35	El contenido del registro W se hace un OR con los 8 bits de K, el resultado es puesto en W	11 1000 kkkk kkkk	1	Z	Instruccion		1	No	No
684	PIC16F87X	190	IORWF	f	d			[label] IORWF f,d	(W).OR.(f) ---> (destino)	IORWF RESULT,0	OR inclusive del registro W con el registro "f". Si el segundo argumento es 0 el resultado es almacenado en W si es 1 en"f".	00 0100 dfff ffff	1	Z	Instruccion		1	No	No
715	PIC16F87X	193	LIST	list_option				LIST[<list_option>, ..., <list_option>]		LIST p=17c84, f=INHX32, r=DEC	Opciones de lista				Pseudolnstruccion			No	SI
726	PIC16F87X	193	LOCAL	argumento				LOCAL <label>[,<label>...]		LOCAL lari, label	Declaración de variable local de MACRO				Pseudolnstruccion			No	SI
750	PIC16F87X	193	MESSG	argumento				MESSG "<message_text>"		messg "msgg_macro-001 invoked without argument"	Crea mensajes definidos por el usuario				Pseudolnstruccion			No	No
685	PIC16F87X	190	MOVF	f	d			[label] MOVF f,d	(f) --->(destino)	MOVF FSR,0	El contenido del registro "f" es movido al destino dependiendo del estado de "d". Si el segundo operando "d" es 0 el destino es W, si es 1 el destino es "f"	00 1000 dfff ffff	1	Z	Instruccion		1	No	No
702	PIC16F87X	192	MOVLW	K				[label] MOVLW K	K --->(W)	MOVLW 0x5A	Mueve a W los 8 bits de K	11 00xx kkkk kkkk	1	---	Instruccion		1	No	No
686	PIC16F87X	190	MOVWF	f				[label] MOVWF f	(W)--->(f)	MOVWF OPTION_REG	Mueve datos del registro W al registro "f".	00 0000 1fff ffff	1	---	Instruccion		1	No	No
751	PIC16F87X	193	NOEXPAND					NOEXPAND			Terminar Macro Expansion				Pseudolnstruccion			No	No
752	PIC16F87X	193	NOLIST					NOLIST			Terminar Listing Output				Pseudolnstruccion			No	No
687	PIC16F87X	190	NOP					[label] NOP	Ninguna oceración	NOP	No operación	00 0000 0xx0 0000	1	---	Instruccion		1	No	No
712	PIC16F87X	193	ORG	valor				[<label>] ORG <expr>		ORG 06H	Directive de Inicio de Programa				Pseudolnstruccion			No	No
753	PIC16F87X	193	PAGE					PAGE			Insert Listing Page Eject				Pseudolnstruccion			No	No
754	PIC16F87X	193	PAGESEL	argumento				PAGESEL <label>			Genera codigo de selección de pag ROM				Pseudolnstruccion			No	No
721	PIC16F87X	193	PROCESSOR	macro				PROCESSOR <processor_type>		PROCESSOR 16F84	Directiva del MPASM				Pseudolnstruccion			No	No
716	PIC16F87X	193	RADIX	sistema				RADIX sistema		RADIX HEX	Formato de Numeración empleado en el programa. el radix por omisión es hex, Los valores permitidos son: hex, dec, oct				Pseudolnstruccion			No	No
724	PIC16F87X	193	RES	argumento				[<label>] RES <mem_units>		RES 2	Memoria de reserva				Pseudolnstruccion			No	No
708	PIC16F87X	192	RETFIE					[label] RETFIE	TOS ---> PC ; 1--->GIE	RETFIE	Retorno de interrupción.	00 0000 0000 1001	2	---	Instruccion		1	No	No
709	PIC16F87X	192	RETLW	K				[label] RETLW K	K--->(W)	RETLW VALOR2	Retorno subrutina y carga W = k	11 01xx kkkk kkkk	2	---	Instruccion		1	No	No
710	PIC16F87X	192	RETURN					[label] RETURN	TOS--->PC	RETURN	Retorno de subrutina	00 0000 0000 1000	2	---	Instruccion		1	No	No
688	PIC16F87X	190	RLF	f	d			[label] RLF f,d	<---C <--- Registro f <--- C	RLF REG1,0	El contenido del registro "f" es rotado un bit a la izquierda a través del acarreo. Si "d" es 0 el resultado es puesto en el registro W. Si "d" es 1 el resultado va a "f".	00 1101 dfff ffff	1	C	Instruccion		1	No	No
689	PIC16F87X	190	RRF	f	d			[label] RRF f,d	C---> Registro direccion ---> C	RRF REG1,0	El contenido del registro "f" es rotado un bit a la derecha a través del acarreo. Si "d" es 0 el resultado es puesto en el registro W. Si "d" es 1 el resultado va a "f".	00 1100 dfff ffff	1	C	Instruccion		1	No	No
755	PIC16F87X	193	SET	argumento				<label> SET <exp>			A la etiqueta <label> es asignada la expresión valida MPASM dada en <exp> Funcionalmente es similar a la directiva EQU excepto que el valor asignado puede ser alterado por una subsiguiente directiva SET				Pseudolnstruccion			SI	No
711	PIC16F87X	192	SLEEP					[label] SLEEP	00h ---> WDT ; 0---> WDT prescaler; 1-->/TO; 0-->/PD	SLEEP	Pasa al modo de reposo. El procesador es puesto en modo sleep con el oscilador detenido	00 0000 0110 0011	1	/TO, /PD	Instruccion		1	No	No
758	PIC16F87X	193	SPACE	argumento				SPACE <expr>			Inserta un número de líneas en blanco dado por <expr> en el "listing File" o archivo de texto que indica el código de máquina generado para cada instrucción, directiva MPASM o macro encontrado en el archivo fuente.				Pseudolnstruccion			No	No
703	PIC16F87X	192	SUBLW	K				[label] SUBLW K	K- (W) ---> (W)	sublw 0x02	Se resta los bits W de K (con el método complemento de 2). El resultado es puesto en el registro W.	11 0x kkkk kkkk	1	C,DC,Z	Instruccion		1	No	No
757	PIC16F87X	193	SUBTITLE	argumento				SUBTITLE "<sub_text>"		SUBTITLE "diagnostic section"	Especifica subtitulo de programa <sub_text> constituye strings ASCII dentro de comillas, de longitud máxima 60 caracteres				Pseudolnstruccion			No	SI

Id	Modelo	Tipo	Mnemonico	Arg1	Arg2	Arg3	Arg4	Sintaxis	Operacion Simbolica	Ejemplo	Descripcion	Codigo Maquina	Ciclos	Señalizador	Pseudoinstruccion	Operandos	Palabras	Exigir Etiqueta	Operandos Opcionales
690	PIC16F87X	190	SUBWF	f	d			[label] SUBWF f,d	(f) - (W) ---> (destino)	SUBWF REG1,1	Resta W de f. Si 'd' es 0 el resultado es guardado en el registro W, si es 1 el resutado es guardado en 'f'	00 0010 dfff ffff	1	C,DC,Z	Instruccion		1	No	No
691	PIC16F87X	190	SWAPF	f	d			[label] SWAPF f,d	(f<3:0>) -> (destino <7:4>) ; (f<7:4>) -> (destino <3:0>)	SWAPF REG,0	Intercambia nibbles bajos y altos del registro 'f'; si 'd' es 0 el resultado es almacenado en W si es 1 se almacena en el registro 'f'	00 1110 dfff ffff	1	---	Instruccion		1	No	No
758	PIC16F87X	193	TITLE	argumento				TITLE "<title_text>"		TITLE"operational code, rev 5.0"	Especifica título de programa que aparecerá en la parte superior del archivo de código de máquina (listing file). Su longitud máxima es de 60 caracteres y los caracteres ASCII deben ir entre comillas.				Pseudoinstruccion			No	Si
759	PIC16F87X	193	UDATA_OVR					[<label>] UDATA_OVR [<address>]			Begining overlayed uninitialized data section				Pseudoinstruccion			No	Si
760	PIC16F87X	193	UDATA_SHR					[<label>] UDATA_SHR [<address>]			Begining shared uninitialized data section				Pseudoinstruccion			No	Si
762	PIC16F87X	193	VARIABLE	argumento				VARIABLE <label>[= <expr>,...,<label> [=<expr>]]			Declara variable simbólica				Pseudoinstruccion			No	Si
763	PIC16F87X	193	WHILE	argumento				WHILE <expr>			Realiza un lazo mientras la condición sea verdadera				Pseudoinstruccion			No	No
704	PIC16F87X	192	XORLW	K				[label] XORLW K	(W).XOR.K ->(W)	XORLW 0xAF	OR Exclusiva de los ocho bits de K con el contenido de W. El resultado es puesto en el registro W	11 1010 kkkk kkkk	1	---	Instruccion			No	No
692	PIC16F87X	190	XORWF	f	d			[label] XORWF f,d	(W).XOR.(f) ---> (destino)	XORWF REG,1	OR exclusivo del contenido de W con f. Si 'd' es 0 es almacenado en el registro W y si 'd' es 1 el resultado es almacenado en el registro 'f'.	00 0110 dfff ffff	1	Z	Instruccion		1	No	No
572	PIC16XB4	6	#DEFINE	argumento				#DEFINE <name> [<string>]			Define una etiqueta para sustitución de texto				Pseudoinstruccion			No	Si
603	PIC16XB4	6	#UNDEFINE	argumento				#UNDEFINE <label>			Borra una sustitución de etiqueta				Pseudoinstruccion			No	No
561	PIC16XB4	6	__BANDRAM	argumento				__BANDRAM <expr>[:<expr>] [, <expr>[:<expr>]]			Specify invalid RAM locations				Pseudoinstruccion			No	Si
398	PIC16XB4	6	__CONFIG	valor				__CONFIG <expr> OR __CONFIG <addr>, <expr>		__CONFIG 3FF1H	Configuración de bits específica				Pseudoinstruccion			No	Si
585	PIC16XB4	6	__IDLOCS	argumento				IDLOCS <EXPR>			Specify ID locatios				Pseudoinstruccion			No	Si
589	PIC16XB4	6	__MAXRAM	argumento				__MAXRAM <expr>			Especifica máxima dirección de RAM				Pseudoinstruccion			No	No
165	PIC16XB4	4	ADDLW	K				[label] ADDLW K	(W)+K --->(W)	ADDLW 0x15	El contenido del registro W es sumado a los 8 bits de "K" y el resultado es puesto en el registro W.	11 111x kkkk kkkk	1	C,DC,Z	Instruccion	0<=k<=255	1	No	No
143	PIC16XB4	1	ADDWF	f	d			[label] ADDWF f,d	(W)+(f) --->(destino)	ADDWF 0xC2,0	Suma el contenido del registro W con el registro "f". Si el segundo argumento es 0 el resultado es almacenado en W si es 1 en"f"	00 0111 dfff ffff	1	C,DC,Z	Instruccion	0 <= f <=127 ; d pertenece [0,1]	1	No	No
166	PIC16XB4	4	ANDLW	K				[label] ANDLW K	(W). AND. (K) --->(W)	ANDLW 0x5F	El contenido del registro W se hace un AND con los 8 bits de K	11 1001 kkkk kkkk	1	Z	Instruccion	0<= k <= 255	1	No	No
144	PIC16XB4	1	ANDWF	f	d			[label] ANDWF f,d	(W).AND.(f)--->(destino)	ANDWF 05H,0	AND entre el contenido del registro W con el contenido de "f". Si "d" es 0 se almacena en W si d es 1 en "f"	00 0101 dfff ffff	1	Z	Instruccion	0<=f <=127; d pertenece [0,1]	1	No	No
564	PIC16XB4	6	BANKISEL	argumento				BANKISEL <label>			Genera codigo para selección de banco de RAM para direccionamiento indirecto				Pseudoinstruccion			No	No
624	PIC16XB4	6	BANKSEL	argumento				BANKSEL <label>			Genera código de selección de banco de RAM				Pseudoinstruccion			No	No
159	PIC16XB4	2	BCF	f	b			[label] BCF f,b	0 --->(f<b>)	BCF FLAG_REG,3	Borra (pone en cero) el bit el 'b' del contenido de 'f'	01 0bb bfff ffff	1	---	Instruccion	0<=f<=127 ; 0<=b<=7	1	No	No
160	PIC16XB4	2	BSF	f	b			[label] BSF f,b	1--->(f<b>)	BSF FLAG_REG,4	Pone a 1 el bit 'b' del contenido de 'f '	01 01bb bfff ffff	1	---	Instruccion	0<=f<=127 ; 0<=b<=7	1	No	No
161	PIC16XB4	3	BTFSC	f	b			[label] BTFSC f,b	Brince si (f<b>)=0	BTFSC FLAG,1	Explora un bit de 'f', si el bit 'b' en el registro 'f' es 1 entonces se ejecuta la siguiente instrucción, si el bit b en el registro 'f' es 0 la siguiente instrucción es descartada, y un NOP es ejecutado instantaneamente marcando 2 ciclos	01 10bb bfff ffff	1(2)	---	Instruccion	0<=f<=127 ; 0<=b<=7	1	No	No
162	PIC16XB4	3	BTFSS	f	b			[label] BTFSS f,b	Brince si (Direccion<b>)=1	BTFSS FLAG,1	Explora un bit de 'f', si el bit 'b' en el registro "f" es 0 entonces se ejecuta la siguiente instrucción, si el bit b en el registro "f" es 1 entonces la siguiente instrucción es descartada, y un NOP es ejecutado instantaneamente marcando 2 ciclos	01 11bb bfff ffff	1(2)	---	Instruccion	0<=f<=127 ; 0<=b<=7	1	No	No
171	PIC16XB4	5	CALL	K				[label] CALL K	(PC)+1->TOS, k->PC<10:0>, (PCLATH<4:3>)->PC<12:11>	CALL CALCULAR	Llamada a subrutina. Primero, retorna la dirección (PC)+1 es puesta en el stack. El onceavo bit inmediato de la dirección es cargado en los bits del PC <10:0>. Los bits altos del PC son cargados del PCLATH.	10 0kkk kkkk kkkk	2	---	Instruccion	0<=k<=2047	1	No	No
185	PIC16XB4	6	CBLOCK					CBLOCK [<expr>]		CBLOK	Define un bloque de constantes				Pseudoinstruccion			No	Si
145	PIC16XB4	1	CLRF	f				[label] CLRF f	00h ---> (f) ; 1 ---> Z	CLRF FLAG_REG	El contenido del registro "f" es borrado (Pone los bits a 0) y el bit Z es puesto en 1	00 0001 1fff ffff	1	Z	Instruccion	0 <=f<=127	1	No	No
395	PIC16XB4		CLRW					[label] CLRW	00h ---> (W) ; 1 ---> Z	CLRW	Borra el registro W. Zero bit (Z) es puesto en 1.	00 0001 0xxx xxxx	1	Z	Instruccion		1	No	No
172	PIC16XB4	5	CLRWDT					[label] CLRWDT	00h --->WDT ; 0 --->WDT prescaler ; 1 ---> /TO ; 1--->/PD	CLRWDT	Borra o refresca el Wath Dog Timer (WDT). También resetea el prescaler del WDT. Los bits de estado /TO y /PD son seteados (puestos en 1).	00 0000 0110 0100	1	/TO, /PD	Instruccion		1	No	No
566	PIC16XB4	6	CODE					[<label>] CODE [<address>]			Inicio de sección de código ejecutable				Pseudoinstruccion			No	Si

Id	Modelo	Tipo	Mnemonico	Arg1	Arg2	Arg3	Arg4	Sintaxis	Operación/Simbolica	Ejemplo	Descripción	CodigoMaquina	Ciclos	Señalizador	SeudoInstruccion	Operandos	Palabras	Exige Etiqueta	Operandos Decimales
147	PIC16XB4	1	COMF	f	d			[label] COMF f,d	Complemento (f) -> (destino)	COMF REG1,0	El contenido del registro 'f' es complementado. Si d es 0 el resultado es almacenado en W. Si d es 1 el resultado es almacenado en el registro 'f'.	00 1001 dfff ffff	1	Z	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
567	PIC16XB4	6	CONSTANT	argumento				CONSTANT <label>=<expr> [, ..,<label>=<expr>]		constant BufLength=512	Declara constantes tipo simbolo				PseudoInstruccion			No	SI
568	PIC16XB4	6	DA	argumento				[<label>] DA <expr> [, <expr2>, ..., <exprn>]		da "abcde"	Almacena "strings" en memoria de programa				PseudoInstruccion			No	SI
569	PIC16XB4	6	DATA	argumento				[<label>] DATA <expr>,[,<expr>, ..,<expr>] or [<label>] DATA "<text string>"[,"<text string>", ...]		DATA 1,2,ext_label ; constants, externals	Crea datos texto y numérico				PseudoInstruccion			No	SI
570	PIC16XB4	6	DB	argumento				[<label>] DB <expr>[,<expr>, ..,<expr>]		db 'f', 0x0f, 'o', 0x0f, 'r', 0x0f, 'f', '\n'	Declara datos de un byte				PseudoInstruccion			No	SI
571	PIC16XB4	6	DE	argumento				[<label>] DE <expr> [, <expr>, ..., <expr>]			Define EEPROM Data				PseudoInstruccion			No	SI
148	PIC16XB4	1	DECR	f	d			[label] DECF f,d	(f) - 1 ->(destino)	DECT CNT,1	Decrementa el registro 'f', Si 'd' es cero el resultado va a W. Si 'd' es 1 el resultado es almacenado en 'f'	00 0011 dfff ffff	1	Z	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
163	PIC16XB4	3	DECFSZ	f	d			[label] DECFSZ f,d	(Direccion) -1 -> (destino) salta si result=0	DECFSZ CNT,1	El contenido del registro 'f' es decrementado; si 'd' es 0 el resultado va a W; si 'd' es 1 el resultado va a 'f'.Si el resultado es 0 salta una instrucción. Si es 1 se ejecuta la siguiente instrucción.	00 1011 dfff ffff	1(2)	...	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
573	PIC16XB4	6	DT	argumento				[<label>] DT <expr> [, <expr>, ..., <expr>]		dt "A Message",0	Genera una serie de instrucciones RETLW. Una por cada <expr>. Cada <expr> puede ser valores de 8 bits				PseudoInstruccion			No	SI
574	PIC16XB4	6	DW	argumento				[<label>] DW <expr>[,<expr>,...,<expr>]		DW 39, "diagnostic $9", (d_list*2+d_offset)	Declara dato de una palabra				PseudoInstruccion			No	SI
575	PIC16XB4	6	ELSE					ELSE			Inicio de alternativa				PseudoInstruccion			No	No
181	PIC16XB4	6	END					END		END	Directiva de Fin de Programa				PseudoInstruccion			No	No
186	PIC16XB4	6	ENDC					ENDC		ENDC	Fin de un bloque				PseudoInstruccion			No	No
577	PIC16XB4	6	ENDIF					ENDIF			Fin de bloque ensamblar condicional				PseudoInstruccion			No	No
402	PIC16XB4		ENDM					ENDM		ENDM	Fin de definición de MACRO				PseudoInstruccion			No	No
578	PIC16XB4	6	ENDW					ENDW			Fin de lazo While				PseudoInstruccion			No	No
180	PIC16XB4	5	EQU	valor				<label> EQU <valor>	Etiqueta <--- valor	DATO1 EQU 05H	El valor indicado en <valor> es asignado a la etiqueta <label>				PseudoInstruccion			Si	No
397	PIC16XB4	6	ERRORLEVEL	argumento				ERRORLEVEL[0|1|2|+<msgnum>|-<msgnum>] [, ..]		ERRORLEVEL 1, -202	Sel de tipos de error mostrados en los archivos de error				PseudoInstruccion			No	SI
579	PIC16XB4	6	EXITM					EXITM			Exit de MACRO				PseudoInstruccion			No	No
580	PIC16XB4	6	EXPAND					EXPAND			Expand Macro Listing				PseudoInstruccion			No	No
581	PIC16XB4	6	EXTERN	argumento				EXTERN <label> [,<label> ...]			Declara etiquetas externas				PseudoInstruccion			No	SI
582	PIC16XB4	6	FILL	argumento				[<label>] FILL<expr>,<count>		FILL 0x1009, 5 ; fill with a constant	Genera <count> ocurrencias de palabras o bytes de programa. Si se coloca entre paréntesis <expr> puede ser una instrucción ejemplar				PseudoInstruccion			No	SI
583	PIC16XB4	6	GLOBAL	argumento				GLOBAL <label> [,<label>]			Exporta una etiqueta definida				PseudoInstruccion			No	SI
173	PIC16XB4	5	GOTO	K				[label] GOTO K	(K--->PC <10:0>, PCLATH<4:3> ---> PC<12:11>	GOTO THERE	Salto incondicional	10 1kkk kkkk kkkk	2	...	Instruccion	0<=K<=2047	1	No	No
628	PIC16XB4	6	IDATA					[<label>] IDATA [<RAM address>]			Inicia la sección de datos inicializados.				PseudoInstruccion			No	SI
586	PIC16XB4	5	IF	argumento				IF <expr>			Inicio de bloque de código ensamblado				PseudoInstruccion			No	SI
587	PIC16XB4	6	IFDEF	argumento				IFDEF <label>			Ejecutar si el simbolo ha sido definido				PseudoInstruccion			No	No
149	PIC16XB4	1	INCF	f	d			[label] INCF f,d	(f) +1 ---> (destino)	INCF CNT,1	Incrementa el contenido del registro 'f', Si 'd' es 0 el resultado va a W. Si 'd' es 1 es almacenado en 'f'	00 1010 dfff ffff	1	Z	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
164	PIC16XB4	3	INCFSZ	f	d			[label] INCFSZ f,d	(f) +1 ---> (destino) salta si result=0	INCFSZ CNT,1	El contenido del registro 'f' es incrementado; si 'd' es 0 el resultado va a W; si 'd' es 1, el resultado va a 'f'.Si el resultado es 0 salta una instrucción.	00 1111 dfff ffff	1(2)	...	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
184	PIC16XB4	6	INCLUDE	valor				INCLUDE <<include_file>> OR ICLUDE "<include_file>"		include "c:\sys\sysdefs.inc"	Incluye un archivo fuente adicional				PseudoInstruccion			No	No
167	PIC16XB4	4	IORLW	K				[label] IORLW K	(W) OR K ---> W	IORLW 0x35	El contenido del registro W se hace un OR con los 8 bits de K , el resultado es puesto en W	11 1000 kkkk kkkk	1	Z	Instruccion	0<=K<=255	1	No	No
150	PIC16XB4	1	IORWF	f	d			[label] IORWF f,d	(W).OR.(f) ---> (destino)	IORWF RESULT,0	OR inclusive del registro W con el registro 'f'. Si el segundo argumento es 0 el resultado es almacenado en W si es 1 en'f'	00 0100 dfff ffff	1	Z	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
182	PIC16XB4	6	LIST	list_option				LIST[<list_option>, ..., <list_option>]		LIST p=17c84, r=INHX32, r=DEC	Opciones de lista				PseudoInstruccion			No	SI
404	PIC16XB4	6	LOCAL	argumento				LOCAL <label>[,<label>...]		LOCAL len, label	Declaración de variable local de MACRO				PseudoInstruccion			No	SI
625	PIC16XB4	6	MACRO					<label> MACRO [<arg>, ..<arg>]		Read MACRO device, buffer, count	Declara definición de Macro				PseudoInstruccion			SI	SI
590	PIC16XB4	6	MESSG	argumento				MESSG "<message_text>"		messg "mssg_macro-001 invoked without argument"	Cres mensajes definidos por el usuario				PseudoInstruccion			No	No
151	PIC16XB4	1	MOVF	f	d			[label] MOVF f,d	(f) --->(destino)	MOVF FSR,0	El contenido del registro 'f' es movido al destino dependiendo del estado de 'd'. Si el segundo operando 'd' es 0 el destino es W, si es 1 el destino es 'f'	00 1000 dfff ffff	1	Z	Instruccion	0<=f<=127; d pertenece [0,1]	1	No	No
168	PIC16XB4	4	MOVLW	K				[label] MOVLW K	K --->(W)	MOVLW 0x5A	Mueve a W los 8 bits de K	11 00xx kkkk kkkk	1	...	Instruccion	0<=K<=255	1	No	No
152	PIC16XB4	1	MOVWF	f				[label] MOVWF f	(W)--->(f)	MOVWF OPTION_REG	Mueve datos del registro W al registro 'f'.	00 0000 1fff ffff	1	...	Instruccion	0<=f<=127	1	No	No
591	PIC16XB4	6	NOEXPAND					NOEXPAND			Terminar Macro Expansion				PseudoInstruccion			No	No
592	PIC16XB4	6	NOLIST					NOLIST			Terminar Listing Output				PseudoInstruccion			No	No
153	PIC16XB4	1	NOP					[label] NOP	Ninguna operación	NOP	No operación	00 0000 0xx0 0000	1	---	Instruccion		1	No	No
179	PIC16XB4	6	ORG	valor				[<label>] ORG <expr>		ORG 05H	Directiva de Inicio de Programa				PseudoInstruccion			No	No
593	PIC16XB4	6	PAGE					PAGE			Insert Listing Page Eject				PseudoInstruccion			No	No
594	PIC16XB4	6	PAGESEL	argumento				PAGESEL <label>			Genera codigo de selección de pag ROM				PseudoInstruccion			No	No
396	PIC16XB4		PROCESSOR	micro				PROCESSOR <processor_type>		PROCESSOR 16F84	Directiva del MPASM				PseudoInstruccion			No	No