UAA
CENTRO DE CIENCIAS BASICAS
DEPARTAMENTO DE SISTEMAS ELECTRONICOS
ACADEMIA DE ELECTRONICA

TUTORIAL DEL 8051

PRESENTACION:
El curso esta dirigido a estudiantes de semestres intermedios y avanzados de Ingeniería en Sistemas o Electrónica, es requisito indispensable tener conocimientos de Lenguaje Ensamblador, Arquitectura de Computadoras y habilidades para hacer interfaces con computadoras

OBJETIVO
Al final del curso el alumno debe ser capaz de armar una aplicación sencilla usando el microntrolador 8051, apoyándose en simuladores para lograr una desempeño adecuado.

CONTENIDO

I. INTRODUCCION AL 8051

1.1 Importancia de los microcontroladores

Los microcontroladores son dispositivos diseñados para ser usados en productos que no son usualmente considerados como computadoras en sí, pero que requieren la sofisticación y el control flexible que una computadora puede proporcionar; por ejemplo: una fotocopiadora, o un horno de microondas. En contraste con los microprocesadores típicos, los microcontroladores integran CPU, RAM, ROM, I/O, Timers y en algunos casos, convertidores ADC. Además, debido a la limitante de espacio en ROM, el juego de instrucciones está diseñado en su mayoría con instrucciones de un solo byte.

La mayoría de las aplicaciones de los microcontroladores, caen en una de dos categorías: Sistemas de Control de Lazo Abierto o Sistemas de Control de Lazo Cerrado; aunque también pueden encontrarse en aplicaciones de manipulación de estructura de datos, por ejemplo, en sistemas de robótica, o en sistemas de comunicación.

El uso de microcontroladores, es una solución con una excelente relación costo-beneficio en aplicaciones de producción en masa, por ejemplo, la gran variedad de equipo de consumo-electrónico, como videograbadoras, televisores, equipos de audio, etc. y otras aplicaciones como los modems inteligentes, impresoras, etc.

La sig. figura muestra el diagrama de bloques de una computadora, y también puede ser el diagrama a bloques de un microcontrolador.

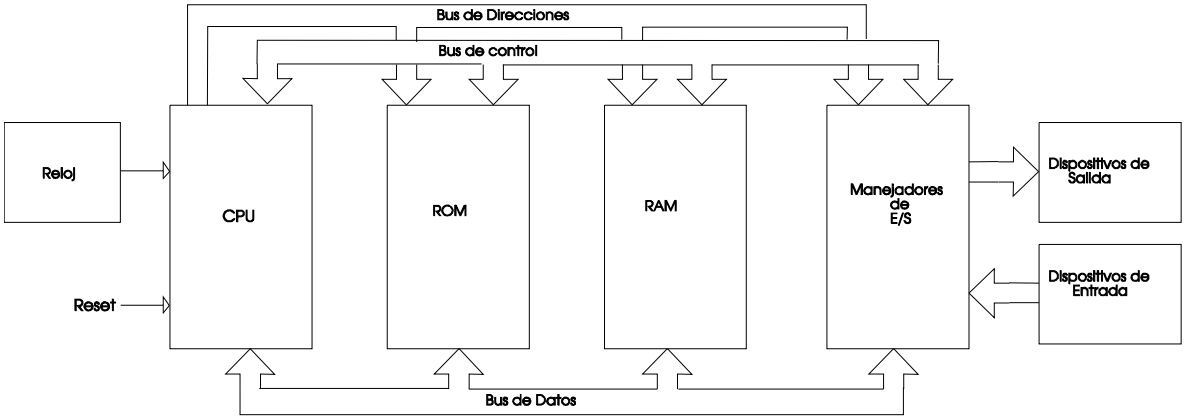


Fig. 1.1 Diagrama a bloques de una micro-computadora.

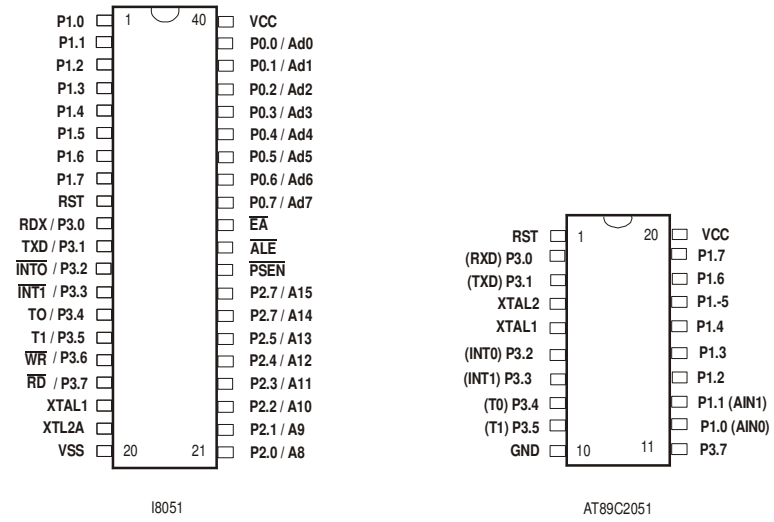


Fig. 1.2 Terminales

1.2 Diagrama de Bloques del 8051

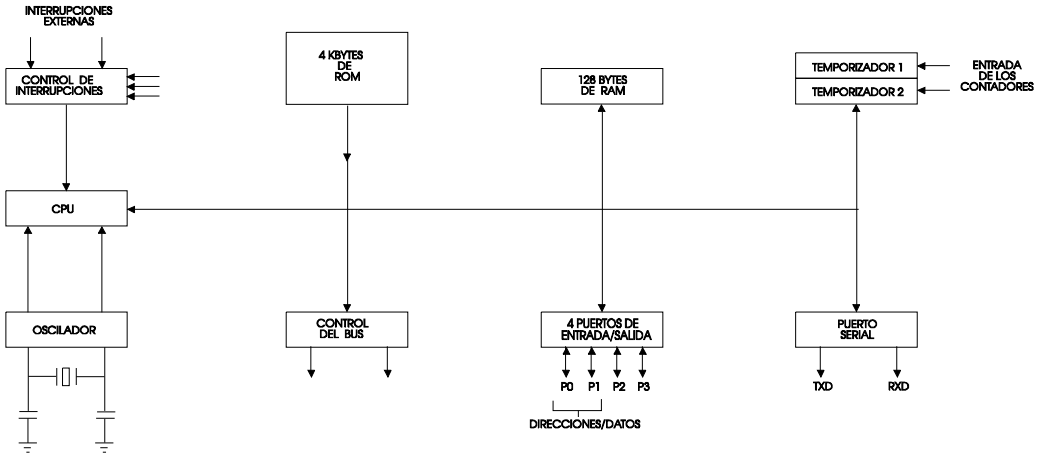


Fig. 1.3 Diagrama de Bloques

1.3 ORGANIZACION DE LA MEMORIA

Separación del espacio de memoria y tamaño.

A diferencia de otros procesadores, el 8051 tiene espacios separados de direcciones de memoria: uno para almacenar programas (ROM), y otro para almacenar datos (RAM). Esto es, Arquitectura de Harvard. El 8051 soporta hasta 64K bytes de almacenamiento de programas, los primeros 4K están dentro del chip, el resto, fuera de él. Además de la RAM interna, el 8051 soporta 64K bytes de memoria de datos, externa. Véase la figura 1.3

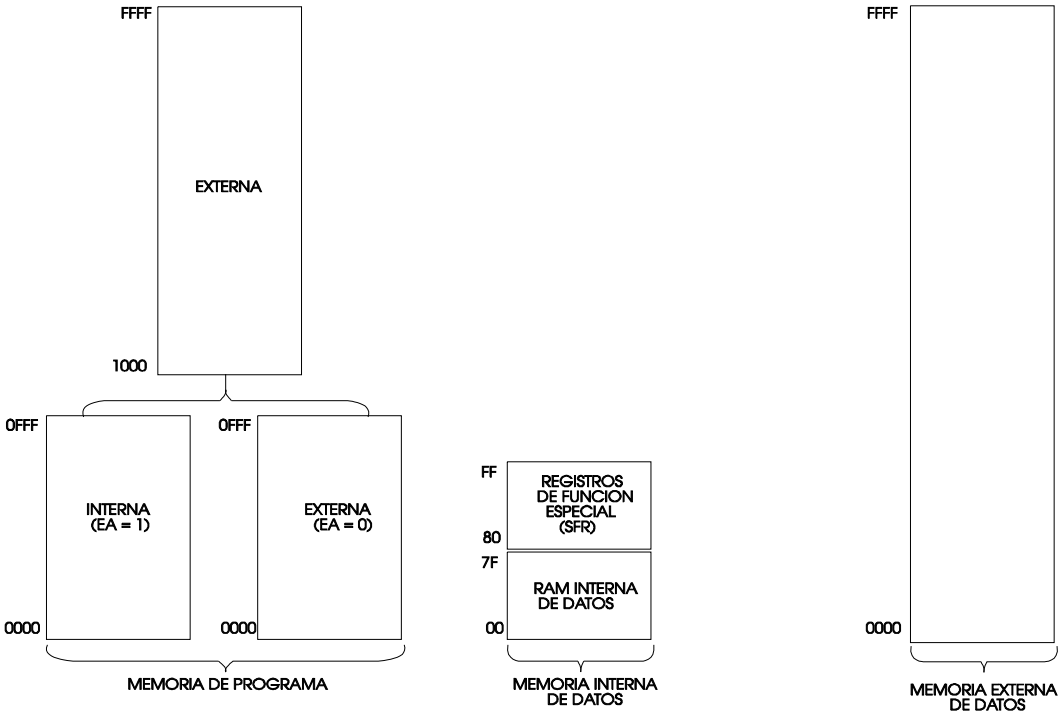


Fig. 1.4 Mapa de memoria.

Almacenamiento de programas.
Después del Reset, el CPU ejecuta la instrucción que se encuentre en la localidad 0000H; la ubicación física de esta localidad, puede ser dentro o fuera del chip, depende de la línea EA (External Access). Si EA es cero, la dirección 0000H y demás localidades serán buscadas fuera. Si EA es uno, las direcciones 0000H hasta 0FFFH se referenciarán dentro del chip.

Además, cada interrupción se asocia a una dirección fija de memoria, (ver figura 1.6); a este conjunto de localidades, se le nombra Vector de Interrupciones.

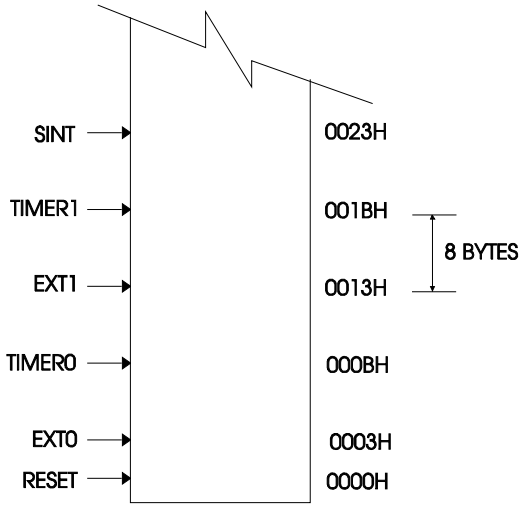


Fig. 1.5 Vector de Interrupciones.

Almacenamiento de Datos.

El almacenamiento interno del 8051, consiste en los 128 bytes bajos de la memoria y los registros de función especial (SFR). 32 bytes bajos, se dividen en cuatro bancos de 8 registros cada uno; sólo uno puede estar activo, y es seleccionado mediante dos bits del registro PSW. Los 8 registros del banco activo, se denominan R0 a R7 y son utilizados por ciertas instrucciones. Después del pulso de Reset, el SP apunta a la localidad 07.
Los 16 bytes arriba de los bancos de registros, forman un bloque que puede ser accesado ya sea por bits, o por bytes; a esta área se le llama "Area accesible por bits". Las direcciones de los bytes, son las localidades desde 20H a 2FH; las direcciones de los bits, son las localidades desde 00H a 7FH. Aunque algunas localidades tengan la misma dirección no hay confusión, ya que las instrucciones determinan si se trata de un byte o de un bit.

ROM en el chip.

El microcontrolador 8051 posee una ROM que es grabada una sola vez (OTP), el 8751, usa una EPROM, borrable con luz ultravioleta y reprogramada aproximadamente 25 veces. La serie 89C51 de Atmel puede ser grabada y borrada 1000 veces.

1.5 Registros de Función Especial (SFR)

Los registros que tienen una función especial, poseen una dirección de memoria asignada de tal manera, que pueden ser accesados por su nombre o por la ubicación en memoria.

Nombre de SFR	Valor al restaurar
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
LP	XXX00000B
LE	0X000000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	indeterminado
PCON (HMOS)	0XXXXXXXB
PCON (CHMOS)	0XXX0000B

Fig. *1.6 Contenido de los SFR´s después del reset*

Acumulador y Registro B.
Cuando se hace referencia al acumulador como una localidad de memoria, se utiliza el mnemónico ACC; cuando se referencía como un registro, se utiliza la A. Tiene la misma función que un acumulador en un microprocesador. En algunas instrucciones funciona como un Registro Indice.
El registro B, tiene una función especial en la multiplicación y división. También puede ser un registro de propósito general.

Program Status Word (PSW).
Este registro contiene las banderas que permiten controlar al circuito, véase la figura 1.7 Este registro es "accesible por bits".

(MSB) (LSB)

CY	AC	F0	RS1	RS0	OV	--	P

Símbolo	Posición	Nombre y Significado
CY	PSW.7	Bandera Carry
AC	PSW.6	Bandera Auxiliary Carry
F0	PSW.5	Bandera 0 (disponible para el usuario para propósito general)
RS1	PSW.4	Bits de control de los bancos de registros. Se enc./apag. por software
RS0	PSW.3	(ver la nota)
OV	PSW.2	Bandera Overflow
--	PSW.1	Bandera definible por el usuario
P	PSW.0	Bandera Parity. Enc./Apag. por el hardware para indicar el número de unos en el ACC, o sea paridad par

NOTA:

El contenido de(RS1, RS0)		habilita a los bancos de la sig. Manera
(0,0) --	Banco 0	(00H - 07H)
(0,1) --	Banco 1	(08H - 0FH)
(1,0) --	Banco 2	(10H - 17H)
(1.1) --	Banco 3	(18H - 1FH)

Fig. 1.7 Program Status Word (PSW)

El Apuntador de Pila (SP).
Debido a su tamaño de 8 bits, la pila (stack) sólo puede medir 256 bytes. A diferencia de otros microprocesadores, este apuntador crece "hacia arriba" con cada instrucción PUSH o CALL; después del reset se guarda un 07H, por lo que la primera posición del stack, es la localidad 08H. En algunos programas es conveniente moverlo de este lugar.

Apuntador de Datos (DPTR).
Es un registro de 16 bits, dividido en dos partes: el byte alto llamado DPH y el byte bajo denominado DPL. Se utiliza para alojar direcciones de 16 bits para ciertas instrucciones.

Latches de los Puertos (P0, P1, P2, P3).
Las 32 líneas de l/O son organizadas en 4 puertos denominados P0 a P3. Estos puertos pueden ser leídos o escritos hacia los SFR´s.

Temporizadores. (T0 y T1)
Son los registros que forman los bytes bajo y alto de los dos registros del Timer/Counter de 16 bits; la función específica depende del origen de los datos; si el conteo es interno, funcionan como un timer, y si es externo, funciona como un contador.

Registros de Control.
Existen varios registros utilizados para controlar al circuito: El Prioritizador de Interrupciones (lP), el Habilitador de Interrupciones (IE), el Modo del Timer (TMOD), el Controlador del Timer (TCON), el Controlador del Puerto Serie (SCON) y el Controlador de Potencia (PCON).

LOS PUERTOS DE l/O

Una de las más útiles características del 8051, son sus 32 líneas de Entrada/Salida, se organizan en cuatro puertos de 8 líneas. Los puertos pueden ser usados como líneas generales de I/O o como líneas de direcciones y datos para memoria

Entrada, Carga y Manejador de Salida.
Los puertos P1, P2 y P3 tienen resistencia de "pull-up". El puerto P0 no la tiene, hay que ponerla externamente (10KOhms). Los puertos P1, P2 y P3 pueden manejar el equivalente de cuatro entradas TTL LS. El puerto 0 maneja ocho de estas mismas cargas.

Funciones Alternas del Puerto P3.
Las líneas del puerto 3 poseen función alterna, según se lista en la figura 1.8. Para habilitar la función alterna, se debe escribir un 1 en el bit correspondiente.

Línea del puerto	Función Alterna
P3.0	RXD (entrada del puerto serie)
P3.1	TXD (salida del puerto serie)
P3.2	lNT0 (interrupción externa 0)
P3.3	lNT1 (interrupción externa 1)
P3.4	T0 (entrada externa del Timer 0)
P3.5	T1 (entrada externa del Timer 1)
P3.6	WR (estrobo de escritura para datos de memoria externa)
P3.7	RD (estrobo de lectura para datos de memoria externa)

Fig. 1.8 Funciones Alternas del Puerto 3

Acceso de memoria externa.
Debido a que el 8051 tiene separadas las áreas de memoria de programa y memoria de datos, utiliza diferentes señales para los dispositivos externos de memoria. La línea PSEN (program storage enable) se utiliza como estrobo para la memoria de programa, y las líneas RD y WR se utilizan para leer y escribir en la memoria de datos. Observe que estas últimas

son funciones alternas del puerto 3. Los puertos 0 y 2 se utilizan como buses de datos y de direcciones.

TEMPORIZADOR/CONTADOR

El 8051 tiene dos registros de 16 bits, que pueden ser utilizados como temporizadores o contadores. Se les denomina Timer0 y Timer1. Son dos pares de registros de 8 bits.

Cuando se utiliza como temporizador, el registro es incrementado una vez por cada ciclo de máquina, lo cual ocurre una vez cada 12 periodos del reloj. Cuando se utiliza como contador, el registro es incrementado en la transición alto-bajo (borde negativo) del pulso que se aplique en la entrada correspondiente. Al 8051 le toma dos ciclos de máquina ver la transición alto-bajo; por ésta razón, la señal de entrada debe permanecer un ciclo de máquina en "alto" y un ciclo de máquina en "bajo".

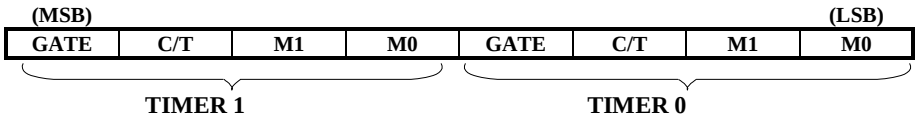
Timer 0 y Timer 1.

El modo en el que operan el Timer 0 y el Timer 1, es determinado por los 8 bits escritos en el registro TMOD; el cual se detalla en la fig. 1.9. Los bits M0 y M1 (un par para cada timer) se usan para seleccionar uno de los cuatro posibles modos de operación: Modo 0, Modo1, Modo 2 y Modo 3. Ambos temporizadores funcionan igual en los modos 0, 1 y 2, pero no en el Modo 3.

Modo 0 y Modo 1.

En modo 0, el timer se configura como un contador de 13 bits; el cual puede ser visto como un contador de 8 bits y un preescalador de 5 bits. Cuando se desborda este registro, se activa la bandera de interrupción (TF0 o TF1). Esta bandera es la conexión entre el hardware del contador y el software.

En la figura 2.11, se observa la lógica de conexión de temporizador/contador. Para que se inicie un conteo, el bit TR respectivo (TR0 o TR1) debe estar encendido; este bit es parte del registro TCON (ver figura 1.10); además debe ser verdadera una de dos condiciones: el bit GATE apropiado es cero, o la línea INT apropiada se mantiene "baja".



GATE	Cuando está encendido, es el control del "gatilleo".
C/T	Selector del Timer o de Counter; se pone en cero para operaciones con el timer, y en uno para el contador.

M1	M0	Modos de Operación
0	0	Timer del 8048, "TLx" sirve de pre-escalador
0	1	Timer/counter de 16 bits, THx y TLx trabajan en cascada
1	0	Timer/counter de 8 bits con "auto-recarga"
1	1	Timer/Counter doble de 8 bits, alojado en TL0 y TH0. Timer 1 detenido.

Fig. 1.9 Registro de control del Modo del Timer (TMOD)

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Simbolo	Posición	Nombre y Significado	Símbolo	Posición	Nombre y Significado
TF1	TCON.7	Bit de desbordamiento del Timer 1. Se limpia con el hardware, cuando se atiende la interrupción.	IE1	TCON.3	Bit del borde (edge) de la interrupción 1. Activado por hardware cuando se detecta el borde de la interrupción externa. Se limpia al atender la interrupción.
TR1	TCON.6	Bit de control de arranque (Timer Run) del timer 1. Se activa por software para arrancar/parar.	IT1	TCON.2	Bit de control del tipo de interrución de Timer 1. Indicar un borde que sube o que baja.
TF0	TCON.5	Bit de desbordamiento del Timer 0.	IE0	TCON.1	Bit del borde (edge) de la interrupción 0.
TR0	TCON.4	Bit de control de arranque (Timer Run) del timer 0.	IT0	TCON.0	Bit de control del tipo de interrupción del Timer 0.

Fig. 1.10 Registro de control del Timer/Counter (TCON)

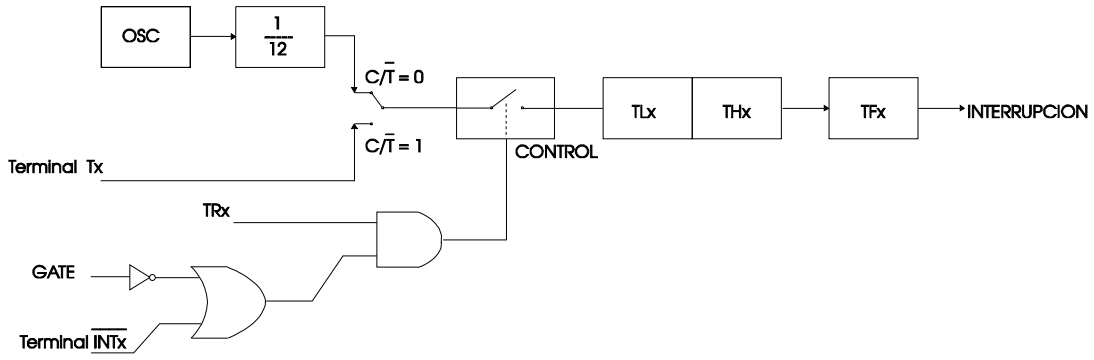


Fig. 1.11 Timer/Counter 0 en Modo 1 (16 bits).

Modo 2.

TL es el contador de 8 bits y TH es el valor de respaldo, cuando se llena el registro TL, se activa la bandera TF, y el contenido de TH se transfiere a TL; TH permanece sin alterar. Este modo se conoce como "recarga".

Modo 3

En este modo, el Timer 1 se deshabilita, pero retiene su cuenta, se "congela". El timer 0 es dividido en dos contadores, el primer contador queda similar al Modo 0, pero sin pre-escalador, y el segundo contador es habilitado con el bit TR1. Cuando se desbordan, el primer contador enciende la bandera TF0 y el segundo levanta la bandera TF1.

II. SOFTWARE DEL 8051

2.1 Modos de Direccionamiento

A pesar de la gran cantidad de componentes del modelo de programación, es muy sencillo programarlo. Posee cinco modos de direccionamiento y cinco grupos de instrucciones.

Direccionamiento Directo

Las instrucciones que utilizan direccionamiento directo, emplean 2 bytes de longitud, uno para el código de operación (op-code) y otro para la dirección. La dirección de la localidad puede estar en la RAM interna o en los SFR´s; el operando puede ser una localidad de memoria o un bit específico del área llamada "accesible por bits".
Ejemplo de Instrucciones Directas:

```
MOV  A,07        ;Mueve el contenido del byte 07 de RAM al acumulador.
CLR 07           ;Limpia el bit 07.
```

Direccionamiento lndirecto.

En este direccionamiento, las instrucciones especifican el registro que contiene el operando efectivo. Es muy útil para la manipulación de datos.

Es posible accesar de esta manera la RAM interna y externa; las direcciones pueden ser de 8 o 16 bits, dependiendo del registro empleado; las direcciones de 16 bits emplean al registro Data-Pointer (DPTR).
Ejemplo de Instrucciones Indirectas:

```
MOV A,@R3        ;Mueve al acumulador el byte de RAM interna apuntado por
                 ;apuntado por el registro R3.
MOVX A,@DPTR     ;Mueve al acumulador el byte de RAM externa apuntado por
                 ;el registro DPTR.
```

Direccionamiento a Registros.

En este modo, las instrucciones utilizan el contenido de un registro, típicamente R0-R7, como el operando. Todas estas instrucciones son eficientes por ser de un byte de longitud. Unas pocas instrucciones son específicas a un registro, las cuales no permiten al usuario definir el registro.
Ejemplo de Instrucciones de Registros:

```
MOV A,R3     ;Mueve el contenido del registro R3 al acumulador.
MUL AB       ;Multiplica el contenido del acumulador por el contenido de B.
```

Direccionamiento Inmediato.

El operador es un valor numérico, el cual va después del código de la instrucción. La constante numérica puede ser de 8 o 16 bits de longitud.

Ejemplo de Instrucciones Inmediatas:

```
MOV A;#NUM8          ;Mueve el número de 8 bits NUM8 al acumulador.
MOV DPTR,#NUM16      ;Mueve el número de 16 bits NUM16 al DPTR.
```

Direccionamiento Indexado.
El 8051 tiene dos usos para el direccionamiento indexado: leer tablas de la memoria de programa e implementar tablas de salto. En ambos casos, un registro de 16 bits contiene la dirección base, y el acumulador contiene un desplazamiento o índice de 8 bits. La dirección efectiva del operando o del salto, se obtiene sumando estas dos cantidades. El registro de 16 bits puede ser el DPTR o el Contador de Programa (PC).
Ejemplo de Instrucciones Indexadas:

```
MOVC A,@A+PC     ;Mueve el byte de código relativo al PC al acumulador.
JMP @A+DPTR;     ;Salta relativo al DPTR.
```

GRUPOS DE INSTRUCCIONES

El 8051 posee 111 tipos de instrucciones: 49 de un byte, 45 de dos bytes y 17 de 3 bytes. Contando las variaciones de cada tipo, existen 255 instrucciones distintas; cada valor de 00H a FFH es una instrucción válida, excepto A5, que está reservada para uso futuro. Estas instrucciones caen en uno de cinco grupos: Aritméticas, Lógicas, Transferencia de Datos, Booleanas y Saltos.

Operaciones Aritméticas.
El 8051 tiene instrucciones usuales de un procesador de 8 bits: suma (ADD), suma con acarreo (ADDC), resta con acarreo (SUBB), incremento (INC), decremento (DEC) y ajuste decimal al acumulador (DA). Posee además dos instrucciones que no son típicas en micros de 8 bits: multiplicación (MUL AB) y división (DlV AB).

Operaciones Lógicas.
Las operaciones lógicas incluyen: AND (ANL), OR (ORL), OR-EXCLUSIVA (XRL), borrar y complementar (CLR y CPL) y rotaciones (RL, RLC, RR, RRC). Incluye además la instrucción para intercambio SWAP A, la cual intercambia los nibbles del acumulador.

Transferencia de Datos.
La instrucción básica para transferir datos es "move", la cual tiene tres formas: MOV, MOVC y MOVX. También se incluyen PUSH, POP y XCH (exchange). MOVC indica "move code" para transferir datos de la ROM, y MOVX indica "move external" para transferir datos desde/hacia RAM externa.

Operaciones Booleanas
Este grupo de instrucciones se asocia con la capacidad del procesador Booleano de un bit implementado en hardware en el 8051; el grupo incluye: SET, CLEAR, AND OR y complementos. Además se dispone de saltos condicionales que prueban bits individuales.

Esta capacidad lo hace muy adecuado para monitorear condiciones externas, sin necesidad de leer todo el puerto.

Saltos.
Además de llamadas (CALL) y retornos (RET's) de subrutinas, se incluyen algunos saltos condicionales e incondicionales. Debido a que el salto es con signo, es posible saltar hacia adelante o hacia atrás.

La instrucción JUMP tiene tres formas básicas: saltos cortos (SJMP), saltos largos (LJMP) y saltos absolutos (AJMP).

Una instrucción importante en este grupo es CJNE, la cual llena el vacío de instrucciones de comparación que había en el grupo de instrucciones lógicas.

2.2 Juego de Instrucciones

Ciclos de máquina ——————¡
Número de bytes ——————| |
Código de operación ——————¡ | |

Instrucciones de movimiento de datos

MOV dir, A	F5 2 1			MOV Rn, A	F8 - FF 1 1
MOV dir, Rn	88 – 8F 2 2	MOV A, Rn	E8 - EF 1 1		
MOV dir, dir	85 3 2	MOV A, dir	E5 2 1	MOV Rn,. dir	A8 – AF 2 2
MOV dir, dir	86 , 87 3 2	MOV A, @Ri	E6 , E7 1 1		
MOV dir, @Ri	75 3 2	MOV A, #dat	74 2 1	MOV Rn, #dat	78 – 7F 2 1
MOVC A,@A + DPTR	93 1 2	MOV X, @Ri	E2, E3 1 2	MOVX @Ri, A	F2, F3 1 2
MOVC A,@A + PC	83 1 2	MOVX A,@DPTR	E0 1 2	MOVX @DPTR, A	F0 1 2
XCHG A, Rn	C8 – CF 1 1	MOV @Ri, A	F6, F7 1 1	PUSH dir	C0 2 2
XCHG A, dir	C5 2 1	MOV @Ri, dir	A6, A7 2 2	POP dir	D0 2 2
XCHG A, @Ri	C6 – C7 1 1	MOV @Ri, #dat	76 77 2 1		
XCHD A, @Ri	D6, D7 1 1	MOV DPTR,#dat	16 90 3 2		

Instrucciones Aritméticas

ADD A, Rn	28 – 2F 1 1	ADDC A, Rn	38 –3F 1 1	SUBB A, Rn	98 –9F 1 1
ADD A, dir	25 2 1	ADDC A, dir	35 2 1	SUBB A, dir	95 2 1
ADD A, @Ri	26, 27 1 1	ADDC A, @Ri	36, 37 1 1	SUBB A, @Ri	96, 97 1 1
ADD A, #dat	24 2 1	ADDC A, #dat	34 2 1	SUBB A, #dat	94 2 1
INC A	04 1 1	DEC A	14 1 1	DA A	D4 1 1
INC Rn	08 – 0F 1 1	DEC Rn	18 – 1F 1 1	MUL AB	A4 1 4
INC dir	05 02 1	DEC dir	15 2 1	DIV AB	84 1 4
INC @Ri	06, 07 1 1	DEC @Ri	16, 17 1 1		
INC DPTR	A3 1 2				

Instrucciones Lógicas

ANL A, Rn	58 – 5F 1 1	ORL A, Rn	48 – 4F 1 1	XRL A, Rn	68 – 6F 1 1
ANL A, dir	55 2 1	ORL A, dir	45 2 1	XRL A, dir	65 2 1
ANL A, @Ri	56 – 57 1 1	ORL A, @Ri	46 – 47 1 1	XRL A, @Ri	66 – 67 1 1
ANL A, #dat	54 2 1	ORL A, #dat	44 2 1	XRL A, #dat	64 2 1
ANL dir, #dat	53 3 1	ORL dir, #dat	43 3 1	XRL dir, #dat	63 3 1
ANL dir, A	52 2 1	ORL dir, A	42 2 1	XRL dir, A	62 2 1
CLR A	E4 1 1	RL A	23 1 1		
CPL A	F4 1 1	RLC A	33 1 1		
SWAP A	C4 1 1	RR A	03 1 1		
		RRC A	13 1 1		

Instrucciones de Manipulación de bit

CLR C	C3 1 1	SETB C	D3 1 1	CPL C	B3 1 1
CLR bit	C2 2 1	SETB bit	D2 2 1	CPL bit	B2 1 2
MOV C, bit	A2 2 1	ANL C, bit	82 2 2	ORL C, bit	72 2 2
MOV bit, C	92 2 2	ANL C, / bit	B0 2 2	ORL C , / bit	A0 2 2

Instrucciones de Control de Programa

LJMP addr16	02 3 2	LCALL addr16	12 3 2	JMP @A + DPTR	73 1 2
AJMP addr11	2 2	ACALL addr11	2 2		
SJMP rel	80 2 2			JBC bit, rel	10 3 2
JZ rel	60 2 2	JC rel	40 2 2	JB bit, rel	20 3 2
JNZ rel	70 2 2	JNC rel	50 2 2	JNB bit, rel	30 3 2
CJNE A, dir, rel	B5 3 2	DJNZ Rn, rel	D8 – DF 2 2	RET	22 1 2
CJNE A, #dat, rel	B4 3 2	DJNZ dir, rel	D5 3 2	RETI	32 1 2
CJNE Rn, #dat, rel	B8 – BF 3 2			NOP	00 1 1
CJNE @ Ri, #dat, rel	B6 , B7 3 2				

2.3 Programas en ensamblador

Ejemplo 1.
Péndulo electrónico.
Escriba un programa que encienda luces en el puerto P0, simulando un pendulo, esto es:

1000 0000
0100 0000
0010 0000
• •
• •
0000 0010
0000 0001
0000 0010
• •

Solución a)

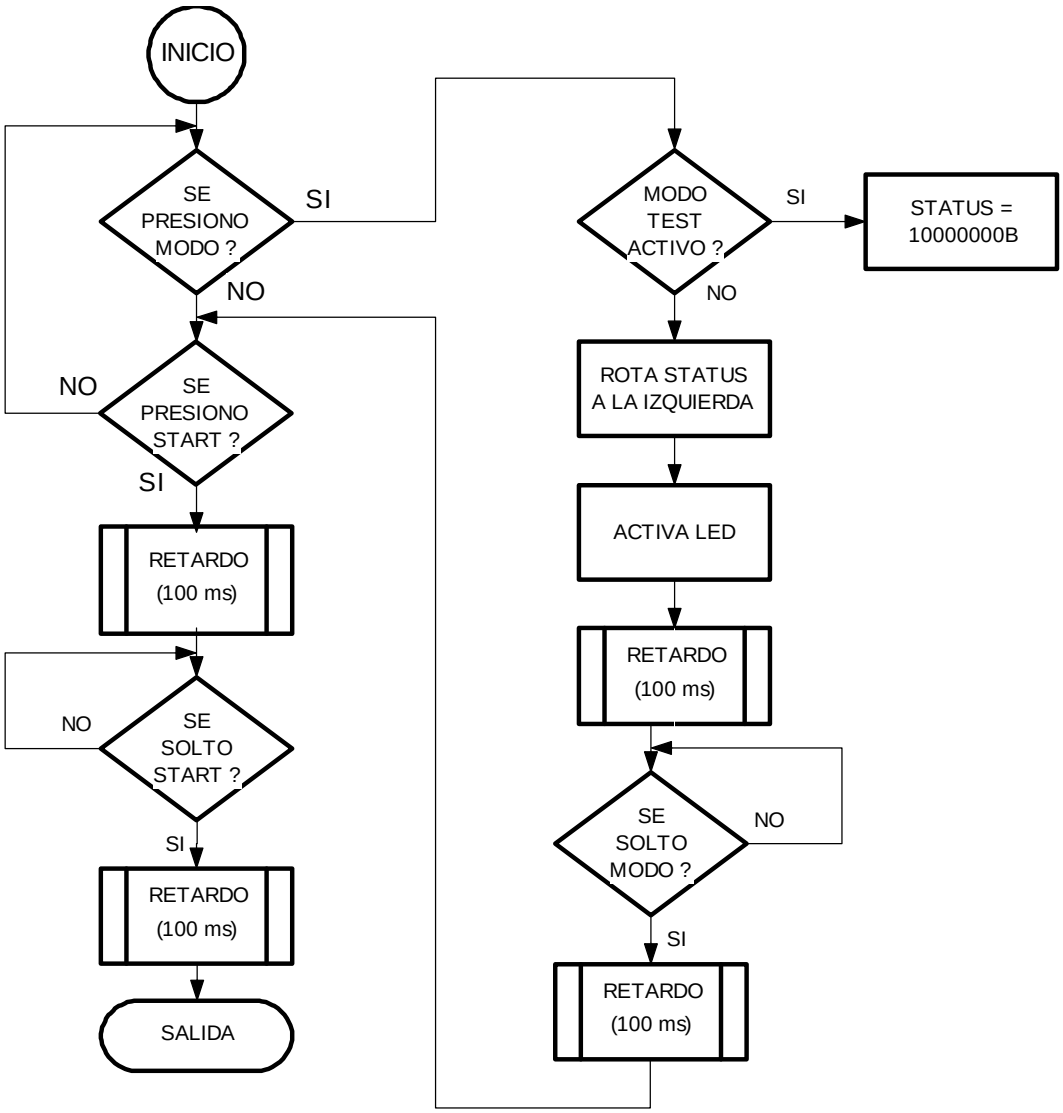
```
INICIO:   MOV P1,#10000000B
          MOV P1,#01000000B
          MOV P1,#00100000B
          MOV P1,#00010000B
          MOV P1,#00001000B
          MOV P1,#00000100B
          MOV P1,#00000010B
          MOV P1,#00000001B
          MOV P1,#00000010B
          MOV P1,#00000100B
          MOV P1,#00001000B
          MOV P1,#00010000B
          MOV P1,#00100000B
          MOV P1,#01000000B
          SJMP INICIO
```

Solución b)

```
INICIO    MOV A,#1000000B
          MOV R0,#7
OTRO:     MOV P1,A
          RR A
          DJNZ R0,OTRO
          MOV R0,#7
SIGUE:    MOV P1,A
          RL A
          DJNZ R0,SIGUE
          SJMP OTRO
```

Ejemplo 2.
Control de un panel electrónico.
Escriba un programa que lea dos botones de un panel de un cargador de baterías que tiene cuatro modos de operación: Carga_Rápida, Carga_Lenta, Rejuvencedor y Prueba. Un botón selecciona el Modo, el otro es de Arranque-Paro. Cada modo se indica con cuatro LEDs.

Solución. Diagrama de Flujo



Cada que se pulsa una tecla se debe efectuar el siguiente procedimiento:

i) Realizar inmediatamente la acción que la tecla tenga destinada.
ii) Encender un zumbador.
iii) Llamar a un retardo de 100 ms para evitar rebotes de la tecla.
iv) Apagar el zumbador
v) Preguntar si ya se soltó la tecla, y si no es así, esperar a que se libere.
vi) Cuando la tecla se liberó llamar al retardo para evitar rebotes al soltar.

```
;Programa LEE_TECLAS
;Función: Monitorea las teclas Modo y Arranque/Paro.
;       Salidas: localidad STATUS actualizada con el modo seleccionado
;       Llamadas: RETARDO
;       Destruye: R0, R1, R2,
LEE_TECLAS:
        JNB MODO,CAMB_MODO ;Se presiono modo?. Si afirmativo cambia de modo LISTO:
LISTO:                          JB      START , LEE_TECLAS      ;Negativo. Se presiono Arranque/pa
        SETB  ZUMBADOR          ;Si. Enciende tono
        CALL  RETARDO           ; Retardo de 100 ms  para evitar rebotes de la tecla
        CLR    ZUMBADOR         ;Apaga tono
        JNB    START,$          ;Espera a que se libere la tecla
        CALL  RETARDO           ; Retardo para evitar rebotes al soltar
        SJMP SALIR

CAMB_MODO:
        JNB    07BH,OTRO_MODO ;Esta Modo TEST activo? (7B es el bit 3 del STATUS) ;
        MOV  STATUS,#00001000B          ;Si. Prepara para siguiente rotación OTRO_MODO:  ;
        MOV  A,STATUS           ;Trae STATUS
        RL  A                   ;para rotarlo
        MOV  STATUS,A           ;Salva ya rotado
        MOV  REL_LED,STATUS;Copia STATUS a los LEDs
        SETB  ZUMBADOR          ;Activa tono
        CALL RETARDO            ;Evita rebotes
        CLR    ZUMBADOR         ;Apaga tono
        JNB    MODO,$           ;Espera a que liberen la tecla
        CALL RETARDO            ;Evita rebotes

        SJMP LISTO              ;Pregunta por otra

        END
```

III MANEJO DE SIMULADORES

3.1 Uso del macroensamblador AVMAC

Sintaxis de los programas.
Vea el siguiente ejemplo:

```
 DEFSEG  BIEN,CLASS=CODE,START=00H
;-------------------------------------------------------------------------
; UNO.ASM
; PRUEBA EL MICROCONTROLADOR 8051,
; ENVIA SECUENCIA ASCENDENTE A LOS 4 PUERTOS
; APROXIMADAMENTE CADA SEGUNDO.
; (CON CRISTAL DE 8 MHZ CADA 0.999999 SEGUNDOS)
;-------------------------------------------------------------------------
;
 SEG BIEN
;
 ORG  00H
START:    INC A              ;Incrementa el acumulador
          MOV P0,A           ;Lo escribe en
          MOV P1,A           ; cada uno de los
          MOV P2,A           ; cuatro puertos
          MOV P3,A           ; puertos
          MOV R1,#06         ;Inicializa para tres
FUERA:    MOV R2,#0d8H       ; lazos anidados
MEDIO:    MOV R3,#0FFH       ; los cuales se usan como
DENTRO:   DJNZ R3,DENTRO     ; un control burdo
          DJNZ R2,MEDIO      ; que tarda 0.99xxxx segundos
          DJNZ R1,FUERA
          MOV R1,#04         ;Inicializa para dos
ACA:      MOV R2,#0DEH       ; lazos anidados
          DJNZ R2,$          ;Aquí se agrega el tiempo
          DJNZ R1,ACA        ; que falta para hacer un segundo
          NOP
          NOP
          NOP
          SJMP START
          RET

          END
```

Para Ensamblar y ligar ejecute los siguientes comandos:

avmac51 <archivo>
avlink <archivo>.hex=<archivo>.obj
del <archivo>.mxp
del <archivo>.obj
del <archivo>.map
del <archivo>.prn

Si existen errores de sintaxis corrijalos hasta lograr cero errores.

3.2 Uso del simulador AVSIM51

Para probar los programas antes de ingresarlos a la memoria del microcontrolador, verifique el funcionamiento. Teclee:

AVSIM51 –C0

El comando C0 permite ver la simulación en colores, (ver comando SETCOLOR)

Aparece una pantalla como la siguiente:

AVSIM 8051 Simulator/Debugger

Serial # 00479 Licensed by Avocet Systems, Inc.

Copyright (C) 1984,1985,1986,1987 by Ken Anderson Software
All Rights Reserved

Intel 8051 Family Microcomputers

HMOS ROM	HMOS ROMless	CMOS ROM
A: 8051/8751	C: 8031	E: 80C51
B: 8052/8752	D: 8032	F: 80C31

Choose a CPU for simulation:

Elija la opción A, enseguida aparece la pantalla principal del simulador:

```
LABEL   OPERATION      8051/8751 AVSIM 8051 Simulator/Debugger      V1.32

RESET   no  memory     CPU REGISTERS      FLAGS   SCL  SPD  DSP  SKP  CURSOR
0001H   no  memory     C  Accumulator    AC F0 OV P   OFF HI    ON   OFF  MENU
0002H   no  memory     0  00000000:00:_   0   0  0  0             Cycles:
0003H   no  memory        addr    data
0004H   no  memory      PC:0000 » FF FF FF FF  Timers TH/TL  TF/TR  G/T/M1/M0
0005H   no  memory      SP:  07 »   00 00 00 00      T0:   00 00    0  0    0 0    0    0
0006H   no  memory                  00 00 00 00      T1:   00 00    0  0    0 0    0    0
0007H   no  memory      DP:0000 » FF FF FF FF
0008H   no  memory      R0:00:_  » 00:_  RB:00  Ints A S T1 X1 T0 X0 Edg IT IE
0009H   no  memory      R1:00:_  » 00:_   B:00  En   0 0 0    0    0  0  X0:   0  0
000AH   no  memory      R2:00   R4:00    R6:00  Pr       0 0    0    0  0  X1:   0  0
000BH   no  memory      R3:00   R5:00    R7:00 SBUF:  In   Out  PCON:0xxxxxxx
000CH   no  memory      Data Space                    00:_ 00:_ SCON:00000000
000DH   no  memory      0000   00 00 00 00 00 00 00 00 _________  Ports
000EH   no  memory      0008   00 00 00 00 00 00 00 00 _________  P0    11111111
000FH   no  memory      0010   00 00 00 00 00 00 00 00 _________  FF:_: 11111111
0010H   no  memory      0018   00 00 00 00 00 00 00 00 _________  P1    11111111
0011H   no  memory      Data Space                               FF:_: 11111111
0012H   no  memory      0020   00 00 00 00 00 00 00 00 _________  P2    11111111
0013H   no  memory      0028   00 00 00 00 00 00 00 00 _________  FF:_: 11111111
0014H   no  memory      0030   00 00 00 00 00 00 00 00 _________  P3    11111111
0015H   no  memory      0038   00 00 00 00 00 00 00 00 _________  FF:_: 11111111
>Select Command - or use arrow keys
Dump   Expression   commandFile   Help   IO   Load        --space--  ESC to screen
```

Cargue el programa con la siguiente secuencia de comandos: (Nota solo digite la primer letra):

Load
Program
UNO.HEX (o como se llame, siempre con la extensión HEX)

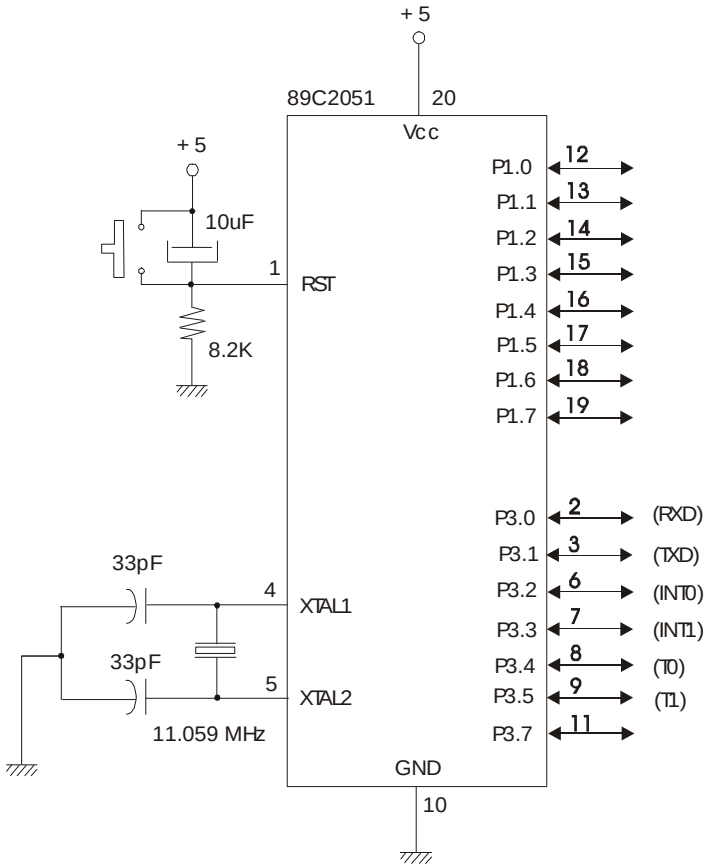
El programa aparecerá en el lado izquierdo de la pantalla, aparecen los contenidos de memoria y los mnemonicos correspondientes. Los comandos principales son:

F1 Corre el programa completo
F5 Cambia la velocidad de simulación
F9 Deshace los cambios
F10 Corre paso a paso.

La tecla ESC cambia del Menú al área del edición de los registros del microcontrolador. En caso de problemas, para salir del algún sub-menu teclear CTRL –C.
El simulador permite interactuar con los contenidos de los registros, lo cual es muy conveniente, es posible incluso, simular entradas, tecleando los valores adecuados en los puertos de I/O

IV. ARMADO DE UN SISTEMA MINIMNO

4.1 Diagrama eléctrico



4.3 Armado de un contador

Ahora es posible armar una sencilla aplicación, un contador de eventos con salida a un exhibidor multiplexado.

V. COMUNICACIÓN SERIE

5.1 Registros de control del puerto serie del 8051

El registro SBUF es el buffer tanto de entrada como de salida. Realmente son dos registros con la misma dirección. Permite comunicación Full-Duplex.

(MSB)							(LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Simbolo	**Nombre y Significado**
SM0 – SM1	Determinan el modo de operación del puerto serie

Modo	M1	M0	Descripción/Velocidad
0	0	0	Shift Register / reloj ÷12
1	0	1	UART de 8 bits / variable
2	1	0	UART de 9 bits / reloj ÷ 64 o ÷ 32
3	1	1	UART de 9 bits /variable

SM2	Habilita el modo muiltiprocesamiento en 8052
REN	Cuando = "1" permite la recepción de caracteres
TB8	Es el noveno bit a transmitir (bit de paridad en Modo 2 y 3)
RB8	En modos 2 y 3 es el noveno bit En modo 1 es el bit de stop En modo 0 no se utiliza
TI	Bandera de Interrupción de Transmisión. Debe borrarse por programa
RI	Bandera de Interrupción de Recepción. Debe borrarse por programa

Fig. 5.1 Registro de control del Puerto Serie (SCON)

Velocidad en baudios	Frecuencia de reloj en MHz	SMOD	Timer 1		
			C / T	Modo	Valor
Modo 0 max 1 MHz	12	x	x	x	x
Modo 2 max 375 KHz	12	1	x	x	x
Modos 1, 3 max. 62.5KHz	11.059	1	0	2	FFh
19,200	11.059	1	0	2	FDh
9,600	11.059	0	0	2	FDh
4,800	11.059	0	0	2	FAh
2,400	11.059	0	0	2	F4h
1,200	11.059	0	0	2	E8h
137,500	11.059	0	0	2	1Dh
110,000	6	0	0	2	72h
110,000	12	0	0	2	FEEBh

Fig 5.2 Velocidad del Puerto Serie

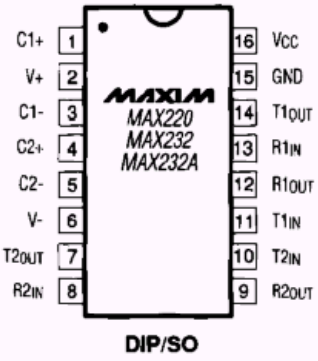
5.2 Comunicación con una PC via RS232
Ejemplo 5.1

```
 DEFSEG  BIEN,CLASS=CODE,START=00H
;*************************************************************************
;ARCHIVO:   ENVIA2.ASM.
;
;DESCRIPCION: Prueba del Puerto Serie de la computadora. Envía  una secuencia
;                    ascendente de datos en ASCII. La velocidad del UART es de 9600 BPS.
;                   La  visualización de  la secuencia va desde 00 hasta 99.
;*************************************************************************
 SEG BIEN
 ORG 00H
              ACALL INIPORT    ;Inicialización del UART.
CICLO:        ACALL ENVIO      ;Transmisión del dato numérico ascendente.
              SJMP CICLO       ;Se cicla el programa.
;*************************************************************************
;Subrutina de inicialización del UART (9600 bps, long. de 8 bits sin paridad)
;Definición de los valores para fijar la velocidad del UART:
;Velocidad:   / Valor del Timer1(TH1):
;1200 baudios / 0E8H
;2400 baudios / 0F4H
;4800 baudios / 0FAH
;9600 baudios / 0FDH

INIPORT:
              MOV SCON,#50h    ;UART modo 1 (8 bits, sin paridad)
              MOV TMOD,#21h    ;Timer1 en modo 2
              MOV TH1,#0FDh    ;9600 baudios (F3 en XTAL=12 MHz.)
              MOV TL1,#0FDh    ;Autorecarga en el TIMER 1
              SETB TR1         ;Pone en marcha el TIMER 1
              RET              ;Fin subrutina de inicialización
;*************************************************************************
;Subrutina de transmisión de secuencia ascendente:
;El registro B se usa como contador
ENVIO:
              MOV A,B          ;Carga el contador con el valor anterior
              ADD A,#1         ;Incrementa el contador
              DA A             ;Ajuste a decimal del ACC para suma en BCD
              MOV B , A        ;Guarda el valor del contador
              ANL A,#0F0H      ;Enmascara
              CLR C            ; el
              RRC A            ; dato
              RRC A            ; a
              RRC A            ; enviar
              RRC A            ; por el puerto
              ADD A,#30H       ;Suma para conversión a ASCII
```

```
MOV SBUF,A      ;Pone el dato en puerto serie
JNB TI , $      ;Espera a que se envíe el dato
CLR TI          ;Desactiva la interrupción del puerto
CLR C           ;Enmascara el dato
MOV A,B         ; a enviar
ANL A,#0Fh      ; por el puerto
ADD A,#30h      ;Suma para conversión a ASCII
MOV SBUF , A    ;Pone el dato en puerto serie
JNB TI , $      ;Espera a que se envíe el dato
CLR TI          ;Desactiva la interrupción del puerto
MOV A , #32     ;Carga el contador con espacio en blanco
MOV SBUF , A    ;Pone el espacio en blanco en puerto serie
JNB TI , $      ;Espera a que se envíe el dato
CLR TI          ;Desactiva la interrupción del puerto
RET             ;Fin de subrutina de transmisión

END             ;Fin del programa
```

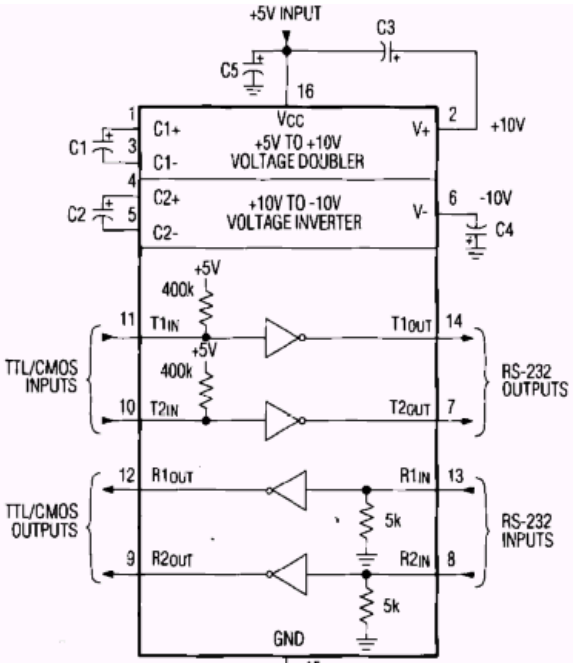
5.3 Uso del manejador MAX232

Para convertir los niveles de TTL a RS232 se usa el MAX232, que permite con una sola fuente de +5 Volts generar +12 y –12 Volts.



DIP/SO

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



Ejemplo 5.2

```
DEFSEG YES,CLASS=CODE,START=00H
;*****************************************************************************
;ARCHIVO: SUMAOK.ASM
;DESCRIPCION: Recibe dos datos por el puerto serie y los suma. Cada uno de los datos es
;            devuelto por la salida serial para verificar que se reciben correctamente,
;            enviando enseguida un caracter espacio en blanco (ASCII 20H). Realiza la
;            suma hasta que recibe un Retorno de Carro (ASCII 0DH). Enseguida una
;            bandera (CRFLAG) es activada y se realiza la suma de los dos  datos; la
;            suma se envía por la salida serial seguida un espacio en blanco (ASCII 20H).
;*****************************************************************************
;Definiciones de variables
CRFLAG EQU 20H.1            ;Bandera de fin de línea.
;*****************************************************************************
 SEG YES
 ORG 00H
START:
START:    CLR RI            ;Limpia
          CLR TI            ; todas las
          CLR ES            ; interrupciones
          CLR CRFLAG        ; y banderas.
          MOV R0,#21H       ;Sitúa el puntero en el buffer de recepción.
          MOV R1,#22H       ;R1 es un apuntador aux. cuando de reciben mas de 2
                            ; datos.
;La sig parte inicializa la UART a 9600 BPS, 8 bits sin paridad
          MOV SCON,#50H     ;Configura en Modo1 la UART(8 bits, sin paridad).
          MOV TMOD,#21H     ;Fija el TIMER 1 en modo 2 (Auto-Recarga).
          MOV TH1,#0FDH     ;Fija una vel.. de 9600 baudios
          MOV TL1,#0FDH     ;
          SETB ES           ;Habilita la interrupción serial.
          SETB EA           ;Habilita todas las interrupciones.
          SETB TR1          ;Pone en marcha el TIMER 1.

          SJMP SIGUE        ;Salta área de interrupción
;*****************************************************************************
;Vector de interrupción del Puerto Serie.
 ORG 23H
          PUSH ACC          ;Salva el ACC
          PUSH PSW          ; y el estado de programa.
          MOV A,SBUF        ;Obtiene el byte de entrada serial.
          CJNE A,#0DH,NOTCR      ;Checa si el dato es Retorno de Carro.
          SETB CRFLAG       ;Activa la bandera de fin de línea.
          CLR RI            ;Limpia la entrada de recepción.
          SJMP RESTAURA     ;Sale
NOTCR:
          ACALL TRANS
```

```
                ANL A,#0FH          ;Obtiene valor numérico del dato ASCII.
                MOV @R0,A           ;Mueve byte (dato) al  rea de almacenamiento.
                CJNE R0,#21H,IFR0E22   ;Checa si hay almacenado dos números a sumar.
                INC R0              ;Apunta al sig. byte de almacenamiento.
                SJMP RESTAURA       ;Sale
IFR0E22:
                CJNE R0,#22H,R0EQU23    ;Checa si hay almacenado m s de dos números.
                INC R0              ;Apunta al sig. byte de almacenamiento.
                SJMP RESTAURA       ;Sale
R0EQU23:
                MOV A,@R1           ;Mueve al ACC lo que hay en la localidad 22H.
                DEC R1              ;Apunta R1 a la localidad 21H.
                MOV @R1,A           ;Mueve el dato de la loc. 22H a la loc. 21H.
                INC R1              ;Apunta R1 a la localidad 22H.
                MOV A,@R0           ;Mueve al ACC lo que hay en la localidad 23H.
                MOV @R1,A           ;Mueve el dato de la loc. 23H a la loc. 22H.
RESTAURA:
                POP PSW             ;Restaura PSW
                POP ACC             ; y el ACC.
                RETI                ;Sale de la interrupción
;---------------------------------------------------------------------
SIGUE:
;Envia el mensaje "HOLA".
                MOV A,#48H          ;Pone el caracter "H" en el ACC.
                ACALL ENVIA         ;Envía el caracter por la salida serial.
                MOV A,#4FH          ;Pone el caracter "O" en el ACC.
                ACALL ENVIA         ;Envía el caracter por la salida serial.
                MOV A,#4CH          ;Pone el caracter "L" en el ACC.
                ACALL ENVIA         ;Envía el caracter por la salida serial.
                MOV A,#41H          ;Pone el caracter "A" en el ACC.
                ACALL ENVIA         ;Envía el caracter por la salida serial.
                MOV A,#20H          ;Pone el caracter " " en el ACC.
                ACALL ENVIA         ;Envia el caracter por la salida serial.
;---------------------------------------------------------------------
CICLO:
                JNB CRFLAG , $      ;Bucle para esperar la recepción de línea.
                CJNE R0,#21H,IF1NUM   ;Checa si solo se tecleo retorno de carro.
                CLR CRFLAG          ;Limpia la bandera de retorno de carro.
                SJMP CICLO          ;Salta a recibir la siguiente línea.
IF1NUM:
                CJNE R0,#22H,SUMA     ;Checa si se tecleo un número solamente.
                CLR CRFLAG          ;Limpia la bandera de retorno de carro.
                SJMP CICLO          ;Salta a recibir la siguiente línea.
SUMA:
                DEC R0              ;Mueve el puntero al
                DEC R0              ; primer valor a sumar.
                MOV A,@R0           ;Carga reg. ACC con primer valor a sumar.
```

```
                INC R0          ;Mueve el puntero al segundo dato a sumar.
                MOV B,@R0       ;Carga reg. B con segundo valor a sumar.
                INC R0          ;Reinicializa puntero a localidad inicial.
                ADD A,B         ;Suma el primer y el segundo valores.
                DA A            ;Ajuste a decimal del ACC para suma en BCD.
                MOV B,A         ;Guarda la suma en el reg. B.
                ANL A,#0F0H     ;Enmascara
                CLR C           ; el
                RRC A           ; dato
                RRC A           ; a
                RRC A           ; enviar
                RRC A           ; por el puerto.
                ADD A,#30H      ;Suma para conversión a ASCII.
                CALL ENVIA
                CLR C           ;Enmascara el dato
                MOV A,B         ; a enviar
                ANL A,#0FH      ; por el puerto.
                ADD A,#30H      ;Suma para conversión a ASCII.
                ACALL ENVIA
                MOV A,#20H      ;Carga el ACC con espacio en blanco.
                ACALL ENVIA
                CLR CRFLAG      ;Limpia la bandera de retorno de carro.
                SJMP CICLO
;*************************************************************************
;Subrutina de transmisión del caracter recibido.
TRANS:
                CLR RI          ;Limpia la entrada de recepción.
                MOV B,A
                ACALL ENVIA
                MOV A,#20H      ;Carga el ACC con espacio en blanco.
                ACALL ENVIA
                MOV A,B
                RET             ;Fin subrutina de transmisión de caracter.
;*************************************************************************
;Subrutina de envío de caracter por la salida serial.
ENVIA:
                MOV SBUF,A      ;Pone el dato en puerto serie.
                CLR ES          ;Deshabilita la interrupción serial.
                JNB TI , $      ;Espera a que se envíe el dato.
                CLR TI          ;Limpia bandera de interrupción.
                SETB ES         ;Habilita interrupción serial.
                RET             ;Regresa.

                END             ;Fin del programa.
```