

MANUAL DEL MICROCONTROLADOR 8051

Dr. Alejandro Vega

Agosto-diciembre 1999

INDICE

	Página
INTRODUCCIÓN	3
PRIMERA PARTE.	
CAPITULO 1, CARACTERÍSTICAS DEL MICROCONTROLADOR 8051	5
1.1 CARACTERÍSTICAS DEL μ C-8051	6
1.2 DESCRIPCIÓN DE LAS LINEAS DEL μ C-8051	6
1.3 CARACTERÍSTICAS ESPECÍFICAS DEL μ C-8051	9
1.3.1 Descripción de los espacios de memoria.	9
1.4 MEMORIA DEL PROGRAMA.	11
1.4.1 Localidades de asignadas a las interrupciones	11
1.4.2 Memoria de programa interna y externa.	11
1.5 MEMORIA DE DATOS.	12
CAPITULO 2, REGISTROS DE FUNCIONES ESPECIALES (SFR).	14
2.1 LOCALIDADES DE LOS REGISTROS DE FUNCIONES ESPECIALES.	15
2.2 REGISTRO DE PALABRA DEL ESTADO DEL PROGRAMA.	16
2.3 REGISTRO DE CONTROL DE POTENCIA.	17
2.4 REGISTRO DE INTERRUPCIONES.	17
2.4.1 Registro Habilitador de Interrupciones (IE).	18
2.4.2 Registro de Prioridades.	19
CAPITULO 3, MANEJO DE LOS TEMPORIZADORES Y CONTADORES.	20
3.1 TIMER / CONTADOR	21
3.2 REGISTRO DE CONTROL DEL PUERTO TIMER/CONTADOR.	21
3.3 REGISTRO DE MODO DE CONTROL DEL TIMER/CONTADOR.	22
3.3.1 Modo 0 del Timer/Contador.	23
3.3.2 Modo 1 del Timer/Contador.	24
3.3.3 Modo 2 del Timer/Contador.	24
3.3.4 Modo 3 del Timer/Contador.	25
3.4 UTILIZACION DEL TIMER 1 COMO GENERADOR DEL "BAUD RATE" PARA LA TRANSMISION SERIE.	25
CAPITULO 4, CONTROL DEL PUERTO SERIE.	26
4.1 PUERTO SERIE	27
4.2 REGISTRO DE CONTROL DEL PUERTO SERIE SCON	27
4.3 MODOS DE CONTROL DEL PUERTO SERIE.	28
4.3.1 Transmisión serie, utilizando el modo de control 0.	29
4.3.2 Transmisión serie, utilizando el modo de control 1.	30
4.3.3 Transmisión serie, utilizando los modos 2 y 3.	32

CAPITULO 5, MODOS DE DIRECCIONAMIENTO.	35
5.1 MODOS DE DIRECCIONAMIENTO.	36
5.2 DIRECCIONAMIENTO DIRECTO.	36
5.3 DIRECCIONAMIENTO INDIRECTO.	36
5.4 DIRECCIONAMIENTO INMEDIATO.	37
5.5 DIRECCIONAMIENTO INDEXADO.	37
5.6 DIRECCIONAMIENTO POR REGISTRO.	37
5.7 TRANSFERENCIA DE DATOS.	38
5.7.1 RAM interna.	38
5.7.2 RAM externa.	39
5.7.3 Movimientos de tablas localizadas en la memoria del programa	39
5.8 INSTRUCCIONES BOOLEANAS.	39
5.9 INSTRUCCIONES DE SALTO.	40
5.9.1 Saltos condicionados.	40
5.9.2 Saltos incondicionados.	41

SEGUNDA PARTE

CONJUNTO DE INSTRUCCIONES DEL μC- 8051.	43
---	-----------

TERCERA PARTE

APLICACIONES DEL MICROCONTROLADOR 8051.	96
--	-----------

BIBLIOGRAFÍA	42
---------------------	-----------

ANEXOS

INTRODUCCIÓN

Hoy en día, el incremento competitivo en el mercado de la industria electrónica, crea la necesidad de diseñar sistemas con mejores características, de menor tamaño, bajos requerimientos de energía, mejor realización, teniendo un especial énfasis sobre todo en la facilidad de duplicidad del sistema diseñado. La lógica definida por el usuario, y realizada por el fabricante, permite individualizar a los sistemas diseñados, así como también apegarse más a los requerimientos específicos del usuario. Esto, tiene repercusión en el costo, realización, compactabilidad, desempeño y seguridad del diseño.

Con el fin de permitir la construcción de circuitos lógicos usando el concepto de lógica programable, los fabricantes de circuitos integrados han visto la necesidad de producir dispositivos, de alta velocidad, con los que se puedan desarrollar funciones lógicas de toda clase.

En estos circuitos, el usuario puede programar, en un sólo "chip", funciones que, de otra forma, con la circuitería tradicional de compuertas, utilizarían muchos componentes, además del espacio físico de los mismos.

En cursos anteriores ya hemos visto los PLD's ó "**Dispositivos Lógicos Programables**". Los diseñadores de circuitos digitales han utilizado tradicionalmente los PLD's para obtener funciones lógicas que generalmente no se encuentran disponibles como componentes estándares. Los PLD's son una excelente opción en sistemas donde el tamaño y complejidad de las tarjetas de circuito impreso, la confiabilidad, el número de componentes o la velocidad son factores críticos.

Como sucede con la familia de las memorias ROM, algunos PLD's se programan una sola vez, otros se pueden programar y borrar las veces que sea necesario. Los PLD's contienen compuertas y Flip-Flops, pero estos componentes no se encuentran interconectados en configuraciones fijas, sino que, se tienen pequeños fusibles que se funden o abren para interconectar dicha circuitería interna. Creando de esta manera cualquier configuración requerida.

Para muchas de las aplicaciones los PLD's son una excelente opción, desafortunadamente, sus aplicaciones se ven limitadas cuando es necesario transmisión de información, adquisición y tratamiento de datos, temporizaciones, etc. Es aquí donde hacen sus aparición los "**microcontroladores**", que son dispositivos mas versátiles que nos permitan llevar a cabo estas nuevas aplicaciones.

¿Pero, qué es un microcontrolador ?

Un Microcontrolador es todo un "sistema mínimo" dentro de un sólo dispositivo, lo cual ofrece un enorme panorama hacia el mundo de la compatibilidad. Este dispositivo contiene: Un CPU (basado principalmente en un microprocesador de 4, 8 ó 16 bits), puertos paralelos de entrada y salida, puerto serie, timers,

contadores, memorias, y en algunos casos hasta convertidores analógicos digitales, todo esto dentro de un solo chip.

¿Por qué un microcontrolador y no un microprocesador ?

Un microcontrolador está encaminado básicamente hacia aplicaciones concretas en donde, el espacio, y número de componentes es mínimo, además, los cambios o ampliaciones futuras del sistema son casi nulos. Por otro lado, un microprocesador se destina a sistemas donde su expansión a corto o mediano plazo es factible. A pesar de que un microprocesador es más rápido que un microcontrolador para la ejecución de sus instrucciones, en la mayoría de los casos es necesario interconectarlo con dispositivos periféricos

Un microcontrolador, puede ser utilizado con un mínimo número de componentes en trabajos específicos y en un amplio rango de aplicaciones, tales como; los sistemas de control de alarmas, tableros de control en la industria automotriz, en la instrumentación médica, en los teclados de computadora, en los sistemas portátiles de almacenamiento de datos, en equipos de laboratorio, etc.

En éste manual nos concretaremos específicamente al microcontrolador 8051, estudiaremos sus características, así como también algunas de sus aplicaciones.

El manual se divide básicamente en tres etapas fundamentales:

La primera, está enfocada hacia la descripción detallada de cada una de las líneas del microcontrolador 8051 ($\mu\text{C-8051}$), que nos permitirán la interconexión con los sistemas periféricos o directamente con los sistemas a controlar. También se estudiará cada uno de los registros internos, sus utilidades y sus aplicaciones.

En la segunda parte, veremos cada una de las instrucciones, las cuales reforzaremos con breves ejemplos específicos. Por último en la tercera etapa, presentaremos algunos diseños que se han tratado de detallar de la manera más completa, desde el planteamiento hasta la realización de sus programas.

1.1 CARACTERÍSTICAS DEL 8051.

La Familia de μ C-8051 es variada, y se encuentra en diversas presentaciones, la selección de uno o de otro tipo de microcontrolador dependerá principalmente de las necesidades a satisfacer. En este manual hablaremos de manera particular del 8051 el cual se presenta en tres versiones, con ROM interna (8051) la cual es programada directamente por el fabricante, con EPROM interna (8751) que puede ser programada por el usuario y sin PROM ni EPROM (8031), cuando el programa se selecciona de manera externa.

El μ c-8051 está basado en los microprocesadores de 8 bits, contiene internamente un CPU de 8bits, 3 puertos de entrada y salida paralelos, un puerto de control, el cual a su vez contiene; un puerto serie, dos entradas para Timer/Contador de 16 bits, dos entradas para interrupciones externas, las señales de RD y WR para la toma o almacenamiento de datos externos en RAM, la señal de PSEN para la lectura de instrucciones almacenadas en EPROM externa. Gracias a estas tres señales el μ c-8051 puede direccionar 64 K de programa y 64K de datos separadamente, es decir un total de 128Kb. Además cuenta con 128 bytes de memoria RAM interna.

Además el μ C-8051 puede generar la frecuencia (Baud Rate) de Transmisión/Recepción de datos por el puerto serie de manera automática partiendo de la frecuencia del oscilador general, por medio de la programación del Timer 1. Dicha frecuencia de transmisión puede ser cambiada en cualquier momento con solo cambiar el valor almacenado en el control o también se puede duplicar o dividir la frecuencia con solo escribir sobre el bit 7 (SMOD) del registro de control (PCON).

A continuación comenzaremos a ver con mayor detalle todo lo referente a sus conexiones así como también las características especiales del μ c-8051.

1.2 DESCRIPCION DE LAS LINEAS (PINS) DEL 8051.

El elemento más básico de la familia 8051 es el 8031, que carece de EPROM o PROM, el cual es direccionado externamente.

El 8031 es fundamentalmente un chip de 40 líneas.

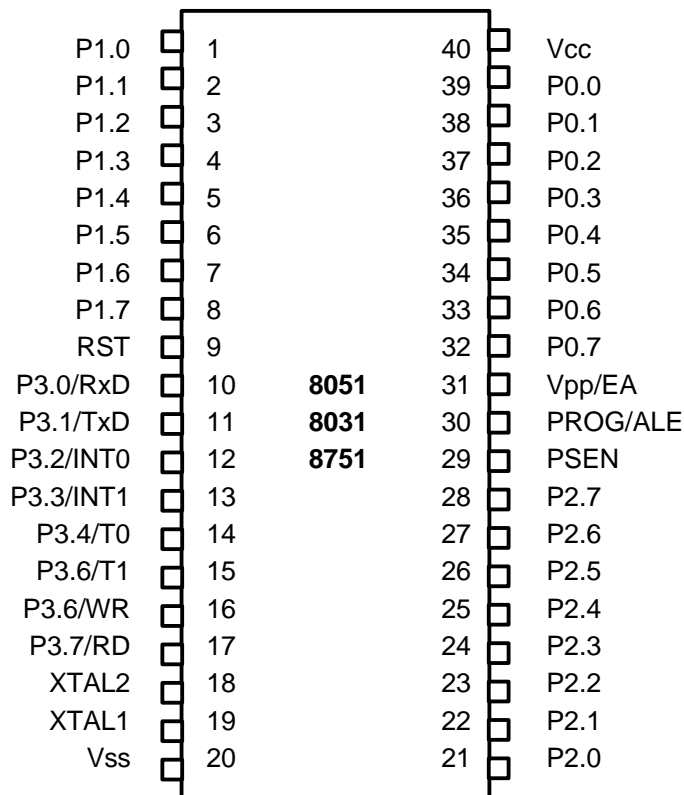


Fig.1 Microcontrolador 8051

Descripción de conexiones.

Nemónico Conex		Tipo	Nombre y función
Vss	20	ENT	Tierra 0V referencia.
P0.0-P0.7	39-32	E/S	Port 0. Es un puerto bidireccional con salidas en colector abierto. Cuando el puerto tiene 1's escritos, las salidas están flotadas y pueden servir como entradas en alta impedancia. El puerto 0 es también multiplexado para obtener el DATO y la parte baja de la dirección.
P1.0-P1.7	1-8	E/S	Port 1. Es un puerto quasidireccional, cuando se escribe 1's en el puerto, el puerto puede ser utilizado como entrada.
P2.0-P2.7	21-28	E/S	Port 2. Es un puerto quasi-bidireccional con fijadores de nivel internos (pull-up). Cuando se escriben 1's sobre el puerto, las líneas pueden ser utilizadas como entradas o salidas. Como entradas, las líneas que son externamente colocadas en la posición baja proporcionaran una corriente hacia el exterior. El puerto 2 es utilizado además para direccionar memoria externa. Este puerto, emite el byte más alto de la dirección durante la búsqueda de datos en la memoria del programa externo y durante el acceso a memorias de datos externos que usan direccionamientos de 16 bits. Durante el acceso a una memoria de dato externa, que usa direcciones de 8 bits, el puerto dos emite el contenido del registro del correspondiente a este puerto, que se encuentra en el espacio de funciones especiales.
P3.0-P3.7	10-17	E/S	Port 3. Es un puerto quasi-bidireccional con fijadores de nivel internos (PULL-UP). Cuando se escriben 1's sobre el puerto, las líneas pueden ser utilizadas como entradas o como salidas. Como entradas las líneas que son externamente colocadas en la posición baja proporcionarán una corriente. El puerto 3 se utiliza además para producir señales de control de dispositivos externos como son los siguientes:
	10	E	RxD(P3.0): Puerto serie de entrada.
	11	S	TxD(P3.1): Puerto serie de salida.

	12	E	INT0(P3.2): Interrupción externa.
	13	E	INT1(P3.3): Interrupción externa.
	14	E	T0(P3.4): Entrada externa timer0.
	15	E	T1(P3.5): Entrada externa timer1.
	16	S	WR(P3.6): Habilitador de escritura para memoria externa de datos.
	17	S	RD (P3.7): habilitador de lectura para la memoria externa de datos.
RST	9	E	Reset. Una entrada alta en esta línea durante dos ciclos de maquina mientras el oscilador está corriendo detiene el dispositivo. Un resistor interno conectado a Vss permite un alto en la fuente usando solamente un capacitor externo a VCC.
ALE	30	E/S	Address Latch Enable. Un pulso positivo de salida permite fijar el byte bajo de la dirección durante el acceso a una memoria externa. En operación normal, ALE es emitido en un rango constante de 1/6 de la frecuencia del oscilador, y puede ser usada para cronometrar. Note que un pulso de ALE es emitido durante cada acceso a la memoria de datos externos.
PSEN	29	S	Program Store Enable. Habilitador de lectura para memoria de programas externos. Cuando el 8031B/8051 está ejecutando un código de una memoria de programas externos, PSEN es activada dos veces cada ciclo de máquina, excepto cuando se accesa la memoria de datos externos que omiten las dos activaciones del PSEN externos. PSEN tampoco es activada cuando se usa la memoria de programas internos.
EA	31	E	External Access Enable. EA debe mantenerse externamente en posición baja para habilitar el mecanismo que elige el código de las localizaciones de la memoria de programas externos, 0000H y 0FFFFH. Si EA se mantiene en posición alta, el dispositivo ejecuta los programas que se encuentran en la memoria interna ROM, a menos que el contador del programa contenga una dirección mayor a 0FFFFH.
XTAL1	19	E	Crystal 1. Es la entrada del cristal para el circuito oscilador (generador del reloj interno) que amplifica e invierte la entrada.
XTAL2	18	0	Crystal 2. Es la salida del amplificador oscilador inversor.

1.3 CARACTERISTICAS ESPECIFICAS DEL 8051.

El 8051 contiene las siguientes características:

- 1 CPU de 8 bits como parte central.
- 32 líneas bidireccionales de entrada y salida (4 puertos)
- 128 bytes de memoria RAM
- 2 Controladores / Timers de 16 bits
- 1 UART completo
- 5 estructuras de interrupción con dos niveles de prioridad
- 1 circuito de reloj
- 64 Kbytes de espacio para programa.
- 64 Kbytes de espacio para datos.

1.3.1 DESCRIPCION DE LOS ESPACIOS DE MEMORIA.

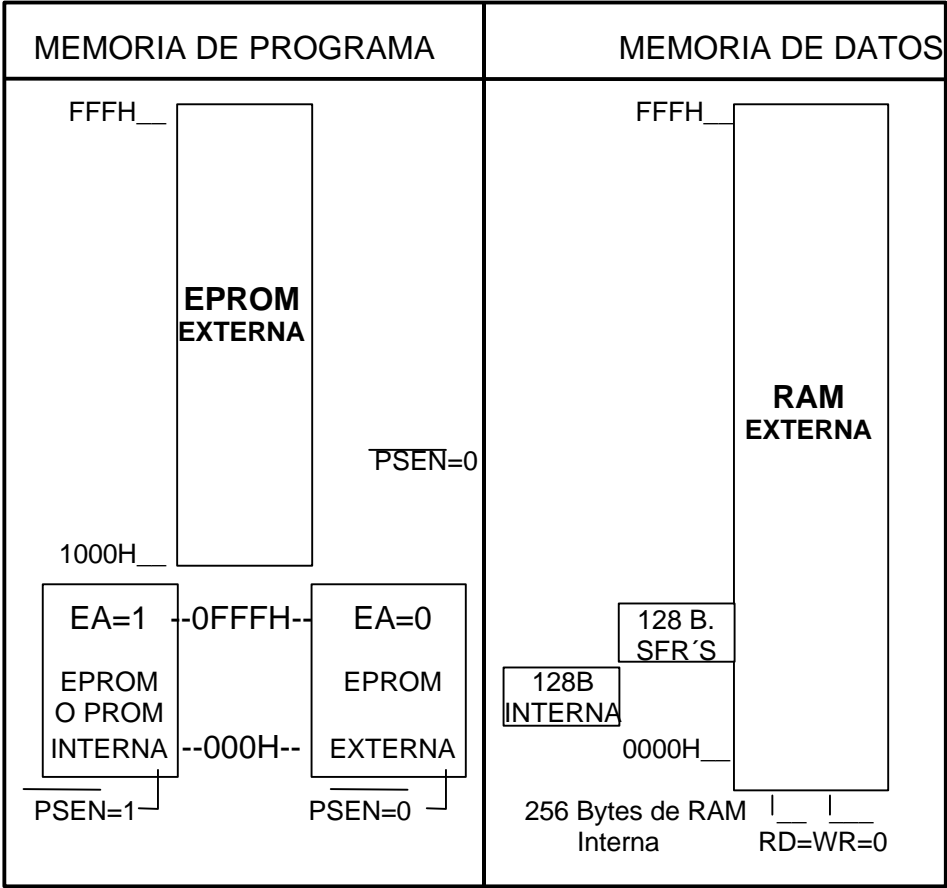
La memoria del sistema del 8051 se clasifica en tres partes fundamentales: (fig. 2):

La primera, llamada memoria de programa, en donde se encuentran todas las instrucciones que van a ser ejecutadas por el μ C-8051, es decir, el programa de trabajo. Algunas versiones del 8051 cuentan con memoria de programa interna (de 2 a 4 Kb). Cuando se requiere trabajar con una localidad arriba de ésta, la memoria del programa (externa) es seleccionada mediante la activación de la señal PSEN (estado bajo). El máximo espacio de memoria de programa que se puede acceder es de 64KB.

El segundo espacio de memoria denominado, memoria de datos es accedido mediante la activación de las señales RD y WR, durante la lectura o escritura de datos respectivamente. En este espacio del μ C toma todos valores que se encuentran en memoria como DATOS, es decir, el μ C no puede ejecutar ninguna instrucción que se encuentre aquí almacenada. El 8051 puede direccionar también 64KB de memoria de datos.

El tercer espacio de memoria es denominado como memoria RAM interna, el cual se subdivide en 128 bytes de memoria bajos y en 128 bytes de memoria altos. En los primeros 128, se encuentran 4 bancos de 7 registros cada uno. Estos registros son de gran ayuda para la simplificación de los programas, debido a que cada uno de ellos nos permiten almacenar datos momentáneamente y realizar un vasto número de instrucciones del 8051. También dentro de este espacio, se encuentran 16 bytes (del 20H al 2FH) que pueden ser direccionados directamente por bit.

En la parte alta de la memoria RAM interna, se encuentran el contenido de los Registros de Funciones Especiales, formado por Puertos, Registros de Control, Acumuladores, Registros de interrupción, etc. Todos estos registros los veremos detalladamente posteriormente.



1.4.1 LOCALIDADES ASIGNADAS A LAS INTERRUPCIONES.

A continuación veremos de manera más detallada el espacio destinado al programa de trabajo.

La tabla 1, muestra las localidades que han sido asignadas por el fabricante, para dar servicio a las rutinas de interrupción.

FUENTE DE INTERRUPCIÓN	VECTOR DE DIRECCIONES
IE0 (Interrupción 0 externa)	0003H
TF0 (Interrupción del timer 0)	000BH
IE1 (Interrupción 1 externa)	0013H
TF1 (Interrupción del timer 1)	001BH
R1 y T1 (Interrupción serial)	0023H
TF2 y EXF2 (Sólo para el 8052)	002BH

Tabla 1. Interrupciones

Una interrupción puede ser causada de manera externa o interna, es decir puede ser producida por un dispositivo periférico o por programación respectivamente. La interrupción con mayor alto orden es el RESET el cual no puede ser mascarable. Cuando el RESET ocurre el programa comienza a partir de la dirección 0000H del programa.

Cuando una interrupción es producida, el Contador del Programa (PC) almacena su contenido temporalmente dentro del SP (apuntador de apilamiento) y se carga con la dirección de la localidad donde se encuentra la rutina de servicio de la interrupción correspondiente. Una vez posicionado en esa localidad deberá de comenzar la ejecución de la rutina de servicio, hasta que encuentre la instrucción RETI, que le permitirá al PC recuperar nuevamente su valor original almacenado en el SP, y continuar con el programa anterior a la interrupción.

Por ejemplo a la interrupción 0, se le asigna la localidad 0003H, si la interrupción no se utiliza, esta localidad puede utilizarse para propósitos generales del programa, si la interrupción ha sido permitida, (estableciendo el bit correspondiente dentro del registro de control IE), en el momento que exista una activación de la interrupción (estado bajo en la línea INTO) el PC se cargará con 0003 y saltará a esa localidad para comenzar a ejecutar la rutina de servicio.

Estas localidades de memoria de los servicios de interrupción están separadas en intervalos de 8 bytes, entre sí. Cuando un servicio de interrupción es corto, éste puede estar contenido en los 8 bytes. En el caso de que fuese largo se puede ejecutar un salto a otra localidad de memoria para continuar con la secuencia de interrupción. El término del servicio de interrupción deberá de realizarse mediante la ejecución de la instrucción de la instrucción RETI.

1.4.2 MEMORIA DE PROGRAMA INTERNA Y EXTERNA.

Cuando se utilizan elementos de la familia del 8051 con memoria interna ROM (o 16K), esta puede ser accesada mediante la conexión de la línea EA =1 (Vcc). Si la memoria interna es de 4 Kbytes y EA = 1, el CPU seleccionará internamente el ROM, desde 0000H hasta 0FFFH y de manera externa automáticamente a partir de 1000H hasta FFFFH.

Por el contrario, si la línea EA = 0, el CPU seleccionará de forma externa el ROM, desde la dirección 0000H hasta FFFFH. En el caso del 8031 ésta línea se conecta siempre a 0 Volts (Vss).

La línea PSEN (Program Store Enable), que sirve para leer el ROM externo, es activado en todas las búsquedas (Fetches) del programa. PSEN NO SE ACTIVA en búsquedas (fetches) del ROM interno. La fig. 1.3 muestra un conexionado a una EPROM externo.

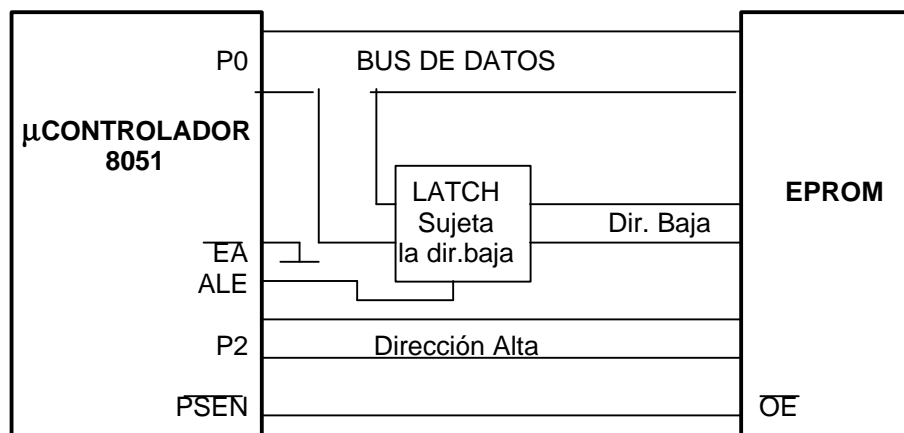


Fig. 1.3 Conexión de una memoria externa

1.5. MEMORIA DE DATOS (DATA MEMORY)

El espacio de memoria RAM interno está dividido en tres espacios, el primer bloque es referido como la parte baja de 128 bytes, el segundo (se tiene sólo en algunas versiones del 8051 v.gr. 8052), la parte alta de 128 bytes y el tercero, llamado espacio SFR (Registros de Funciones Especiales).

Las direcciones de la Memoria Interna de Datos siempre son de un byte (de 00H a FFH). Sin embargo los modos de direccionamiento para la memoria interna pueden acomodar hasta 384 bytes, como se ve en la versión 8052, lo cual es posible debido a que el modo de direccionamiento directo accede un espacio de memoria diferente físicamente al permitido por el modo de direccionamiento indirecto.

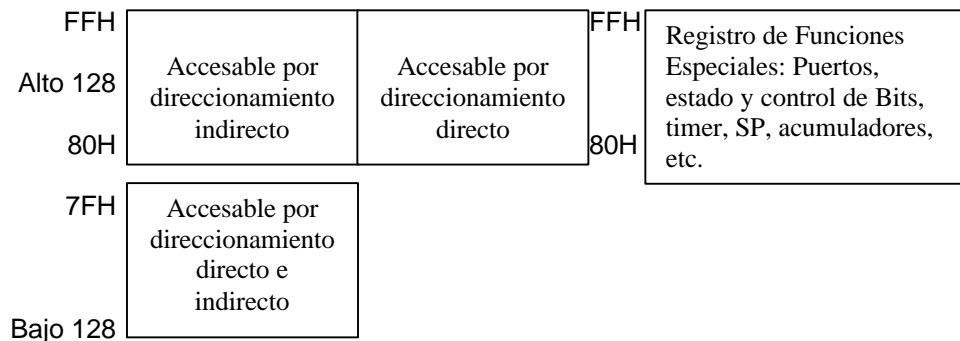


Fig. 1.4 Estructura de la memoria interna.

Los primeros 128 bytes, son presentados en todos los dispositivos de la familia MCS-51, que está mapeados como se presenta en la fig 1.5.

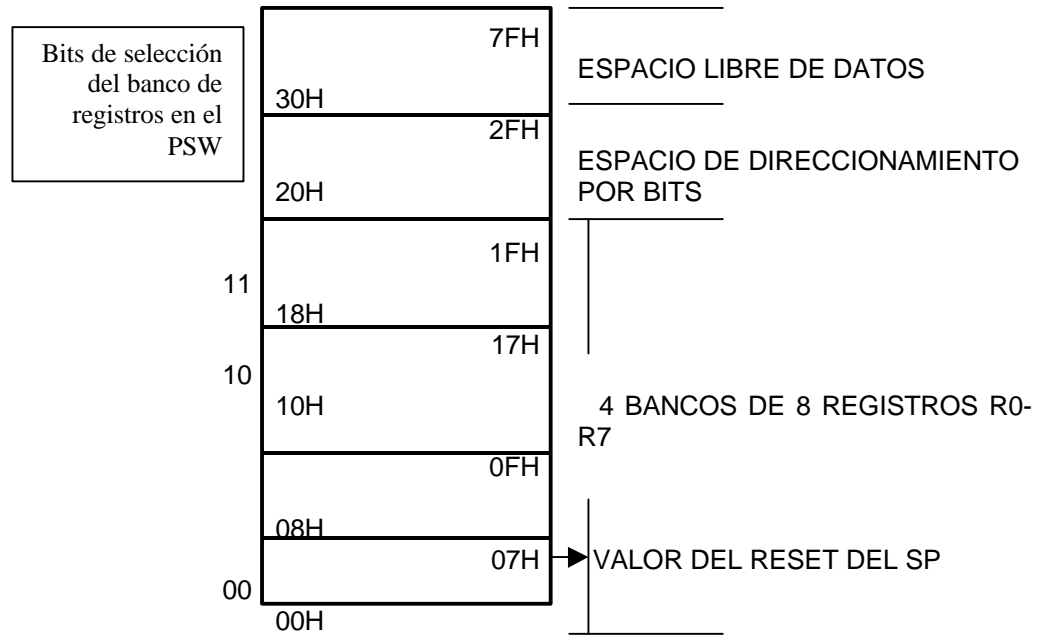


Fig. 1.5 Distribución de los 128 Bytes más bajos de la memoria RAM interna.

Como se puede apreciar en la figura anterior, los 128 bytes más bajos son divididos en 4 bloques de 8 registros cada uno, que contienen los valores de los registros R0 A R7, los bloques pueden ser seleccionados mediante la escritura en los bits 3 y 4 del registro PSW (palabra del estado del programa), el cual veremos más adelante. La utilización de registros permiten un uso más eficiente del espacio de códigos debido a que sus direccionamientos son de 8 bits únicamente.

Como ya habíamos mencionado anteriormente, existen algunas versiones del 8051, como el 8052, que contienen 128 bytes de memoria interna que puede ser direccionada indirectamente. Por otro lado, todas las versiones del 8051 contienen un espacio de 128 bytes en la parte alta de la memoria que son direccionados directamente, en este espacio se localizan los Registros de Funciones Especiales (SFR). Estos registros especiales, tienen sus localidades bien establecidas, y son utilizados por el microcontrolador para realizar las distintas operaciones internas que ejecuta el microcontrolador, así como también para el control y acceso de los diferentes puertos de entrada y salida.

2.1 REGISTROS DE FUNCIONES ESPECIALES (SFR)

En el capítulo anterior habíamos visto que los SFR se encuentran en la parte alta (128 bytes) de la memoria RAM interna del 8051. Las direcciones de los SFR es mostrado en la Tabla 2.

SÍMBOLO	NOMBRE	DIRECCIÓN
ACC	Acumulador	0EOH
B	Registro B	0F0H
PSW	Program Status Word (Palabra de estado del programa)	0DOH
SP	Stack Pointer (apuntador de apilamiento)	81H
DPTR	Data Pointer (apuntador de datos)16bits	
DPL	Data Pointer low byte	82H
DPH	Data Pointer high byte	83H
P0	Puerto 0	80H
P1	Puerto 1	90H
P2	Puerto 2	0A0H
P3	Puerto 3	0B0H
IP	Control de Prioridad de Interrup.	0B8H
IE	Control de Validación de Interrup.	0A8H
TMOD	Modo de control Timer/Contador	89H
TCON	Control del Timer/Contador	88H
T2CON(8052)	Control 2 del Timer/Contador	0C8H
TH0	Byte alto del T/C 0	8CH
TL0	Byte bajo del T/C 0	8AH
TH1	Byte alto del T/C 1	8DH
TL1	Byte bajo del T/C 1	8BH
TH2(8052)	Byte alto del T/C 2	0CDH
TL2(8052)	Byte bajo del T/C 2	0CCH
RCAP2H(8052)	Byte alto del registro de captura T/C2	0CBH
RCAP2L(8052)	Byte bajo del registro de capt. del T/C2	0CAH
SCON	Control serie	98H
SBUF	Buffer de datos serie	99H
PCON	Control de Potencia	87H

Tabla 2 Espacio de los registros de funciones especiales.

En la Fig. 2.1 se muestra el mapa de memoria correspondiente a los registros de funciones especiales.

Mapa de memoria de los SFR (registros de funciones especiales)

F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2		CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TLO	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH			PCON	87

Fig. 2.1 Mapa de memoria de los SFR

A continuación veremos con mayor detalle cada uno de los principales registros de control del 8051.

2.2 REGISTRO DE PALABRA DEL ESTADO DEL PROGRAMA (PROGRAM STATUS WORD)

El registro de palabra del estado del programa contiene algunos bits que reflejan el estado del CPU en ese instante. El PSW se muestra en la fig.2.2.

CY	AC	F0	RS1	RS0	OV	-	P
-----------	-----------	-----------	------------	------------	-----------	----------	----------

Fig. 2.2 Registro PSW (Palabra de Estado del Programa)

CY	PSW.7	Bandera del carry
AC	PSW.6	Bandera del carry auxiliar (operaciones en BCD)
F0	PSW.5	Bandera 0 para usos generales
RS1	PSW.4	Bit 1 selector del banco de registros
RS0	PSW.3	Bit 0 selector del banco de registros
OV	PSW.2	Bandera del overflow
-	PSW.1	Bandera sin definir
P	PSW.0	Bandera de paridad, establece/limpia por hardware, indica si el número de 1's en el acumulador es par o impar.

Este registro como ya se vio, reside en el espacio SFR. El registro contiene; el bit de Carry, El bit Auxiliar (para operaciones BCD), los dos bits de selección del banco de registros, la bandera de overflow, el bit de paridad y dos banderas sin definir.

El bit de Paridad refleja el número de 1's, en el acumulador:

P=1, si el Acumulador contiene un número impar de 1's

P=0, si el Acumulador contiene un número par de 1's, es decir el número de 1's, en el acumulador más P es siempre par.

2.3. REGISTRO DE CONTROL DE POTENCIA (CONSUMO DE ENERGIA).

En la fig.2.3 tenemos el registro PCON, el cual a excepción de la bandera SMOD, sirve para controlar, principalmente el consumo de energía, el cual es utilizado sólo por los dispositivos fabricados con la tecnología CHMOS que permite disminuir dicho consumo de energía, en estados de espera. La bandera PCON.7 (SMOD) sirve para dividir la frecuencia de transmisión o de recepción por el puerto serie, proporcionada ya sea, por la fase 2 de los estados, (1/2 de la frecuencia del oscilador en la transmisión serie en modo 2), o bien, por el timer 1 en los modos 1 y 3.

SMOD	-	-	-	GF1	GF0	PD	IDL
-------------	---	---	---	------------	------------	-----------	------------

Fig.2.3 PCON (Registro de control de potencia)

SMOD	Dobla el "BAUD RATE" para el puerto serie cuando se utiliza el timer para generar el BAUD RATE.	
GF1	Propósitos generales	
GF0	Propósitos generales	
PD	Bajo consumo de energía	} 80C51BH CHMOS
IDL	Bajo consumo de energía	

2.4. REGISTROS DE INTERRUPCIONES.

Las interrupciones son controladas mediante la escritura en los registros IE (Interrupt Enable) e IP (Interrupt Priority) los cuales son físicamente representados en la fig.2.4.

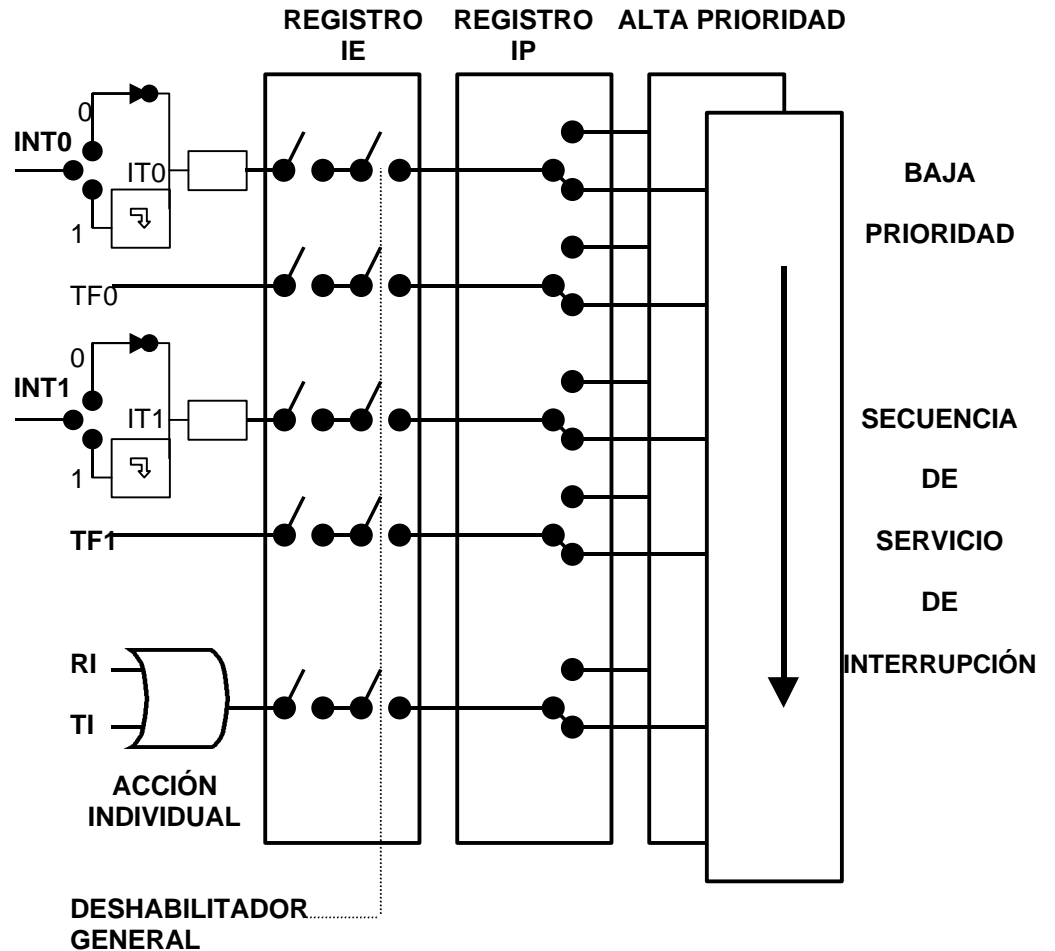


Fig. 2.4 Sistema de control de interrupciones

2.4.1 REGISTRO HABILITADOR DE INTERRUPTONES (IE).

La desactivación general de las interrupciones es efectuada mediante la escritura de un 0 lógico, en la bandera EA (IE.7).

Con la bandera EA=1, el 8051 está en condiciones de aceptar interrupciones, aunque la verdadera aceptación es realizada cuando se escribe un 1 lógico, en la bandera de la interrupción correspondiente del registro de interrupciones, IE (fig.2.5).

EA	-	ET2	ES	ET1	EX1	ET0	EX0
-----------	----------	------------	-----------	------------	------------	------------	------------

Fig. 2.5 Registro IE (Interrupt Enable)

EA	IE.7	Desactiva todas las INTERRUPTONES EA=0.
ET2	IE.5	Activa la interrupción causada por el timer2 (ET2=1)
ES	IE.4	Activa la interrupción causada por el puerto serial.
ET1	IE.3	Activa la interrupción de sobreflujo causada por el timer 1.
EX1	IE.2	Activa la interrupción causada externamente en INT1.
ET0	IE.1	Activa la interrupción de sobreflujo causada por el timer 0.
EX0	IE.0	Activa la interrupción causada externamente en INT 0.

2.4.2 REGISTRO DE PRIORIDAD.

El 8051 tiene dos planos de prioridad para trabajar las interrupciones, llamadas alto y bajo, respectivamente. En la inicialización, todas las interrupciones trabajan en el plano de baja prioridad. Para pasar del plano de baja prioridad al de alta, es necesario escribir un 1 lógico en las banderas correspondientes a las interrupciones que se desean aumentar de prioridad, ubicadas dentro del registro IP (fig.2.6.)

-	-	PT2	PS	PT1	PX1	PT0	PX0
----------	----------	------------	-----------	------------	------------	------------	------------

Fig. 2.6 Registro de Prioridad de Interrupciones

PT2	IP.5	Timer 2 PT2=1 mayor prioridad.
PS	IP.4	Define el nivel de prioridad de la interrupción del puerto serial.
PT1	IP.3	Define el nivel de prioridad de la interrupción del Timer 1.
PX1	IP.2	Define el nivel de prioridad de la interrupción externa 1.
PT0	IP.1	Define el nivel de prioridad de la interrupción del Timer 0.
PX0	IP.0	Define el nivel de prioridad de la interrupción externa 0.

Aunque los registros de control de los puertos del Timer / Contador y Serie pertenecen a los registros de funciones especiales les vamos a dedicar un capítulo completo a cada uno de ellos debido a la importancia que presentan para el desarrollo e interconexión con sistemas periféricos.

3.1 TIMER / CONTADOR .

El 8051 tiene 2 timer/contadores de 16 bits cada uno, llamados Timer 0 y el Timer 1 respectivamente. Ambos pueden ser configurados para operar como temporizadores (timers) o como contadores (counters).

Cuando se trabaja como contador, el registro interno del contador, es incrementado cada vez que existe una transición negativa (de 1 a 0) por la línea de entrada correspondiente a T0 ó T1. En cambio, cuando funciona como temporizador "Timer", el registro es incrementado cada 12 periodos de oscilación es decir su frecuencia de conteo es 1/12 de la frecuencia del oscilador.

En el momento que los bits del registro del contador pasan de todos 1's a todos 0's, se activa la línea de interrupción interna correspondiente a TF0 o TF1, generándose, (si ha sido permitida) una interrupción.

3.2 REGISTRO DE CONTROL DEL PUERTO TIMER/CONTADOR.

El registro de control del Timer/Contador de la fig. 3.1 es direccionable por Bit, para activar o desactivar cada una de sus banderas.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

Fig. 3.1 Registro del control del Timer/Contador

- TF1 TCON.7** Bandera de sobreflujo (overflow) del registro del Timer 1. Activada por hardware cuando el registro que guarda la cuenta del Timer/Contador 1, incrementa su contenido pasando todos sus bits de 1's a 0's. Limpiado por hardware cuando existe el procesamiento de los vectores del servicio de las rutinas de interrupción.
- TR1 TCON.6** Bit de control de activación del timer 1. Habilitado/Deshabilitado por software para colocar el Timer/Contador en Encendido/Apagado.
- TF0 TCON.5** Bandera de sobreflujo (overflow) del registro del Timer 0. Activada por hardware cuando el registro que guarda la cuenta del timer/contador 0, incrementa su contenido pasando todos sus bits de 1's a 0's. Limpiado por hardware cuando existe el procesamiento de los vectores del servicio de las rutinas de interrupción.

TR0 TCON.4 Bit de control de activación del timer 0. Habilitado/Deshabilitado por software para colocar el Timer/Contador en Encendido/Apagado.

IE1 TCON.3 Bandera de transición de la interrupción externa 1. Activada por hardware cuando una transición (de 1 a 0) en la línea de interrupción externa 1, es detectada. Limpiada por hardware cuando la interrupción es procesada. (solamente se acciona si se programó la aceptación de la interrupción por transiente, IT1=1).

IT1 TCON.2 Bit de control del Interrup 1. Activado/Limpiado por software para especificar el tipo de interrupción, por nivel bajo (IT1= 0) o por transiente negativo (IT1=1).

IE0 TCON.1 Bandera de transición de la interrupción externa 0 Activada por hardware cuando una transición (de 1 a 0) en la línea de interrupción externa 0, es detectada. Limpiada por hardware cuando la interrupción es procesada. (solamente se acciona si se programó la aceptación de la interrupción por transiente, IT0=1).

IT0 TCON.0 Bit de control del Interrup 0. Activado/Limpiado por software para especificar el tipo de interrupción, por nivel bajo (IT0= 0) o por transiente negativo (IT0=1). 3.3 registro de modo de control del timer/contador.

3.3 REGISTRO DE MODO DE CONTROL DEL TIMER/CONTADOR

Este registro permite especificar si se van a trabajar como Temporizadores (Timers) o como Contadores (Counters), los puertos denominados Timer 0 y Timer 1.

Existen 4 modos de trabajo para estos puertos, los cuales son definidos por la escritura en los bits M1 y M0 de TMOD fig 3.2.

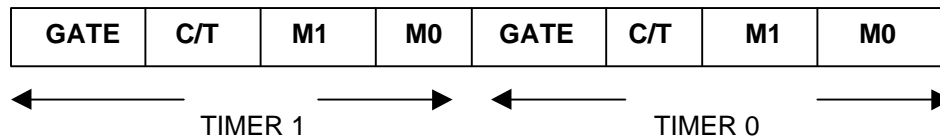


Fig.3.2 TMOD: Registro del Modo de Control del Timer/Counter

GATE Cuando TRx (en TCON) está activada y GATE=1, TIMER/COUNTERx correrá. solamente si la línea INTx está en posición alta (control por hardware). Cuando GATE=0, TIMER/COUNTERx correrá solamente si TRx=1 (control por software)

C/T Selector de Timer o de Contador. Es limpiado por la operación del Timer (entrada del reloj del sistema interno). Es activada por la operación del Contador (entrada de la línea Tx)

M1 Bit selector del modo.
M0 Bit selector del modo.

M1	M0	MODO	ESPECIFICACIÓN
0	0	0	Timer/contador de 13 bits
0	1	1	Timer/contador de 16 bits
1	0	2	Timer/contador de 8 bits recargables
1	1	3	Timer 0, TL0 Timer/contador de 8 bits, controlado por los bits de control del Timer 0 . TH0 Timer de 8 bits controlado por los bits de Control del timer 1. (El Timer 1 no se utiliza)

3.3.1 MODO 0, DEL TIMER/CONTADOR.

En este modo, cualquiera de los 2 Timers, 0 ó 1, trabajan como un contador de 8 bits, al cual le antecede un predivisor de la frecuencia de conteo. En la fig.3.3 se muestra el modo de operación 0 para el timer 1.

El registro del Timer 1 está configurado como un registro de 13 bits, que consisten de los 8 bits de TH1 y los 5 bits menos significativos de TL1. Los 3 bits mas significativos de TL1 no se utilizan en este modo.

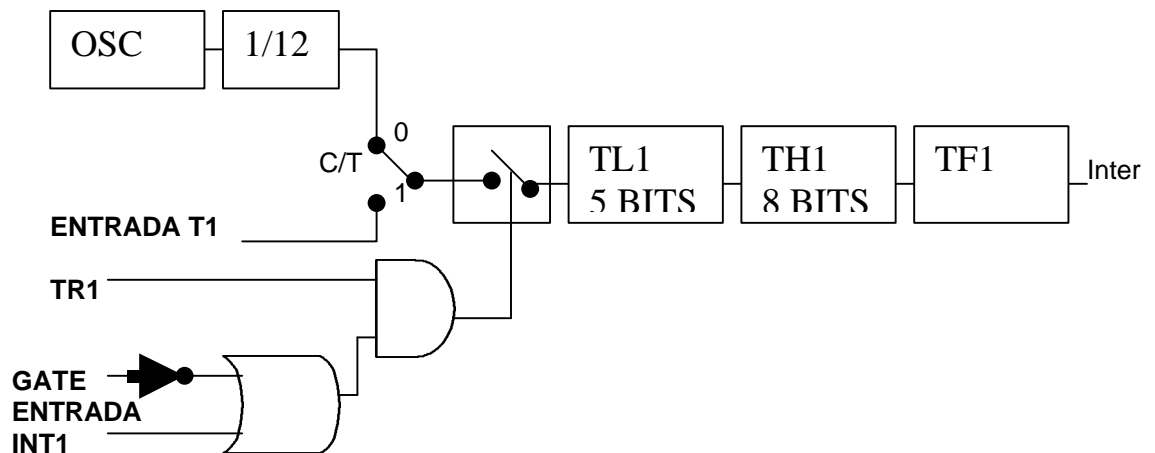


Fig. 3.3 Modo 0, contador de 13 bits con el Timer1

El valor de la cuenta se puede apreciar en el registro TH1, dado que TL1 solamente actúa como divisor de frecuencia, TH1 se puede cargar con cualquier valor, de 1 a 256 cuentas, pudiendo obtener de esta forma varios retardos, solamente detectando la bandera de sobreflujo (overflow TF1).

3.3.2 MODO 1 DEL TIMER/CONTADOR

Este modo es utilizado por cualquiera de los 2 Timers, se caracteriza principalmente por ser un Timer/contador de 16 bits cuyos valores se encuentran cargados en los registros TH y TL de cada uno de los Timers.

En la fig. 3.4 se puede apreciar la disposición de estos registros que se asemejan al modo 0, solo que en el modo 1, actúan en cascada.

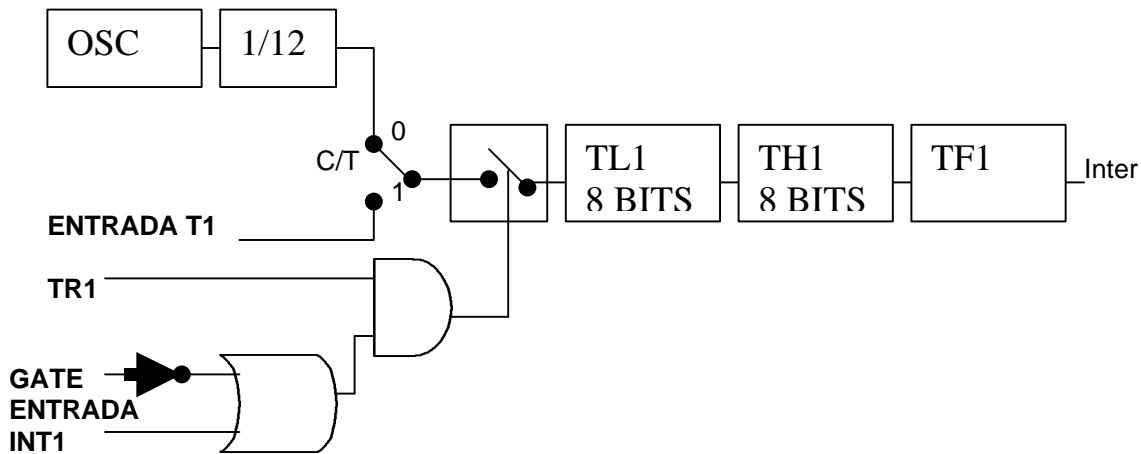


Fig. 3.4 Modo 1 contador de 16 bits con Timer/Contador 1

3.3.3 MODO 2 DEL TIMER/CONTADOR

Este modo puede ser utilizado tanto por el timer 0 como por el 1, tienen un registro de conteo de 8 bits (TLx).

En la figura 3.5 se presenta el manejo del Timer 1, en el modo 2. El registro TL1 es cargado automáticamente con el contenido de TH1, cuando se produce el sobreflujo en TL1, el cual además establece la bandera de TF1.

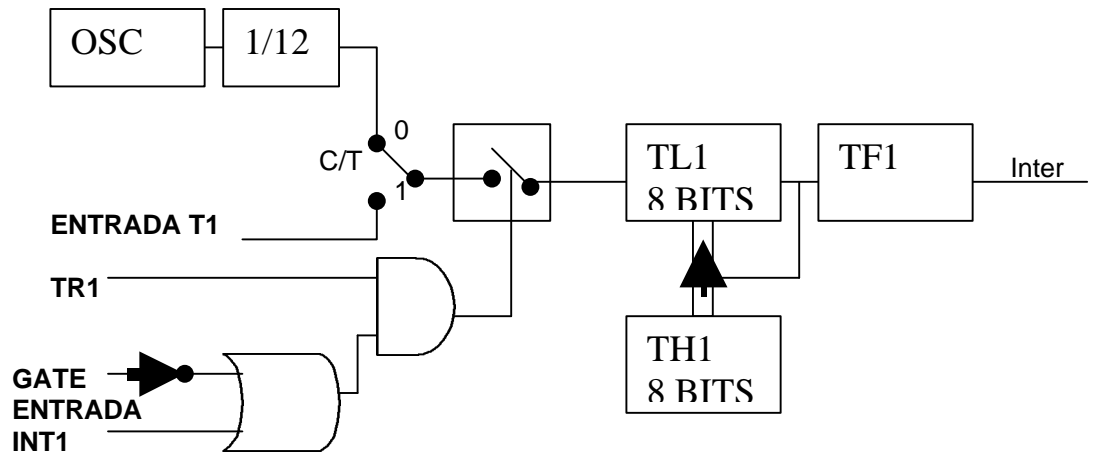


Fig. 3.5 Modo 2, registro de 8 bits autorecargable

3.3.4 MODO 3 DEL TIMER/CONTADOR

El Timer 1, en el modo 3 mantiene su cuenta, es decir, tiene el mismo efecto que cuando se establece la bandera TR1=0.

El Timer 0, en éste modo, establece TL0 y TH0 como dos contadores separados. La figura 3.6 muestra la lógica para el modo 3. TL0 utiliza los bits de control (C/T, GATE, TR0, INT0) del Timer 0. TH0 es bloqueado como temporizador "Timer", el cual emplea las señales de control del Timer 1, TR1 y TF1.

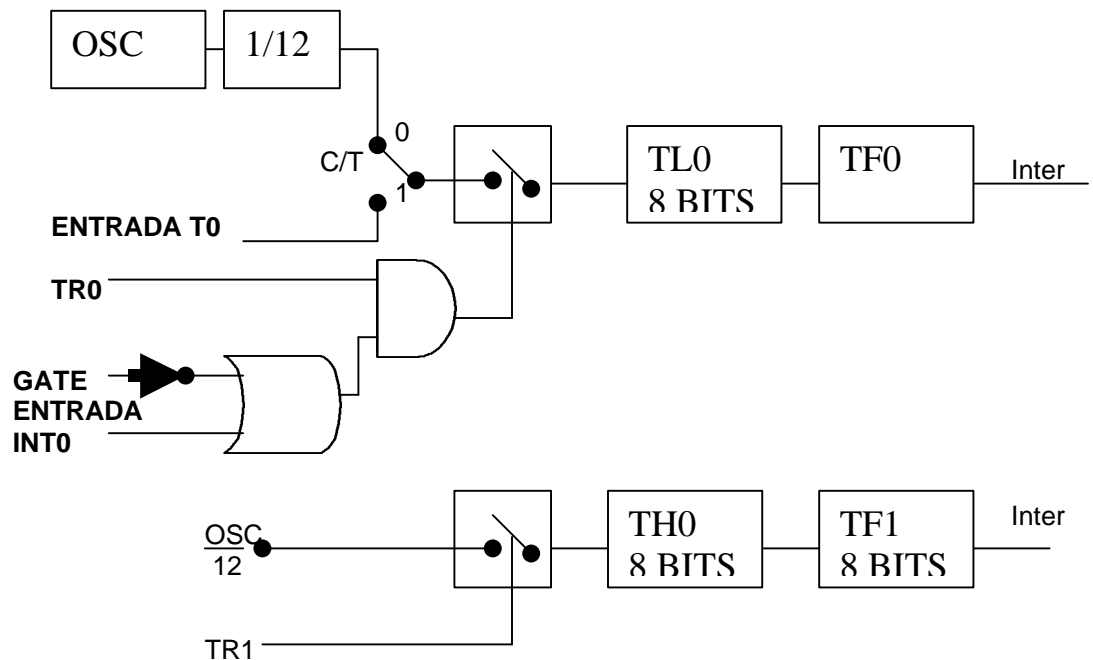


Fig. 3.6 Modo 3, 2 timers de 8 bits

El Timer 1 puede ser activado o desactivado con solo salir o entrar al modo 3 respectivamente o puede permanecer siendo utilizado por el puerto Serie cuando

está generando la frecuencia de oscilación "Baud rate",o en efecto en cualquier aplicación que no se requiere una interrupción.

3.4 UTILIZACIÓN DEL TIMER 1 COMO GENERADOR DEL "BAUD RATE" PARA LA TRANSMISIÓN SERIAL.

El Timer 1 es usado para generar la frecuencia de transmisión / Recepción de datos en serie, cuando el puerto es programado para trabajar en el modo 1 ó 3. La frecuencia de transmisión es obtenida a partir del valor almacenado en TH1 y el valor de SMOD mediante la siguiente ecuación:

$$\text{Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Frec. Del Oscilador}}{12 \times [256 - (\text{TH1})]}$$

El valor $2^{\text{SMOD}} / 32$, es debido a los circuitos divisores de frecuencia que se encuentran en la etapa de control del Puerto Serie, los cuales dividen entre 16 ó 32 dependiendo del bit 7 (SMOD) del registro de control PCON y la frecuencia que nos proporciona la salida del Timer 1 (overflow).

El valor 12 que divide a la frecuencia del oscilador proviene del divisor, que se encuentra en la etapa de control del Timer 1, cuando éste es utilizado como temporizador.

NOTA: Cabe recalcar que el valor que se almacena en TH1 es el valor negativo de la cuenta que se desea, debido a que, el contador se incrementa cada vez que un pulso es detectado, de ahí que en la ecuación se representa como $256-(\text{TH1})$.

La interrupción del Timer 1 en éste caso no tendría mucha aplicación por lo que se podría deshabilitar.

El Timer 1 actúa en modo 2, es decir en modo recargable, el valor de conteo se encuentra fijo en el registro TH1, el cual se recarga cada vez que existe un overflow.

La figura 3.7 muestra una tabla de valores de TH1, para generar el Baud Rate, tomando en cuenta la frecuencia del oscilador.

BAUD RATE	F osc	SMOD	TIMER1		
			C/T	MOD0	VALOR DE TH1
19.2 kHz	11.059 MHz	1	0	2	FDH
9.8 kHz	11.059 MHz	0	0	2	FDH
4.8 kHz	11.059 MHz	0	0	2	FAH
2.4 kHz	11.059 MHz	0	0	2	F4H
1.2 kHz	11.059 MHz	0	0	2	E8H
137.5 Hz	11.986 MHz	0	0	2	1DH
110 Hz	6.000 MHz	0	0	2	72H

FIG. 3.7 Tabla de valores para generar el Baud Rate

4.1 PUERTO SERIE

El puerto serie es un puerto "FULL DUPLEX", lo cual significa que puede transmitir y recibir datos simultáneamente. El receptor contiene un almacén "Buffer", que le permite comenzar a recibir un segundo dato sin necesidad de que el primero haya sido completamente leído del registro Buffer. Sin embargo si el primer byte permanece sin ser leído hasta el final de la recepción del segundo dato, éste se perderá.

El dato de la Recepción y de la Transmisión se encuentra en el registro SBUF del SFR (espacio de funciones especiales).

4.2 REGISTRO DE CONTROL DEL PUERTO SERIE SCON

El puerto Serie puede ser operado en 4 modos diferentes que son especificados mediante la escritura en los bits SM0 y SM1 del registro de Control del Puerto Serie (Fig. 4.1)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

Fig. 4.1 Registro de control del puerto serie. Bit direccionables.

SM0	SCON.7	Especifica el modo de control del puerto serie.
SM1	SCON.6	Especifica el modo de control del puerto serie.
SM2	SCON.5	Habilita la comunicación del tipo "multiprocesador" utilizado en los modos 2 y 3. En estos modos, si SM2 = 1, RI no es activado si el noveno dato recibido (RB8) es 0. En modo 1, RI no es activado si no se recibe un bit de stop. En el modo 0, SM2 será 0.
REN	SCON.4	Establece la recepción serie, cuando REN = 0 se desactiva la recepción (por software).
TB8	SCON.3	Almacena el noveno bit que será transmitido en los modos 2 y 3.
RB8	SCON.2	Es el noveno bit que fue recibido en los modos 2 y 3. En el modo 1, si SM = 0, RB8 es el bit de stop recibido. En el modo 0 RB8 no es usado.
TI	SCON.1	Bandera de interrupción de la transmisión. Activada por hardware al final del octavo bit en el modo 0, o al principio del bit de stop en los otros modos. Debe ser limpiado por software.
RI	SCON.0	Bandera de interrupción de la recepción. Activada por hardware al final del octavo bit en el modo 0, o al medio tiempo de transmitido el bit de stop en los otros modos. Debe ser limpiado por software.

SM0	SM1	MOD0	ESPECIFICACIÓN	BAUD RATE
0	0	0	Registro de corrimiento	F. Osc. /12
0	1	1	UART 8 bits	Variable.
1	0	2	UART 9 bits	F.Osc./32 o /64
1	1	3	UART 9 bits	Variable.

Descripción del funcionamiento de cada modo.

- MODO 0:** Los datos de recepción o transmisión son enviados mediante 8 corrimientos con una frecuencia de 1/12 de la frecuencia de oscilación.
- MODO 1:** 10 bits son los que se transmiten por la línea (TxD) o se reciben por línea (RxD). Un bit de inicio (Start bit de nivel 0 lógico). En recepción el bit de Fin (Stop), se almacena en RB8 de SCON, si SM2=0. El Baud Rate (frecuencia de transmisión o recepción) es variable.
- MODO 2:** 11 bits son transmitidos (TxD) o recibidos (RxD), un bit de inicio (Start bit de nivel 0 lógico), 8 bits de datos, un noveno bit de datos programable, en la transmisión es TB8, en la recepción es RB8 de SCON, y un bit de Fin (Stop bit). Su Baud Rate es de 1/32 o 1/64 de la frecuencia de oscilación.
- MODO 3:** 11 bits son transmisión (TxD) o recibidos (RxD), en la misma forma que el modo 2, sólo que aquí la frecuencia de transmisión/recepción (Baud Rate) es variable. Se utiliza el Timer 1 para generar el Baud Rate.

En los 4 modos, la transmisión es inicializada cuando SBUF es utilizado como registro destino. La recepción en el modo 0 comienza cuando RI=0, en los otros modos, cuando se detecta el bit de inicio (START) si REN=1.

4.3 MODOS DE CONTROL DEL PUERTO SERIE

Veamos con mayor detalle los 4 modos de control para la transmisión/recepción de datos por el Puerto Serie, comenzando por el Modo 0.

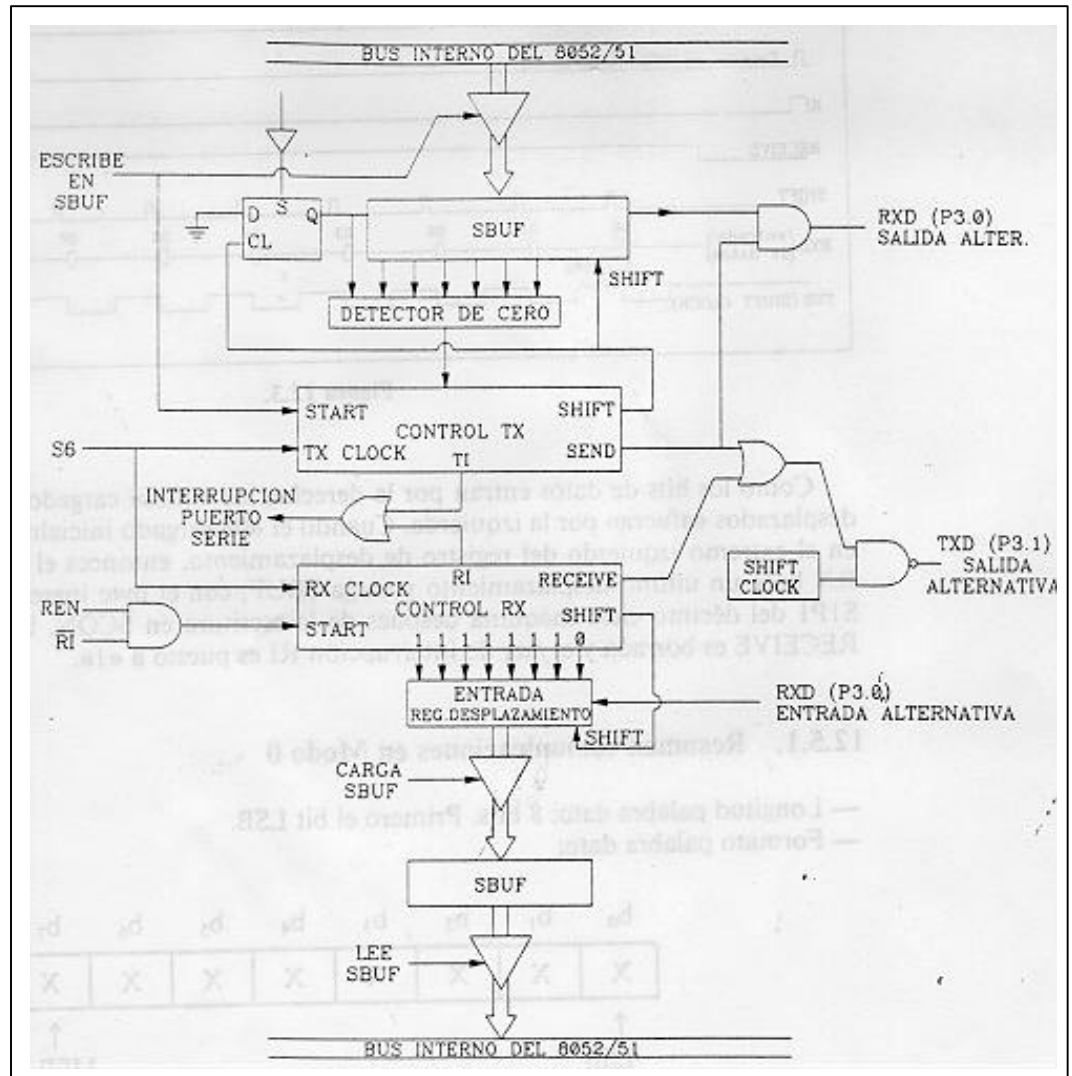


Fig. 4.2 Transmisión serie en Modo 0.

4.3.1 TRANSMISIÓN SERIE UTILIZANDO EL MODO DE CONTROL 0.

Los datos serie que entran o salen a través de RxD o TxD respectivamente, son efectuados mediante 8 desplazamientos, con una frecuencia de transmisión (Baud Rate) de 1/12 de la frecuencia del oscilador. La figura 4.2 muestra el Puerto Serie en Modo 0.

La transmisión es iniciada por cualquier instrucción que utilice SBUF como registro de "Destino", v.gr. MOV SBUF,A . La escritura a SBUF se produce durante la fase 2 de estado 6 "S6P2" del ciclo de máquina, también se carga un 1 dentro de la 1ª 9na. Posición del registro de corrimiento y llama al bloque de control de Tx para que comience a transmitir. Un ciclo completo de la máquina se efectuará entre la escritura en SBUF y la activación de la señal de SEND.

SEND acciona la salida del registro de corrimiento la función alterna de la línea P3.0 de salida. También acciona el reloj de corrimiento "Shift Clock" a la función alterna de la línea P3.1 de salida. El reloj de corrimiento esta en estado bajo, durante los estados S3, S4 y S5 de cada ciclo de máquina y en estado alto durante S6, S1 y S2. En la fase 2 del estado 6, S6P2, de cada ciclo de máquina la señal SEND es activada y el contenido del registro de corrimiento de transmisión es recorrido hacia la derecha una posición (Fig.4.3).

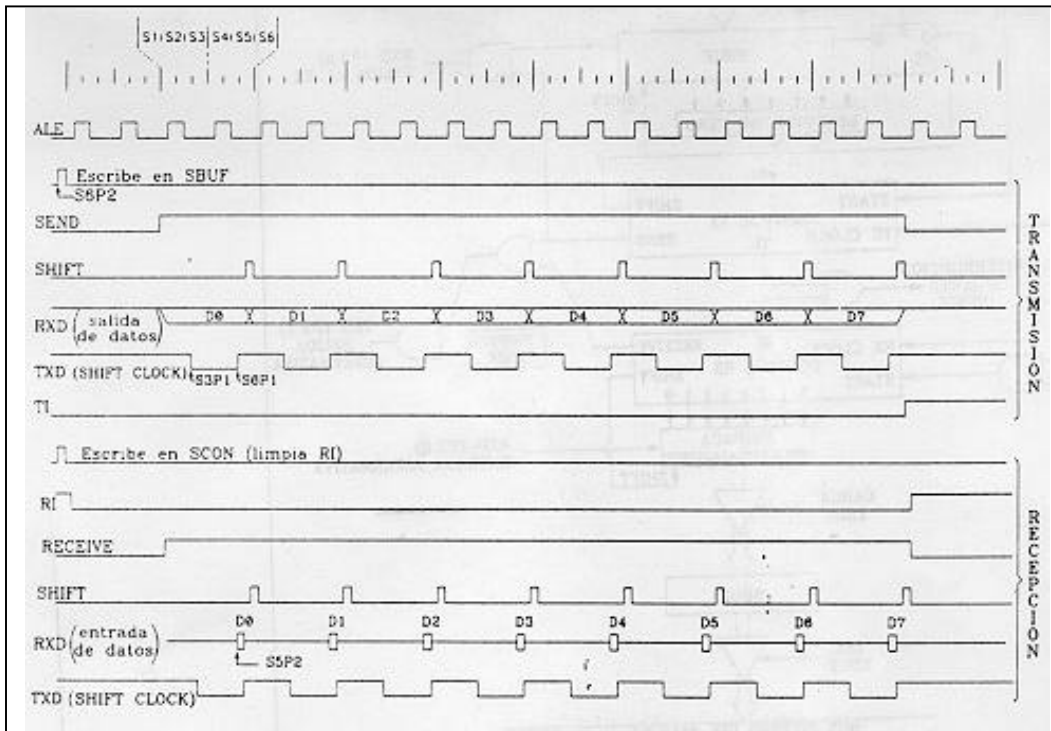


Fig. 4.3 Diagrama de estados del Modo 0.

Cuando los datos van saliendo por la derecha, ceros van entrando por la izquierda. Cuando el bit más significativo MSB, del carácter, está en la posición de salida del registro de corrimiento, el 1 que fue inicialmente cargado en la 9na. Posición está justo a la izquierda del MSB y todas las demás posiciones a la izquierda se encuentran cargadas de ceros. Esta condición establece al bloque de control Tx a realizar un último corrimiento y después desactiva SEND y establece TI. Ambas acciones ocurren en S1P1 del 10mo. Ciclo de la máquina después de la escritura en SBUF.

La recepción es iniciada por la condición REN=1 y RI=0. En S6P2 del siguiente ciclo de máquina, la unidad de Control Rx escribe los bits 11111110 al registro de corrimiento del receptor y en la siguiente fase del reloj activa RECEIVE.

RECEIVE acciona SHIFT CLOCK a la función alterna de la línea P3.1 de salida. SHIFT CLOCK hace las transiciones del S3P1 y S6P2 de cada ciclo de máquina cuando se activa RECEIVE el contenido del registro de corrimiento de recepción, es recorrido a la izquierda una posición. El valor que viene de la derecha es el valor que fue muestreado de la terminal P3.0 y S5P2 del mismo ciclo de máquina.

Como los bits del carácter que se recibe entran por la derecha, 1's saldrán por la izquierda. Cuando el cero que fue inicialmente cargado en la posición más hacia la derecha, llega a la posición más a la izquierda en el registro de corrimiento, esta condición establece en el bloque de Control Rx de realizar un último corrimiento y carga SBUF. En S1P1 del 10mo. Ciclo de máquina (después de haber escrito RI=0 en SCON), RECEIVE es limpiado y RI establecido.

4.3.1 TRANSMISIÓN SERIAL UTILIZANDO EL MODO DE CONTROL 1.

En este modo, 10 bits son transmitidos (por TxD) o recibidos (por RxD). Un bit de inicio "START" en nivel 0, 8 bits de datos (primero el LSB), y un bit de paro "STOP" en nivel 1. En la recepción, el bit de STOP viene en RB8 de SCON. El Baud Rate es determinado por la frecuencia de los "overflows" del Timer 1. La figura 4.4 muestra un diagrama del Puerto Serie en MODO 1.

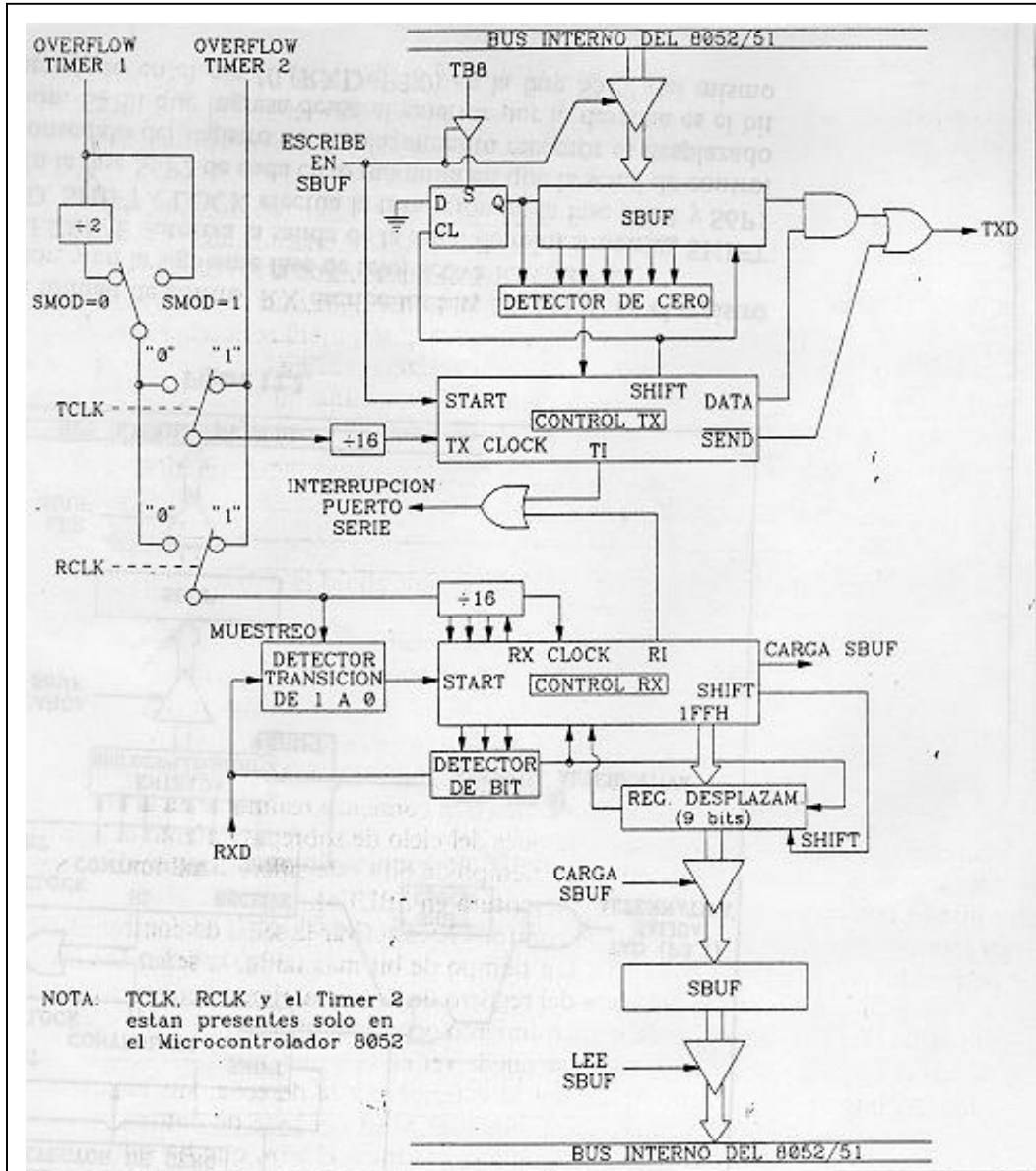


Fig. 4.4 Transmisión serial en Modo 1.

La transmisión es iniciada por cualquier instrucción que escriba en el registro SBUF. La señal de escritura en SBUF, también carga un 1 dentro de la 9na. Posición del registro de corrimiento de transmisión y establece en la unidad de Control Tx una demanda de transmisión. La transmisión comienza en S1P1 del ciclo de máquina siguiente, en el primer pulso proporcionado por el circuito divisor ($\%16$ de la frecuencia dada por el Timer 1). Es decir, los datos de entrada están sincronizados por el circuito divisor $\%16$, y no por la escritura en SBUF.

La transmisión comienza con la activación de la señal $\overline{\text{SEND}}$, la cual introduce el bit de START en TxD. Un periodo ($1/16$ de la frecuencia dada por el Timer 1) más tarde la señal DATA es activada, la cual permite la salida de los bits que se transmitirán por TxD. El primer pulso de corrimiento ocurre un período después de esto.

La recepción es inicializada por una transición de 1 a 0 en RxD. La línea RxD es muestreada a una frecuencia de 16 veces la frecuencia del "Baud Rate" que ha sido establecida. Cuando un transiente es detectado el divisor entre 16 es inmediatamente restablecido y 1FFH es escrito en la entrada del registro de corrimiento.

Cada tiempo que dura el bit recibido, es dividido en 16 períodos. Durante los períodos 7mo. 8vo. 9mo., el valor del bit muestreado, el valor aceptado, es el que se obtuvo en las últimas dos muestras. Esto se hace para eliminar ruido. Si el valor aceptado del primer bit es 1, el circuito receptor es restablecido y la unidad receptora espera otro transiente de inicio. Si el bit de comienzo es correcto, es decir 0, la recepción continúa.

Cuando RI=0, SM2=0 o el bit de stop es = 1, el bit de stop se introducirá en RB8, los 8 bits de datos en SBUF y RI se activa. A partir de este momento la unidad receptora espera otra transición negativa en la línea RxD.

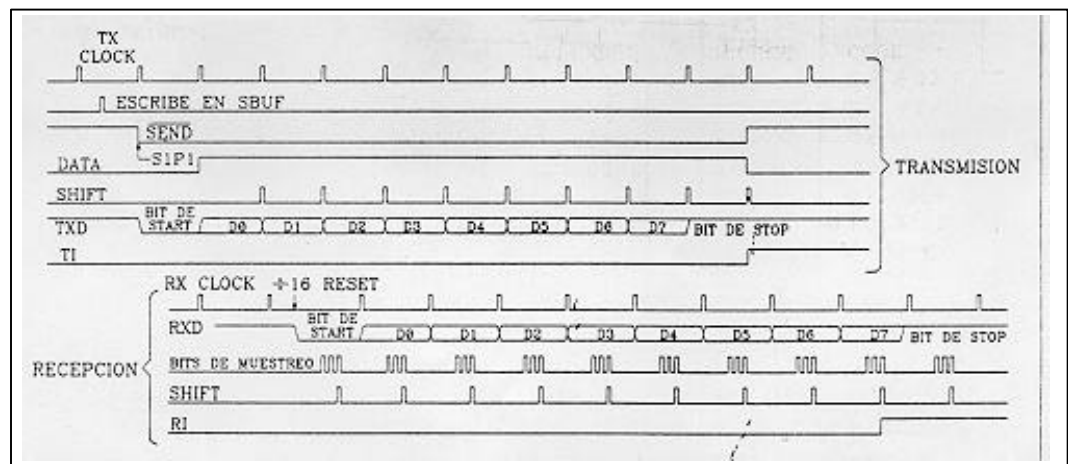


Fig. 4.5 Diagrama de estados del Modo 1.

4.3.1 TRANSMISIÓN SERIE UTILIZANDO LOS MODOS DE CONTROL 2 Y 3

11 Bits son transmitidos (por la línea TxD) o recibidos (por la línea RxD); un bit de inicio (0), 8 bits de datos, un bit programable (9no. Bit) y un bit de stop (1).

En la transmisión el 9no. Bit transmitido es TB8, el cual se le puede asignar al valor 0 o 1. En la recepción del 9no. Bit recibido en RB8 de SCON.

En el modo 2 el “Baud Rate” es programable a 1/32 o 1/64 de la frecuencia de oscilación. (Fig. 4.6).

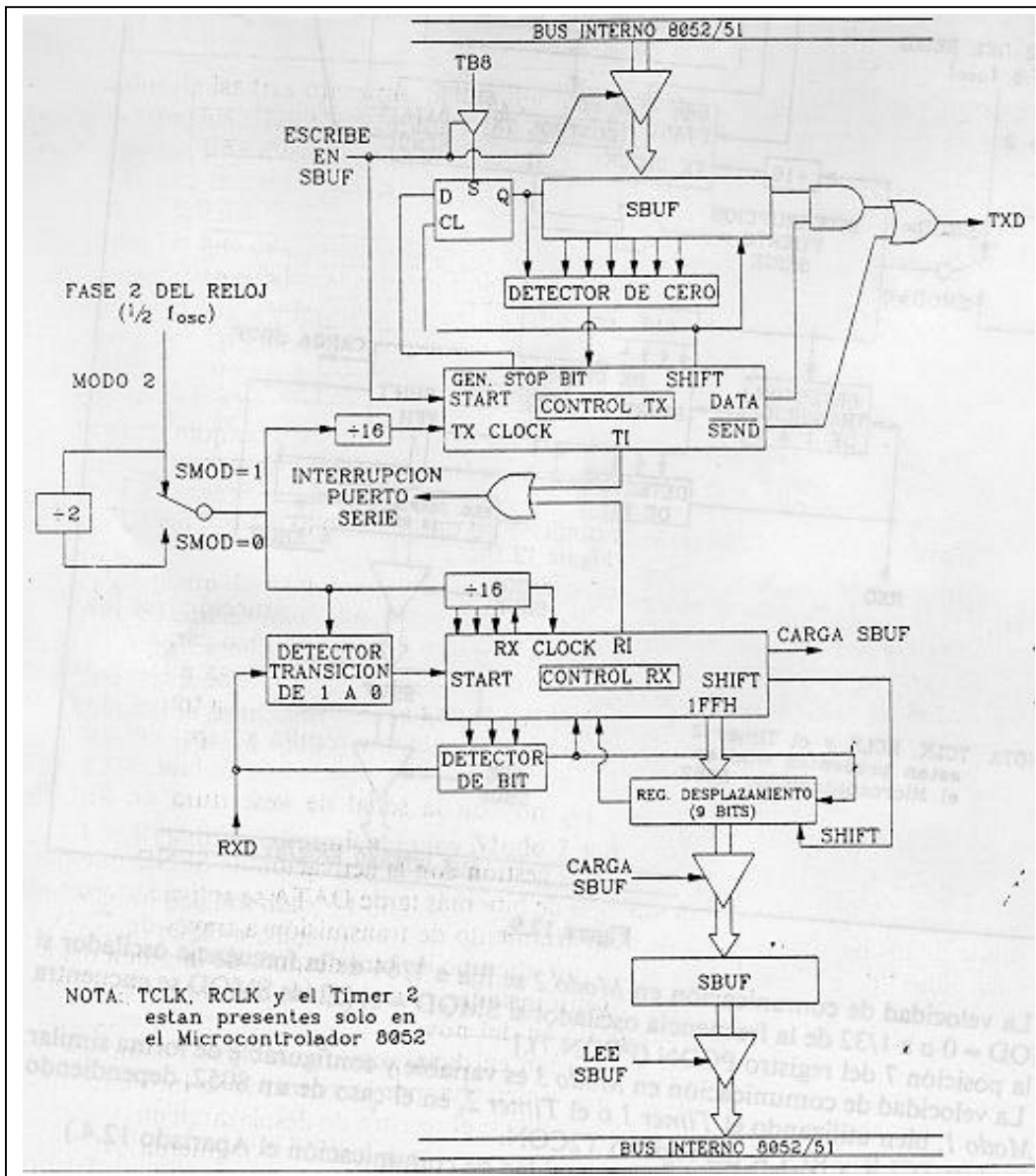


Fig. 4.6 Transmisión en Modo 2.

En el modo3 el "Baud Rate" es variable y es generado por el Timer 1. (Fig. 4.7)

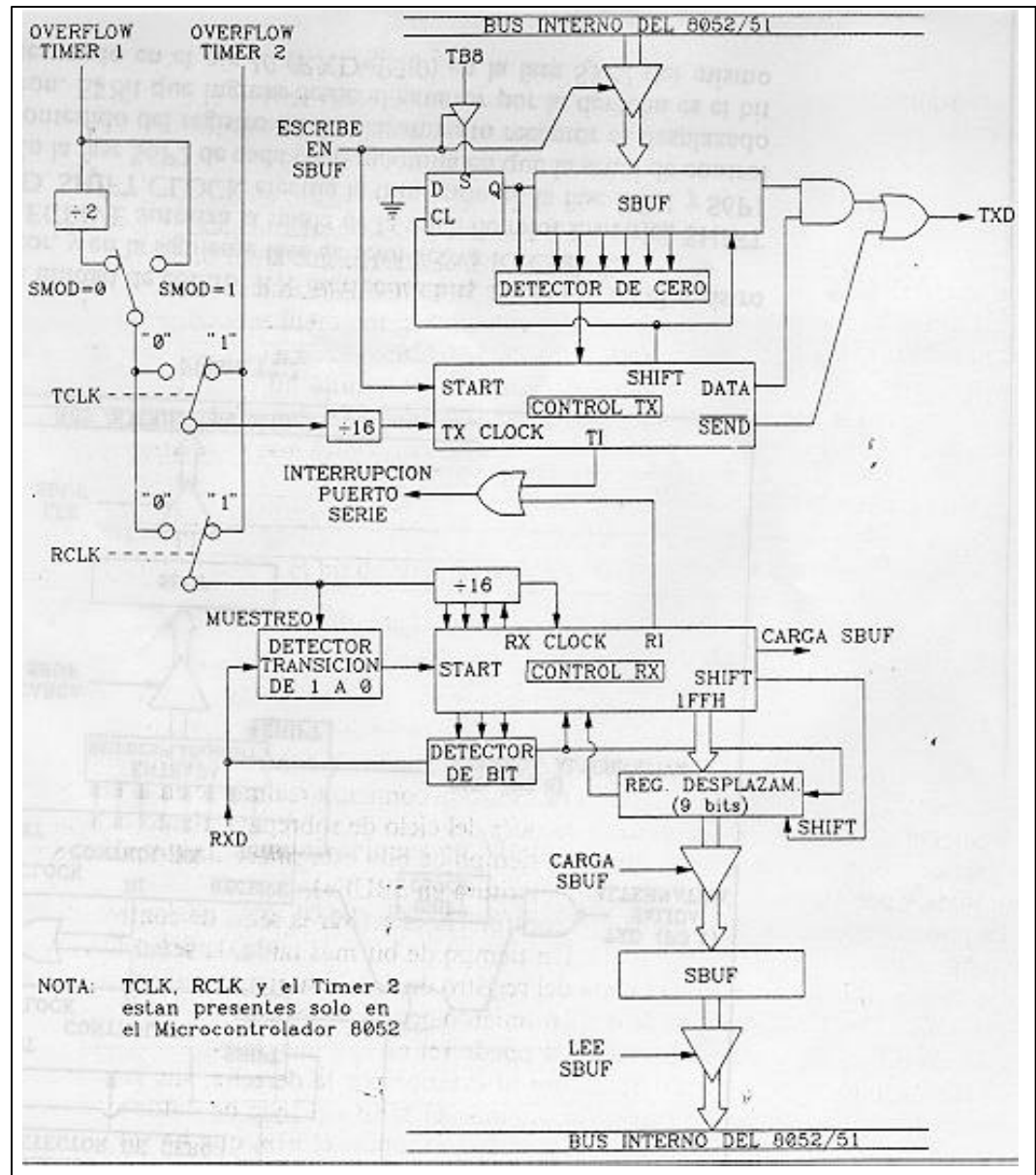


Fig. 4.7 Transmisión Serie en Modo 3

La transmisión se inicia cuando se escribe en SBUF, en este momento también TB8 se carga en el 9no. Bit del registro de corrimiento.

La transmisión comienza con la activación de SEND la cual pone el bit de inicio en la línea TxD. En un período (tiempo que dura un bit transmitiéndose o recibiendo), mas tarde, DATA es activada, lo cual habilita el bit de salida del registro de transmisión a la línea TxD.

El primer pulso de corrimiento ocurre un período mas tarde. El primer pulso de reloj pone un 1 (bit de stop), en la posición del 9no. bit. Después de esto, solo ceros sean introducidos por la izquierda mientras que los bits de datos salen por la derecha.

Cuando TB8 está en la posición de salida del registro de corrimiento, a la izquierda de éste se encontrará el bit de STOP y a la izquierda del bit de STOP todos los demás bits del registro serán igual a ceros. Esta condición establece la unidad de control TX la cual será un último corrimiento y entonces desactivará la señal de SEND y establecerá TI.

Esto ocurre en el 11vo. Período después de la escritura en SBUF (Fig. 4.8).

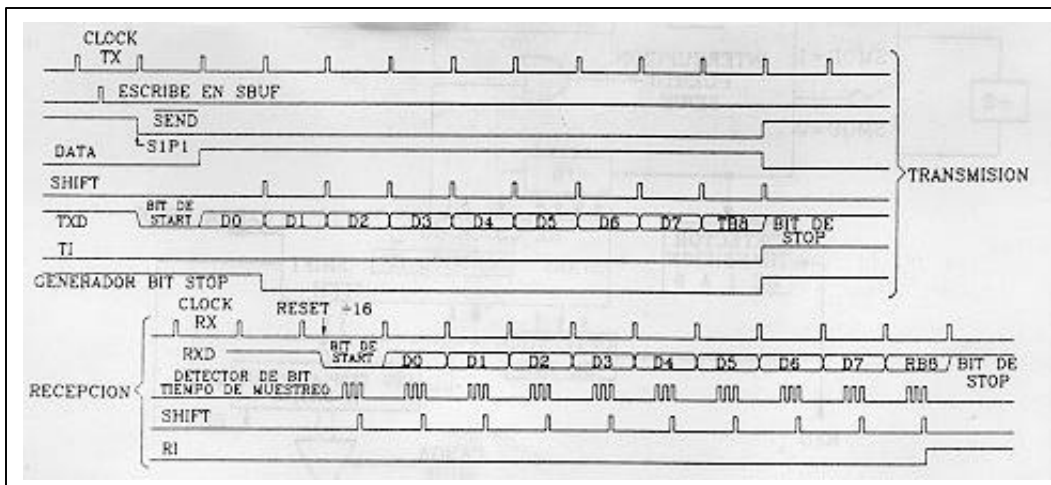


Fig. 4.8 Diagrama de estados de los Modos 2 y 3

La recepción es iniciada por la detección de un transiente negativo (de 1 a 0) en la línea RxD, la cual es muestreada a una frecuencia de 16 veces la frecuencia de "Baud Rate" que ha sido establecida. Cuando un transiente es detectado el divisor entre 16 es inmediatamente reestablecido y 1FFH es escrito en la entrada del registro de corrimiento.

Cada tiempo que dura el bit recibido, es dividido en 16 períodos. Durante los períodos 7mo., 8vo. Y 9no, el valor del bit es muestreado, el valor aceptado, es el que se obtuvo en las últimas dos muestras. Esto se hace para eliminar ruido. Si el valor aceptado del primer bit es 1, el circuito receptor es restablecido y la unidad receptora espera otro transiente de inicio. Si el bit de comienzo es correcto, es decir 0, continúa la recepción.

Cuando RI=0 y SM2=0 o el 9no. bit de datos = 1, el 9no. bit de datos se introducirá en RB8, los 8 bits de datos en SBUF y RI se activa. A partir de este momento la unidad receptora espera otra transición negativa en la línea RxD.

5.1 MODOS DE DIRECCIONAMIENTO.

Los modos de direccionamiento que la familia del MCS-51 trabaja son los siguientes:

5.2 DIRECCIONAMIENTO DIRECTO.

En este direccionamiento el operando es especificado por una dirección de 8 bits en la instrucción. Solamente los datos de la RAM interna y los del campo del SFR pueden ser directamente direccionados.

EJEMPLO :

ADD A,7FH ;El Acumulador es sumado al dato que se encuentra en la dirección 7FH de la RAM interna y el resultado será almacenado en el Acumulador.

MOV A,2EH ;El Acumulador es cargado con el dato que se encuentra en la dirección 2EH de la memoria RAM interna.

MOV 3DH,4EH;El contenido de la dirección 3DH es cargado con el dato que se encuentra en la dirección 4EH.

5.3 DIRECCIONAMIENTO INDIRECTO.

En este direccionamiento se utiliza un registro en el cual se encuentra la dirección del operando. Toda la memoria RAM interna y externa puede ser direccionada indirectamente.

Los registros de direcciones de 8 bits, pueden ser los registros R0 y R1 del banco de registros, o el SP.

El registro de dirección de 16 bits puede ser solamente el registro DPTR.

EJEMPLO :

ADD A,@R0 ;El Acumulador es sumado con el contenido de la dirección que esta apuntando R0.

MOV A,@R0 ;El Acumulador es cargado con el dato que se encuentra en la dirección apuntada por R0.

MOVX A,@DPTR ;El Acumulador es cargado con el dato que se encuentra en la dirección apuntada por el DPTR.

MOVX @DPTR,A ;El contenido del acumulador es guardado en la dirección apuntada por el D

5.4 DIRECCIONAMIENTO INMEDIATO.

El valor de una constante sigue al código de operación en el programa.

EJEMPLO :

MOV A,#64H ;El acumulador es cargado con el dato 64H inmediatamente.

ADD A,#120 ;El acumulador es sumado al número decimal 120 y el resultado se almacena en el acumulador.

MOV DPTR,#1245H ;El DPTR es cargado con el dato 1245H en forma inmediata.

5.5 DIRECCIONAMIENTO INDEXADO.

Solamente la memoria del programa puede ser accesada mediante este modo de direccionamiento y sólo en lecturas.

Este modo de direccionamiento es utilizado en las lecturas de tablas de la memoria del programa o datos que se encuentran como constantes.

Un registro de 16 bits (el DPTR o el PC), apunta la base de la tabla y mediante el Acumulador se establece el número de la entrada de la tabla. La dirección de la entrada de la tabla en la memoria del programa está formada por la suma del Acumulador y el Apuntador de Base (DPTR o PC).

Otro tipo de Direccionamiento indexado, es usando la instrucción "Salto de casillero". En este caso la dirección del destino el salto es calculada como la suma del apuntador de base más el Acumulador.

EJEMPLO :

MOVC A,@A+DPTR;Mueve una constante que se encuentra en la memoria del programa. El Acumulador es cargado con el dato que se encuentra apuntado por la dirección formada por la suma del Acumulador A y el Apuntador de Datos.

MOVC A,@A+PC ;El Acumulador es cargado con dato que se encuentra en la dirección formada por la suma del mismo Acumulador A y el Contador del Programa (PC).

5.6 DIRECCIONAMIENTO POR REGISTRO

Los 8 registros pueden ser accedidos mediante ciertas instrucciones que simplifican sus códigos de operación (opcode) y en la mayoría de los casos son más rápidas.

Existen 4 bancos de registros, cada banco contiene los 8 registros (R0 a R7). Estos bancos pueden ser accedidos mediante los bits 3 y 4 del PSW.

EJEMPLO :

ADD A,R7 ;El acumulador es cargado con el resultado de suma del Acumulador y el contenido del registro R7.

DEC R0 ;Decrementa el registro R0.

5.7 TRANSFERENCIA DE DATOS.

5.7.1 RAM INTERNA.

Para poder mover datos de la memoria interna RAM o SFR, existen 8 instrucciones principales:

MOV A, <fuente>	;	A = <fuente>
MOV <destino>,A	;	<destino> = A
MOV <destino>,<fuente>	;	<destino> = <fuente>
MOV DPTR,#dato 16 bits	;	DPTR = constante de 16 bits inmediata.
PUSH <fuente>	;	INC SP → <@SP> ← (fuente)
POP <destino>	;	(destino) ← <@SP> ; DEC SP
XCH A, <byte>	;	ACC y <byte> intercambia sus datos.
XCHD A, @Ri	;	ACC y @Ri intercambian nibble bajo.

La instrucción **MOV <dest>,<fuente>** ; permite transferir datos de la memoria interna RAM y del SFR sin pasar a través del Acumulador, mientras que las instrucciones **MOV A,<fuente>** y **MOV <dest>,A** utilizan al Acumulador para el movimiento de datos dentro de la memoria interna RAM.

La instrucción **PUSH**, primeramente incrementa el Stack Pointer (SP), y después guarda el dato dentro de la localidad de memoria apuntada por el Stack.

La instrucción **POP**, primero toma el dato de la memoria y después decrementa el SP. Cabe señalar que si el SP apunta en la localidad 7FH en los dispositivos que no tienen implementados los 128 bytes altos de memoria interna RAM, NO SE PODRÁN SEGUIR ALMACENANDO DATOS, DEBIDO A QUE ESTOS SE PERDERÍAN.

Arriba de la dirección 80H los datos almacenados con la instrucción PUSH se pierden y los datos tomados con POP, son indeterminados.

La instrucción **XCH A, <dirección>** permite al Acumulador y al dato apuntado por la dirección de intercambiarse entre si. La instrucción **XCHD A, @R0** intercambia los 4 bits menos significativos (low nibble), de Acumulador con los 4 bits menos significativos del dato apuntado por el registro R0.

5.7.2 RAM EXTERNA.

Las instrucciones para la transferencia de datos en la memoria RAM externa, son 4 básicamente:

MOVX A, @Ri	;	A	←	<@Ri>
MOVX @Ri, A	;	<@Ri>	←	A
MOVX A, @DPTR	;	A	←	<@DPTR>
MOVX @DPTR, A	;	<@DPTR>	←	A

Las habilitaciones de lectura y escritura en RAM externa son activadas solamente durante la activación de una instrucción **MOVX**.

5.7.3 MOVIMIENTO DE TABLAS LOCALIZADAS EN LA MEMORIA DEL PROGRAMA

La mayoría de las veces dentro de nuestro programa principal tenemos algunas constantes, como son palabras de control, contadores, banderas iniciales, tablas de datos, etc. Por tal motivo, el 8051 utiliza las siguientes instrucciones solamente en lectura (ya que en escritura no tendrá caso por ser una memoria EPROM o ROM),

MOVC A, @A + DPTR	;	A ← <@A + DPTR>	Dirección en memoria de programa
MOVC A, @A + PC	;	A ← <@A + PC>	Dirección en memoria de programa.

5.8 INSTRUCCIONES BOOLEANAS.

El 8051 contiene un completo procesador Booleano (por bits), el cual permite ejecutar instrucciones de movimiento, limpieza, establecimiento, complementación de un sólo bit, y operaciones de AND y OR entre bits.

ANL C,bit	;	$C \leftarrow C \text{ .AND. bit}$
ANL C,/bit	;	$C \leftarrow C \text{ .AND. .NOT. bit}$
ORL C,bit	;	$C \leftarrow C \text{ .OR. bit}$
ORL C,/bit	;	$C \leftarrow C \text{ .OR. .NOT. bit}$
MOV C,bit	;	$C \leftarrow \text{bit}$
MOV bit,C	;	$\text{bit} \leftarrow C$
CLR C	;	$C \leftarrow 0$
CLR bit	;	$\text{bit} \leftarrow 0$
SETB C	;	$C \leftarrow 1$
SETB bit	;	$\text{bit} \leftarrow 1$
CPL C	;	$C \leftarrow \text{.NOT. } C$
CPL bit	;	$\text{bit} \leftarrow \text{.NOT. bit}$
IC	.	Principio de C = 1

EJEMPLO :

```

CONTROL      DATA      20H
BANDERA      BIT         CONTROL,7
ORG 100H
MOV C,BANDERA
MOV P1.0,C

```

La etiqueta **BANDERA** representa, un bit direccionable de los 128 bits que se encuentran en la memoria baja RAM interna (desde la dirección 20H a la 2FH), o de los 128 bits de algunos registros del espacio SFR. En este caso particular, pertenece al bit 7 de la palabra CONTROL que se encuentra en la dirección 20H de la RAM interna, la línea 0 del Puerto 1 es establecida o limpiada dependiendo del valor de BANDERA.

Otro ejemplo, sería la implementación de la función Booleana OR Exclusivo por programación. Partiremos de la función siguiente:

$C = \text{bit 1} \text{ .XOR. bit 2}$	C	bit2	bit1
	0	0	0
	1	0	1
	1	1	0
	0	1	1

Como podemos apreciar si el bit 2 es igual a 0 el bit 1 tiene el mismo valor de C, y si el bit 2 es igual a 1, el bit 1 tiene el valor negado de C, de tal forma que su implementación sería:

```

MOV C,bit 1      ;El carry C = bit 1
JNB bit 2,conti:  ;Es bit 2 = 0?, sino salta
CPL C            ;Se complementa C=bit 1

```

conti: RET ;Regresa al programa

5.9 INSTRUCCIONES DE SALTO.

5.9.1 SALTOS CONDICIONADOS.

El juego de instrucciones con saltos que están condicionados a la activación o desactivación de algunas de las banderas del PSW son las siguientes:

JZ rel	;	Salta si A = 0
JNZ rel	;	Salta si A ≠ 0
DJNZ <byte>, rel	;	Decrementa y salta si no es igual a 0
CJNE A,<byte>,rel	;	Salta si A ≠ <byte>
CJNE <byte>,#dato,rel	;	Salta si <byte> ≠ #dato

DJNZ <byte>, rel ;Decrementa y salta si no es igual a 0

Como se vió anteriormente, en el PSW no se tiene la bandera Z, que prueba el CERO, más sin embargo, aquí se muestran las instrucciones JZ y JNZ, las cuales prueban todo el dato del Acumulador para esta condición.

La instrucción DJNZ (decrementa y salta sino es igual a cero), ejecuta un lazo N veces, hasta que un "contador" es igual a cero.

EJEMPLO :

MOV CONTADOR,#10 ; N = 10

LAZO: (comienzo del conjunto de instrucciones)

DJNZ CONTADOR,LAZO
(continúa)

De la misma forma podemos apreciar la instrucción CJNE, la cual compara y salta si el BYTE (registro, Acumulador o dirección) es igual al dato comparado. El salto es relativo.

En los saltos relativos el rango del desplazamiento es de -128 a +127 bytes con respecto al primer byte siguiente a la instrucción.

5.9.2 SALTOS INCONDICIONADOS.

Las instrucciones que permiten hacer los saltos incondicionados son las siguientes:

JMP dirección	;	Salta a la dirección.
JMP @A + DPTR	;	Salta a la dirección A + DPTR
CALL dirección	;	Llama a la subrutina "dirección".
RET	;	Regreso a la subrutina.
RETI	;	Regreso de la interrupción.
NOP	;	Sin operación.

En la tabla se muestra una sola instrucción de Salto JMPdirec pero en realidad existen 3 tipos de saltos; **SJMP**, **LJMP** y **AJMP** los cuales difieren en el formato de la dirección de destino.

JMP es un Mnemónico genérico el cual puede ser usado (en algunos compiladores) durante la programación sin tener en cuenta de ue forma el salto será codificado.

La instrucción **SJMP**, codifica la dirección como un desplazamiento relativo. La instrucción es de 2 bytes, es decir el PCODE y el salto relativo. Su límite es de -128 a +127 bytes a partir de instrucción siguiente al SJMP.

La instrucción **LJMP**, codifica la dirección como una constante de 16 bits. La instrucción es de 3 bytes de largo, el OPCODE y la dirección en 2 bytes. Este salto puede desplazarse en todos los 64K de memoria.

La instrucción **AJMP** codifica la dirección como una constante de 11 bits. La instrucción es de 2 bytes de largo, el OPCODE el cual contiene 3 bits de los 11 bits de direcciones y el otro byte de 8 bits de direcciones. Cuando la instrucción es ejecutada esos 11 bits son simplemente sustituidos por los 11 bits más bajos en el PC. Los 5 bits mas altos permanecen sin alterarse. Estos saltos pueden ser hasta de 2K.

En cualquiera de los casos el programador especifica la dirección al ensamblador de la misma manera; como una etiqueta o constante de 16 bits. El ensamblador deberá poner la dirección dentro del formato correcto de forma automática.

La instrucción **RETI** es usada para el regreso de una rutina de servicio de interrupción. la diferencia con la instrucción RET es que RETI llama al sistema de control de interrupción mientras que la interrupción está en proceso. Si no existe ninguna interrupción en proceso, entonces la instrucción RETI es igual a RET.

ACALL dirección 11 (llamada absoluta)
--

ACALL llama incondicionalmente a una subrutina localizada en la dirección indicada. Durante esta instrucción se realizan los siguientes eventos: El contenido del PC se incrementa dos veces y apunta la dirección de la siguiente instrucción; el apuntador de apilamiento (Stack Pointer) se incrementa una vez, introduciendo el byte bajo del PC; incrementa nuevamente el SP para introducir el byte alto del PC; por último, el PC es cargado con el contenido de la **DIRECCIÓN DESTINO**. La ejecución de las instrucciones de la subrutina comienza en ésta dirección, hasta que encuentre la instrucción RET, la cual restablece el PC que habrá sido almacenado en el SP, continuando nuevamente con el programa inicial.

La **DIRECCIÓN DESTINO** es obtenida por concatenación sucesiva de los cinco bits de más alto orden del PC incrementado; más los 3 bits de más alto orden y el byte bajo del código de operación (OPCODE). Colocados respectivamente en ese orden a partir del byte alto, para formar la nueva dirección del PC. En otras palabras, las operaciones que se llevan a cabo son las siguientes:

OPERACIÓN: ACALL

(PC)	←	(PC)+2
(SP)	←	(SP)+1
((SP))	←	(PC ₇₋₀)
(SP)	←	(SP)+1
((SP))	←	(PC ₁₅₋₈)

El PC final es creado por:

(PC ₁₅₋₁₁)	←	(PC ₁₅₋₁₁ INCREMENTADO)
(PC ₁₀₋₈)	←	(OP CODE ₁₅₋₁₃) # de la pág.
(PC ₇₋₀)	←	(OP CODE ₇₋₀) direc. en la pág.

A continuación se presenta un ejemplo detallado de ésta instrucción.

EJEMPLO:

```

                                ORG 00H

0000 5100      ACALL LAZO1 ; 0101 0001 0000 0000
0002 E8      LAZO2 : MOV A, R0      pág.2      Direc. pág.
0003 C6              XCH A, @R0
0004 E9              MOV A, R1
0005 22              RET
0006 02 0200      LJMP 0200H

                                ORG 0200H

0200 F8 LAZO1: MOV R0, A
0201 22              RET
0202 1102      ACALL LAZO2; 000 1 0001 0000 0010
0204              END              pág.0      Direc. Pág.

```

En el ejemplo anterior se muestra en la dirección 0202 el llamado a la subrutina LAZO2, con el código de operación 1102, por lo tanto, la dirección a la cual debe de saltar se forma de la siguiente manera:

$$\begin{array}{rcl}
 \text{PC}_{\text{INC}} = 0204 & & \text{OPCODE} = 1102 \\
 \hline
 \begin{array}{c} 0000\ 0\ 010\ 0000\ 0100\ \text{B} \\ \text{A} \end{array} & \begin{array}{c} 000\ 1\ 0001\ 0000\ 0010\ \text{B} \\ \text{B} \qquad \qquad \text{C} \end{array} & \\
 \hline
 \text{PC}_{\text{FINAL}} = \overbrace{0000\ 0\ 000}^{\text{A}} \overbrace{0000\ 0010}^{\text{B}} \text{B} = 0002\text{H} & &
 \end{array}$$

La dirección con la cual se carga el PC es 0002. Cabe aclarar que los bits 8 al 12 del OPCODE permanecen siempre constantes (10001), en cualquier llamado del tipo ACALL.

Como puede observarse también, el valor del nibble de más alto orden del PC INC y el PC FINAL es el mismo, ello nos limita la longitud del salto.

El destino debe por lo tanto estar dentro del mismo block de 2K de memoria del programa donde se encuentra la instrucción ACALL.

ADD suma la variable (SRC-BYTE) indicada y el Acumulador, dejando el resultado en el Acumulador. Las banderas de acarreo y acarreo auxiliar son establecidas, si hay un acarreo hacia afuera del bit 7 o del bit 3, de otro modo son limpiadas.

**ADD A,<SRC –
BYTE>**

Cuando se suman enteros sin signo, la bandera de acarreo indica que ocurrirá un sobreflujo.

OV se establece si hay un acarreo del bit 6 al 7 pero sin existir acarreo del bit 7 hacia afuera, o un acarreo hacia afuera del bit 7 pero no del bit 6 al 7; de otro modo OV es limpiada. Cuando se suman enteros signados, el OV indica un número negativo producido como la suma de dos operandos positivos, o uno positivo producido por la suma de dos operandos negativos.

Cuatro modos de fuentes de operandos dirigidos están permitidos: registro, directo, registro indirecto, o inmediato.

EJEMPLO:

El Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B). La instrucción, ADD A,R0, dejaría 6DH (01101101B) en el acumulador con la bandera del acarreo auxiliar limpiada y las banderas de acarreo y sobreflujo establecidas.

ADD A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	1 r r r
---------	---------

OPERACIÓN: **ADD** $(A) \leftarrow (A) + (Rn)$ **ADD A, @Ri**

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	1 1 1 i
---------	---------

OPERACIÓN: **ADD** $(A) \leftarrow (A) + ((Ri))$ **EJEMPLO:**

RAM INTERNA

A=21	R0=10
0000 28	ADD A,R0
0001 26	ADD A,@R0

0000
0010

10
12

En la primera instrucción se suman el contenido de A más el contenido de R0, es decir : $21 + 10 = 31$, el resultado es almacenado en el acumulador, ahora $A=31$.

En la segunda instrucción se suman el contenido del acumulador y el contenido de la localidad de memoria (dirección) apuntada por el registro R0, que en este caso es la 10, es decir: $31 + 12 = 43$, el resultado se almacena en el acumulador. Al término de las instrucciones el estado del programa sería $A=43$, $C=0$, $AC=0$, $OV=0$.

ADD A, directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **ADD**

(A) ← (A)+(directo) **ADD A, #dato**

ADD A, #dato

BYTES: 2CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **ADD**

(A) ← (A) + #dato

PSW7	PSW6	PSW2	BANDERAS QUE SE ACTIVAN
CY	CY	OV	

EJEMPLO:

	ORG 00H	RAM INTERNA
0000 7421	MOV A,#21H	R0 R1-----R7
0002 7810	MOV R0,#10H; A←21H	0000 10
0004 2500	ADD A, 00H ; A←31H	0009
0006 751012	MOV 10H,#12H	0010 12
0009 2412	ADD A,#12 ; A←43H	0018
000B	END	

ADDC simultáneamente suma el byte variable indicado, la bandera de acarreo y el contenido del Acumulador, dejando el resultado en el Acumulador. Las banderas de acarreo-auxiliar y la de acarreo están establecidas, respectivamente, si hay acarreo hacia el exterior del bit 3 ó del bit 7. De otra forma ellas están limpias.

Cuando se suman enteros signados negativamente la bandera de acarreo, indica que ocurrió un sobreflujo. OV se establece, si hay un acarreo del bit 6 al bit 7 pero no del bit 7

**ADDC A, <src-byte>
sumar con acarreo
(carry).**

hacia afuera, o del bit 7 hacia afuera pero no del bit 6 al bit 7. de otro modo OV está limpio. Cuando se suman enteros signados, el OV indica un número negativo producido como la suma de dos operandos positivos ó un número positivo producido por la suma de dos operandos negativos. Cuatro modos de direccionamiento de operandos fuente están permitidos: registro directo, registro indirecto ó inmediato.

EJEMPLO:

El Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B) con la bandera de acarreo fija. La instrucción, **ADDC A, R0**, dejará 6EH (01101110B) en el acumulador con AC limpiado y las banderas de acarreo y OV establecidas.

ADDC A, Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0011	1rrr
------	------

OPERACIÓN: **ADDC**

(A) ← (A)+(C)+(Rn)

EJEMPLO:

A = C3	R0=AA C=1	C3 = 11000011
		AA= 10101010
		CT= <u> 1 </u>
		1 01101110

ADDC A,R0	⇒	A ← 6E	CY=1
		AC ← 0	OV=1

ADDC A, @Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0011	011i
------	------

OPERACIÓN: **ADDC**

$$(A) \leftarrow (A)+(C)+((Ri))$$

ADDC A, directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **ADDC**

$$(A) \leftarrow (A)+(C)+(\text{directo})$$

ADDC A, #dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 1 0 1	dato inmediato
---------	---------	----------------

OPERACIÓN: **ADDC**

$$(A) \leftarrow (A)+(C)+(\#dato)$$

AJMP transfiere la ejecución del programa a la dirección formada por la concatenación de los 5 bits de más alto orden del PC incrementado, más los bits del 7 al 5 del código de operación y el segundo byte que forma ésta instrucción AJMP. El destino debe por lo tanto estar dentro del mismo block de 2K de memoria del programa donde se encuentra la instrucción **AJMP**.

AJMP dirección 11
(salto absoluto)

EJEMPLO :

La etiqueta "SALTDR" está en la dirección del programa 0123H. La instrucción, **AJMP SALTDR** está en la localización 0345H y cargará el PC con 0123H.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

a10 a9 a8 0	0 0 0 1	a7	a6	a5	a4	a3	a1	a1
-------------	---------	----	----	----	----	----	----	----

OPERACIÓN: **AJMP**

$$(PC) \leftarrow (PC)+2$$

El PC final es creado por:

$$\begin{aligned} (PC_{15-11}) &\leftarrow (PC_{15-11} \text{ INCREMENTADO}) \\ (PC_{10-8}) &\leftarrow (OP \text{ CODE }_{15-13}) \# \text{ de la pág.} \\ (PC_{7-0}) &\leftarrow (OP \text{ CODE }_{7-0}) \text{ direc. en la pág.} \end{aligned}$$

EJEMPLO :

```

                                ORG 00H
0000 C100    AJMP ETA ;ETA =0600H = 0000 0 110 0000 0000B
0600 7450    ETA: MOV A,#50
END

```

ANL <dest.>,<fuente>

ANL Ejecuta la operación lógica AND, bit a bit entre las variables indicadas y guarda los resultados en la variable destino. Las banderas no son afectadas.

Esta operación lógica permite 6 combinaciones de direccionamientos. Cuando el destino es el acumulador, los direccionamientos pueden ser por registro, directo, registro-indirecto, ó inmediato. Cuando el destino es una dirección, la fuente puede ser el acumulador o el dato inmediato.

NOTA :

Cuando ésta instrucción es usada para modificar un puerto de salida, el valor usado como dato del puerto será leído del latch del puerto NO DE LA PATA DEL MISMO.

EJEMPLO :

Si el acumulador contiene 0C3H (11000011B) y el registro contiene 55H (01010101B) luego la instrucción, **ANL A,R0** dejará 41H (01000001B) en el acumulador. Cuando el destino es un byte direccionado directamente, ésta instrucción limpiará combinaciones de bits en cualquier localidad RAM o registro.

El byte máscara que determina el patrón de bits que serán limpiados puede ser una constante contenida en la instrucción o un valor calculado en el acumulador. La instrucción, **ANL P1,#01110011B**, limpiará los bits 7, 3, y 2 del latch del puerto de salida 1.

ANL A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	1 r r r
---------	---------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge (R_n)$$
ANL A,Directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge (\text{directo})$$
ANL A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 1 1 i
---------	---------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge ((R_i))$$
ANL A,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: ANL

$$(A) \leftarrow (A) \wedge \#dato$$
ANL directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 0 1 0
---------	---------

dirección directa

OPERACIÓN: **ANL**

$$(\text{directo}) \leftarrow (\text{directo}) \wedge (A)$$
ANL directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 0 1 1
---------	---------

dirección directa	dato inmediato
-------------------	----------------

OPERACIÓN: **ANL**

$$(\text{directo}) \leftarrow (\text{directo}) \wedge \# \text{dato}$$
EJEMPLO :

A=C3 R0=55

ANL A,R0 ; 11000011 = A
 01010101 = R0
 01000001 = A x R0 = 41

ANL P1,#01110011B ; si P1 es puerto de lectura

P1 = 11111111

P1 = 01110011

Limpia los bits 7, 3 y 2 del puerto de salida 1.

ANL C, bit AND lógico para bits variables
--

Si el valor Booleano del bit fuente es un 0 lógico, limpia la bandera de acarreo; de otro modo deja la bandera de acarreo en su estado corriente. Un slash ("/") precediendo al operando en el lenguaje de ensamblador indica que el complemento lógico del bit direccionado es usado como el valor fuente, pero el bit fuente por si mismo no es afectado. Ninguna otra bandera es afectada.

Sólo el direccionamiento directo es permitido por el operando fuente.

EJEMPLO :

Se coloca la bandera de acarreo si y solo si , P1.0=1, ACC.7=1 y OV=0:

MOV C,P1.0; C ← P1.0 ; establece el CARRY
 ANL C,ACC.7; C = C x ACC.7 ; Y-lóg. del bit ACC.7 con el CARRY
 ANL C,/OV; C = C x OV ; Y - lógico con el inverso de la bandera OV C se establece en los 3 casos.

ANL C,bit

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 0 1 0
---------	---------

bit direccionado

OPERACIÓN: **ANL**

(C) ← (C)^(bit)

ANL C,bit

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 0 0 0
---------	---------

bit direccionado

OPERACIÓN: **ANL**

(C) ← (C)^(bit)

CJNE. Compara las magnitudes del primero y segundo operando, y brinca si sus valores no son iguales. El salto es calculado por la suma algebraica del valor actual del PC y el desplazamiento relativo (operando REL). La bandera de acarreo es establecida si el valor entero sin signo de la <palabra destino> es menor que el valor entero sin signo de la <palabra fuente>; por otro lado el acarreamiento es limpiado. Ningún operando es afectado.

CJNE
 <dest>,<fuente>
Compara y brinca
si no es igual

Los primeros dos operandos permiten cuatro combinaciones de modos de direccionamiento; el acumulador puede ser comparado con cualquier palabra directamente direccionada ó un dato inmediato, y cualquier localidad de RAM indirecta ó registro trabajando que puede ser comparado con una constante inmediata.

EJEMPLO :

El acumulador contiene 34H. El registro 7 contiene 56H. La primera instrucción secuencia,

CJNE R7,#60H,NO_IGUAL**;R7=60H****NO_IGUAL: JC RIG_BAJO****;si R7<60H****;si R7>60H**

establece la bandera de acarreo y brinca a la instrucción con la etiqueta NO_IGUAL. Probando la bandera de acarreo esa instrucción determina si R7 es mayor o menor que 60H.

Si el dato que ha sido presentado al puerto 1 es también 34H entonces la instrucción;

ESPERA: CJNE A,P1,ESPERA

limpia la bandera de acarreo y continua con la siguiente instrucción, debido a que el acumulador es igual al dato leído del puerto P1. Si es cualquier otro valor el programa deberá de girar en este punto hasta que el puerto 1 tenga el dato 34H.

NOTA :

El dato es tomado directamente de las terminales del puerto.

CJNE A, DIRECTO,REL

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

dirección directa	dirección relativa
-------------------	--------------------

OPERACIÓN: (PC) ← (PC) + 3

Si (A) <> (directo)

Entonces:

(PC) ← (PC) + salto relativo

Si (A) < (directo)

Entonces:

(C) ← 1

De lo contrario:

(C) ← 0

CJNE A,#DATO,REL

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 1 0 1	dato inmediato	dirección relativa
---------	---------	----------------	--------------------

OPERACIÓN: (PC) \leftarrow (PC) + 3Si (A) \neq DATO

Entonces:

(PC) \leftarrow (PC) + salto relativo

Si (A) < dato

Entonces:

(C) \leftarrow 1

Además:

(C) \leftarrow 0**CJNE Rn #DATO, REL**

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	1 r r r	dato inmediato	dirección relativa
---------	---------	----------------	--------------------

OPERACIÓN: **CJNE**(PC) \leftarrow (PC) + 3Si (Rn) \neq dato

Entonces:

(PC) \leftarrow (PC) + salto relativo

Si (Rn) < dato

Entonces:

(C) \leftarrow 1

De lo contrario:

(C) ← 0

CJNE @Ri,#DATO,REL

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 1 1 i
---------	---------

dato inmediato	dirección relativa
----------------	--------------------

OPERACIÓN: (PC) ← (PC) + 3

Si ((Ri)) <> dato

Entonces:

(PC) ← (PC) + salto relativo

Si ((Ri)) < dato

Entonces:

(C) ← 1

De lo contrario:

(C) ← 0

<p>CLR A Limpia el acumulador</p>
--

CLR A. El acumulador es limpiado (todos los bits se colocan en cero). Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene 5CH (01011100B). La instrucción, **CLR A** dejará el acumulador colocado a 00H (00000000B).

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 1 0 0
---------	---------

OPERACIÓN: **CLR**

(A) ← 0

<p>CLR bit limpia el bit.</p>
--

CLR. El bit indicado es limpiado (se convierte a cero). Ninguna otra bandera es afectada. CLR puede operar sobre una bandera de acarreo o cualquier bit directamente direccionable.

EJEMPLO :

El puerto 1 ha sido previamente escrito con 5DH (01011101B). La instrucción,

CLR P1.2

dejará el puerto colocado en 59H (01011001B El).

CLR C

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 0 1 1
---------	---------

OPERACIÓN: **CLR**

(C) \leftarrow 0

CLR bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 0 1 0	dirección del bit
---------	---------	-------------------

OPERACIÓN: **CLR**

(bit) \leftarrow 0

CPL A. Cada bit del acumulador está lógicamente complementado (complemento a unos). Los bits que previamente contienen un 1 son cambiados a 0 y viceversa. Las banderas no son afectadas.

<p>CPL A complemento del acumulador.</p>

EJEMPLO:

El acumulador contiene 5CH (01011100B). La instrucción, **CPL A** dejará el acumulador colocado a 0A3H (10100011B).

CPL A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

OPERACIÓN: CPL

$$(A) \leftarrow \neg(A)$$

Complemento del acumulador

CPL BIT
Complemento del bit

CPL BIT. La variable bit especificada es complementada. Un bit el cual ha estado en 1 es cambiado a 0, y viceversa. Ninguna otra bandera es afectada. CPL puede operar sobre la bandera de acarreo (carry) o cualquier bit direccionable.

NOTA :

Cuando ésta instrucción es utilizada para modificar una terminal de salida, el valor usado como dato original será leído del latch de salida (cerrojo), no de la entrada de la terminal del puerto 1;

EJEMPLO :

El puerto 1 ha sido previamente escrito con 5DH (01011101B).
La instrucción secuencia,

CPL P1.1

CPL P1.2

dejará el puerto colocado a 5BH (01011011B).

CPL C

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 1 1	0 0 1 1
---------	---------

OPERACIÓN: **CPL**(C) \leftarrow /(C)**CPL bit**

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

dirección del bit

OPERACIÓN:

(bit) \leftarrow /(bit)

Complemento del acarreo

DA A, Ajusta el valor de los 8 bits del acumulador, resultado de una suma anterior de dos variables (cada una en formato BCD), produciendo dos dígitos de cuatro bits. Cualquier instrucción ADD o ADDC puede haber sido usada para ejecutar la suma.

Si en el acumulador los bits de 3-0 son, más grandes que 9 (XXXX1010-XXXX1111), o si la bandera AC es 1, 6 es sumado al acumulador produciendo el dígito en BCD del nibble de bajo orden. Esta suma interna establecerá la bandera de acarreo si un acarreo hacia el exterior del campo de los cuatro bits de alto orden, del o contrario, limpiará la bandera de acarreo.

Si la bandera de acarreo es ahora establecida ó si los cuatro bits de alto orden ahora excedieron de nueve (1010XXXX-1111XXXX), estos bits de alto orden son incrementados por seis, produciendo el dígito BCD en el nibble de alto orden. De igual forma, se establecería la bandera de acarreo si hubiera un acarreo hacia el exterior de los bits de alto orden, de lo contrario la bandera será limpiada. Así la bandera de acarreo indica si la suma de las dos variables BCD, originales es más grande que 100, permitiendo sumar múltiples decimales con precisión. OV no es afectado.

Todo esto ocurre durante el ciclo de una instrucción. Esencialmente, esta instrucción ejecuta la conversión decimal por adición de 00H, 06H, 60H, ó 66H al acumulador, dependiendo del acumulador inicial y las condiciones del PSW.

NOTA:

DA A no puede simplemente covertir un número hexadecimal en el acumulador a notación BCD; no se aplica DA A a substracción decimal.

EJEMPLO:

DA A
Ajuste decimal del
acumulador por
adición

El acumulador contiene el valor 56H (1010110B) representando el paquete de dígitos BCD del número decimal 56. El registro 3 contiene el valor 67H (01100111B) representando el paquete de dígitos del número decimal 67. La bandera de acarreo está establecida. La secuencia de instrucciones,

ADDC A,R3

DA A

ejecutará una suma estándar en complemento a dos binario, resultando el valor 0BEH (10111110) en el acumulador. Las baderas de acarreo y acarreo auxiliar serán limpiadas.

La instrucción de ajuste decimal alterará entonces el acumulador al valor 24H (00100100B), indicando el paquete de dígitos del número decimal 24, los dos dígitos de bajo orden de la suma decimal de 56, 67, y el acarreo. La bandera de acarreo será establecida por la instrucción de ajuste decimal, indicando que ocurrió un sobreflujo. La verdadera suma 56, 67, y 1 es 124.

Las variables BCD pueden ser incrementadas o decrementadas por adición de 01H ó 99H respectivamente. Si el acumulador inicialmente contiene 30H (representando los dígitos del 30 decimal), entonces la secuencia de las instrucciones,

ADD A,#99H

DA A

dejarán el acarreo establecido y 29H en el acumulador, puesto que $30 + 99 = 129$. El byte de bajo orden de la suma puede ser interpretado como $30 - 1 = 29$. En este caso es convencional si se incluye o no el valor del acarreo como parte del resultado.

BYTES:1 CICLOS: 1

CODIGO DE OPERACIÓN:

1 1 0 1	0 1 0 0
---------	---------

OPERACIÓN: **DA**

Los contenidos del acumulador están contenidos en BCD

SI $[(A_{3-0}) > 9] \vee [(AC) = 1]$
Entonces $(A_{3-0}) \leftarrow (A_{3-0}) + 6$

SI $[(A_{7-4}) > 9] \vee [(C) = 1]$
Entonces $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

DEC. La variable indicada es decrementada en 1. Cuando el valor original del BYTE es 00H al decrementarse pasará a 0FFH y existirá un sobreflujo. Ninguna otra bandera es afectada. Cuatro operandos de modos de direccionamiento son permitidos: acumulador, registro, directo ó registro indirecto.

NOTA:

Cuando ésta instrucción es usada para modificar un puerto de salida, el valor usado como el dato original del puerto será leído del latch de salida, no de la terminal de entrada.

EJEMPLO:

El registro 0 contiene 7FH (01111111B). Las localidades 7EH y 7FH de la RAM interna contienen 00H y 40H, respectivamente. La secuencia de instrucciones,

```
DEC  @R0
DEC  R0
DEC  @R0
```

Dejará el registro 0 colocado a 7EH y la localidad de la RAM interna 7EH y 7FH colocada a 0FFH y 3FH. Se indicará además que hubo un sobreflujo, OV = 1.

DEC A

BYTES:1 CICLOS:1

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 1 0 0
---------	---------

OPERACIÓN: **DEC**
 (A) $\leftarrow (A) - 1$

DEC Rn

BYTES:1 CICLOS: 1

CODIGO DE OPERACIÓN:

0 0 0 1	1 r r r
---------	---------

OPERACIÓN: **DEC**

(Rn) \leftarrow (Rn) - 1

DEC directo

BYTES:2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **DEC** $(\text{directo}) \leftarrow (\text{directo}) - 1$ **DEC @Ri**

BYTES:1 CICLOS:1

CODIGO DE OPERACIÓN:

0 0 0 1	0 1 1 1
---------	---------

OPERACIÓN:

 $((Ri)) \leftarrow ((Ri)) - 1$
DIV AB
Divide

DIV AB divide el entero no signado del acumulador entre el entero no signado del registro B. El acumulador recibe la parte entera del cociente; el registro B el residuo. El carry y la bandera OV serán limpiadas.

EXCEPCIÓN:

Si B contiene originalmente 00H, los valores del acumulador y el registro B será indefinido, la bandera de sobreflujo será establecida. La bandera de acarreo de limpiada en cualquier caso.

EJEMPLO:

El acumulador contiene 251 (0FBH ó 11111011B) y B contiene 18 (12H ó 00010010B). La instrucción,

DIV AB

Dejará 13 en el acumulador (0DH ó 00001101B) y el valor 17 (11h ó 00010001B) en B, puesto que $251 = (13 \times 18) + 17$. Carry y OV serán limpiados.

BYTES: 1 CICLOS: 4

CODIGO DE OPERACIÓN:

1 0 0 0	0 1 0 0
---------	---------

OPERACIÓN: **DIV**

$$\begin{array}{ccc} (A)_{15-8} & & \\ & \leftarrow & (A) / (B) \\ (B)_{7-0} & & \end{array}$$

DJNZ decrementa en uno la localidad indicada, si el byte no es igual a cero, salta a la dirección formada por la suma algebraica del PC incrementado y el desplazamiento relativo <REL>, de otra manera continúa con la siguiente instrucción.

**DJNZ <byte>,<rel>
decrementa y brinca
si no es cero**

Un valor original de 00H pasará a 0FFh. Las banderas no son afectadas. La localidad decrementada puede ser un registro o un byte direccionado directamente.

NOTA:

Cuando la instrucción es usada para modificar un puerto de salida, el valor usado como dato original del puerto será leído del latch de salida, no de las terminales de entrada.

EJEMPLO:

Las localidades de la RAM interna 40H, 50H y 60H contienen el valor 01H, 70H y 15H, respectivamente. La secuencia de instrucciones,

```
DJNZ 40H,LABEL_1
DJNZ 50H,LABEL_2
DJNZ 60H,LABEL_3
```

Causará un salto a la instrucción con la etiqueta LABEL_2 con los valores 00H, 6FH y 15H, en las tres localidades de la RAM. El primer salto no se efectúa porque el resultado es cero.

La instrucción provee una forma simple de ejecutar el lazo de un programa un número dado de veces, o provocar un tiempo de retraso moderado (de 3 a 512 ciclos de máquina) con una sola instrucción. La secuencia de instrucciones,

```
MOV R2,#8
TOGGLE: CPL P1.7
DJNZ R2,TOGGLE
```

Alternará el P1.7 ocho veces, causando cuatro pulsos de salida que aparecerán en el bit 7 del puerto de salida 1. Cada pulso durará tres ciclos de máquina; dos por DJNZ y uno por alterar la terminal.

DJNZ Rn,rel

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **DJNZ** $(PC) \leftarrow (PC) + 2$ $(Rn) \leftarrow (Rn) - 1$ Si $(Rn) > 0$ ó $(Rn) < 0$

Entonces:

 $(PC) \leftarrow (PC) + rel$ **DJNZ directo,rel**

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 1 0 1	Dirección directa	Dirección relativa
---------	---------	-------------------	--------------------

OPERACIÓN: **DJNZ** $(PC) \leftarrow (PC) + 2$ $(directo) \leftarrow (directo) - 1$ Si $(directo) > 0$ ó $(directo) < 0$

Entonces:

 $(PC) \leftarrow (PC) + rel$

INC<byte> incremento.
--

INC incrementa el byte en 1. Si éste se encuentra inicialmente en 0FFH pasará a 00H. Las banderas no son afectadas. Cuatro modos de direccionamiento son permitidos: Acumulador, registro, directo, ó registro-indirecto.

NOTA :

Cuando esta instrucción es usada para modificar un puerto de salida, el valor usado como el dato original del puerto será leído del latch de salida, no de la terminal de entrada.

EJEMPLO :

El registro 0 contiene 7EH(01111110B). Las localidades de la RAM interna 7EH y 7FH contienen 0FFH y 40H, respectivamente.

La instrucción secuencia,

INC @R0

INC R0

INC @R0

dejará el registro 0 colocado a 7FH y las localidades de la RAM interna 7EH y 7FH conteniendo (respectivamente) 00H y 41H.

INC A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 0 0
---------	---------

OPERACIÓN: **INC**

$(A) \leftarrow (A) + 1$

NOTA:

En esta instrucción de INC la bandera de paridad se ve afectada.

INC Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 r r r
---------	---------

OPERACIÓN: **INC**

$(Rn) \leftarrow (Rn) + 1$

INC directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **INC**

$$(\text{directo}) \leftarrow (\text{directo}) + 1$$
INC @Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 1 1 i
---------	---------

OPERACIÓN: **INC**

$$((Ri)) \leftarrow ((Ri)) + 1$$

INC DPTR
incrementa el apuntador de
datos.

INC DPTR. Incrementa el valor del apuntador de datos en 1. El incremento en un registro de 16 bits es optimizado; un sobreflujo del byte de bajo orden del apuntador de datos (D L) de 0FFH a 00H incrementará el byte de alto orden (D H). Las banderas no son afectadas.

Este es el único registro de 16 bits que puede ser incrementado.

EJEMPLO :

Los registros DPH y DPL contienen 12H y 0FEH, respectivamente. La secuencia de instrucciones,

INC DPTR
INC DPTR
INC DPTR

cambiará DPH y DPL a 13H y 01H respectivamente.

INC

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 0 1 1
---------	---------

OPERACIÓN: **INC**

$$(DPTR) \leftarrow (DPTR) + 1$$

JB. Si el bit indicado es uno, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. El bit marcado no se modifica. Las banderas no son afectadas.

EJEMPLO :

El dato presente en el puerto de entrada 1 es CA (11001010B). El acumulador contiene 56 (01010110B). La secuencia de instrucciones,

JB 1,ETIQUETA1
JB ACC.2,ETIQUETA2

causará que el programa en ejecución salte a la instrucción marcada con ETIQUETA2.

JB

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 0 0 0	dirección del bit	dirección relativa
---------	---------	-------------------	--------------------

OPERACIÓN: **JB**

Si (bit) = 1
 Entonces:

$$(C) \leftarrow (C) + 3$$

$$(C) \leftarrow (C) + \text{rel}$$

JBC. Si el bit marcado es uno, lo limpia y salta a la dirección formada por la suma algebraica del C incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. Las banderas no son afectadas.

JBC bit,rel
Si el bit esta establecido
lo limpia y salta.

NOTA : Cuando esta instrucción es usada para examinar una terminal de salida, el valor usado como dato original será leído del latch de salida, no de la terminal de entrada.

EJEMPLO :

El acumulador contiene 56H (01010110B). La secuencia de instrucciones,

JBC ACC.3,ETIQUETA1
JBC ACC.2,ETIQUETA2

causará que la ejecución del programa continúe en la instrucción identificada por ETIQUETA2, con el acumulador modificado a 52H (01010010B).

JBC

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 0 0 0
---------	---------

dirección del bit	dirección relativa
-------------------	--------------------

OPERACIÓN: **JBC**

Si (bit) = 1
Entonces:

$$(PC) \leftarrow (C) + 3$$

$$(bit) \leftarrow 0$$

$$(PC) \leftarrow (PC) + rel$$

JC. Si la bandera de acarreo está establecida, salta a la dirección formada por la suma algebraica del C incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. Las banderas no son afectadas.

EJEMPLO:

La bandera de acarreo es limpiada. La instrucción secuencia,

```

JC    ETIQUETA 1
CPLC  C
JC    ETIQUETA 2

```

colocará el carry y causará la ejecución del programa para continuar con la instrucción identificada por la etiqueta LABEL 2.

JC

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 0 0 0
---------	---------

dirección relativa

OPERACIÓN: **JC**

Entonces:

$$(C) \leftarrow (C) + 2$$

$$(C) \leftarrow (C) + rel$$

JMP @A+DPTR salto indirecto.
--

JUMP @A+DPTR. Suma los ocho bits no signados contenidos en el acumulador con los 16 bits del apuntador de datos y carga la suma resultante al contador del programa. Esta será la nueva dirección para las siguientes búsquedas de las instrucciones.

La suma de los 16 bits es optimizada. El carry de salida de los ocho bits de bajo orden se propaga a los bits de alto orden. Ni el acumulador ni el dato del apuntador son alterados. Las banderas no son afectadas.

EJEMPLO :

Un número par de 0 a 6 está en el acumulador. La siguiente secuencia de instrucciones saltará a una de cuatro instrucciones AJMP en un salto empezando por **JMP_TBL**:

```

MOV      DPTR,#JMP_TBL
JMP      @A+DPTR
JMP_TBL: AJMP ETIQT0
          AJMP ETIQT1
          AJMP ETIQT2
          AJMP ETIQT3

```

Si el acumulador es igual a 04H cuando empieza ésta secuencia, la ejecución de la instrucción **JMP @A+DPTR**, saltará a la etiqueta ETIQT2. Recuerda que AJMP es una instrucción del byte 2, así la instrucción empieza en cada dirección.

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 0 1 1
---------	---------

OPERACIÓN: **JMP**

(PC) ← (A) + (DPTR)

JNB. Si el bit indicado es un cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción.

JNB bit,rel salta si el bit no es colocado

El bit examinado no es modificado. Las banderas no son afectadas.

EJEMPLO:

El dato presente en el puerto de entrada 1 es 11001010B. El acumulador contiene 56H (01010110B). La instrucción secuencia,

```

JNB P1.3,ETIQT1
JNB ACC.3,ETIQT2

```

causará la ejecución del programa para continuar con la instrucción o etiqueta ETIQT2.

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 0 0 0	bit de dirección	dirección relativa
---------	---------	------------------	--------------------

OPERACIÓN: **JNB**

(PC) \leftarrow (PC) + 3

Si (bit) = 0

Entonces (PC) \leftarrow (PC) + rel

JNC. Si el bit de acarreo es un cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. La bandera de acarreo no es modificada.

EJEMPLO :

La bandera de acarreo está establecida. La secuencia de instrucciones:

JNC ETIQT1
CPL C
JNC ETIQT2

limpiará el acarreo y continuará la ejecución del programa en la instrucción identificada con la etiqueta ETIQT2

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 1	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **JNC**

(PC) \leftarrow (PC) + 2

Si (C) = 0

Entonces (PC) \leftarrow (PC) + rel

JNZ rel salta si el acumulador no es cero.

JNZ. Si cualquier bit del acumulador es un 1, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. La bandera de acarreo no es modificada. de otra manera procede con la siguiente instrucción.

EJEMPLO :

El acumulador originalmente contiene 00H. La secuencia de instrucciones,

```

JNZ ETIQUETA1
INC A
JNZ ETIQUETA2

```

colocará el acumulador a 01H y saltará a la instrucción identificada con ETIQUETA2.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **JNZ**

(PC) ← (PC) + 2

Si (A) ≠ 0

Entonces (PC) ← (PC) + rel

JZ. Si todos los bits del acumulador son cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo, REL; de otra manera procede con la siguiente instrucción. El acumulador no se modifica. Las banderas no son afectadas.

JZ rel
salta si el acumulador
es cero.

EJEMPLO :

El acumulador originalmente contiene 01H. La secuencia de instrucciones,

```

JZ ETIQUETA 1
DEC A
JZ ETIQUETA 2

```

cambiará el acumulador a 00H y causará que la ejecución del programa continúe en la instrucción identificada por ETIQUETA 2.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **JZ**

(PC) ← (PC) + 2
 Si (A) = 0
 Entonces (PC) ← (PC) + rel

**LCALL dirección 16
 llamada larga.**

LCALL llama una subrutina que puede empezar en cualquier parte de la memoria de programa (64K bytes). La instrucción suma tres al contador del programa para que apunte a la dirección de la siguiente instrucción, luego incrementa el SP introduciendo el byte bajo del PC e incrementa nuevamente el SP para introducir el byte alto del PC. El PC se carga con el segundo y tercer byte de la instrucción **LCALL**. La ejecución de las instrucciones de la subrutina comienza en ésta dirección, hasta que encuentre la instrucción **RET**, la cual restablece el PC que había sido almacenado en el SP, continuando nuevamente con el programa inicial. Las banderas no son afectadas.

EJEMPLO :

Inicialmente el apuntador de apilamiento es igual a 07H. La etiqueta "**SBRTN**" es asignada a la memoria del programa en la localidad 1234H. Después de la ejecución la instrucción, **LCALL SUBRTN** se localiza en 0123H, el apuntador de apilamiento contendrá 09H, las localidades de la RAM interna 08H y 09H contendrá 26H y 01H, y la PC contendrá 1235H.

BYTES: 3

CICLOS: 2

0 0 0 1	0 0 1 0
---------	---------

direc. 15-direc. 8	direc. 7-direc. 0
--------------------	-------------------

OPERACIÓN: **LCALL**

(PC) ← (PC) + 3
 (SP) ← (SP) + 1
 ((SP)) ← (PC₇₋₀)
 (SP) ← (SP) + 1
 ((SP)) ← (PC₁₅₋₈)
 (PC) ← direc.15-0

**LJMP dirección 16
 salto largo.**

LJMP causa un salto incondicional a cualquier parte en el espacio de memoria del programa (64K bytes). El PC se carga con los 2 últimos bytes de la instrucción y salta para continuar con la ejecución del programa a partir de esa dirección. Las banderas no son afectadas.

EJEMPLO :

La etiqueta "**SALTO**" es asignada a la instrucción del programa de memoria en la localidad 1145H.

0156 021145 LJMP SALTO

1145 7827 SALTO: MOV R0,#27

La instrucción LJMP se localiza en 0156H, cargará el contador del programa con 1145H y saltará a dicha dirección continuando la ejecución del programa en ese punto.

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 0 1 0	direc. bits 15-8	direc. bits 7-0
---------	---------	------------------	-----------------

OPERACIÓN: **LJMP**

(PC) ← direc.15-0

MOV. La variable indicada por el segundo operando "Byte Fuente" es copiada en la localidad especificada por el primer operando "Byte destino". La palabra fuente no es afectada. Ningún otro registro o bandera es afectado.

Esta es la operación más extensa y flexible con que cuenta el microcontrolador. Quince combinaciones de modos de direccionamiento de fuente y destino son permitidos.

EJEMPLO :

La localidad de la RAM interna 30H contiene el dato 40H. El valor de la RAM interna 40H es 10H. El dato presente en el puerto de entrada 1 es 11001010B (0CAH).

```

00          ORG 00H
0000 7830   MOV R0,#30H ;      R0 ← 30H
0002 E6     MOV A,@R0 ;      A ← 40H
0003 F9     MOV R1,A ;      R1 ← 40H
0004 87F0   MOV B,@R1 ;      B ← 10H
0006 A790   MOV @R1,P1 ;      RAM(40H) ← 0CAH
0008 8590A0 MOV P2,P1 ;      P2 #0CAH
0000        END

```

deja el valor 30H en el registro 0, 40H en ambos el acumulador y el registro 1, 10H en el registro B, y 0CAH (11001010B) ambos en la localidad 40H de la RAM y sobre el puerto 2 de salida.

MOV A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	1 r r r
---------	---------

OPERACIÓN: **MOV**

(A) ← (Rn)

MOV A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **MOV**

(A) ← (directo)

MOV A,ACC no es una instrucción válida.

MOV A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 1 1 i
---------	---------

OPERACIÓN: **MOV**

(A) ← ((Ri))

MOV A,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 1 0 0
---------	---------

dato inmediato

OPERACIÓN: **MOV**

(A) ← #dato

MOV Rn,A

BYTES: 1

CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 1	1 r r r
---------	---------

OPERACIÓN: **MOV**(Rn) \leftarrow (A)**MOV Rn,directo**

BYTES: 2

CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	1 r r r
---------	---------

dirección directa

OPERACIÓN: **MOV**(Rn) \leftarrow (directo)**MOV Rn,#dato**

BYTES: 2

CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 1	1 r r r
---------	---------

dato inmediato

OPERACIÓN: **MOV**(Rn) \leftarrow #dato**MOV directo,A**

BYTES: 2

CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **MOV**

(directo) ← (A)

MOV directo,Rn

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	1 r r r	dirección directa
---------	---------	-------------------

OPERACIÓN: **MOV**

(directo) ← (Rn)

MOV directo,directo

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 1 0 1	direc. directa(dest.)	direc.directa(fuen.)
---------	---------	-----------------------	----------------------

OPERACIÓN: **MOV**

(directo) ← (directo)

MOV directo,@Ri

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 1 1 i	dirección directa
---------	---------	-------------------

OPERACIÓN: **MOV**

(directo) ← ((Ri))

MOV directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 1 0 1	dirección directa	dato inmediato
---------	---------	-------------------	----------------

OPERACIÓN: **MOV**

(directo) ← #dato

MOV @Ri,A

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 1 1 i
---------	---------

OPERACIÓN: **MOV**

((Ri)) ← (A)

MOV @Ri,directo

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 1 1 i	dirección directa
---------	---------	-------------------

OPERACIÓN: **MOV**

((Ri)) ← (directo)

MOV @Ri,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 1 1 i	dato inmediato
---------	---------	----------------

OPERACIÓN: **MOV**

((Ri)) ← #dato

MOV. El bit indicado por el segundo operando es copiado en la localidad especificada por el primer operando. Uno de los operandos debe ser la bandera de acarreo; el otro puede ser cualquier bit direccionable directamente. Ningún otro registro o bandera es afectado.

EJEMPLO :

<p>MOV <bit destino> <bit fuente> movimiento de bit</p>

La bandera de acarreo es originalmente establecida a 1 lógico. El dato presente en el puerto 3 de entrada es C5 (11000101B). El dato previamente escrito al puerto 1 de salida es 35H (00110101B). Las instrucciones,

MOV P1.3,C
MOV C,P3.3
MOV P1.2,C

dejarán el acarreo limpiado y cambiará el puerto 1 a 39H (00111001B).

MOV C,bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 0 1 0
---------	---------

dirección del bit

OPERACIÓN: **MOV**

(C) ← (bit)

MOV bit,C

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 0 1 0
---------	---------

dirección del bit

OPERACIÓN: **MOV**

(bit) ← (C)

MOV DPTR,#DATO 16 carga al apuntador de datos con una constante de 16 bits.
--

MOV DPTR.El apuntador de datos es cargado con un dato de 16 bits contenidos en el segundo y tercer bytes de la instrucción, que corresponden al byte de alto orden (DPH) y al byte de bajo orden (DPL) del DPTR respectivamente. Las banderas no son afectadas. Esta es la única instrucción que mueve un dato de 16 bits.

EJEMPLO :

La instrucción, **MOV DPTR,#35A4H** cargará el valor 35A4H en el apuntador de datos : DPH almacenará 35H y DPL almacenará A4H.

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 0 0 0
---------	---------

dato inmediato 15-8	dato inmediato 7-0
---------------------	--------------------

OPERACIÓN: **MOV**

DPH ← #dato₁₅₋₈
 DPL ← #dato₇₋₀

La instrucción **MOVC** carga el acumulador con un byte código, ó constante del programa de memoria. La dirección del byte escogido es la suma de los 8 bits del contenido original del acumulador y los 16 bits del registro base, el cual puede ser el apuntador de datos (DPTR) o el contador del programa (PC). En el último caso, el PC es incrementado a la dirección de la siguiente instrucción antes de ser sumado con el acumulador. En cualquiera de los casos el registro base no es alterado. La suma de los 16 bits es optimizada. Un acarreo de salida del byte de bajo orden se propaga al byte de alto orden. Las banderas no son afectadas.

EJEMPLO :

Un valor entre 0 y 3 está en el acumulador. La siguiente instrucción trasladará el valor en el acumulador a uno de cuatro valores definidos por el DB (byte definido) directivo.

```
REL_PC:    INC    A
            MOVC  A,@A + PC
            RET
            DB    66H
            DB    77H
            DB    88H
            DB    99H
```

Si la subrutina es llamada con el acumulador igual a 01H, retornará con 77H en el acumulador. La instrucción INC A es necesitada antes de la instrucción MOVC para saltar por encima de la instrucción RET a las constantes de la tabla.

MOVC A,@A + DPTR

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 0 1 1
---------	---------

OPERACIÓN: **MOVC**

(A) ← ((A) + (DPTR))

MOVC A,@A + PC

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 0 1 1
---------	---------

OPERACIÓN: **MOVC**

$$\begin{array}{lcl} (\text{PC}) & \leftarrow & (\text{PC}) + 1 \\ (\text{A}) & \leftarrow & ((\text{A}) + (\text{PC})) \end{array}$$

La instrucción **MOVX** transfiere el dato entre el acumulador y un byte de la memoria externa. Existen dos tipos de instrucciones; las que proveen un direccionamiento indirecto de 8 bits y las que lo hacen con 16 bits para acceder un dato de la RAM externa.

En el primer caso, el contenido de R0 y R1 proveen los 8 bits de la parte baja de la dirección dada por el puerto P0. Ocho bits son suficientes para una expansión externa I/O decodificada o para un arreglo RAM relativamente pequeño. Para algunos arreglos grandes, cualquier puerto de salida o terminal puede ser usado para direccionar bits de salida de alto orden. Estas terminales deberán ser controladas por instrucciones de salida que precedan la instrucción MOVX.

ATENCIÓN:

En el direccionamiento indirecto de 8 bits, en el momento de acceder el Dato, el contenido del P2 que se encuentra en el Espacio de Funciones Especiales será enviado sin excepción a las terminales de salida del puerto.

Debe tenerse especial cuidado cuando se accese una memoria de 256 bytes, de utilizar las terminales restantes del puerto de manera correcta, o bien, cuando se utilice una memoria mayor y se desee acceder por páginas, por medio de los registros R0 o R1, asegurarse que el P2 señale el número de página adecuado.

En el segundo caso, el apuntador de datos DPTR, genera una dirección de 16 bits. Por el puerto P2 saldrá el byte de alto orden (el contenido de DPH) de la dirección y por el puerto P0 el byte de bajo orden (DPL) el cual se multiplexa con el dato. En el registro de funciones especiales el P2 retiene su contenido previo mientras el buffer de salida del P2 está emitiendo el contenido de DPH. Esta forma es más rápida y más eficiente cuando accesa arreglos de datos muy grandes (hasta 64K bytes), dado que no necesita instrucciones adicionales en el P2 para acceder el dato.

Es posible en algunas situaciones mezclar los dos tipos de MOVX. Un arreglo de RAM grande con sus líneas de dirección de alto orden manejadas por P2 puede ser direccionado por vía del apuntador de datos, o guardando primeramente en el puerto P2 la dirección de alto orden, seguida por una instrucción MOVX usando R0 o R1.

EJEMPLO :

Una RAM externa de 256 bytes, usando un multiplexor de líneas dato/dirección (v.gr. el 74HC373), es conectada al puerto 0 del 8051. El puerto 3 provee líneas de control por la RAM externa. El puerto 1 y 2 son usados por I/O normal. El registro 0 y 1 contienen 12H y 34H.

La localidad 34H de la RAM externa contiene el valor 56H. La secuencia de instrucciones,

```
MOVX  A,@R1
MOVX  @R0,A
```

copia el valor 56H en el acumulador y la localidad 12H de la RAM externa.

MOVX A,@Ri

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 0 1 i
---------	---------

OPERACIÓN: **MOVX**

(A) ← ((Ri))

MOVX A,@DPTR

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 0	0 0 0 0
---------	---------

OPERACIÓN: **MOVX**

(A) ← ((DPTR))

MOVX @Ri,A

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 0 1 i
---------	---------

OPERACIÓN: **MOVX**

((Ri)) ← (A)

MOVX @DPTR,A

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 1 1	0 0 0 0
---------	---------

OPERACIÓN: **MOVX**

(DPTR) ← (A)

MUL AB multiplica los ocho bits del acumulador y los del registro B. El byte de bajo orden del producto de 16 bits, se almacena en el acumulador, y el byte de alto orden en B. Si el producto es más grande que 255 (0FFH) la bandera de sobreflujo es establecida, de lo contrario es limpiada.

EJEMPLO :

Originalmente el acumulador contiene el valor 80 (50H). El registro B contiene el valor 160 (0A0H). La instrucción, **MUL AB** dará el producto 12,800 (3200H), así B es cambiado a 32H (00110010B) y el acumulador es limpiado. La bandera de sobreflujo es colocada, el acarreo es limpiado.

BYTES: 1 CICLOS: 4

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 1 0 0
---------	---------

OPERACIÓN: **MUL**

(A)₇₋₀ ← (A) X (B)

(B)₁₅₋₈

<p>NOP No operación</p>

NOP. La ejecución continúa con la instrucción siguiente. No se afecta, ni el contador del programa, ni registros o banderas.

EJEMPLO :

Se desea producir un pulso bajo en la terminal de salida del bit 7 del puerto 2, exactamente durante los últimos 5 ciclos. Una secuencia SETB/CLR generaría un pulso de un ciclo, de tal manera, que se necesitarían 4 ciclos adicionales. Esto puede realizarse de la siguiente manera (se asume que no existen interrupciones):

CLR P2.7
NOP
NOP

NOP
NOP
SETB P2.7

NOP

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 0 0 0
---------	---------

OPERACIÓN: **NOP**

(PC) ← (PC) + 1

ORL ejecuta la operación OR lógica entre las variables indicadas, guardando el resultado en el byte destino. Las banderas no son afectadas.

Los dos operandos permiten 6 combinaciones de modos de direccionamiento. Cuando el destino es el acumulador, la fuente puede provenir de los siguientes direccionamientos; por registro, directo, registro indirecto, o direccionamiento inmediato; cuando el destino es una dirección directa, la fuente puede ser el acumulador o dato inmediato.

NOTA :

Cuando ésta instrucción es usada para modificar un puerto de salida, el valor usado como dato original, será leído del latch de salida del puerto y no de sus terminales.

EJEMPLO :

Si el acumulador contiene 0C3H (11000011B) y el R0 contiene 55H (01010101B) entonces la instrucción, **ORL A,R0** dejará el acumulador conteniendo el valor 0D7H (11010111B).

Con el direccionamiento directo, la instrucción puede colocar combinaciones de bits en cualquier localidad RAM interna o Registro. El patrón de bits que serán establecidos es determinado por un byte máscara, el cual puede ser también un dato o valor constante en la instrucción o una variable calculada en el acumulador. La instrucción, **ORL 1,#00110010B** colocará los bits 5, 4 y 1 del puerto de salida 1.

ORL A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	1 r r r
---------	---------

OPERACIÓN: ORL

(A) ← (A) V (Rn)

ORL A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **ORL**

(A) ← (A) V (directo)

ORL A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 1 1 i	dirección directa
---------	---------	-------------------

OPERACIÓN: **ORL**

(A) ← (A) V ((Ri))

ORL A,#dato

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **ORL**

(A) ← (A) V #dato

ORL directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 0 1 0
---------	---------

dirección directa

OPERACIÓN: **ORL**

(directo) ← (directo) V (A)

ORL directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 0 0	0 0 1 1
---------	---------

dirección directa	dato inmediato
-------------------	----------------

OPERACIÓN: **ORL**

(directo) ← (directo) V #dato

ORL. Establece la bandera de acarreo si el valor Booleano es un 1 lógico; de otra manera no lo cambia. Un slash (/o/o) precediendo el operando en el lenguaje ensamblador, indica que el complemento lógico del bit de direccionado es usado como el valor fuente, pero el bit original no es afectado. Las banderas no son afectadas.

EJEMPLO :

Coloca la bandera de acarreo sí y solo sí P1.0 = 1, ACC.7 = 1, ó OV = 0:

MOV C,P1.0 ;CARGA EL ACARREO CON LA TERMINAL DE ENTRADA P10**ORL C,ACC.7** ;O ACARREO CON EL ACC. BIT 7**ORL C,/OV** ;O ACARREO CON EL INVERSO DE OV.**ORL C,bit**

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 1	0 0 1 0
---------	---------

bit de dirección

OPERACIÓN: **ORL**

(C) ← (C) V (bit)

ORL C,/bit

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 1 0	0 0 0 0
---------	---------

bit de dirección

OPERACIÓN: **ORL**

(C) ← (C) V /(bit)

POP. El contenido de la localidad de la RAM interna direccionada por el apuntador de apilamiento es leído, y el apuntador de apilamiento es decrementado por 1. El valor leído es transferido al byte indicado por el direccionamiento directo. Las banderas no son afectadas.

EJEMPLO :

El apuntador de apilamiento originalmente contiene el valor 32H, y las localidades de RAM interna de la 30H a la 32H contienen los valores 20H, 23H y 01H, respectivamente. La instrucción secuencia,

POP DPH**POP DPL**

dejará el apuntador de apilamiento igual al valor 30H y el apuntador de datos se colocará a 0123H. En éste punto la instrucción, **POP SP** dejará el apuntador de apilamiento colocado a 20H. Nota que en éste caso especial el apuntador de apilamiento estaba decrementado a 2FH antes de ser cargado con el valor extraído (20H).

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 0 0 0
---------	---------

Dirección directa

OPERACIÓN: **POP**

(directo)---((SP))

(SP)---(SP) - 1

PUSH directo almacenamiento en la pila.
--

Push. El apuntador de apilamiento es incrementado por 1. El contenido de la variable indicada es luego almacenada en la localidad direccionada por el apuntador de apilamiento en la RAM interna. Las banderas no son afectadas.

EJEMPLO :

Al entrar en una rutina de interrupción el apuntador de apilamiento contiene 09H. El dato apuntador contiene el valor 0123H. La instrucción secuencia,

PUSH DPL
PUSH DPH

dejará el apuntador de apilamiento colocado a 0BH y guarda 23H y 01H en las localidades de la RAM interna 0AH y 0BH, respectivamente.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 0 0 0	dirección directa
---------	---------	-------------------

OPERACIÓN: **PUSH**

(SP) \leftarrow (SP) + 1
((SP)) \leftarrow (directo)

RET extrae de la pila los bytes de bajo y alto orden del PC, decrementando dos veces el apuntador de apilamiento. Una vez que el PC es cargado con la nueva dirección, continúa con la ejecución del programa principal, en la instrucción siguiente a la instrucción que llamó a la subrutina (ACALL o LCALL). Las banderas no son afectadas.

EJEMPLO :

El apuntador de apilamiento originalmente contiene el valor 0BH. Las localidades de la RAM interna 0AH y 0BH contienen los valores 23H y 01H, respectivamente. La instrucción, **RET** dejará el apuntador de apilamiento con el valor 09H. La ejecución del programa continuará en la localidad 0123H.

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 0 1 0
---------	---------

OPERACIÓN: **RET**

(PC₁₅₋₈) \leftarrow ((SP))
(SP) \leftarrow (SP) - 1
(PC₇₋₀) \leftarrow ((SP))
(SP) \leftarrow (SP) - 1

RETI retorno de interrupción.

RETI extrae de la pila, los bytes de alto y bajo orden del PC y restablece la lógica de Interrupción que le permite aceptar interrupciones adicionales de la misma o menor prioridad. El apuntador de apilamiento es decrementado dos veces. Los otros registros no son afectados; el estado de pre-interrupción del PSW no es automáticamente resguardado. La ejecución del programa continúa a la dirección resultante, que generalmente es la instrucción siguiente al punto en el cual la interrupción fue detectada.

NOTA :

Una interrupción de más alta prioridad puede interrumpir el servicio de interrupción de una de baja prioridad. Si una interrupción de nivel bajo ó del mismo nivel de prioridad está pendiente cuando la instrucción RETI es ejecutada entonces una instrucción deberá ser ejecutada antes de que dicha interrupción pendiente sea procesada.

EJEMPLO :

El apuntador de apilamiento originalmente contiene el valor 0BH. Una interrupción fue detectada durante la instrucción finalizando en la localidad 0122H. La localidad de la RAM interna 0AH y 0BH contienen el valor 23H y 01H, respectivamente. La instrucción, **RETI** dejará el apuntador de apilamiento igual a 09H y regresará al programa ejecutando la localidad 0123H.

BYTES: 1 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 0 1 0
---------	---------

OPERACIÓN: **RETI**

(PC ₁₅₋₈)	←	((SP))
(SP)	←	(SP) - 1
(PC ₇₋₀)	←	((SP))
(SP)	←	(SP) - 1

RL. Los ocho bits en el acumulador son rotados un bit a la izquierda. El bit 7 es rotado a la posición del bit 0. Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B). La instrucción, **RLA** deja el acumulador conteniendo el valor 8BH (10001011B) con el acarreo no afectado.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 0	0 0 1 1
---------	---------

OPERACIÓN: **RL**

$$\begin{array}{l} (An + 1) \\ (A0) \end{array} \leftarrow (An) \ n = 0 - 6$$

RLC. Los ocho bits en el acumulador y la bandera de acarreo son rotados un bit a la izquierda. El bit 7 se mueve a la bandera de acarreo; el estado original de la bandera de acarreo se mueve a la posición del bit 0. Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B), y el acarreo es cero. La instrucción, **RLC A** deja el acumulador conteniendo el valor 8AH (10001010B) con el acarreo establecido.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 1 1	0 0 1 1
---------	---------

OPERACIÓN: **RLC**

$$\begin{array}{l} (An + 1) \\ (A0) \\ (C) \end{array} \leftarrow \begin{array}{l} (An) \ n = 0 - 6 \\ (C) \\ (A7) \end{array}$$

RR. Los ocho bits en el acumulador son rotados un bit a la derecha. El bit 0 es rotado a la posición del bit 7. Las banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B). La instrucción, **RR A** deja al acumulador conteniendo el valor 0E2H (11100010B) con el acarreo no afectado.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 0	0 0 1 1
---------	---------

OPERACIÓN: **RR**

$$\begin{array}{l} (An) \\ (A7) \end{array} \leftarrow \begin{array}{l} (An + 1) \ n = 0 - 6 \\ (A0) \end{array}$$

RRC A rota el acumulador a la derecha a través de la bandera de acarreo.

RRC. Los ocho bits en el acumulador y la bandera de acarreo son rotados un bit a la derecha. El bit 0 se mueve a la bandera de acarreo; el valor original de la bandera de acarreo se mueve a la posición del bit 7. Las otras banderas no son afectadas.

EJEMPLO :

El acumulador contiene el valor 0C5H (11000101B), el acarreo es cero. La instrucción, **RRC A** deja al acumulador conteniendo el valor 62 (01100010B) con el acarreo establecido.

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 0 0 1	0 0 1 1
---------	---------

OPERACIÓN: **RRC**

(An)	←	(An + 1) n = 0 - 6
(A7)	←	(C)
(C)	←	(A0)

SETB coloca el bit indicado en uno. SETB puede operar sobre la bandera de acarreo ó cualquier bit direccionable directamente. Las otras banderas no son afectadas.

EJEMPLO :

La bandera de acarreo es limpiada. El puerto 1 de salida ha sido escrito con el valor 34H (00110100B). Las instrucciones,

SETB C
SETB P1.0

dejan la bandera de acarreo colocada a 1 y cambian el dato de salida en el puerto 1 a 35H (00110101B).

SETB C

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 0 1 1
---------	---------

OPERACIÓN: **SETB**

(C)	←	1
-----	---	---

SETB bit

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 0 1 0	bit de dirección
---------	---------	------------------

OPERACIÓN: **SETB**

(bit) \leftarrow 1

SJMP. El programa de control salta incondicionalmente a la dirección indicada. El destino del salto es calculado por la suma del desplazamiento REL, más el contenido del Contador del Programa PC incrementado. El rango de destinaciones permitido es de 128 bytes hacia atrás y de 127 bytes hacia adelante.

EJEMPLO :

La etiqueta “RELAD” es asignado a una instrucción en la localidad del programa de memoria 0123H. La instrucción,

0100 8021 SJMP RELAD
102 \leftarrow EL PC APUNTA A ESTA LOCALIDAD

0123 7420 RELAD: MOV A,#20 \leftarrow NUEVA INSTRUCCIÓN A EJECUTAR

ensamblará en la localidad 0100H. Después la instrucción es ejecutada, el PC contendrá el valor 0123H.

NOTA :

Por debajo o por arriba de las condiciones la instrucción siguiente SJMP estará en 102H. Además, el byte de desplazamiento de la instrucción será el offset relativo (0123H-0102H) = 21H. De otra forma, un SJMP con un desplazamiento de 0FEH sería una instrucción de lazo infinito.

BYTES: 2 CICLOS: 2

CÓDIGO DE OPERACIÓN:

1 0 0 0	0 0 0 0	dirección relativa
---------	---------	--------------------

OPERACIÓN: **SJMP**

(PC) \leftarrow (PC) + 2

$$(PC) \leftarrow (PC) + rel$$

SUBB subtrae la variable indicada y la bandera de acarreo juntas del acumulador, dejando el resultado en el acumulador. **SUBB** establece la bandera de acarreo C, (borrow), si un préstamo es necesitado por el bit 7, por otro lado limpia C. AC es establecido si es necesitado un préstamo por el bit 3, de otra manera es limpiado. OV es establecido si es necesitado un préstamo por el bit 6, pero no por el bit 7, o por el bit 7, pero no por el bit 6.

Cuando se substraen enteros signados la señal de OV, indica un número negativo producido cuando un valor negativo es substraído de un valor positivo, o un resultado positivo cuando un número positivo es substraído de un número negativo.

El operando fuente permite cuatro modos de direccionamiento: Por registro, directo, registro-indirecto e inmediato.

EJEMPLO :

El acumulador contiene 0C9H (11001001B), el registro 2 contiene 54H (01010100B), y la bandera de acarreo es establecida. La instrucción, **SUBB A,R2** dejará el valor 74H (01110100B) en el acumulador, con la bandera de acarreo y AC limpiados pero OV establecido.

Note que 0C9H menos 54H es 75H. La diferencia entre éste y el resultado siguiente es debido a la bandera de acarreo (borrow) que ha sido establecida antes de la operación. Si el estado del acarreo no es conocido antes de empezar una substracción simple o de precisión múltiple, él debe de ser limpiado por una instrucción CLR C.

SUBB A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	1 r r r
---------	---------

OPERACIÓN: **SUBB**

$$(A) \leftarrow (A) - (C) - (Rn)$$

SUBB A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 1 0 1	dirección directa
---------	---------	-------------------

OPERACIÓN: **SUBB**

$$(A) \leftarrow (A) - (C) - (\text{directo})$$

SUBB A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 1 1 i
---------	---------

OPERACIÓN: **SUBB** $(A) \leftarrow (A) - (C) - ((Ri))$ **SUBB A,#dato**

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 0 0 1	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **SUBB** $(A) \leftarrow (A) - (C) - \#dato$

SWAP A intercambia los “NIBBLES” (campos de 4 bits) de alto y bajo orden del acumulador (bits 3-0 y bits 7-4). La operación puede también ser vista como una instrucción de rotación de 4 bits sin acarreo. Las banderas no son afectadas.

EJEMPLO:

El acumulador contiene el valor 0C5H (11000101B). La instrucción, **SWAP A** deja al acumulador conteniendo el valor 5CH (01011100B).

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 0	0 1 0 0
---------	---------

OPERACIÓN: **SWAP** $(A_{3-0}) \rightarrow/\leftarrow (A_{7-4})$

XCH carga el acumulador con el contenido de la variable indicada, al mismo tiempo que escribe el contenido original del acumulador a la variable indicada. El operando fuente/destino puede usar direccionamiento por registro, directo, o registro indirecto.

SWAP A
intercambia la parte
alta y parte baja del
acumulador.

EJEMPLO:

R0 contiene la dirección 20H. El acumulador contiene el valor 3FH (00111111B). La localidad de la RAM interna 20H contiene el valor 75H (01110101B). La instrucción, **XCH A,@R0** dejará la localidad 20H de la RAM conteniendo los valores 3FH (00111111B) y 75H (01110101B) en el acumulador.

XCH A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1	1	0	0	1	r	r	r
---	---	---	---	---	---	---	---

OPERACIÓN: **XCH**

(A) →/← (Rn)

XCH A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1	1	0	0	0	1	0	1	dirección directa
---	---	---	---	---	---	---	---	-------------------

OPERACIÓN: **XCH**

(A) →/← (directo)

XCH A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1	1	0	0	0	1	1	i
---	---	---	---	---	---	---	---

OPERACIÓN: **XCH**

(A) →/← ((Ri))

XCHD intercambia el NIBBLE de bajo orden del acumulador (bits 3-0), con el NIBBLE bajo del dato de la RAM interna indirectamente direccionada por el registro especificado. Las banderas no son afectadas.

EJEMPLO:

R0 contiene la dirección 20H. El acumulador contiene el valor 36H (00110110B). La localidad 20H de la RAM interna contiene el valor 75H (01110101B). La instrucción, **XCHD A,@R0** dejará la localidad 20H de la RAM conteniendo el valor 76H (01110110B) y en el acumulador 35 (00110101B).

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

1 1 0 1	0 1 1 i
---------	---------

OPERACIÓN: **XCHD**

$(A_{3-0}) \quad \rightarrow/\leftarrow \quad ((Ri_{3-0}))$

XRL ejecuta la operación lógica OR-Exclusivo entre las variable indicadas, guardando el resultado en el destino. Las banderas no son afectadas. Los dos operandos permiten 6 combinaciones de modos de direccionamiento. Cuando el destino es el acumulador, la fuente puede usar direccionamiento directo, por registro, por registro indirecto, e inmediato; cuando el destino es una dirección directa, la fuente puede ser el acumulador o un dato inmediato.

NOTA :

Cuando esta instrucción es usada para modificar un puerto de salida, el valor usado como dato original del puerto será leído del latch de salida, no de la terminal de entrada.

EJEMPLO:

Si el acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B) entonces la instrucción, **XRL A,R0** dejará al acumulador conteniendo el valor 69H (01101001B).

Cuando el destino es un byte directamente direccionado, ésta instrucción puede complementar combinaciones de bits en cualquier localidad RAM interna o registro. El patrón de bits que será complementado es entonces determinado por un byte máscara, ya sea una constante contenida en la instrucción o una variable calculada en el acumulador en tiempo real .

La instrucción, **XRL P1,#00110001B** deberá complementar los bits 5, 4 y 0 del puerto de salida 1.

XRL A,Rn

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	1 r r r
---------	---------

OPERACIÓN: **XRL**

(A) ← (A) A (Rn)

XRL A,directo

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 1 0 1
---------	---------

dirección directa

OPERACIÓN: **XRL**

(A) ← (A) A (directo)

XRL A,@Ri

BYTES: 1 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 1 1 i
---------	---------

OPERACIÓN: **XRL**

(A) ← (A) A ((Ri))

XRL A,#DATA

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 1 0 0	dato inmediato
---------	---------	----------------

OPERACIÓN: **XRL**

(A) \leftarrow (A) A #dato

XRL directo,A

BYTES: 2 CICLOS: 1

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 0 1 0	dirección directa
---------	---------	-------------------

OPERACIÓN: **XRL**

(directo) \leftarrow (directo) A (A)

XRL directo,#dato

BYTES: 3 CICLOS: 2

CÓDIGO DE OPERACIÓN:

0 1 1 0	0 0 1 1	dirección directa	dato inmediato
---------	---------	-------------------	----------------

OPERACIÓN: **XRL**

(directo) \leftarrow (directo) A #dato.

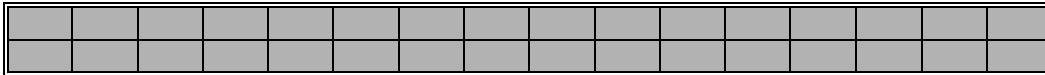
INTERCONEXIÓN DE UN EXHIBIDOR ALFANUMÉRICO, DE CRISTAL LIQUIDO DE 2 LINEAS POR 16 CARACTERES, AL MICROCONTROLADOR 8051

En la mayoría de los sistemas a desarrollar, resulta indispensable el uso de un exhibidor, el cual, nos muestre mediante mensajes escritos, las demandas del aparato, los requerimientos, los mandatos externos, las señalizaciones, las alarmas, etc. El uso de un exhibidor alfanumérico nos proporciona la solución a esta necesidad, facilitando el manejo y aprendizaje del aparato o dispositivo que ha sido implementado.

En esta sección se propone utilizar el Modulo de cristal líquido de 2 líneas por 16 caracteres por línea, el cual es fabricado por diversas compañías, tales como, Philips, Sharp, AND y algunas otras.

Este exhibidor puede interconectarse directamente, con el bus de datos de cualquier microprocesador o microcontrolador, gracias a que tiene un bus de datos con tres estados. Además contiene una memoria RAM que le permite almacenar hasta 128 caracteres y una memoria ROM con 160 caracteres matriciales de 5x7 puntos, y 30 caracteres de 5x10 puntos. La ventana del exhibidor permite ver 32 caracteres a la vez, en 2 líneas de 16 caracteres cada una.

EXHIBIDOR ALFANUMÉRICO DE 2 LINEAS X 16 CARACTERES

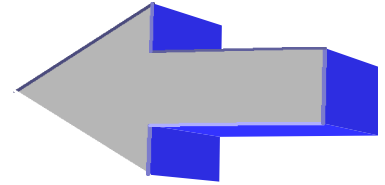


14° 13° 12° 11° 10° 3° 2° 1°

ASIGNACION DE LAS TERMINALES

TERMINAL	SEÑAL	FUNCION
1	GND	TIERRA 0 Volts.
2	VDD	5 Volts
3	Vo	VOLTAJE DE CONTRASTE
4	RS	RS=1 ENTRADA DE DATO, RS=0 ENTRADA DE CONTROL
5	R/W	R/W=1 LECTURA R/W=0 ESCRITURA
6	E	SEÑAL DE HABILITACION DEL CIRCUITO
7	DB0	BIT MENOS SIGNIFICATIVO DEL BUS DE DATOS
8	DB1	
9-13	DB2	BUS DE DATOS 8 BITS
14	DB7	BIT MÁS SIGNIFICATIVO DEL BUS DE DATOS

El procedimiento para inicializar el exhibidor es el siguiente:



1. _ Primeramente se establece el tipo de interfase a la cual el exhibidor se va a conectar, en nuestro caso, se trata de un microcontrolador con un bus de datos de 8 bits, el cual se conecta directamente.

La primera palabra de control que se envía al exhibidor es el número 38H, el cual significa lo siguiente:

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
38H	0	0	0	0	1	1	1	0	0	0

Se envía la palabra de control al exhibidor (RS=0 y R/W=0), los bits DB5 y DB4 especifican el tamaño del bus, y el bit DB3 el número de líneas del exhibidor.

Se espera un lapso de tiempo de 40 μ s antes de enviar la siguiente instrucción.

NOTA: Cada instrucción, toma un cierto tiempo de ejecución que va de 40 μ s a 1.64 ms. (Ver tiempos de ejecución en la Tabla1.)

2. - Se limpia toda la memoria del exhibidor y se regresa la pantalla del exhibidor a su posición inicial.

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
01H	0	0	0	0	0	0	0	0	0	1

Esta instrucción toma un tiempo de 1.64 ms,

3. -Se establece el movimiento del cursor hacia la derecha, la pantalla del exhibidor permanece fija con la entrada de los caracteres.

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
06H	0	0	0	0	0	0	0	1	1	0

Esta instrucción toma un tiempo de 40 μ s.

4.-Se prende la pantalla del exhibidor, se activa el cursor señalando la posición del próximo caracter de entrada y se desactiva el parpadeo.

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0EH	0	0	0	0	0	0	1	1	1	0

5._Se posiciona el cursor en el primer caracter y la primera línea.

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
80H	0	1	0	0	0	0	0	0	0	0

Esta instrucción toma un tiempo de 40 μ s

6.-A partir de aquí se puede comenzar a enviar los caracteres que se desean exhibir dejando un tiempo entre cada uno de ellos de 40 μ s. mínimo, y con RS=1

Por ejemplo se enviar n las letras A y B, por lo tanto se escribir el siguiente código ASCII para la letra A, el cual se ejecuta en 40 μ s.

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
41H	1	0	0	1	0	0	0	0	0	1

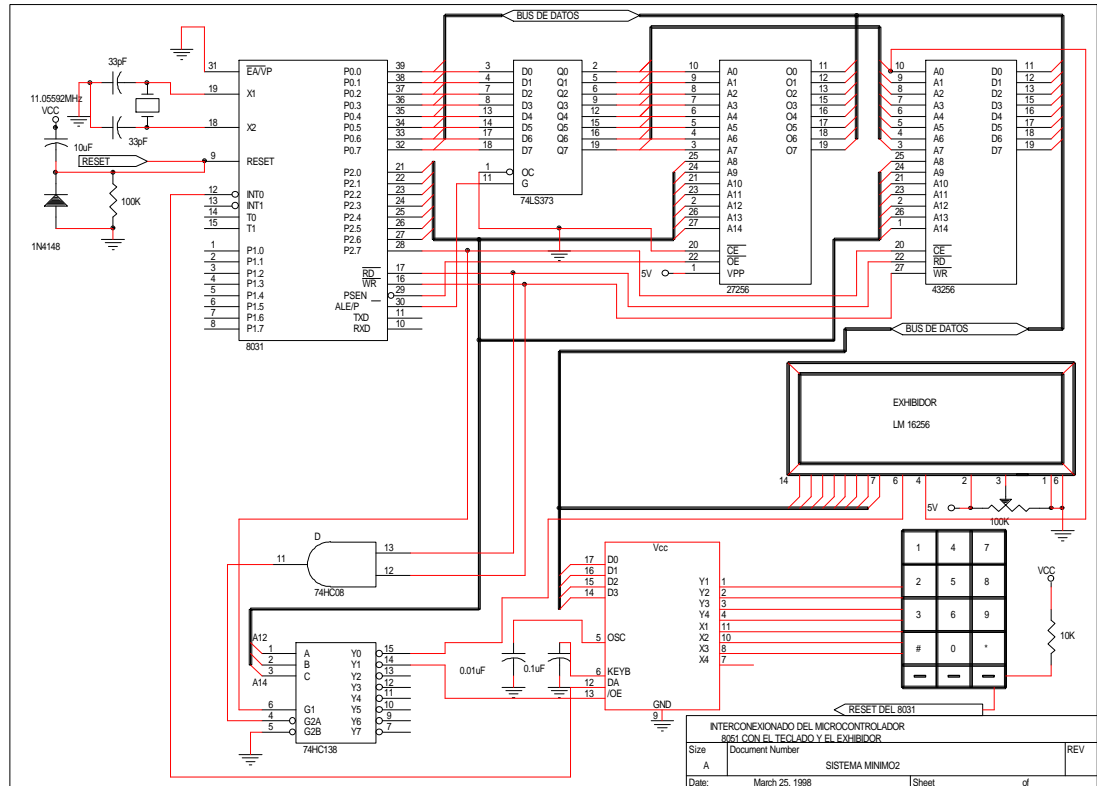
A															

Ahora se envía el código respectivo de la letra B.

Código	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
42H	1	0	0	1	0	0	0	0	1	0

Se escribe la letra B y se incrementa el cursor.

A	B														



PROGRAMA PARA MANEJO DEL EXHIBIDOR ALFANUMÉRICO

; ESTE PROGRAMA COMIENZA A PARTIR DE LA LOCALIDAD 4000H POR
 ; SER LA LOCALIDAD DE INICIO DEL EMULADOR, PERO PUEDE COMENZAR
 ; A PARTIR DE CUALQUIER LOCALIDAD.

```

4000                                ORG 4000H
4000                                ;LAS DIRECCIONES DEL EXHIBIDOR SON
4000                                ;LA 8000H PARA CONTROL DEL EXHIBIDOR
4000                                ;LA 8001H PARA EXHIBICIÓN DEL CARACTER
4000                                ;APUNTADAS POR LOS REGISTROS R0 Y P2.

```

```

4000 904075    EXHIBE: MOV DPTR,#CONTEX    ;CONTROL DEL
4003 7800      MOV R0,#00H                ;EXHIBIDOR
4005 75A050    MOV P2,#80h
4008 124046    LCALL XCBDOR

```

```

400B          ;ENVIA LOS CARACTERES DE
              :CONTROL AL EXHIB

```

```

400B
400B A3          INC DPTR
400C 124046      LCALL XCBDR

400F            ;SE ENVIA EL CARACTER 01 DE CONTROL
400F            ;(LIMPIA PANTALLA), EL CUAL REQUIERE
400F            ;1.64mSEG. PARA SU EJECUCION.

400F 124056      LCALL LIMPIA

4012            ;CARGA EL APUNTADOR CON EL PRIMER
4012            ;MENSAJE

4012 90407D BIEN: MOV DPTR,#BIENV

4015            ;SE ENVIA A LA RUTINA QUE EXHIBE
4015            ;EN DOS LINEAS EN UNA VEZ.

4015 124030      LCALL DOBLEX
4018 124064      LCALL TIME

401B            ;SE ENVIA EL CARACTER 01 DE CONTROL
401B            ;(LIMPIA PANTALLA), EL CUAL REQUIERE
401B            ;1.64mSEG. PARA SU EJECUCION.

401B 7800        MOV R0,#00
401D 124056      LCALL LIMPIA

4020            ;SE ENVIA UN SOLO MENSAJE A LA PRIMERA
4020            ;LINEA A PARTIR DE LA COLUMNA 5.

4020 90409F      MOV DPTR,#MEDIO
4023 7485        MOV A,#85H    ;ESCRIBE EN LA 1era
4025 12403A      LCALL COEXH   ;LINEA (5ta. COLUMNA)
4028 124064      LCALL TIME
402B 124056      LCALL LIMPIA

402E 80E2        TEREH: SJMP BIEN

4030            ;*****
4030            ;**** SUBROUTINA DE CONTROL Y PRE-****
4030            ;**** SENTACION DEL EXHIBIDOR ****
4030            ;*****

4030 7450        DOBLEX: MOV A,#80    ;ESCRIBE EN LA 1era.
4032 113A        ACALL COEXH   ;LINEA (1era. COLUMNA)
4034 A3          INC DPTR
4035 74C0        MOV A,#0C0H    ;ESCRIBE EN LA 2da.
4037 113A        ACALL COEXH   ;LINEA (1era. COLUMNA)
4039 22          RET
403A 7800        COEXH: MOV R0,#00H
403C 75A080      MOV P2,#80H
403F F2          MOVX @R0,A
4040 1151        ACALL QARNTA
4042 08          INC R0
4043 1146        ACALL XCBDR
4045 22          RET

4046 E4          XCBDR: CLR A
4047 93          MOVC A,@A+DPTR
4048 6006        JZ TERMIN
404A F2          MOVX @R0,A

```

```

404B 1151          ACALL QARNTA      ;TIEMPO 40uSEGS
404D A3            INC DPTR
404E 80F6          SJMP XCBDOR
4050 22            TERMIN: RET

4051              ;*****
4051              ;*** SUBROUTINA DE TIEMPO DE 40uSEGS ***
4051              ;*****
4051 7F14          QARNTA: MOV R7,#20
4053 DFFE          TIEMPO: DJNZ R7,TIEMPO
4055 22            RET

4056

              ;*****
              ;** SUBROUTINA QUE ENVIA EL CARACTER ***
4056              ;** 01 DE CONTROL, Y ADEMAS CONSUME ***
4056              ;** LOS 1.64 mSEGS PARA SU EJECUCION ***
4056              ;*****

4056 90407B        LIMPIA: MOV DPTR,#CLEAR
4059 124046        LCALL XCBDOR
405C 7E28          MOV R6,#40
405E 124051        TI1600: LCALL QARNTA      ;TIEMPO DE 40uSEGS
4061 DEFB          DJNZ R6,TI1600
4063 22            RET
4064
4064              ;*****
4064              ;*** SUBROUTINA DE TIEMPO DE 2 SEGS ***
4064              ;*****

4064 7902          TIME:  MOV R1,#02
4066 785A          E2:    MOV R0,#90
4068 7E63          E3:    MOV R6,#99
406A 7F32          E4:    MOV R7,#50
406C DFFE          WAIT:  DJNZ R7,WAIT
406E DEFA          DJNZ R6,E4
4070 D8F6          DJNZ R0,E3
4072 D9F2          DJNZ R1,E2
4074 22            RET

4075              ;*****
4075              ;*** TABLA DE MENSAJES DEL EXHIBIDOR ***
4075              ;*****

4075 38            CONTEX: DB 38H
4076 00            DB 00H
4077 06            DB 06H
4078 0E            DB 0EH
4079 80            INIEXH: DB 80H
407A 00            DB 00H
407B 01            CLEAR:  DB 01H
407C 00            DB 00H

407D 20424945      BIENV:  DB ' BIENVENIDOS AL '
408D 00            DB 00H
408E 20534953      DB ' SISTEMA 200000 '
409E 00            DB 00H
409F 454E4D45      MEDIO   DB 'ENMEDIO'
40A6 00            DB 00H
0000              END

---- TABLA SIMBOL ---

```

BIEN	4012	DOBLEX	4030	FEXH	4039
QARNTA	4051	TIME	4064	BIENV	407D
E2	4066	INIEXH	4079	TEREXH	402E
WAIT	406C	CLEAR	407B	E3	4068
LIMPIA	4056	TERMIN	4050	XCBDOR	4046
COEXH	403A	E4	406A	MEDIO	409F
TI1600	405E	CONTEX	4075	EXHIBE	4000
P2	00A0	TIEMPO	4053		

INTERCONEXIONANDO DEL MICROCONTROLADOR CON UN TECLADO MATRICIAL DE 12 TECLAS

La utilización de un teclado para seleccionar una acción de control o para introducir las variables solicitadas por el sistema, es siempre indispensable.

Existen dos tipos de teclado los denominados matriciales y los que tienen un punto en común con todas las teclas. En este ejemplo se trabajará con uno del tipo matricial de 3 columnas por 4 renglones.

Cada vez que se oprime una tecla existen los denominados rebotes, los cuales son indeseables, debido a que se genera varias veces la demanda de la tecla oprimida. Lo cual ocasiona que el mismo valor de la tecla sea leído varias veces.

En algunos sistemas este inconveniente es anulado mediante un programa que verifica los valores eliminando los rebotes, en otros sistemas se realiza mediante circuitería. En este ejemplo haremos uso del circuito 74C922 que tiene la ventaja de eliminar internamente los rebotes, ayudado por dos capacitores que se le conectan externamente.

Este circuito tiene además la característica de poder manejar hasta un teclado matricial de 16 teclas (el 74C923 es de 20 teclas), y de conectarse directamente al bus de datos del microcontrolador ó microprocesador. Asimismo, mediante una señal de activación DA, nos indica cuando el dato está estable guardándolo durante todo el tiempo que así se requiera en fijadores internos.

El diagrama que muestra su conexión con el microcontrolador se presenta conjuntamente con el del exhibidor.

A continuación se muestra un pequeño programa que toma datos del teclado y los envía al exhibidor.

PROGRAMA PARA MANEJO DE TECLADO

```
; *****
; **  PROGRAMA PARA TOMAR DATOS DEL TECLADO  **
; **  Y ENVIARLOS AL EXHIBIDOR                **
; **  DIRECCIONES DEL EXHIBIDOR                **
; **      8000H = CONTROL DEL EXHIBIDOR        **
; **      8001H = EXHIBICION DEL DATO          **
; **  DIRECCION DEL TECLADO = 9000H            **
; **  EL DA ACTIVA LA INT0 DE MICROCONTR.     **
```



```

;*****
0000                                ORG 00H
9000          TEC:    EQU 9000H
8000          EXHI:   EQU 8000H
0000 020064          LJMP TECLAD
0003                                ORG 03H
0003 D200            SETB 20H.0
0005 909000          MOV DPTR,#TEC
0008 E0              MOVX A,@DPTR
0009 540F            ANL A,#0FH
000B 2430            ADD A,#30H ;VALOR ASCII
000D 32              RETI

;*****
;*****          PROGRAMA PRINCIPAL          *****
;*****
0064                                ORG 100
0064 758801          TECLAD: MOV TCON,#01H
0067 9000AE          MOV DPTR,#CONTRL
006A 7800            MOV R0,#00H
006C 1191            ACALL SUBEXH
006E 9000B6          MOV DPTR,#TEXT0
0071 118F            ACALL SUBEX1
0073 74C0            MOV A,#0C0H
0075 11A5            ACALL POSCUR
0077 9000C7          MOV DPTR,#TEXT1
007A 118F            ACALL SUBEX1
007C 3000FD          ESPTEC: JNB 20H.0,ESPTEC
007F C200            CLR 20H.0
0081 908001          MOV DPTR,#8001H
0084 F0              MOVX @DPTR,A
0085 7F20            MOV R7,#20H
0087 DFFE            TEX:   DJNZ R7,TEX

0089 7410            MOV A,#10H ; MUEVE EL CURSOR A LA
                                IZQ.
008B 11A5            ACALL POSCUR
                                ;SE REPITE EL DATO CONTINUAMENTE
008D 80ED            SJMP ESPTEC

;*****
;*****          SUBROUTINA DE EXHIBICION          *****
;*****
008F 7801            SUBEX1: MOV R0,#01H
0091 75A080          SUBEXH: MOV P2,#80H
0094 E4              SUBEX:  CLR A
0095 93              MOVC A,@A+DPTR
0096 600C            JZ FINEXH
0098 F2              MOVX @R0,A
0099 7A10            MOV R2,#10H
009B 79FF            LAZEX2: MOV R1,#0FFH
009D D9FE            LAZEXH: DJNZ R1,LAZEXH
009F DAFA            DJNZ R2,LAZEX2
00A1 A3              INC DPTR
00A2 80ED            SJMP SUBEXH
00A4 22              FINEXH: RET

00A5 908000          POSCUR: MOV DPTR,#EXHI
00A8 F0              MOVX @DPTR,A
00A9 79FF            MOV R1,#0FFH
00AB D9FE            LAZPOS: DJNZ R1,LAZPOS
00AD 22              RET

;*****

```

```
;*****          TEXTOS          *****
;*****
00AE 38010206  CONTRL: DB 38H,01H,02H,06H,0FH,80H,00H
00B5 00          DB 00H
00B6 4F505249  TEXTO:  DB 'OPRIMA UNA TECLA'
00C6 00          DB 00H
00C7 5445434C  TEXT1:  DB 'TECLA --> '
00D1 00          DB 00H
0000          END
```

---- TABLA SIMBOL ----

CONTRL	00AE	LAZEX2	009B	POSCUR	00A5	TCON
0088	TEXT1	00C7	ESPTEC	007C	LAZEXH	009D
SUBEX	0094	TEC	9000	TEXTO	00B6	EXHI
8000	LAZPOS	00AB	SUBEX1	008F	TECLAD	0064
FINEXH	00A4	P2	00A0	SUBEXH	0091	TEX
0087						

INTERCONEXIONANDO UN MOTOR DE PASOS CON EL MICROCONTROLADOR 8051

En muchas aplicaciones de Control Automático, es necesario el accionamiento de válvulas o sistemas de engranes con una exactitud y precisión muy alta. En Robótica, son indispensables éstas características, donde las manos y brazos mecánicos deben de ejecutar movimientos de gran precisión. Existen muchas otras ramas de la electrónica donde la utilización de dispositivos de posicionamiento mecánico son indispensables.

Un motor de pasos resuelve en gran medida este problema, ya que su principio de funcionamiento le permite realizar pequeños movimientos (pasos), con gran exactitud y repetibilidad.

El motor de pasos es un motor eléctrico cuyo eje gira una cantidad específica por cada pulso de entrada que recibe, lo cual permite el control de posición, velocidad, y sentido (dirección).

Existen diferentes tipos de motores de pasos, de los cuales veremos el funcionamiento de uno ellos, el Motor de Magneto Permanente.

En la figura 1, se muestra un diagrama del Motor de pasos de Magneto Permanente, el cual está construido de un rotor que incluye polos magnéticos de polaridad contraria colocados uno junto al otro. El estator contiene bobinas alineadas de tal forma que sus energización secuencial provoca que el rotor se desplace a las posiciones de mínima reluctancia magnética.

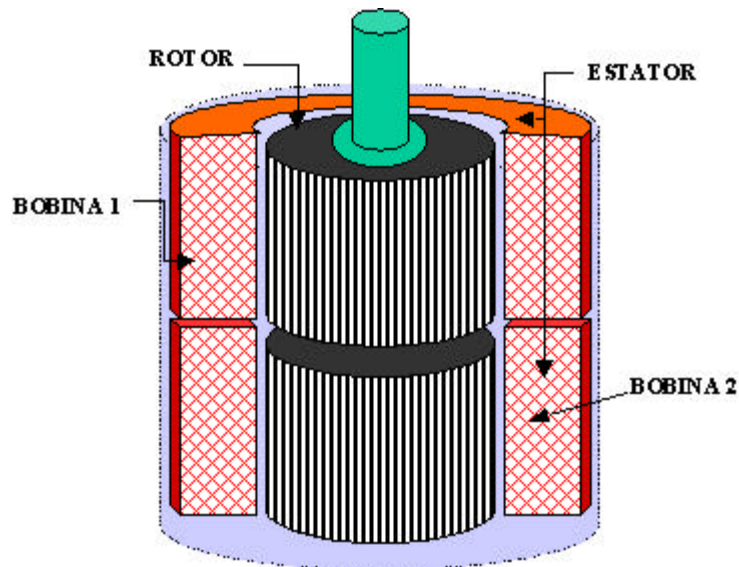


Fig. 1 Motor de imán Permanente

El número de pasos varía según sea la aplicación que se requiera. Existen en el mercado desde, 0.1 a 120 grados. Los ángulos más comunes son de 1.8, 2.0, 2.5, 5.0, 15 y 30 grados, que respectivamente dan 200, 180, 144,72, 24 y 12 pasos/revolución. Los motores de pasos son alimentados con fuentes de corriente directa y manejados con circuitería lógica.

PRINCIPIO DE OPERACIÓN DEL MOTOR.

Su diagrama se muestra en la figura 2, y su principio de funcionamiento es el siguiente:

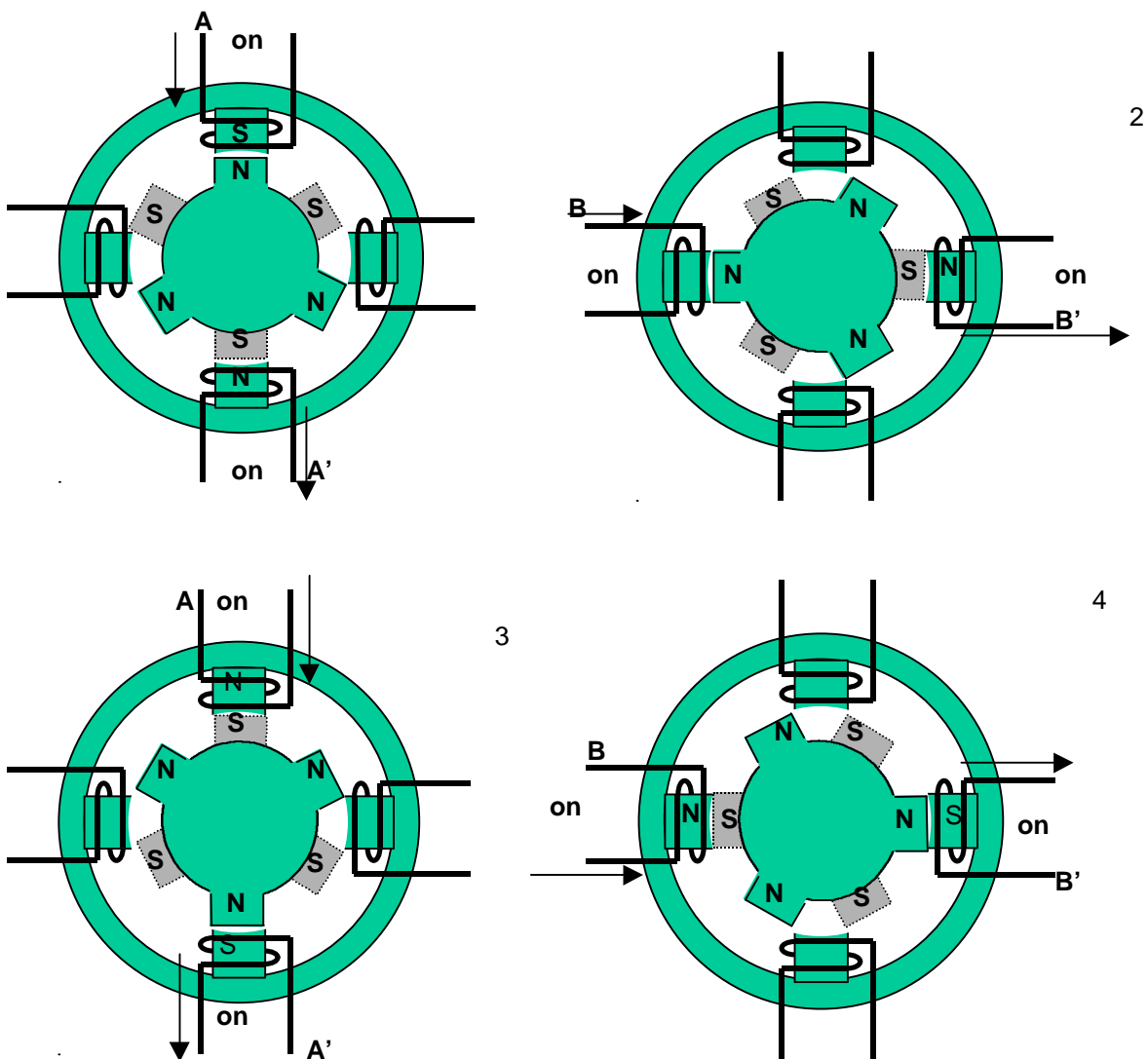


Fig. 2 Principio de funcionamiento de un motor de pasos

Consideremos primeramente, la bobina A-A' la cual se energiza haciendo circular la corriente de A hacia A' por lo cual el polo inferior se polariza positivamente, atrayendo al polo sur del rotor (parte inferior del rotor), y el superior negativamente el cual atrae, al polo norte del rotor, (parte superior del rotor), de tal manera que se realiza el primer paso .

Para que realice un segundo paso en el mismo sentido consideremos, ahora, la desactivación de la bobina A-A' y la activación de las bobinas B-B', al circular la corriente la bobina B polariza negativamente lo que atrae al polo positivo del rotor (N) y la bobina B' se polariza positivamente atrayendo al polo negativo (S) del rotor. Esto hace que gire un otro paso.

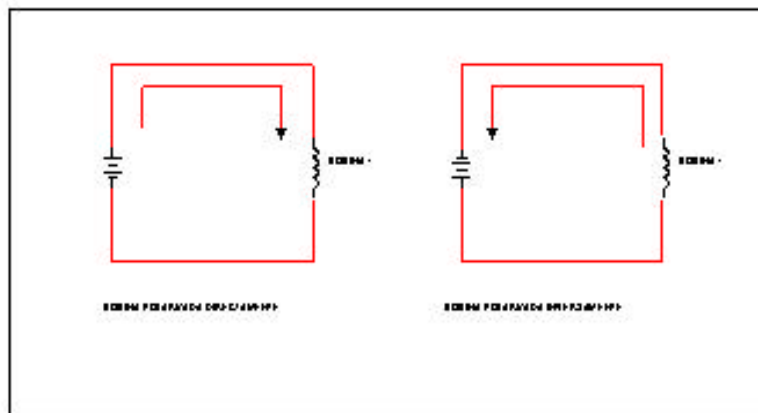
Para que gire un tercer paso, desenergizamos las bobinas B-B' y ahora volvemos a activar las bobinas A-A', pero ahora haciendo circular la corriente de A' hacia A por lo cual el polo superior se polariza positivamente, atrayendo al polo sur del rotor (parte inferior del rotor), y el inferior negativamente, el cual atrae, al polo norte del rotor (parte superior del rotor).

Siguiendo con un cuarto paso en el mismo sentido, sea la bobina B-B', ahora alimentada en sentido contrario, es decir haciendo circular la corriente de B' hacia B, con la bobina A-A' desactivada. El polo de la derecha se polariza positivamente atrayendo al polo sur del rotor (parte inferior del rotor), y el de la izquierda negativamente, el cual atrae, al polo norte del rotor, (parte superior del rotor).

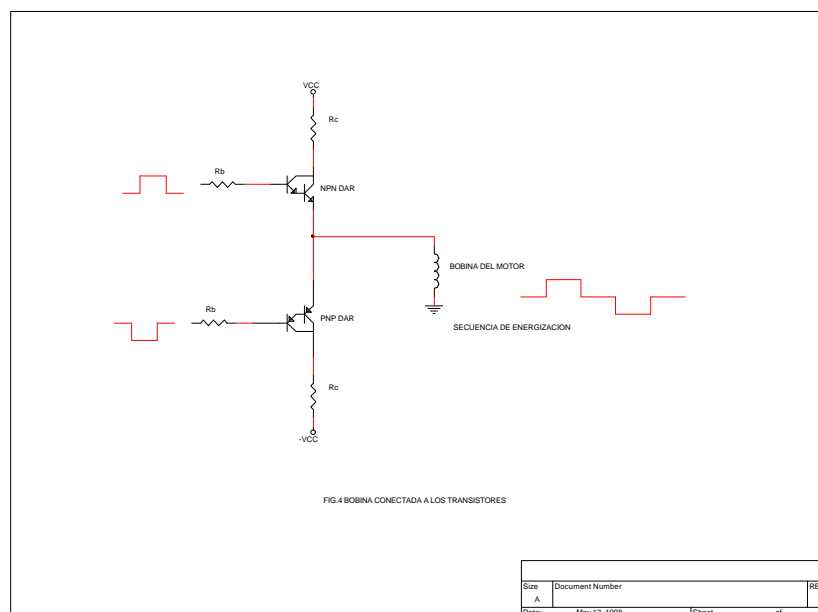
Por último para completar el ciclo, volvamos a conectar las cuatro bobinas como al inicio por lo cual el rotor girará un cuarto de paso. Si el ciclo se vuelve repetitivo podremos observar el movimiento del rotor en sentido de las manecillas del reloj. La velocidad dependerá de la activación y desactivación de las bobinas. Como podemos observar 4 pasos fueron necesarios para que el motor girara 90 grados de su posición original, si quisiéramos que el motor girara una vuelta completa tendríamos que darle 12 pasos, es decir repetir el ciclo 4 veces.

De aquí se observa que, el número de pasos de un motor está condicionado al número de polos del imán permanente (rotor), o bien al número de bobinas, (estator). V. gr. para un motor de 2 bobinas y 200 pasos, se requiere que el rotor tenga 50 polos.

Como cada una de las bobinas deben ser energizadas en los dos sentidos, (fig. 3).



Un circuito que se propone para la activación de estas bobinas es la figura 4.



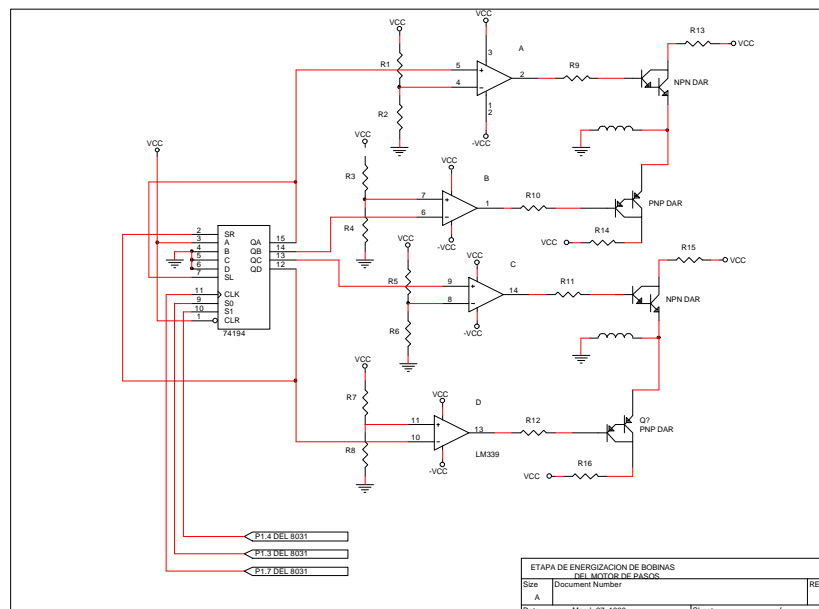
Cuando el transistor 1, es activado la parte A de la bobina queda energizada positivamente y la A' negativamente. Cuando el transistor 1 se desactiva y el transistor 2, se activa, la parte A se energiza negativamente y la A' positivamente, de esta manera se invierten las polaridades. Este circuito se presenta para cada una de las bobinas.

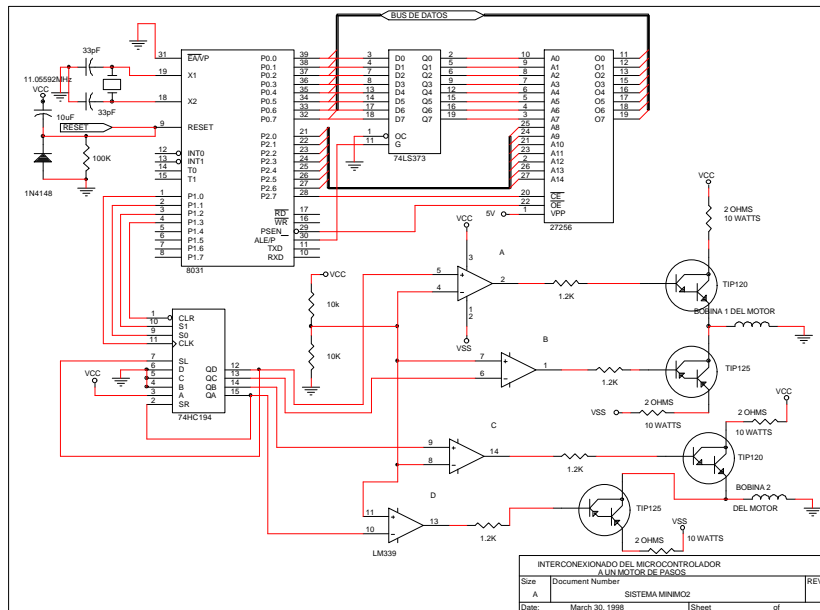
El arreglo en general del circuito se presenta en la figura 4, en donde, se puede apreciar el circuito 74194 el cual es un registro de corrimiento de 4 bits con su siguiente lógica de control:

CLR	S0	S1	CLK	DEFINICIÓN
0	X	X	X	Deshabilita el circuito
1	0	0	X	No cambian las salidas
1	0	1	↑	Desplaza a la Derecha en serie
1	1	0	↑	Desplaza a la izquierda en serie
1	1	1	↑	Carga el dato en paralelo

TABLA DE ESTADOS LÓGICOS

Ahora interconectaremos el microcontrolador 8051 al circuito de control del motor de pasos (fig. 5), como podemos apreciar se utilizan las líneas de P1.0 a P1.3, para poder controlar el CLR, S0, S1 y CLK respectivamente. La primera línea, CLR, desactiva el circuito, es decir, ninguna de las dos bobinas se encuentran energizadas. Mediante S0=1, S1=1 y un transiente positivo por el CLK, el dato paralelo es cargado. Un "1" es cargado en una sola de las líneas de salida y las demás con "0" con el fin de que solo sea energizada una bobina por paso y en un solo sentido. El sentido de giro del motor se debe al valor de de S0 y S1.





PROGRAMA DE CONTROL DE MOTORES A PASOS

```

;*****
; * PROGRAMA PARA CONTROL DE MOTORES DE PASOS**
; *** Se utiliza el T0 como base de tiempo ***
; *** T0 = 10,000 mseg. ***
; *** 100 pasos por segundo, 1 Revol.= 2seg***
; *** Primero girara durante 10 segundos a ***
; ** la derecha,despues parara durante 2seg ***
; *a continuaci " n, girara hacia la izquierda*
; ** Repetira la secuencia indefinidamente ***
; *** P1.0 = SEÑAL DE CONTROL DEL CLK *****
; *** P1.1 = CONTROL DE LA SEÑAL S1 *****
; *** P1.2 = CONTROL DE LA SEÑAL S0 *****
; *** P1.3 = CONTROL DE LA SEÑAL DE CLR *****
;*****

0000          ORG 00H
0000 020012   LJMP MOTOR
000B          ORG 0BH
000B 758CD8   MOV TH0,#0D8H ; SE RECARGA EL T0 CON
000E 758AF0   MOV TL0,#0F0H ;LA BASE DE T=-10,000
0011 32       RETI
0012 758CD8   MOTOR: MOV TH0,#0D8H ; SE CARGA CON LA BASE
0015 758AF0   MOV TL0,#0F0H ;DE TIEMPO =
                                ;10000useg.
0018 C293     CLR P1.3 ;SE DESACTIVA EL MOTOR
001A D293     SETB P1.3 ;SE VUELVE ACTIVAR
001C C290     CLR P1.0 ;SE CARGA EL VALOR INICIAL
001E D290     SETB P1.0 ;DE CORRIMIENTO "0001"

;*****
; **EL MOVIMIENTO SE REALIZARA A LA DERECHA **
;*****
;***** S0=1 y S1=0 *****
;*****
0020 758901   MOV TMOD,#01H ;SE ESTABLECE T0 COMO

```



```

0023 758810      MOV TCON,#10H ;TEMPORIZADOR EN MODO 1
0026 75A882      MOV IE,#82H  ;SE PERMITE LA INTER.T0

0029 C291      REPITE: CLR P1.1    ;S1=0

002B D292      SETB P1.2    ;S0=1
                                ;SE LLAMA A LA RUTINA DE ;PASOS
002D 113B      ACALL PSS
                                ;SE LLAMA A LA RUTINA DE PARO
002F 114B      ACALL PARO
0031
                ;*****
;* EL MOVIMIENTO SE REALIZARA A LA IZQUIERDA*
;*****      S0=0 y S1=1      *****
;*****
0031 C292      CLR P1.2    ;S0=0
0033 D291      SETB P1.1    ;S1=1
                ;SE LLAMA A LA RUTINA DE PASOS
0035 113B      ACALL PSS
                ;SE LLAMA A LA RUTINA DE PARO
0037 114B      ACALL PARO
                ;SE REPITE LA SECUENCIA DE MANERA INDEFINIDA
0039 80EE      SJMP REPITE

                ;*****
;*** RUTINA DE MOVIMIENTO DEL MOTOR *****
;*****


003B 7B07      PSS:   MOV R3,#07H    ;R2R3 COMO CONTADORES DE
003D 7AD0      PAS1:  MOV R2,#0D0H   ;2000 PASOS=10 VUELTAS
003F 308DFD    PASO:  JNB TCON.5,PASO ; ESPERA 10 mseg.
0042 C290      CLR P1.0    ;SE ENVIA UN PULSO DE
0044 D290      SETB P1.0    ;RELOJ "CLK"
0046 DAF7      DJNZ R2,PASO
0048 DBF3      DJNZ R3,PAS1
004A 22        RET

                ;*****
;***** RUTINA DE PARO DEL MOTOR *****
;*****

004B C293      PARO:  CLR P1.3    ; SE DESACTIVA EL MOTOR
004D 7C14      MOV R4,#20    ; CONTADORES PARA 2 SEG.
004F 7D00      MOV R5,#00    ; DE PARO TOTAL DEL
0051 7E00      MOV R6,#00    ; MOTOR.
0053 DEFE      TIEMPO: DJNZ R6,TIEMPO
0055 DDFC      DJNZ R5,TIEMPO
0057 DCFA      DJNZ R4,TIEMPO
0059 D293      SETB P1.3    ;SE VUELVE ACTIVAR EL M.
005B C290      CLR P1.0    ;SE CARGA EL VALOR INICIAL
005D D290      SETB P1.0    ;DE CORRIMIENTO "0001"
005F 22        RET
0000          END

```

EJEMPLO DE TRANSMISIÓN DE DATOS EN SERIE (INTERFASE RS-232C), DEL MICROCONTROLADOR 8051 HACIA UNA COMPUTADORA DEL TIPO PC-COMPATIBLE CON IBM.



Regreso al
menú principal

Un parámetro importantísimo, que debe ser tomado en cuenta, al inicio de todos los diseños, es la capacidad de almacenamiento de información de los dispositivos principalmente en los sistemas de adquisición de datos, donde, el tratamiento, análisis y cálculo de resultados es indispensable.

El almacenar datos en memoria RAM, es relativamente sencillo y fácil de realizar, la rapidez de acceso de estos circuitos, permite procesar datos en tiempo real, más sin embargo, su capacidad de almacenamiento es relativamente limitada. Por otro lado, aunque de menor velocidad de acceso, los discos flexibles y discos duros, ofrecen una capacidad mucho mayor de almacenamiento.

Por tal motivo, en este ejemplo se plantea la transmisión de datos, de la memoria RAM hacia una computadora del tipo IBM-PC compatible, haciendo uso del puerto serie del microcontrolador y su interconexión a la interfase del tipo RS-232C.

El protocolo de transmisión de datos en serie, contempla lo siguiente:

El primer bit que se transmite es el bit de inicio (Start bit), el cual tiene la característica de ser siempre un “0” lógico.

Posteriormente los bits que le siguen son los 8 bits de datos, comenzando con el menos significativo “D0”.

La transmisión termina con el envío de un último bit, denominado bit de paro (Stop bit), que siempre será un “1” lógico.

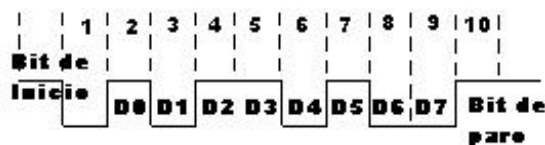


Fig. 1 Flujo de datos transmitidos en serie

El microcontrolador 8051, cuenta con 4 modos de transmisión de datos en serie, de los cuales utilizaremos, el modo 1, es decir, con el “Baud Rate” variable dependiendo del valor almacenado en TH1, con un reloj de 11.0592 MHz.

Las características de transmisión son las siguientes:

FRECUENCIA DE TRANSMISIÓN	9600 Hz.
NÚMERO DE BITS POR CARCTER	8 BITS
BITS DE INICIO	1 BIT
BITS DE PARO	1 BIT
SIN PARIDAD	1 BIT

El ICL-232 es un doble emisor/receptor, es decir contiene dos emisores que convierten los niveles de entrada TTL/CMOS a niveles de salida RS-232C (-/+ 10V), y dos receptores que efectúan la operación inversa a niveles TTL/CMOS (0-5V). Su ventaja principal es que para su funcionamiento requiere solamente de una fuente de alimentación, (5V) ya que internamente contiene dos convertidores de voltaje de CD/CD (fig. 2).

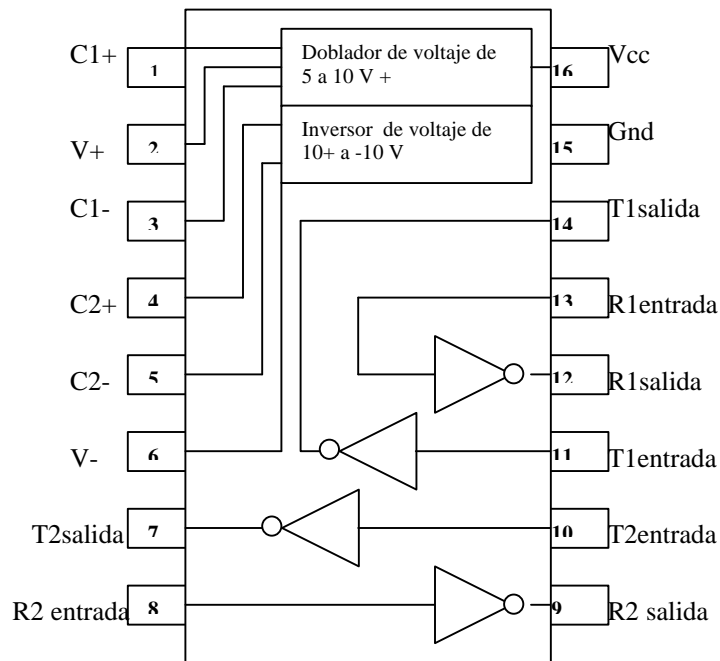
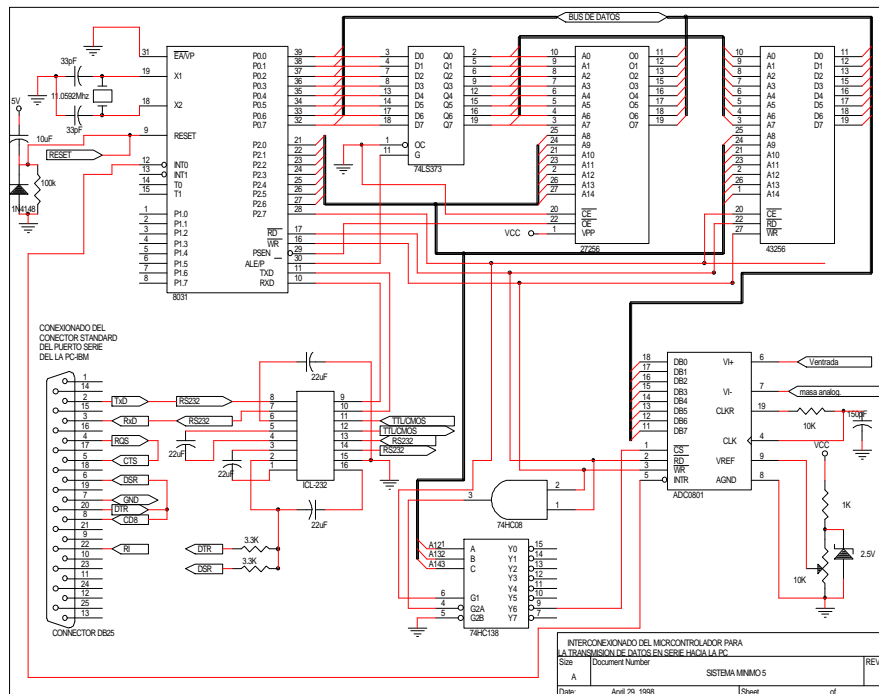


Fig. 2 ICL232 (de intel)

El ejemplo propone la adquisición de datos por medio de un convertidor analógico digital el ADC0801, el cual adquiere datos cada 3 segs. Y el microcontrolador los almacena en una memoria RAM de 32k. Cada 24 horas los datos son enviados a la computadora vía el puerto serie, el número total de datos enviados es 28800 (20xmin * 60xhora * 24 horas).

El diagrama 1, muestra el interconexionado del microcontrolador 8051 con el C D/A ADC0801, el circuito reforzador ICL232 o MAX-232 y la conexión con el puerto serie de la computadora IBM-PC o compatible.



PROGRAMA DE TRANSMISIÓN DE DATOS POR EL PUERTO SERIE

```

;*****
;*****  PROGRAMA DE TRANSMISION DE DATOS  *****
;* POR EL PUERTO SERIE DEL MICROCONTROLADOR *
;** CON EL CONVERTIDOR A/D SE TOMAN DATOS  **
;** CADA 3 SEGUNDOS Y SE ALMACENAN EN RAM  **
;** DURANTE 24 HORAS SE TIENEN 28800 DATOS **
;** DESPUES SON TRANSMITIDOS POR EL PUERTO **
;* SERIE. SE UTILIZA UN RELOJ DE 11.0592MHZ *
;*****

```

```

0000      ORG 00H
0000      BANDER: EQU 20H.0
0001      BANTEM: EQU 20H.1
00E0      CAD:    EQU 0E0H ;DIRECCION DEL C-A/D
0000      MEMRAM: EQU 0000H
0000 020100      LJMP TRANSMI

```

```

;*****

```

```

;*****      RUTINA DE SERVICIO DE LA      *****
;*****      INTERRUPCION 0      *****
;*****
0003      ORG 03H
0003 D200      SETB BANDER
0005 E2      MOVX A,@R0 ;SE ADQUIERE EL DATO CA/D
0006 F0      MOVX @DPTR,A ;SE ALMACENA EL DATO RAM
0007 32      RETI

;*****
;*****      RUTINA DE SERVICIO DE LA      *****
;*****      INTERRUPCION DEL TIMER 0      *****
;*****
000B      ORG 0BH
000B 758C1F      MOV TH0,#01FH ;SE CARGA LA BASE DE
000E 758A00      MOV TL0,#00H ;TIEMPO 62,500 uSeg.
0011 DC04      DJNZ R4,SALE
0013 D201      SETB BANTEM
0015 7C30      MOV R4,#30H ;30H=48
0017 32      SALE: RETI

;*****
;*****      RUTINA DE SERVICIO DE LA      *****
;*****      INTERRUPCION DEL PUERTO SERIE      *****
;*****
0023      ORG 23H
0023 E0      MOVX A,@DPTR
0024 A3      INC DPTR
0025 32      RETI

0100      ORG 100H
0100 75A0E0      TRANSMI:MOV P2,#CAD
0103 7800      MOV R0,#00H
0105 7C30      MOV R4,#30H
0107 758DFD      MOV TH1,#0FDH ;PARA GENERAR EL BAUD
010A 758BFD      MOV TL1,#0FDH ;RATE A 9600
010D 900000      REPIT: MOV DPTR,#MEMRAM
0110 758C1F      MOV TH0,#01FH ;SE CARGA LA BASE DE
0113 758A00      MOV TL0,#00H ;TIEMPO 62,500 uSeg.
0116 758921      MOV TMOD,#21H ;CON RELOJ DE 11.059
;SE ACTIVA EL T0 EN MODO 1 Y T1 EN MODO 2
0119 758851      MOV TCON,#51H ;INT0 POR TRANSCIENTE
011C 75A883      MOV IE,#83H ;SE PERMITE LA INT T0
011F 7F70      MOV R7,#70H ;R7R6 CONTADORES DE
0121 7E80      LAZ0: MOV R6,#80H ;28800 DATOS
0123 3001FD      TIAD: JNB BANTEM,TIAD ;TIEMPO DE ADQUIS.
0126 C201      CLR BANTEM ;DE 3 seg.
0128 E2      MOVX A,@R0 ;COMIENZA LA CONVERSI.
0129 3000FD      TICONV: JNB BANDER,TICONV ;TIEMPO DE CONVER.
012C C200      CLR BANDER
012E A3      INC DPTR
012F DEF2      DJNZ R6,TIAD

```

```
0131 DFEE          DJNZ R7,LAZ0
0133 C2A8          CLR IE.0  ;DEACTIVA LAS INTERRUPT.
0135 C2A9          CLR IE.1  ;DEL T0 Y INT0
                   ;SE TERMINA LA ADQUISICION DE DATOS Y
                   ;COMIENZA LA TRANSMISION DE DATOS.
0137 759840        MOV SCON,#40H ;MOD01 DE TRANSMISION
013A D2AC          SETB IE.4   ;SE PERMITE INTERRUPTCION
013C 900000        MOV DPTR,#MEMRAM ;DEL PUERTO SERIE
013F E0           MOVX A,@DPTR ;SE CARGA 1er.DATO
0140 7F70          MOV R7,#70H ;SE RECARGAN LOS CONTADS.
0142 7E80          LAZ2: MOV R6,#80H ;CON 28800
0144 F599          LAZ3: MOV SBUF,A  ;SE TRANSMITE EL DATO
0146 3099FD        ESPTR: JNB SCON.1,ESPTR
0149 C299          CLR SCON.1
014B DEF7          DJNZ R6,LAZ3
014D DFF3          DJNZ R7,LAZ2
                   ;EL CICLO SE VUELVE A REPETIR INDEFINIDAMENTE
014F 80BC          SJMP REPIT
0000              END
```

IMPRESIÓN DE DATOS MEDIANTE EL MICROCONTROLADOR 8051, HACIENDO USO DE LA INTERFASE PARALELO TIPO CENTRONICS

Regreso al
menú principal

En muchos sistemas de adquisición y control de procesos es necesario tener por escrito una hoja de resultados, que nos indique los valores de las variables medidas o características del proceso, v.gr., si hubo algún artefacto o nivel sobrepasado, el tiempo en que se realizó tal o tal evento, el número de muestras, etc.

Por tal motivo una solución, es el envío de los resultados obtenidos por nuestro sistema, hacia una computadora, en la cual mediante un programa específico, adecue la información, para posteriormente ser enviada hacia una impresora obteniéndose la hoja deseada.

Desafortunadamente, el tiempo que se pierde en enviar los datos hacia la computadora, aunado a la necesidad de tener una computadora libre cada vez que se requieran imprimir los resultados, hace de esto, un medio tedioso y algunas veces molesto para los usuarios.

Por tal motivo, en éste proyecto se propone interfazar directamente la impresora al sistema basado en el micro- controlador 8051.

En la tabla 1 se puede apreciar las señales características de la interfase paralelo del tipo Centronics: Asimismo en la fig. 1 se puede apreciar las señales que se activan en el momento de la impresión de datos.

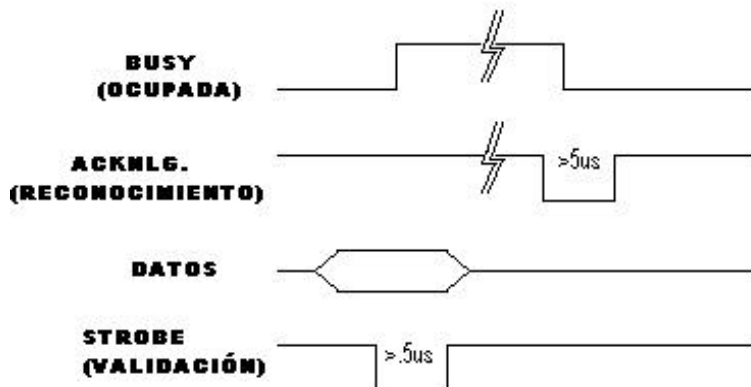


fig 1 Señales de activación de la impresora

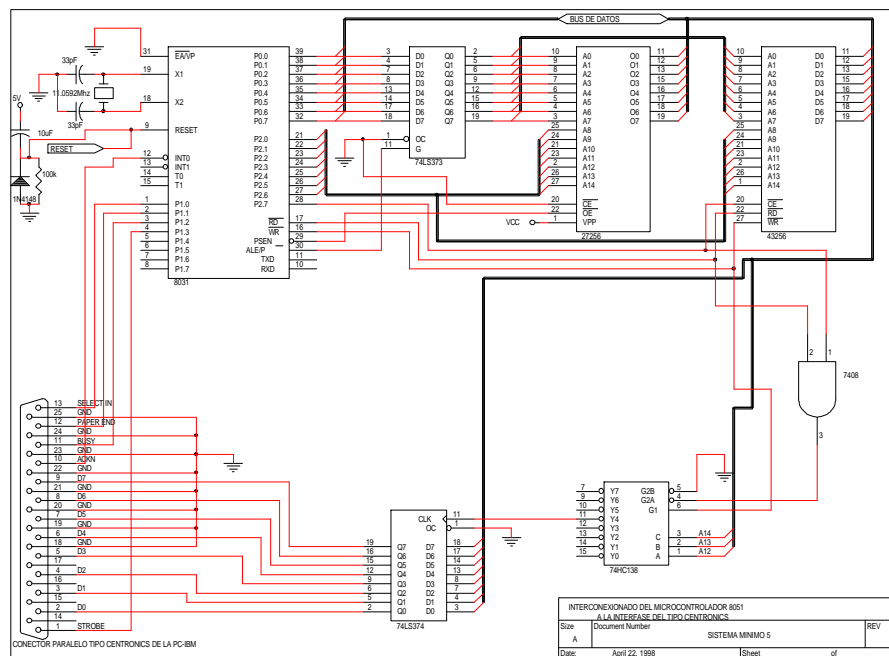
Terminal	Nombre	Función
1	Stro	Pasa de alto a bajo ($>.5\mu s$) cuando se activa
2	D0	Dato 0 del bus de datos
3	D1	Dato 1 del bus de datos
4	D2	Dato 2 del bus de datos
5	D3	Dato 3 del bus de datos
6	D4	Dato 4 del bus de datos
7	D5	Dato 5 del bus de datos
8	D6	Dato 6 del bus de datos
9	D7	Dato 7 del bus de datos
10	ACKN	Pulso de reconocimiento de dato $5\mu s$ bajo

11	BUSY	Señal de ocupada Baja si la impresora está lista
12	P.E.	Alto cuando no existe papel (Paper End)
13	SELE	Alto cuando la impresora está en línea
14	AUTO	Autoalimentación, sólo en algunas impresoras
15	ERRO	Señal de error. La impresora no puede continuar
16	INIT	Cuando se baja se inicializa la impresora
17	SELE	Selección. Alta si la impresora está en línea
18	GND	Tierra
19-25	GND	Tierra

Tabla 1. Terminales del conector paralelo.

A continuación se muestra el diagrama de interconexión del puerto paralelo tipo Centronics, con el microcontrolador 8031, al igual que cada una de sus terminales.

Posteriormente, se muestra el programa para imprimir tres letreros, utilizando control de carro CR, alimentadores de línea, así como tabuladores horizontales.



PROGRAMA PARA IMPRESIÓN DE DATOS

```

;*****
;**** PROGRAMA PARA IMPRESIÓN DE DATOS ****
;**** UTILIZANDO UN CONECTOR PARALELO ****
;**** TIPO CENTRONICS ****
;**** UTILIZA CUATRO TERMINALES DE P1 ****
;**** Y UN FIJADOR (74374) COMO PERIFE- ****
;**** RICO DE SALIDA DE DATOS. DIR=C000H****
;**** P1.0 = SELECTOR ****
;**** P1.1 = PAPER END (NO HAY PAPEL ****
;**** P1.2 = BUSY (OCUPADO) ****
;**** P1.3 = STROBE (VALIDACION) ****
;**** INT0 = ACKNOWLEDGE (RECONOCIM) ****
;*****

0000 ORG 00H
0000 010B AJMP IMPRES
0003 ORG 03H

0003 D200 SETB 20H.0
0005 F2 MOVX @R0,A
0006 C293 CLR P1.3 ;SE ESTABLECE EL
STROBE
0008 D293 SETB P1.3
000A 32 RETI

;*****
;** ESTE PROGRAMA CONTROLARA LA IMPRESORA **
;** PRIMERO SALTARA 4 LINEAS, DESPUÉS ES- **
;** CRIBIRÁ "BIENVENIDOS AL SISTEMA 2000 **
;** VOLVERÁ A SALTAR 2 LINEAS DARA UN TAB **
;** Y ESCRIBIRA "ESTA ES UNA PRUEBA DE **
;** IMPRESIÓN DE DATOS", VOLVERÁ A SALTAR **
;** 2 LINEAS DARA UN TAB Y ESCRIBIRA "PARA**
;** IMPRIMIR LOS CARACTERES HAY QUE ENVIAR**
;** SU CÓDIGO ASCII", POR ÚLTIMO SALTARÁ 2**
;** LÍNEAS DARA UN TAB Y ESCRIBIRA **
;** "123456789" **
;*****

000B 901000 IMPRES: MOV DPTR,#TABLA
000E 75A0C0 MOV P2,#0C0H
0011 1140 ACALL LF
0013 1140 ACALL LF
0015 1140 ACALL LF
0017 1140 ACALL LF
0019 1152 ACALL HT
001B 1164 ACALL IPRE
001D 1140 ACALL LF
001F 1140 ACALL LF

```

```

0021 90101C      MOV DPTR,#PRLE
0024 1152        ACALL HT
0026 1164        ACALL IPRE
0028 1140        ACALL LF
002A 1140        ACALL LF
002C 1152        ACALL HT
002E 901044      MOV DPTR,#SELE
0031 1164        ACALL IPRE
0033 1140        ACALL LF
0035 1140        ACALL LF
0037 1152        ACALL HT
039 90107F      MOV DPTR,#NUMER
003C 1164        ACALL IPRE
003E 80FE      TERMIN: SJMP TERMIN

0040 740D      LF:      MOV A,#0DH
0042 F2        MOVX @R0,A
0043 C293      CLR P1.3      ;SE ESTABLECE EL STROBE
0045 D293      SETB P1.3
0047 740A      MOV A,#0AH
0049 2092FD    OCUP0:   JB P1.2,OCUP0 ;OCUPADA LA IMPRESORA
004C 3000FD    REC0:   JNB 20H.0,REC0 ;ESPERA RECONOCIMIENTO
004F C200      CLR 20H.0
0051 22        RET

0052 740D      HT:      MOV A,#0DH
0054 F2        MOVX @R0,A
0055 C293      CLR P1.3      ;SE ESTABLECE EL STROBE
0057 D293      SETB P1.3
0059 7409      MOV A,#09H
005B 2092FD    OCUP:   JB P1.2,OCUP   ;OCUPADA LA IMPRESORA
005E 3000FD    REC:   JNB 20H.0,REC   ;ESPERA RECONOCIMIENTO
0061 C200      CLR 20H.0
0063 22        RET

0064 E0        IPRE:    MOVX A,@DPTR
0065 6017      JZ FIN
0067 F2        CARGA:   MOVX @R0,A
0068 C293      CLR P1.3      ;SE ESTABLECE EL STROBE
006A D293      SETB P1.3
006C 3090FD    ESPLIN:  JNB P1.0,ESPLIN ;ESPERA QUE SE PONGA
                                ;EN LINEA LA IMPRESORA

006F 2091FD    NHPAP:   JB P1.1,NHPAP ;NO HAY PAPEL
0072 2092FD    OCUPA:   JB P1.2,OCUPA ;OCUPADA LA IMPRESORA
0075 3000FD    RECO:   JNB 20H.0,RECO ;ESPERA RECONOCIMIENTO
0078 C200      CLR 20H.0
007A A3        INC DPTR
007B E0        MOVX A,@DPTR
007C 70EE      JNZ ESPLIN
007E 22        FIN:    RET
007F

```

```
1000                                ORG 1000H
1000 4249454E  TABLA:  DB 'BIENVENIDOS AL SISTEMA 2000 '
101B 00                                DB 00H
101C 45535441  PRLE:  DB 'ESTA ES UNA PRUEBA DE IMPRESIÓN'
103B 44452044                                DB 'DE DATOS '
1043 00                                DB 00H
1044 50415241  SELE:  DB 'PARA IMPRIMIR LOS CARACTERES HAY '
1064 51554520                                DB 'QUE ENVIAR SU CÓDIGO ASCII '
107E 00                                DB 00H
107F 30                NUMER:  DB 30H
1080 31                                DB 31H
1081 32                                DB 32H
1082 33                                DB 33H
1083 34                                DB 34H
1084 35                                DB 35H
1085 36                                DB 36H
1086 37                                DB 37H
1087 38                                DB 38H
1088 39                                DB 39H
1089 00                                DB 00H
0000                                END
```