

CAPITULO 1

CARACTERÍSTICAS DEL AT89C52

La familia de microcontroladores de Intel conocida como MCS-51 represento el despegue en el uso y aplicación de los microcontroladores, los miembros de esta familia se encuentran en diversas presentaciones, tanto en formas físicas como en características, la selección de uno o de otro tipo de microcontrolador dependerá principalmente de las necesidades a satisfacer; diversos fabricantes de semiconductores tienen, basados en esta familia, sus propios derivados. En este documento se tratará de manera particular del ATMEL AT89C52, el cual se presenta con 8 Kbytes en la versión EEPROM.

Es necesario relacionar a este microcontrolador con los de la familia MCS-51, que están basados en los microprocesadores de 8 bits; contienen internamente un CPU de 8 bits, 4 puertos de entrada y salida paralelo completamente programables en forma individual como salidas o entradas, de los cuales el puerto 3 tiene también funciones de control con las particularidades siguientes: un puerto serie, 2 entradas para Timer/Contador de 16 bits, 2 entradas para interrupciones externas, las señales de RD y WR para la toma o almacenamiento de datos en RAM externa y en el puerto 1 se encuentra el timer 2. Además, cuenta también con 256 bytes de memoria RAM interna.

El AT89C52 es capaz de 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 o del 2. Dicha frecuencia de transmisión puede ser cambiada en cualquier momento con sólo cambiar el valor almacenado en el contador, o también se puede duplicar o dividir la frecuencia con sólo escribir directamente sobre el bit 7 (bit SMOD) del registro de control PCON.

En la tabla 1 se muestra la familia de microcontroladores MCS-51, en la que se listan las principales características de estos dispositivos.

| Nombre | Versión sin ROM | Versión EEPROM | Versión EPROM | Bytes en ROM | Bytes en RAM | Timers 16-Bits | Tipo de Circuito |
|--------|--------------------|-------------------|------------------|-----------------|-----------------|-------------------|---------------------|
| 8051 | AT89C52 | | - | 4k | 128 | 2 | NMOS |
| 80C51 | 80C31 | AT89C51 | 87C51 | 4k | 128 | 2 | HMOS |
| 8052 | 8032 | | - | 8k | 256 | 3 | NMOS |
| 80C52 | 80C32 | AT89C52 | 87C52 | 8k | 256 | 3 | CMOS |

Tabla 1. Familia de microcontroladores de la familia MCS-51 incluyendo a los ATMEL.

La figura 1 muestra el diagrama a bloques de un miembro de esta familia de microcontroladores. Se trata del μC AT89C52; este dispositivo por su bajo costo es el que se usará como elemento de control en los experimentos que aquí se proponen.

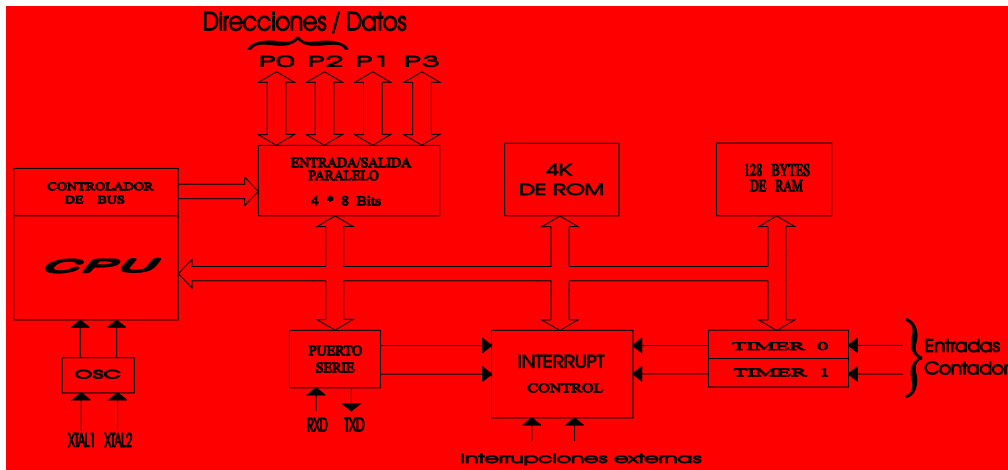


Figura 1. Diagrama a Bloques del μC 80C31.

1.1

DESCRIPCIÓN DE LAS LÍNEAS (PINES) DEL AT89C52

La presentación de este dispositivo puede encontrarse en tres tipos de encapsulado:

- **DIP** Dual In Line Package (figura 2).
- **PLCC** Plastic Leaded Chip Carrier
- **PQFP/TQFP**

La presentación **DIP** es utilizada para montaje de **Pin In Hole (PIH)**, en tanto que las presentaciones en **PLCC** y **PQFP/TQFP** son especiales para la Tecnología de Montaje Superficial (SMT).

La distribución de pines en el microcontrolador del tipo DIP, que en proyectos a nivel de aprendizaje es el mas apropiado se muestra en la siguiente figura 2.

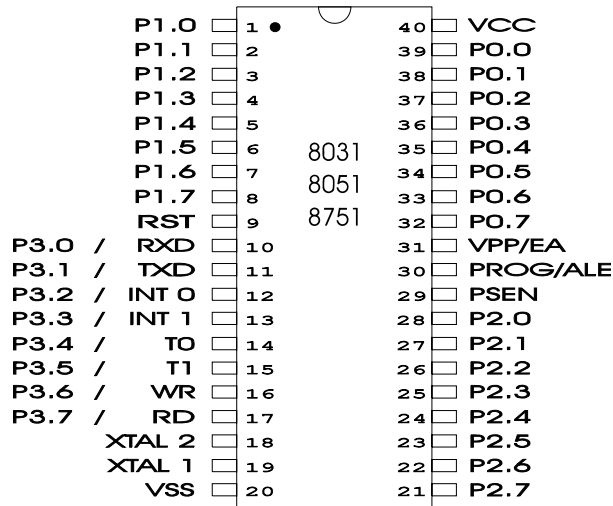


Figura 2. Distribución de los pines del AT89C52.

EL AT89C52 es un microcontrolador de 8 bits con 8 Kbytes de memoria flash de bajo consumo y gran desempeño (PEROM). Este dispositivo es fabricado usando la tecnología de memoria no volátil de ATMEL y es compatible con el estándar, muy usado en la industria, 80C51 y 80C52, en cuanto a sus pines de conexión y a su set de instrucciones.

La memoria flash contenida internamente permite que el programa contenido pueda ser modificado en “el sistema” o por cualquier convencional programador. Combinando la versátil CPU de 8 bits y la memoria Flash en un solo chip, este microcontrolador permite soluciones flexibles y de bajo precio a muchas aplicaciones de control.

Entre sus características más relevantes tenemos:

- Compatibilidad con los productos de la familia MCS-51
- 8K bytes de memoria reprogramable flash interna
- hasta 1000 ciclos de escritura/lectura
- operación completamente estática, desde 0 HZ hasta 24 MHZ
- tres niveles de seguridad para la protección de código
- Memoria interna RAM de 256 x 8-bit
- 32 líneas de entrada/salida programables
- Tres Timers/Contadores de 16 bits
- 8 fuentes de interrupción con 2 niveles de prioridad
- 1 puerto de comunicación serial
- 1 modos IDLE y de bajo consumo de energía

La Tabla 2 muestra la descripción de los pines que conforman al μC AT89C52, en el entendido que los miembros de la familia MCS-51 tienen las mismas características.

| MNEMÓNICO | CONEXIÓN | TIPO | NOMBRE Y FUNCIÓN |
|-----------|----------|------|--|
| VCC | 40 | | Alimentación positiva +5V. |
| VSS | 20 | E/S | Tierra referencia 0 Volts . |
| P0.0-P0.7 | 39-32 | E/S | PUERTO 0. Es un puerto I/O 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. Puede activar hasta 8 compuertas TTL y es el que recibe los datos en la programación necesita de resistencias pull-ups en la programación y en la verificación |
| P1.0-P1.7 | 1-8 | E/S | PUERTO 1. Es un puerto I/O bidireccional, con pullups internas. Puede activar hasta 4 compuertas TTL, cuando se escriben 1's en el puerto, éste puede ser utilizado como entrada en alta impedancia., además P1.0 y P1.1 pueden ser configurados como entradas del timer/contador 2, también recibe la dirección baja durante la programación y la verificación |
| P2.0-P2.7 | 21-28 | E/S | PUERTO 2. Es un puerto I/O bidireccional, con pullups internas. Puede activar hasta 4 compuertas TTL, cuando se escriben 1's en el puerto, éste puede ser utilizado como entrada en alta impedancia. Como entradas, las líneas que son externamente colocadas en la posición baja, proporcionarán 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 de programa externa y durante el acceso a memoria de datos externa que usan direccionamientos de 16 bits. Durante el acceso a una memoria de datos externa que usa direcciones de 8 bits, el puerto 2 emite el contenido del registro correspondiente a este puerto, que se encuentra en el espacio de funciones especiales. Recibe también las direcciones altas durante la programación y la verificación |
| P3.0-P3.7 | 10-17 | E/S | PUERTO 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 en alta impedancia. Como entradas, las líneas que son externamente colocadas en la posición baja proporcionarán una corriente hacia el exterior. El puerto 3 es utilizado además, para producir señales de control de dispositivos externos tales como: |
| RxD | 10 | E | (P3.0) Puerto serie de entrada. |
| TxD | 11 | S | (P3.1) Puerto serie de salida. |
| INT0 | 12 | E | (P3.2) Interrupción externa 0. |
| INT1 | 13 | E | (P3.3) Interrupción externa 1. |
| TO | 14 | E | (P3.4) Entrada externa timer 0. |

| | | | |
|-------|----|-----|---|
| T1 | 15 | E | (P3.5) Entrada externa timer 1. |
| WR | 16 | S | (P3.6) Habilitador de escritura para memoria externa de datos. |
| RD | 17 | S | (P3.7) Habilitador de lectura para memoria externa de datos. |
| RST | 9 | E | RESET. Una entrada alta en esta línea durante dos ciclos máquina, mientras el oscilador está funcionando, 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 del ALE es emitido durante cada acceso a la memoria de datos externos. Durante la programación este pin es la entrada del pulso de programación. |
| PSEN | 29 | S | PROGRAM STORE ENABLE. Habilitador de lectura para memoria de programa externa. Cuando el AT89C52 está ejecutando un código de una memoria de programa externa, PSEN es activada dos veces cada ciclo de máquina, excepto cuando se accesa a la memoria de datos externa que omiten las dos activaciones del PSEN externos. PSEN no es activado cuando se usa la memoria de programa interna. |
| EA | 32 | E | EXTERNAL ACCESS ENABLE. EA debe mantenerse externamente en posición baja para habilitar el mecanismo que elige el código de direccionamiento para memoria de programa externa, 0000H y 0FFFH. 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 exceda la dirección 1FFFH. Este pin recibe el pulso de 12 volts de Vpp durante la programación. |
| XTAL1 | 19 | E | CRISTAL 1. Esta es la entrada del cristal para el circuito oscilador (generador del reloj interno) que amplifica e invierte la entrada. |
| XTAL2 | 20 | S | CRISTAL 2. Es la salida del amplificador oscilador inversor |

Tabla 2. Configuración de los pines del μ C AT89C52.

XTAL1 Y XTAL2 son la entrada y la salida, respectivamente, de un amplificador inversor que puede ser configurado para su uso como un chip oscilador, como se muestra en la siguiente figura (Figura 3). Se puede utilizar indistintamente un cristal de cuarzo o un resonador cerámico.

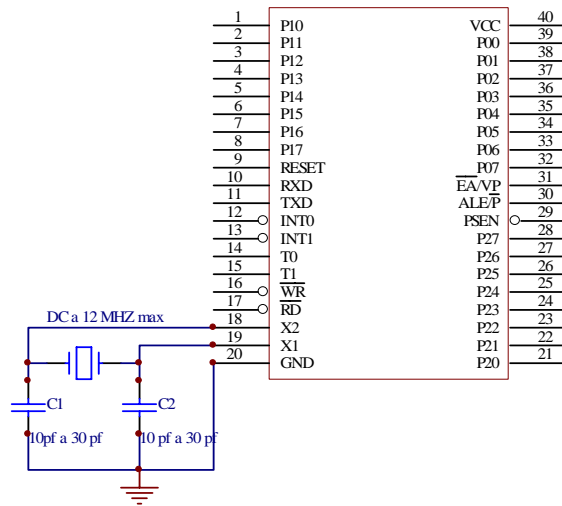


Figura 3. Circuito Oscilador

El RESET automático del microcontrolador, se produce al aplicar la tensión de alimentación (Vcc) al pin 9 (RST) a través de un capacitor de 10 μ F y una resistencia de 8.2 kohms, como se indica en la figura 4.

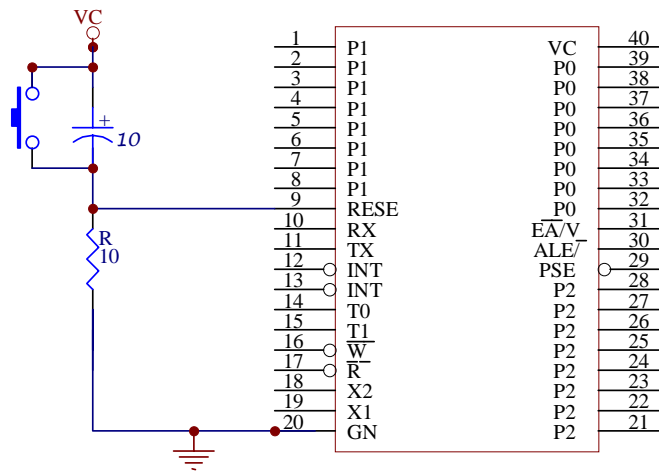


Figura 4. Reset del sistema.

1.3**CARACTERÍSTICAS ESPECÍFICAS DEL AT89C52**

Entre sus características mas relevantes tenemos:

- Compatibilidad con los productos de la familia MCS-51
- 8K bytes de memoria reprogramable flash interna
- hasta 1000 ciclos de escritura/lectura
- operación completamente estática, desde 0 HZ hasta 24 MHZ
- tres niveles de seguridad para la protección de código
- Memoria interna RAM de 256 x 8-bit
- 32 líneas de entrada/salida programables
- Tres Timers/Contadores de 16 bits
- 8 fuentes de interrupción con 2 niveles de prioridad
- 1 puerto de comunicación serial
- 1 modos IDLE y de bajo consumo de energía

1.4**DESCRIPCIÓN DE LOS ESPACIOS DE MEMORIA**

La memoria del sistema del AT89C52 se clasifica en tres tipos fundamentales como se describe a continuación.

La primera, llamada **MEMORIA DE PROGRAMA**, en donde se encuentran todas las instrucciones que van a ser ejecutadas por el μ C, es decir, el programa de trabajo. Algunas versiones del AT89C52 cuentan con memoria de programa interna (de 4KB en el AT89C51 y de 8KB en el AT89C52). 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 64 Kb, utilizando para esto los puertos P0 y P2.

El segundo espacio de memoria, denominado **MEMORIA DE DATOS**, es accedido mediante la activación de las señales RD y WR (P3.6 y P3.7 respectivamente), durante la lectura o escritura de datos respectivamente. En este espacio el μ C toma todos los valores que se encuentran en memoria como DATOS, es decir, el μ C no puede ejecutar ninguna instrucción que se encuentre aquí almacenada. El AT89C52 puede direccionar también 64 Kb de memoria de datos.

El tercer espacio de memoria es el denominado **MEMORIA RAM INTERNA**, el cual se subdivide en 256 bytes de memoria bajos y en 256 bytes de memoria altos. Los primeros 256 bytes, son registros utilizables por el usuario y 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 basto número de instrucciones del AT89C52. 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.

1.5

MEMORIA DE PROGRAMA

Como ya se mencionó, el AT89C52 cuenta con 8Kb de memoria PEROM interna, la cual puede ser accesada mediante la conexión de la línea EA a Vcc, de esta manera el CPU, automáticamente, leerá el programa del ROM interno, desde 0000H hasta 1FFFH, y si en algún caso el Program Counter llegara a exceder esta última dirección, las siguientes instrucciones serán buscadas en la memoria externa (a partir de 2000H hasta FFFFH).

Por el contrario, si la línea EA es conectada a tierra, el AT89C52 buscará el total de las instrucciones en la ROM externa (desde la dirección 0000H hasta FFFFH). **En el caso del entrenador presentado en este libro esta línea se conecta siempre a 5 volts (Vcc) ya que siempre se usa la memoria ROM INTERNA.**

En la figura 5 se muestra de una manera gráfica la disposición de ambos tipos de memorias.

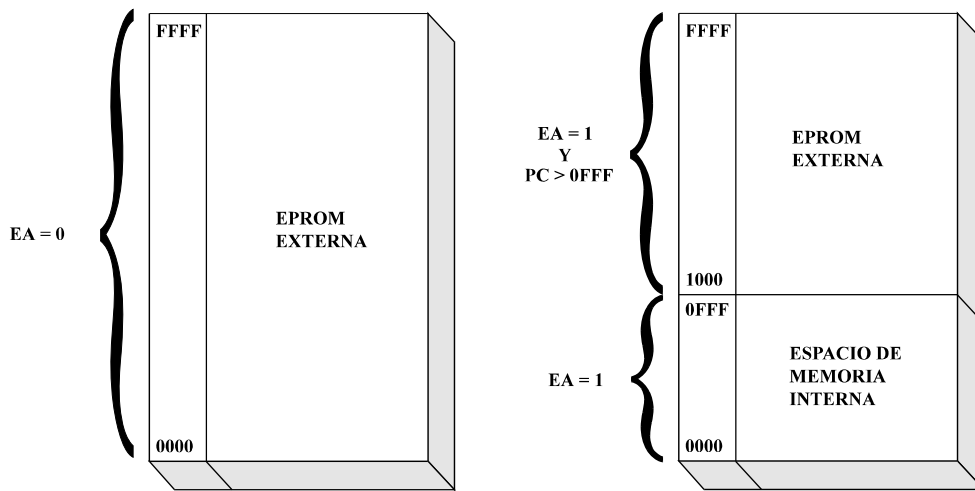


Figura 5. Organización de la Memoria de Programa

La línea PSEN (Program store enable), que sirve para leer el ROM externo, es activado en todas las búsquedas (fetches) del ROM externo. La figura 6, en el caso que se requiera, muestra una conexión a una EPROM externa.

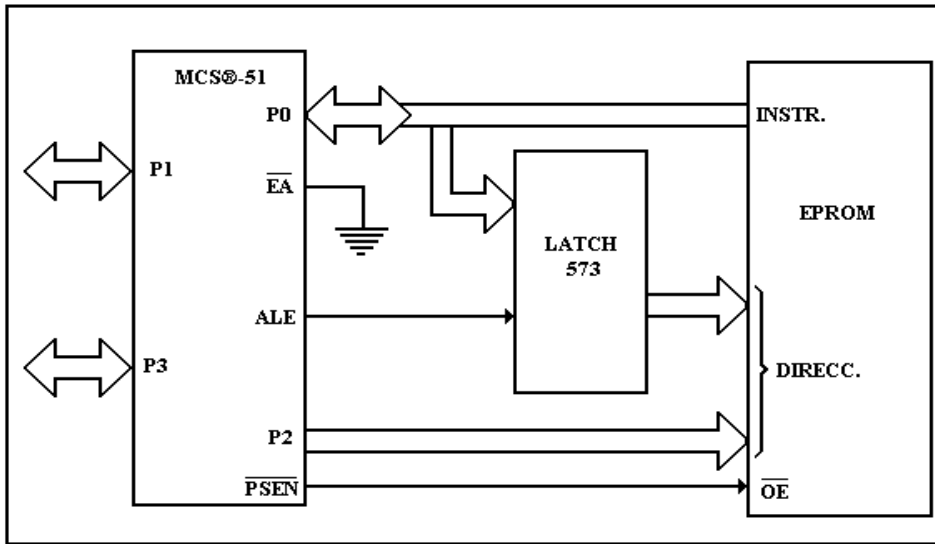


Figura 6. Conexión de una memoria externa.

Dentro de la memoria de programa se deberán encontrar localidades asignadas (vectores) para la atención de interrupciones, cuando una interrupcion se origina, el CPU acudirá a la localidad asignada por el fabricante, pero programada y considerada por el usuario en la memoria de programa, con el propósito de ejecutar las instrucciones que se desea se lleven a cabo al presentarse esta situación.

En la siguiente tabla 3 se muestran las fuentes de interrupción, su respectiva dirección en memoria.

| 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 serie) | 0023H |

Tabla 3. Interrupciones.

Una interrupción puede ser causada de manera externa o interna, es decir, puede ser producida por un dispositivo periférico o por programación, indistintamente. La interrupción

con mayor alto orden es el RESET el cual no puede ser enmascarable. Cuando, el RESET ocurre el programa se reinicia a partir de la dirección 0000H del programa.

Si una interrupción es producida, el Contador del Programa (PC) almacena su contenido temporalmente en la localidad que le señala el SP stack pointer (apuntador de pila) y se carga con la dirección de la localidad donde se encuentra la rutina de servicio de la interrupción correspondiente. Una vez posicionado en esta localidad deberá de comenzar la ejecución de la rutina de servicio, hasta que encuentre la instrucción RETI, que le permitirá al Program Counter (PC) recuperar nuevamente su valor original almacenado en el Stack Pointer (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 emplearse para propósitos generales del programa, si la interrupción ha sido permitida, en el momento que exista una activación de la interrupción (estado bajo en la línea INT0, pin p3.2) el PC se cargará con 0003H 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 estos 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 RETI.

1.6

MEMORIA DE DATOS (DATA MEMORY)

El espacio total 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 AT89C52 v.gr.8052), la parte alta de 128 bytes y el tercero, llamado espacio SFR (Registros de Funciones Especiales), otros 128 bytes.

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 89C52, 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.(Ver figura 7)



Figura 7. Estructura de la memoria RAM interna.

Los primeros 128 bytes son denominados de usuario, es una zona de memoria que normalmente es usada para manipular datos que no son importantes, esto es, que solo se ocupan para la toma de decisiones lógicas o de señalización, estos bytes están presentes en todos los dispositivos de la familia MCS-51, se presenta en la figura 8.

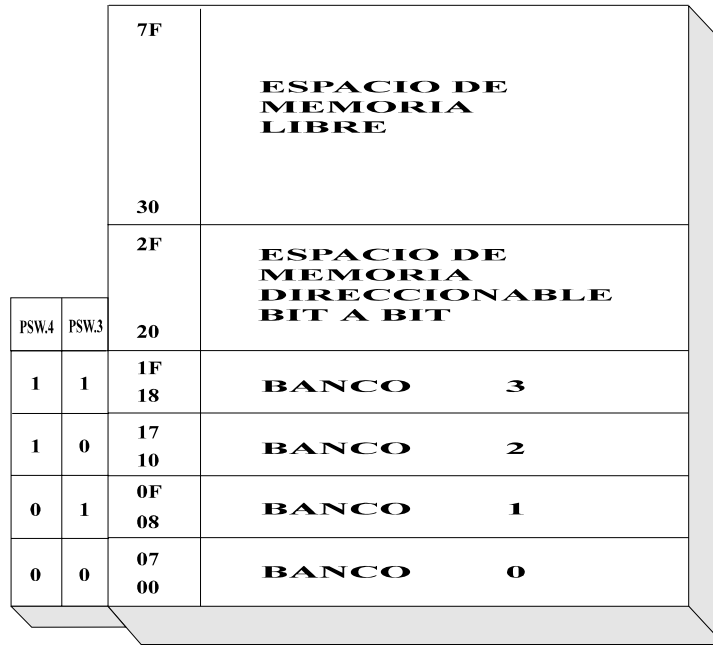


Figura 8. Distribución de los 128 bytes más bajos de la memoria RAM interna.

Como se puede apreciar en la figura 8, 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, estos bloques pueden ser seleccionados mediante la escritura en los bits 3 y 4 del registro PSW (palabra del estado del programa), mediante las combinaciones de unos o ceros. El empleo de registros permite un uso más eficiente del espacio de códigos, debido a que sus direccionamientos son de 8 bits únicamente.

Como ya se mencionó, existen algunas versiones del AT89C51, como el AT89C52, que contienen 128 bytes de memoria interna que pueden ser direccionadas de manera indirecta. Por otro lado, todas las versiones del AT89C52 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 al decodificar la instrucción y ejecutarla, así como también para el control y acceso de los diferentes puertos de entrada y salida.

Direccionamiento Directo.

El operando se especifica en la instrucción por un campo de dirección de 8 bits. Sólo la RAM interna de datos (primeros 128 bytes) y la zona de SFR's se puede direccionar de esta forma.

Ejemplo: `MOV A,45H` ;el contenido de la dirección de memoria 45H es transferido al Acc.

Direccionamiento indirecto.

La instrucción especifica un registro que contiene la dirección del operando. Tanto la memoria RAM interna como la externa se pueden direccionar indirectamente. Los registros para direccionar el mapa de 8 bits pueden ser R0 y R1 del banco de registros seleccionado o el Stack Pointer. El registro para direccionar sobre 16 bits sólo puede ser el puntero de datos (Data Pointer DPTR).

Ejemplo: `MOV @R0,A` ;el contenido del Acc es transferido a la dirección que indica el contenido de R0.

Direccionamiento por registro.

Se tienen 4 bancos de registros seleccionados por los bits 3 y 4 del PSW y cada banco de registro tiene 8 registros R0 a R7. El propio código de operación de la instrucción especifica con cual registro opera, es decir, cuando la instrucción es ejecutada, se accede a uno de los 8 registros del banco seleccionado.

Ejemplo: `MOV R5,A` ;el contenido del Acc es transferido a R5 (dirección de RAM 05H)

Direccionamiento implícito.

Algunas instrucciones especifican implícitamente, el registro sobre el cual van a operar, como el acumulador, el puntero de datos, etc. No necesitan especificar el operando porque está implícito en el código de operación.

Ejemplo: `INC A` ;el contenido del Acc es incrementado en 1.

Direccionamiento inmediato.

Al código de operación le sigue una constante en la memoria de programas.

Ejemplo: `MOV R1,#34H` ;el dato 34H es transferido a R1 (dirección 01H de la RAM)

Direccionamiento indexado.

Este direccionamiento sólo es posible en la memoria de programas y sólo permite la lectura. Es utilizado para la lectura de tablas. Un registro base de 16 bits (el DPTR o el contador de programas) apunta a la base de la tabla y el contenido del acumulador es el offset que permite acceder a la lectura de esa posición de la tabla. Es decir, la dirección de la tabla que se va acceder esta formada por la suma del acumulador y el puntero base.

Ejemplo: `MOVC A,@A+DPTR` ;el contenido de la dirección de memoria de programa indicada por la suma del valor actual del Acumulador y el Data Pointer es transferido a el Acumulador.

Se puede utilizar otro tipo de direccionamiento indexado en los saltos de instrucción. En este caso la dirección de destino del salto se computa como la suma del puntero base y el dato del acumulador.

CAPITULO 2

REGISTROS DE FUNCIONES ESPECIALES

A continuación se describen los registros y funciones con los cuales se puede controlar en su totalidad al AT89C52. Las direcciones de los Registros de Funciones Especiales (que en lo sucesivo se denominarán SFR) se muestran en la tabla 4.

| SÍMBOLO | NOMBRE | DIRECCIÓN |
|---------|------------------------------------|-----------|
| ACC | Acumulador | 0E0H |
| B | Registro B | 0F0H |
| PSW | Program Status Word | 0D0H |
| SP | Stack Pointer | 81H |
| DPTR | Data Pointer (16 bits) | 82H,83H |
| DPL | Byte bajo del Data Pointer | 82H |
| DPH | Byte alto del Data Pointer | 83H |
| IP | Prioridad de interrupciones | 0B8H |
| IE | Habilitador de interrupciones | 0A8H |
| TMOD | Modo de control del Timer/Contador | 89H |
| TCON | Control del Timer/Contador | 88H |
| TH0 | Byte alto del Timer/Contador 0 | 8CH |
| TL0 | Byte bajo del Timer/Contador 0 | 8AH |
| TH1 | Byte alto del Timer/Contador 1 | 8DH |
| TL1 | Byte bajo del Timer/Contador 1 | 8BH |
| SCON | Control del puerto serie | 98H |
| SBUF | Buffer de datos del puerto serie | 99H |

| | | |
|------|---------------------|------|
| PCON | Control de potencia | 87H |
| P0 | Puerto 0 | 80H |
| P1 | Puerto 1 | 90H |
| P2 | Puerto 2 | 0A0H |
| P3 | Puerto 3 | 0B0H |

Tabla 4. Registros de Funciones Especiales.

Cabe mencionar que estos registros se encuentran en la parte alta de la memoria RAM interna del AT89C52. A continuación se analiza cada uno de los registros, así como los bits más importantes de cada uno y la función que desempeñan en el control del AT89C52.

En la siguiente tabla (Tabla 5) se observa el mapa de memoria de los Registros de Funciones Especiales.

| 8 BYTES | | | | | | | | | |
|---------|------|------|-----|-----|-----|-----|--|------|----|
| F8 | | | | | | | | | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW | | | | | | | | D7 |
| C8 | | | | | | | | | 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 | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON | 87 |

↑
Registros direccionados por bit

Tabla 5. Mapa de memoria de los SFR.

2.1

ACUMULADOR

A_{CC} es el registro **Acumulador**. En las instrucciones específicas que se refieren a todo el byte, el mnemónico de éste se conoce simplemente como A. Este registro es de los más importantes, puesto que es utilizado como un registro procesador y en torno a él se realizan la mayoría de las operaciones, tanto aritméticas como lógicas, es por esto que en su arquitectura es el registro más complicado. Es importante mencionar que después de realizada una operación aritmética, el resultado de esta operación aparecerá en el acumulador.

2.2

REGISTRO B

El registro B es usado durante las operaciones de multiplicación y división, por lo que se toma como un acumulador auxiliar. Pero además, puede ser utilizado como cualquier otro registro de propósito general.

2.3

STACK POINTER

El Stack Pointer es un registro de 8 bits. Contiene la dirección de memoria RAM interna, en la que se guardará el dato al utilizar la instrucción PUSH o en su defecto, de donde el dato contenido en esta dirección, será obtenido al ejecutar la instrucción POP. Después del reset el SP apunta a la dirección 07H, por lo que la primera dirección disponible, por default, será la dirección 08H (R0 del segundo banco de registros). Cabe mencionar que el SP puede ser movido a cualquier dirección de memoria (RAM interna) y que en todas las llamadas a subrutina o interrupción es usado recuperando los datos almacenados con las instrucciones de retorno RET y RETI.

2.4

DATA POINTER

El Data Pointer (DPTR) consiste en un byte alto (DPH) y un byte bajo (DPL). Está diseñado para retener direcciones de 16 bits. Puede ser manipulado como un registro de 16 bits, o como dos registros independientes de 8 bits. Este registro se utiliza para mover datos hacia y desde la memoria ROM interna utilizando 16 bits de direccionamiento.

2.5

PUERTOS 0 AL 3

P0, P1, P2 y P3 son los latches de los SFR correspondientes a los Puertos 0, 1, 2 y 3 respectivamente. Escribir un 1 en un bit de un puerto (P0, P1, P2 o P3) causa la correspondiente salida de un estado alto por el pin del puerto. Escribir un cero causa la salida de un estado bajo por el mismo. Cuando se usan como entrada, el estado externo del pin es tomado en el registro correspondiente al puerto (por ejemplo, si el estado externo de un pin es bajo, el bit correspondiente al puerto SFR contiene un 0; si es alto contendrá un 1).

2.6

BUFFER DE DATOS SERIALES

El buffer serial son, en realidad, dos registros separados, un buffer para transmitir y un buffer para recibir. Cuando un dato es movido a SBUF, este va al buffer para transmitir y es usado para la transmisión serial. Moviendo un dato al SBUF es como inicia la transmisión serial. Cuando un dato es movido desde el SBUF, viene del buffer de recepción.

2.7

REGISTROS BÁSICOS DE LOS TIMER DEL AT89C52

El AT89C52 cuenta con tres registros timers (T0, T1 y T2), los cuales a su vez, están compuestos de dos registros de 8 bits cada uno, los registros pares TH0, TL0, TH1, TL1, TH2 y TL2 respectivamente.

2.8

PROGRAM STATUS WORD (PSW)

El registro PSW (palabra de estado de programa) muestra en sus diferentes bits el estado en el que se encuentra el microcontrolador en cualquier momento. A continuación se muestra dicho registro y la función de cada uno de sus bits (llamadas comúnmente banderas).

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

Registro PSW (Palabra de estado del programa)

| SÍMBOLO | BIT | FUNCIÓN |
|---------|-------|--|
| CY | PSW.7 | Bandera del Carry |
| AC | PSW.6 | Bandera del Carry Auxiliar (operaciones en BCD) |
| F0 | PSW.5 | Bandera 0 disponible por el usuario |
| RS1 | PSW.4 | Bit 1 del Selector del banco de registros |
| RS0 | PSW.3 | Bit 0 del Selector del banco de registros |
| OV | PSW.2 | Bandera de Overflow |
| -- | PSW.1 | Bandera disponible por el usuario |
| P | PSW.0 | Bandera de paridad del acumulador (ésta se afecta por hardware). |

Este registro, reside en el espacio de los SFR. El registro contiene: el bit del Carry, el Bit de carry 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, es decir:

P = 1, si el acumulador contiene un número impar de 1's (paridad impar).

P = 0, si el acumulador contiene un número par de 1's (paridad par).

Si se desea programar ó modificar este registro, se deberá utilizar la instrucción

MOV PSW,#DATO

donde DATO es la palabra de control en hexadecimal generada a partir de acomodar 1's y 0's en el registro PSW. Si es necesario modificar un solo bit de este registro, se utilizan las siguientes instrucciones:

| | |
|-------------------|------------------|
| SETB PSW.x | PSW.x = 1 |
| CLR PSW.x | PSW.x = 0 |

Donde: x = número de bit del registro PSW.

La bandera de *overflow* (de desbordamiento) sólo se activa cuando existe acarreo en el bit 6 o en el bit7 (nunca ambos a la vez) en la realización de operaciones aritméticas de:

- SUSTRACCIÓN
- MULTIPLICACIÓN.- Cuando el resultado sobrepasa la capacidad del acumulador (255). El carry siempre será igual a cero.
- DIVISIÓN.- Tanto el carry como el overflow siempre son igual a cero, sólo el overflow es activado cuando el divisor es igual a cero.

2.9

REGISTRO DE CONTROL DE POTENCIA (PCON)

El registro PCON, 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 SMOD sirve para dividir la frecuencia de transmisión o de recepción en 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 |
|------|----|----|----|-----|-----|----|-----|

Registro PCON (Registro de control de potencia).

| SÍMBOLO | BIT | FUNCIÓN |
|---------|--------|---|
| SMOD | PCON.7 | Dobla el BAUD RATE para el puerto serie cuando se utiliza el timer para generar el BAUD RATE. |
| GF1 | PCON.3 | Propósitos generales. |
| GF0 | PCON.2 | Propósitos generales. |
| PD | PCON.1 | Bajo consumo de energía. |
| IDL | PCON.0 | Bajo consumo de energía. |

Si se desea programar este registro o modificarlo, se deberá utilizar la instrucción MOV PCON,#DATO, donde DATO es la palabra de control en hexadecimal generada a partir de acomodar 1's y 0's en el registro PCON. Si específicamente un bit se desea modificar, deberá usarse la instrucción siguiente: SETB PCON.x, en donde x es el número de bit del registro PCON (pone un 1 en el bit PCON.x); y CLR PCON.x (borra el bit PCON.x).

2.10

REGISTRO HABILITADOR DE INTERRUPCIONES (IE)

Las interrupciones son controladas mediante la escritura tanto en el registro IE (Interruption Enable) como en el registro IP (Interruption Priority):

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

Con la bandera EA = 1, el AT89C52 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.

| | | | | | | | |
|----|----|-----|----|-----|-----|-----|-----|
| EA | -- | ET2 | ES | ET1 | EX1 | ET0 | EXO |
|----|----|-----|----|-----|-----|-----|-----|

Registro IE (Registro habilitador de interrupciones).

| SÍMBOLO | BIT | FUNCIÓN |
|---------|------|---|
| EA | IE.7 | Desactiva todas las INTERRUPCIONES EA=0. |
| ET2 | IE.5 | Activa la interrupción causada por el timer2 (ET2=1). |
| ES | IE.4 | Activa la interrupción causada por el puerto serie. |
| ET1 | IE.3 | Activa la int. de sobreflujo causada por el timer1. |
| EX1 | IE.2 | Activa la int. causada externamente en INT 1. |
| ET0 | IE.1 | Activa la int. se sobreflujo causada por el timer 0. |
| EX0 | IE.0 | Activa la int. causada externamente en INT 0. |

Si se desea programar una interrupción o modificar este registro, se deberá utilizar la instrucción MOV IE,#DATO la cual ya ha sido explicada con anterioridad en el PSW. Y de igual manera éste registro es modificable bit a bit.

Nota. Los bits de este registro así como los de todos los SFR's se les puede llamar también con el nombre especificado en sus tablas. Por ejemplo en este registro el bit 1 es nombrado ET0, se le podrá programar si el símbolo anterior esta definido en el archivo 8051.DEF (que se estudiará más adelante) o en el mismo programa.

2.11

REGISTRO DE PRIORIDAD (IP)

El AT89C52 tiene dos planos de prioridad para trabajar las interrupciones (figura 9), llamados alto y bajo, respectivamente. En la inicialización, todas las interrupciones trabajan en el plano de baja prioridad. Para poder 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.

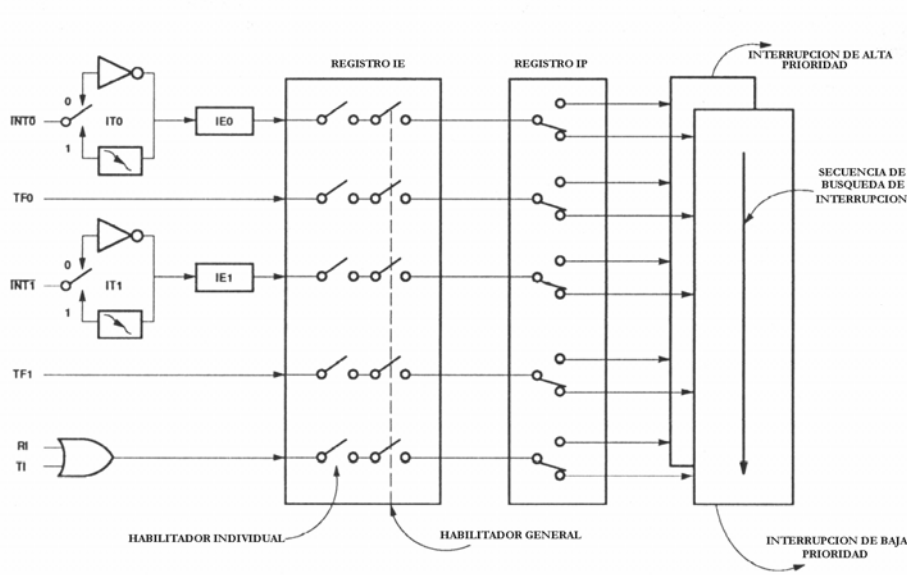


Fig. 9. Sistema de Interrupciones y Prioridades del AT89C52

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

Registro IP (Registro de prioridad de interrupciones).

| SÍMBOLO | BIT | FUNCIÓN |
|---------|------|---|
| PT2 | IP.5 | Timer 2 PT2=1 mayor prioridad. |
| PS | IP.4 | Define el nivel de prioridad de la int. puerto serie. |
| PT1 | IP.3 | Define el nivel de prioridad de la int. del timer 1. |
| PX1 | IP.2 | Define el nivel de prioridad de la int. 1 externa. |
| PT0 | IP.1 | Define el nivel de prioridad de la int. del timer 0. |
| PX0 | IP.0 | Define el nivel de prioridad de la int. 0 externa. |

Si se desea programar alguna prioridad en la interrupciones o modificar este registro, será posible con la instrucción MOV que también es posible modificarlo bit a bit.

2.12

TIMER/CONTADOR (T/C)

El AT89C52 tiene 2 Timer/Contadores de 16 bits cada uno, llamados Timer 0 y Timer 1 respectivamente. Ambos pueden ser configurados para operar como temporizadores (timers) o bien, 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 el pin de entrada correspondiente a T0 o T1. En cambio, cuando funciona como temporizador (Timer), el registro es incrementado cada 12 períodos de oscilación; es decir, su frecuencia de conteo es 1 / 12 de la frecuencia del oscilador (es decir que si se tiene un cristal de 12MHZ, el timer será incrementado cada microsegundo).

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 programada, una interrupción.

2.13

**REGISTRO DE CONTROL DEL PUERTO
TIMER/CONTADOR (TCON)**

El registro de control del Timer/Contador, es también direccionable bit a bit, para activar o desactivar cada una de sus banderas.

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

Registro TCON (Registro de control del Timer/Contador).

| SÍMBOLO | BIT | FUNCIÓN |
|---------|--------|---|
| TF1 | TCON.7 | Bandera de sobreflujo del registro del Timer 1. Activada por hardware cuando el registro T/C 1 incrementa su contenido pasando todos sus bits de 1's a 0's. Es limpiado por hardware cuando es atendida la interrupción causada por ésta bandera. |
| 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 del registro del Timer 0. Esta opera exactamente igual que la bandera TF1. |
| SÍMBOLO | BIT | FUNCIÓN |
| 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 es detectada una transición de 1 a 0 en el pin correspondiente a dicha interrupción. Limpiada por hardware cuando la interrupción es atendida (solamente se acciona, si dicha interrupción es habilitada: IE.2=1). |
| IT1 | TCON.2 | BIT de control de interrupción 1. Activado y limpiado por software. Este BIT programa el tipo de transición que activará la interrupción externa 1. IT1=0 la interrupción se activará al detectar un nivel bajo en el pin correspondiente; IT1=1 la interrupción se activará al detectar un flanco de bajada. |
| IE0 | TCON.1 | Bandera de transición de la interrupción externa 0. Opera exactamente igual que IE1, siendo IE.0. (Entiéndase por IE: registro de habilitación de interrupciones). |
| IT0 | TCON.0 | BIT de control de interrupción 0. Aplica lo descrito para IT1, y se entiende que al mencionar IT1 ahora se habla de IT0. |

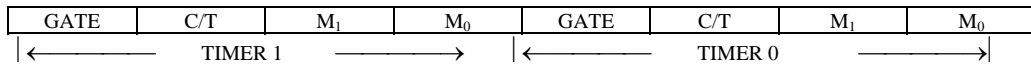
Al igual que todos los registros de este tipo (SFR) éste puede ser modificado con la instrucción MOV o con las instrucciones SETB y CLR.

2.14

**REGISTRO DE MODO DE CONTROL DEL
TIMER/CONTADOR (TMOD)**

Este registro permite especificar si se van a trabajar como temporizadores (times) 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, el registro TMOD se muestra a continuación.



Registro TMOD (Reg. del modo de control del T/C).

| SÍMBOLO | BIT | FUNCIÓN |
|----------------|--------|---|
| GATE | TMOD.7 | Cuando GATE=1, no es suficiente activar TR1 (en TCON) para que el Timer/Counter 1 funcione, si no que es necesario que la línea externa INT1 esté en posición alta (control por hardware). Cuando GATE=0, el funcionamiento del Timer/Counter 1 solamente dependerá de la activación de TR1 (Control por software). |
| C/T | TMOD.6 | Selector de función: Temporizador o Contador. C/T=0 activa la función de Temporizador (contabiliza pulsos del reloj del sistema interno).C/T=1 activa la función de Contador (contabiliza pulsos en la entrada externa T1). |
| M ₁ | TMOD.5 | BIT selector del modo del Timer 1. |
| M ₀ | TMOD.4 | BIT selector del modo del Timer 1. |
| GATE | TMOD.3 | Aplica lo descrito para el GATE del TIMER 1 sólo que ahora al hablar de TR1 e INT1, entiéndase que se trata de TR0 e INT0 respectivamente. |
| C/T | TMOD.2 | Selector de función: Temporizador o Contador. Igual que para C/T en TIMER 1 sólo cambiar T1 por T0. |
| | TMOD.1 | BIT selector del modo del Timer 0. |
| M ₀ | TMOD.0 | BIT selector del modo del Timer 0. |

La siguiente tabla muestra las diferentes combinaciones de los bits selectores de modo de los times y sus funciones.

| M ₁ | M ₀ | 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 timer1. El Timer 1 no se utiliza. |

- ♦ **Modo 0.** En este modo cualquiera de los 2 times, 0 y 1, se configuran como registros de 13 bits, que consisten en los 8 bits del registro de TH (TH1 o TH0) y los 5 bits menos significativos del registro TL (TL1 o TH1). Los 3 bits más significativos de TL no son utilizados en este modo.
- ♦ **Modo 1.** Este modo es utilizado por cualquiera de los dos times, y los configura como Timer/Contador de 16 bits (TH y TL como contadores o times de 8 bits completan los 16 bits).
- ♦ **Modo 2.** Este modo también puede llegar a ser utilizado por los dos times, y los configura para un conteo de 8 bits (Tlx) con recarga automática. Al ser sobrepasada la capacidad de TL, éste es recargado automáticamente, con el contenido de TH y a su vez es activada la bandera de sobreflujo (TF).
- ♦ **Modo 3.** 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 de contadores separados. 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.

El Timer 1 puede ser activado o desactivado con sólo 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.

Como ya se ha visto anteriormente, este registro también es modificable por las instrucciones MOV, SETB y CLR.

2.15

CONTROL DEL PUERTO SERIE (SCON)

El puerto serie contenido en la familia del MCS-51, es un puerto “FULL DUPLEX”, lo cual significa que puede transmitir y recibir datos simultáneamente. El receptor contiene un almacén “Buffer”, que le permite empezar 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 la transmisión se encuentra en el registro SBUF del SFR (Registros de Funciones Especiales, anteriormente descrito).

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 SCON, que se describe a continuación.

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

Registro SCON. (Registro de Control del Puerto Serie).

| SÍMBOLO | BIT | FUNCIÓN |
|---------|--------|---|
| 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 9no. BIT que será transmitido en los modos 2 y 3. |
| RB8 | SCON.2 | En modos 2 y 3, el 9no. BIT recibido es de datos. En el modo 1, si SM2=0, RB8 es el BIT de stop el que fue recibido. En el modo 0, RB8 no es usado. |
| TI | SCON.1 | Bandera de interrupción para transmitir. Se habilita por hardware al final del 8vo. BIT en el modo 0, ó al principio del BIT de stop en los otros modos. Debe ser deshabilitado por software. |
| RI | SCON.0 | Bandera de interrupción para recibir. Se habilita por hardware al final del 8vo. BIT en el modo 0, ó a la mitad del camino del BIT de stop en los otros modos (ver excepción SM2). Debe ser deshabilitado por software. |

Para poder seleccionar el modo en este registro, se encuentran los bits 6 y 7 (SM1 y SM0 respectivamente) y la configuración y descripción es como sigue:

| SM0 | SM1 | Modo | Descripción | Baud Rate |
|-----|-----|------|----------------|---------------------------------|
| 0 | 0 | 0 | Shift Register | $F_{osc} / 12$ |
| 0 | 1 | 1 | 8 BIT UART* | Variable |
| 1 | 0 | 2 | 9 BIT UART | $F_{osc} / 64$ ó $F_{osc} / 32$ |
| 1 | 1 | 3 | 9 BIT UART | Variable |

*UART: Universal Asincronic Receive-Transmit.

2.16

UTILIZACIÓN DEL TIMER 1 COMO GENERADOR DEL "BAUD RATE" PARA LA TRANSMISIÓN SERIE

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 ecuación siguiente:

$$\text{BAUD RATE} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{FREC. DEL OSCILADOR}}{12 \times [256 - (\text{TH1})]}$$

O en su defecto si ya se desea una velocidad en particular, con un cristal determinado la formula anterior queda:

$$\text{TH1} = 256 - \frac{\text{Baud Rate}}{(32)(12) 2^{\text{SMOD}} (\text{Frecuencia del Oscilador})}$$

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 o 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.

Cabe recalcar en este momento, 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 este 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.

En la tabla 6 se muestran los valores de TH1, para generar el Baud Rate, tomado en cuenta la frecuencia del oscilador o cristal.

| BAUD RATE | F osc | SMOD | TIMER 1 | | |
|-----------|------------|------|---------|------|--------------|
| | | | C / T | MODO | VALOR DE TH1 |
| 19.2 KHz | 11.059 MHz | 1 | 0 | 2 | FDH |
| 9.6 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 |

Tabla 6. Valores para generar el Baud rate

CAPITULO 3**INSTRUCCIONES DEL μ C AT89C52**

3.1**CICLO DE INSTRUCCIÓN**

La ejecución de un ciclo de instrucción comienza en el estado 1 del ciclo máquina, el cual tiene doce estados, cuando el código de operación es almacenado en el registro de instrucción. Como norma general, una instrucción requiere de uno o más ciclos máquina, en función de:

a) El Código de operación.

Por ejemplo la instrucción INC A tiene 1 byte de instrucción y requiere un ciclo máquina, la instrucción INC DPTR también de 1 byte requiere de dos ciclos máquina y la instrucción MUL AB requiere cuatro ciclos máquina para su total ejecución y ocupa 1 byte de memoria.

b) El número de bytes.

Por ejemplo la instrucción MOV A,#dato tiene 2 bytes de instrucción y requiere de un ciclo máquina. En cambio la instrucción MOV directo, #dato al constar de 3 bytes necesita de dos ciclos máquina. Esto no ocurre siempre, o sea a mas bytes mas ciclos máquina. Ante la imposibilidad de establecer una norma o regla que facilite el conocimiento del número de bytes y ciclo de instrucción se han editado tablas con el set completo de instrucciones.

Cada ciclo máquina se producen dos accesos a la memoria, el primero al comienzo del ciclo de instrucción siempre leerá el código de operación, el segundo, que normalmente se descarta, se utiliza para leer el segundo byte operando de la instrucción.

3.2**SET DE INSTRUCCIONES**

El set o conjunto de instrucciones de la familia MCS-51 se puede dividir según las siguientes especialidades o dedicación de las instrucciones según su operación.

| <u>TIPOS DE OPERACIONES</u> | <u>No. MNEMÓNICOS</u> |
|---------------------------------------|-----------------------|
| OPERACIONES ARITMÉTICAS | 5 |
| OPERACIONES LÓGICAS | 10 |
| TRANSFERENCIA DE DATOS | 7 |
| SALTO O CONTROL DEL PROGRAMA | 13 |
| MANIPULACIÓN A BIT O ÁLGEBRA BOOLEANA | 11 |

Las instrucciones aritméticas, lógicas, de transferencia y de salto son comunes a la mayoría de microprocesadores. Los microcontroladores tienen un área especial de aplicación que es el área de control de procesos, en este campo las operaciones están orientadas muy a menudo a BIT. Los microcontroladores leen, procesan, escriben e intercambian información con los sistemas exteriores en formato BIT a BIT o palabra a palabra. Un procesador booleano con un set de instrucciones booleanas muy completo se encarga de realizar este tipo de operaciones, que está orientada a BIT. Esta particularidad, así como su inmunidad al ruido eléctrico, lo hacen valioso en el mundo de control de procesos industriales.

3.3**CONJUNTO DE INSTRUCCIONES DE LA FAMILIA MCS-51®**

A continuación se presenta una tabla (Tabla 7) en la cual se describen todos los conjuntos de instrucciones que conforman a la familia de MCS-51, así como sus periodos de oscilación, los bytes que ocupa cada instrucción y una breve descripción de cada una de las instrucciones.

Instrucciones que afectan a los <<flags>>:

| Instrucción | C | FLAR | |
|-------------|---|------|----|
| | | OV | AC |
| ADD | X | X | X |
| ADDC | X | X | X |
| SUBB | X | X | X |
| MUL | 0 | X | |
| DIV | 0 | X | |
| DA | X | | |
| RRC | X | | |
| RLC | X | | |
| SETB C | 1 | | |
| CLR C | 0 | | |
| CPL C | X | | |
| ANL C, bit | X | | |
| ANL C./bit | X | | |
| ORL C, bit | X | | |
| ORL C./bit | X | | |
| MOV C, bit | X | | |
| CJNE | X | | |

Notas sobre el significado de los operandos de las instrucciones que se describen a continuación:

Rn.....Registros R0-R7 del banco de registros seleccionado.

Direct.....Direccion del dato de 8 bits de la memoria interna. Puede ser un dato de la RAM interna (0- 127 para el 8051 o 0-255 para el 8052) o un registro del SFR (por ejemplo, Puerto de E/S, registro de control, registro de estado (128-255).

@RiSe refiere a los registros R0 y R1, que permiten el direccionamiento indirecto para acceder a los datos de 8 bits de la RAM interna (80-FFH) solapada con el SFR para el 8052 y (00-7FH) para el 8052/8051.

dataOperando constante de 8 bits

#data16 Operando constante de 16 bits.

Addr 16 Dirección destino de 16 bits. Utilizada por las instrucciones de salto LCALL y LJMP para permitir el salto dentro del espacio de 64Kbytes de la memoria de programas.

Addr 11..... Dirección destino de 11 bits. Utilizada por las instrucciones de salto ACALL y AJMP para permitir el salto dentro de la página de 2Kbytes de la memoria de programas a partir del primer byte de la siguiente instrucción.

relSalto relativo en formato de 8 bits en complemento a 2. Utilizado por la instrucción SJMP y todos los saltos condicionales. El rango del salto (8 bits) esta comprendido entre -128 a +127 bytes a partir del primer byte de la siguiente instrucción.

BIT.....Direccionamiento directo BIT a BIT dentro del área de la memoria RAM interna y de los registros SFR que permiten el direccionamiento por BIT.

rrr.....En la columna de codificación indica el registro implicado en la instrucción (Rn), según la siguiente tabla:

| | | | |
|-------|---|---|----------|
| r | r | r | registro |
| 0 | 0 | 0 | R0 |
| 0 | 1 | 1 | R1 |
| 0 | 1 | 0 | R2 |
| ===== | | | |
| 1 | 1 | 1 | R7 |

i.....en la columna de codificación indica el registro implicado en la instrucción de direccionamiento indirecto (@Ri), según se muestra en la siguiente tabla:

| | |
|---|----------|
| i | registro |
| 0 | @R0 |
| 1 | @R1 |

| MNEMÓNICO | DESCRIPCIÓN | BYTE | PERIODOS DE OSCILACIÓN |
|----------------------------------|---|------|------------------------|
| INSTRUCCIONES ARITMÉTICAS | | | |
| ADD A,Rn | Suma registro a Acumulador | 1 | 12 |
| ADD A,direct | Suma un byte directo a Acumulador | 2 | 12 |
| ADD A,@Ri | Suma indirectamente un registro de RAM a Acumulador | 1 | 12 |
| ADD A,#data | Suma dato inmediato a Acumulador | 2 | 12 |
| ADDC A,Rn | Suma registro a Acumulador con carry | 1 | 12 |
| ADDC A,direct | Suma un byte directo a Acumulador con carry | 2 | 12 |
| ADDC A,@Ri | Suma indirectamente un registro de RAM a Acumulador con carry | 1 | 12 |

| MNEMÓNICO | DESCRIPCIÓN | BYTE | PERIODOS DE OSCILACIÓN |
|--|---|------|------------------------|
| ADDC A, #data | Suma dato inmediato a Acumulador con carry | 2 | 12 |
| SUBB A,Rn | Subtrae Registro de ACC con préstamo | 1 | 12 |
| SUBB A,direct | Subtrae un byte directo de ACC con préstamo | 2 | 12 |
| SUBB A,@Ri | Subtrae indirectamente un registro de RAM de ACC con préstamo | 1 | 12 |
| SUBB A,#data | Subtrae un dato inmediato de ACC con préstamo | 2 | 12 |
| INC A | Incrementa Acumulador | 1 | 12 |
| INC Rn | Incrementa Registro | 1 | 12 |
| INC direct | Incrementa directamente un byte | 2 | 12 |
| INC @Ri | Incrementa indirecto un byte en RAM | 1 | 12 |
| DEC A | Decrementa Acumulador | 1 | 12 |
| DEC Rn | Decrementa Registro | 1 | 12 |
| DEC direct | Decrementa directamente un byte | 2 | 12 |
| DEC @Ri | Decrementa indirecto un byte en RAM | 1 | 12 |
| INC DPTR | Incrementa Data Pointer | 1 | 24 |
| MUL AB | Multiplca A y B | 1 | 48 |
| DIV AB | Divide A y B | 1 | 48 |
| DA A | Ajuste decimal del Acumulador | 1 | 12 |
| INSTRUCCIONES LÓGICAS | | | |
| ANL A,Rn | AND Registro a Acumulador | 1 | 12 |
| ANL A,direct | AND Byte directo a Acumulador | 2 | 12 |
| ANL A,@Ri | AND indirecto un byte en RAM a Acumulador | 1 | 12 |
| ANL A,#data | AND Dato inmediato a Acumulador | 2 | 12 |
| ANL direct,A | AND Acumulador a un byte directo | 2 | 12 |
| ANL direct,#data | AND dato inmediato a byte directo | 3 | 24 |
| ORL A,Rn | OR Registro a Acumulador | 1 | 12 |
| ORL A,direct | OR Byte directo a Acumulador | 2 | 12 |
| ORL A,@Ri | OR indirecto un byte en RAM a Acumulador | 1 | 12 |
| ORL A,#data | OR Dato inmediato a Acumulador | 2 | 12 |
| ORL direct,A | OR Acumulador a un byte directo | 2 | 12 |
| ORL direct,#data | OR dato inmediato a byte directo | 3 | 24 |
| XRL A,Rn | OR-Exclusiva Registro a Acumulador | 1 | 12 |
| XRL A,direct | OR-Exclusiva Byte directo a Acumulador | 2 | 12 |
| XRL A,@Ri | OR-Exclusiva indirecto un byte en RAM a Acumulad. | 1 | 12 |
| XRL A,#data | OR-Exclusiva Dato inmediato a Acumulador | 2 | 12 |
| XRL direct,A | OR-Exclusiva Acumulador a un byte directo | 2 | 12 |
| XRL direct,#data | OR-Exclusiva dato inmediato a byte directo | 3 | 24 |
| CLR A | Limpia Acumulador | 1 | 12 |
| CPL A | Complementa Acumulador | 1 | 12 |
| RL A | Rota Acumulador a la izquierda | 1 | 12 |
| RLC A | Rota Acumulador a la izquierda a través del carry | 1 | 12 |
| RR A | Rota Acumulador a la derecha | 1 | 12 |
| RRC A | Rota Acumulador a la derecha a través del carry | 1 | 12 |
| SWAP A | Intercambio de nibbles dentro del ACC | 1 | 12 |
| INSTRUCCIONES DE TRANSFERENCIA DE DATOS | | | |
| MOV A,Rn | Mueve registro a Acumulador | 1 | 12 |
| MOV A,direct | Mueve byte directo a Acumulador | 2 | 12 |
| MOV A,@Ri | Mueve indirecto un byte en RAM a Acumulador | 1 | 12 |

| MNEMÓNICO | DESCRIPCIÓN | BYTE | PERIODOS DE OSCILACIÓN |
|--|---|------|------------------------|
| MOV Rn,A | Mueve Acumulador a registro | 1 | 12 |
| MOV Rn,direct | Mueve byte directo a registro | 2 | 24 |
| MOV Rn,#data | Mueve dato inmediato a registro | 2 | 12 |
| MOV direct,A | Mueve Acumulador a byte directo | 2 | 12 |
| MOV direct,Rn | Mueve registro a byte directo | 2 | 24 |
| MOV direct,direct | Mueve byte directo a byte directo | 3 | 24 |
| MOV direct,@Ri | Mueve indirecto Byte en RAM a byte directo | 2 | 24 |
| MOV direct,#data | Mueve dato inmediato a byte directo | 3 | 24 |
| MOV @Ri,A | Mueve Acumulador a byte indirectamente en RAM | 1 | 12 |
| MOV @Ri,direct | Mueve byte directo a byte indirectamente en RAM | 2 | 24 |
| MOV @Ri,#data | Mueve dato inmediato a byte indirectamente en RAM | 2 | 12 |
| MOV DPTR,#data16 | Carga el Data Pointer con una constante de 16 bits | 3 | 24 |
| MOVC A,@A+DPTR | Mueve Código relativo byte a DPTR a Acumulador | 1 | 24 |
| MOVC A,@A+PC | Mueve Código relativo byte a PC a Acumulador | 1 | 24 |
| MOVX A@Ri | Mueve de RAM externa (dirección de 8 bits) a Acumulador | 1 | 24 |
| MOVX A@DPTR | Mueve de RAM externa (dirección de 16 bits) a Acumulador | 1 | 24 |
| MOVX A,@Ri,A | Mueve Acumulador a RAM externa (dirección de 8 bits) | 1 | 24 |
| MOVX @DPTR,A | Mueve Acumulador a RAM externa (dirección de 16 bits) | 1 | 24 |
| PUSH direct | Empuja byte directo al stack | 2 | 24 |
| POP direct | Suelta byte directo del stack | 2 | 24 |
| XCH A,Rn | Intercambia registro con Acumulador | 1 | 12 |
| XCH A,direct | Intercambia byte directo con Acumulador | 2 | 12 |
| XCH A,@Ri | Intercambia indirecto RAM con A _{CC} | 1 | 12 |
| XCHD A,@Ri | Intercambia dígito de bajo orden indirecto RAM con Acumulador | 1 | 12 |
| MANIPULACIÓN DE VARIABLES BOOLEANAS | | | |
| Nota. La palabra carry se refiere al BIT de acarreo del SFR PSW.7 | | | |
| CLR C | Limpia el carry (PSW.7) | 1 | 12 |
| CLR BIT | Limpia BIT directo | 2 | 12 |
| SETB C | Pone un uno en el carry | 1 | 12 |
| SETB BIT | Pone un uno en el BIT directo | 2 | 12 |
| CPL C | Complementa el carry | 1 | 12 |
| CPL BIT | Complementa BIT directo | 2 | 12 |
| ANL C,bit | AND BIT directo y el carry | 2 | 24 |
| ANL C,/BIT | AND complemento de BIT directo y carry | 2 | 24 |
| ORL C,bit | OR bit directo y carry | 2 | 24 |
| ORL C,/BIT | OR complemento de BIT directo y carry | 2 | 24 |
| MOV C,bit | Mueve BIT directo al carry | 2 | 12 |
| MOV bit,C | Mueve carry a bit directo | 2 | 24 |
| JC rel | Salta si el carry es uno | 2 | 24 |
| JNC rel | Salta si el carry no es uno | 2 | 24 |
| JB rel | Salta si el BIT directo es uno | 2 | 24 |
| JNB rel | Salta si el BIT directo no es uno | 2 | 24 |
| JBC bit,rel | Salta si el BIT directo es uno y limpia el BIT | 3 | 24 |

| RAMIFICANDO EL PROGRAMA | | | |
|-------------------------|--|------|------------------------|
| MNEMÓNICO | DESCRIPCIÓN | BYTE | PERIODOS DE OSCILACIÓN |
| LCALL addr16 | Larga llamada a subrutina | 3 | 24 |
| RET | Retorno de subrutina | 1 | 24 |
| RETI | Retorno de interrupción | 1 | 24 |
| AJMP addr11 | Salto absoluto | 2 | 24 |
| LJMP addr16 | Salto largo | 3 | 24 |
| SJMP rel | Salto corto (dirección relativa) | 2 | 24 |
| JMP @A+DPTR | Salto relativo indirecto a el DPTR | 1 | 24 |
| JZ rel | Salta si el Acumulador es cero | 2 | 24 |
| JNZ rel | Salta si el Acumulador no es cero | 2 | 24 |
| CJNE A,direct,rel | Compara byte directo con ACC y salta si no es igual | 3 | 24 |
| CJNE A,#data,rel | Compara dato inmediato con ACC y salta si no es igual | 3 | 24 |
| CJNE Rn,#data,rel | Compara dato inmediato con registro y salta si no es igual | 3 | 24 |
| CJNE @Ri,#data,rel | Compara dato inmediato con byte indirecto y salta si no es igual | 3 | 24 |
| DJNZ Rn,rel | Decrementa registro y salta si no es cero | 2 | 24 |
| DJNZ direct,rel | Decrementa byte directo y salta si no es cero | 3 | 24 |
| NOP | No operación | 1 | 12 |

Tabla 7. Conjunto de instrucciones del µC AT89C52.

3.4

DESCRIPCIÓN DE LAS INSTRUCCIONES DEL AT89C52

En las siguientes páginas se estudiarán las instrucciones, su significado y sus acepciones de uso en un programa según lo requiere el intérprete de instrucciones llamado ensamblador. El formato que se presenta deberá ser respetado rigurosamente al escribir el programa fuente, y significa según se indica.

$$(PC) \quad \longleftarrow \quad (PC) + 2$$

Ejemplo: $(PC) \quad \longleftarrow \quad (PC) + 2$
 $(SP) \quad \longleftarrow \quad (SP) + 1$

donde: $(PC) =$ contenido del PC

Al ejecutarse la instrucción que tenga los dos renglones anteriores el PC se habrá incrementado en 2 y el SP en 1, lo que indica que una dirección o un dato habrán sido guardados en la pila.

Significa que después de realizada la instrucción el PC de tener un valor cualquiera se encuentra en ese valor más dos localidades de memoria más, si existieran más instrucciones

con este formato en orden, sería lo que al ejecutarse la instrucción se modificaría dentro del microcontrolador.

Para interpretar correctamente la presencia de paréntesis tomar en cuenta que:

- (R0) significa que se está trabajando, directamente, con el contenido de R0 y
- ((R0)) significa que el valor de R0 es la dirección de memoria en la cual está contenido el dato con el que se está trabajando.

esto es:

(A) ← (R0)

Si R0=#03BH el valor del acumulador después de esta operación será de 03BH en cambio, si la operación es:

(A) ← ((R0))

al final de ésta, el contenido del acumulador será el dato que se encuentra en la dirección de memoria 03BH

La palabra Directo se refiere a la dirección en hexadecimal según el mapa de memoria del microcontrolador. Ejemplo: MOV 20H,90H 20H y 90H son direcciones directas de la memoria interna del microcontrolador.

La palabra Relativa es una dirección normalmente mostrada con una etiqueta. Ejemplo: CJNE A,#20H,ENCENDIDO La palabra ENCENDIDO es una dirección relativa y se les llama etiqueta.

DESCRIPCIÓN DE LAS INSTRUCCIONES

ACALL.- (Llamada) 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 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 esta 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.

ACALL
(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP)) ← (PC₇₋₀)
(SP) ← (SP) + 1

((SP)) ← (PC₁₅₋₈)

Ejemplo:

```
ACALL    TIME
MOV     A,R1
```

Esta instrucción solicita la ejecución de la subrutina llamada TIME, para lo cual los requisitos son: 1) Que exista la etiqueta que indica el principio de la subrutina, y 2) Una instrucción que nos indique el final de la subrutina y por lo tanto el regreso a la ejecución normal del programa (instrucción RET). Así pues, al encontrarse con el ACALL el micro acudirá a ejecutar la siguiente subrutina:

```
TIME MOV     R1,#0FFH
      DJNZ   R1,$
      RET
```

Después de haber ejecutado las dos instrucciones que componen la subrutina TIME, y al encontrarse con la instrucción RET, el micro regresa a ejecutar el programa en donde se quedó, lo cual quiere decir que ejecutará la instrucción MOV A,R1 y las siguientes hasta encontrar otra desviación.

ADD.- (Suma) Esta instrucción suma el contenido del acumulador de acuerdo al modo de direccionamiento y deja el resultado en el acumulador, afectando las banderas de carry y carry auxiliar, así como la de sobreflujo. Sus modos de direccionamiento son el directo, indirecto, inmediato y de registro indirecto

```
ADD A, R0      (A) ← (A) + (Rn)
ADD A, directo (A) ← (A) + (directo)
ADD A, @Ri     (A) ← (A) + ((Ri))
ADD A, #dato   (A) ← (A) + #dato
```

Ejemplos:

```
MOV A,#05H
MOV R0,#05H
ADD A,R0
```

Después de la instrucción ADD el contenido del acumulador será 0AH, esto es, el resultado de la suma del contenido, el registro R0 más el contenido del acumulador.

```
MOV 20H,#05H
MOV A,#05H
ADD A,20H
```

Después de la última instrucción, el contenido del acumulador será de 0AH, esto es, el resultado de sumar el contenido del acumulador más el contenido de la dirección 20H.

```
MOV 3BH,#05H
MOV R1,#3BH
MOV A,#05H
ADD A,@R1
```

Después de la última instrucción el contenido del acumulador será 0AH, esto es, el resultado de sumar el contenido de la dirección 3BH más el contenido del acumulador.

```
MOV A,#05H
ADD A,#05H
```

Después de la última instrucción, el contenido del acumulador será el resultado de la suma directa del contenido del acumulador y el número 05H, esto es, 0AH.

ADDC.- (Suma + carry) Esta instrucción es similar a la anterior, con la diferencia de que en la suma se incluye el valor del carry, y de la misma manera puede direccionarse de los cuatro modos anteriores:

| | | |
|-----------------|-------|-----------------------|
| ADDC A, R0 | (A) ← | (A) + (C) + (Rn) |
| ADDC A, directo | (A) ← | (A) + (C) + (directo) |
| ADDC A, @Ri | (A) ← | (A) + (C) + ((Ri)) |
| ADDC A, #dato | (A) ← | (A) + (C) + #dato |

AJMP.- (Salto absoluto) Esta instrucción transfiere la ejecución del programa incondicionalmente hacia donde el dato lo indique, utilizando únicamente 5 bits para la codificación de la instrucción y los otros 3 que sobran del BIT para la página a saltar, ya que el salto máximo por esta instrucción es dentro de una página de 2K.

| | |
|--|---|
| A ₁₀ , A ₉ , A ₈ , 0 0001 | A ₇ , A ₆ , A ₅ , A ₄ , A ₃ , A ₂ , A ₁ , A ₀ |
| AJMP addr11 | (PC) ← (PC) + 2 |

Ejemplo:

```
MOV A,#0FH
MOV R1,#0AH
AJMP FULL
DIV A,#03H
FULL ADD A,R1
END
```

Al encontrarse con la instrucción AJMP salta incondicionalmente a ejecutar la instrucción precedida por la etiqueta FULL (la suma del acumulador y el registro 1), sin ejecutar la instrucción que, en orden sucesivo, sería la siguiente (división del acumulador entre el valor #03H)..

ANL.- (And lógico) Esta es la instrucción lógica AND entre las variables indicadas, no afecta ninguna bandera y puede usar direccionamiento directo, indirecto, de registro o inmediato:

| | | | |
|--------------------|-----------|---|-------------------|
| ANL A, Rn | (A) | ← | (A) ^ (Rn) |
| ANL A, directo | (A) | ← | (A) ^ (directo) |
| ANL A, @Ri | (A) | ← | (A) ^ ((Ri)) |
| ANL A, #dato | (A) | ← | (A) ^ #dato |
| ANL directo, A | (directo) | ← | (directo) ^ (A) |
| ANL directo, #dato | (directo) | ← | (directo) ^ #dato |

Ejemplos:

```
MOV A,#06H
MOV R1,#04H
ANL A,R1
```

Al final de este segmento de programa, el contenido del acumulador será = 02H, que es el resultado de la operación AND entre el acumulador y el registro R1.

```
MOV A,#06H
MOV 020H,#04H
ANL A,020H
```

Después de la última instrucción el contenido del acumulador será 04H, que es el resultado de la operación AND entre el acumulador y el contenido de la dirección 020H.

```
MOV A,#06H
MOV 03BH,#04H
MOV R1,#03BH
ANL A,@R1
```

Después de haber ejecutado esta última instrucción el contenido del acumulador será 04H, que es el resultado de la operación AND entre el acumulador y el valor contenido en la dirección de memoria 03BH.

ANL C.- (And lógico con carry) La misma instrucción que la anterior pero sólo con el carry del acumulador, si en el ensamblaje se pone un “/” entonces el operando se negará sólo para la operación sin ser afectado realmente, es decir sólo en la operación lógica se verá negado; el resultado de esta operación aparecerá en el mismo BIT de carry después de realizada la operación:

| | | | |
|------------|-----|---|--------------|
| ANL C,bit | (C) | ← | (C) ^ (bit) |
| ANL C,/bit | (C) | ← | (C) ^ (-bit) |

Ejemplos:

```
SETB F0 ;Flag del usuario
SETB PSW.7 ;PSW.7 = dirección del carry del acumulador
ANL C,F0
ANL C,/F0
```

Después de la primera instrucción ANL el estado del carry será **1**, pero después de la segunda será **0**.

CJNE.- (Comparación) Esta instrucción se encarga de hacer comparaciones con un byte completo y transfiere el control en caso de no ser igual a la referencia dada. Esta instrucción trabaja con cuatro combinaciones de direccionamiento, el acumulador puede ser comparado con cualquier byte directamente o con cualquier dato inmediatamente y cualquier dirección de RAM indirectamente o un registro puede ser comparado con una constante inmediata:

| | |
|---------------------------------------|------------------|
| CJNE A, directo, relativa | (A) <> (directo) |
| CJNE A, #dato, relativa | (A) <> #dato |
| CJNE Rn, #dato, relativa | (A) <> #dato |
| CJNE @Ri, #dato, relativa | ((Ri)) <> #dato |
| (PC) ← (PC) + 3 | falso |
| (PC) ← (PC) + desplazamiento relativo | verdadero |

Ejemplos:

Cualquiera de los siguientes fragmentos de programa mandará a ejecutar la instrucción precedida por la etiqueta FIN, sin ejecutar la instrucción de suma.

```

MOV A,#05H
MOV 03BH,#04H
CJNE A,03BH,FIN
ADD A,03BH
FIN MOV A,03BH
END
    
```

```

MOV A,#05H
CJNE A,#04H,FIN
ADD A,#04H
FIN MOV A,#04H
END
    
```

```

MOV R1,#05H
CJNE R1,#04H,FIN
ADD A,R1
FIN MOV A,R1
END
    
```

```

MOV R1,#03BH
MOV 03BH,#05H
MOV A,#04H
CJNE A,@R1,FIN
    
```

El siguiente fragmento de programa ejecutará las dos últimas instrucciones del programa, y es aplicable a cualquiera de los cuatro tipos de direccionamiento:

```

MOV A,#05H
CJNE A,#05H,FIN
ADD A,R1
FIN MOV A,R1
END

```

COLOCAR AQUÍ LAS CONSIDERACIONES DE SI ES MAYOR O MENOR CUANDO NO ES IGUAL EN CUANTO AL CARRY

CLR A.- (borra A) Esta instrucción se encarga de limpiar el acumulador y poner ceros a todos sus bits, ejemplo:

```
CLR A          (A) ← 0
```

CLR bit.- (Borra el BIT) Limpia el BIT al igual que la instrucción anterior pero con la particularidad de actuar individualmente en las zonas de memoria direccionables BIT a BIT.

```
CLR C          (C) ← 0
CLR bit       (bit) ← 0
```

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

```
CPL A          (A) ← (-A)
```

Ejemplo:

```
MOV A,#06H
CPL A
END
```

Al finalizar la ejecución de este fragmento de programa el contenido del acumulador será 0F9H, que es el complemento de 06H.

CPL BIT.- (Complementa el BIT) Al igual que la anterior complementa pero únicamente el BIT indicado, sólo actúa en la zona direccionable por BIT, ejemplo:

```
CPL C          (C) ← (-C)
CPL bit       (bit) ← (-bit)
```

DA.- (Ajusta decimal) Se encarga de hacer ajustes de hexadecimal a BCD mediante sumas de 00H, 06H, 60H, o 66H de acuerdo al caso; es decir, divide el byte en 2 nibbles y si alguno es


```

MOV R1,#03H
DJNZ R1,FIN
CPL A
FIN MOV A,R1
END

MOV 03BH,#03H
MOV R1,#02H
DJNZ 03BH,FIN
CPL A
FIN MOV A,R1
END

```

INC.- (Incremento) Encargada de incrementar el valor del byte, no afecta banderas, direcciona mediante forma de registro directo o registro indirecto:

```

INC A      (A)      ← (A) + 1
INC Rn    (Rn)     ← (Rn) + 1
INC directo (directo) ← (directo) + 1
INC @Ri   ((Ri))  ← ((Ri)) + 1

```

INC DPTR.- (Incrementa el DPTR) Función similar a la anterior pero para uso exclusivo del puntero de datos (Data Pointer):

```
INC DPTR (DPTR) ← (DPTR) + 1
```

JB.- (Salto) Si el BIT indicado es 1 salta a la dirección indicada sumándola al PC:

```

JB BIT, relativa      (PC) ← (PC) + 3      FALSO
                      (PC) ← (PC) + relativa VERDADERO

```

En el siguiente ejemplo se omite la instrucción que incrementa el acumulador y brinca a la etiqueta FIN en la cual le transfiere un valor numérico, cabe señalar que si en lugar de la instrucción SETB F0 se colocara una instrucción CLR F0, se ejecutarían ambas instrucciones antes de llegar al final del programa:

```

SETB F0
JB F0,FIN
INC A
FIN MOV A,#02H
END

```

JC.- (Salto) Saltar si el carry está en 1, de otro modo continúa su curso normal:

```

JC relativa      (PC) ← (PC) + 2      FALSO
                 (PC) ← (PC) + relativa VERDADERO

```

En el siguiente ejemplo la ejecución del programa brinca a la etiqueta FIN, al encontrar que el carry está en 1, eludiendo así la instrucción que transfiere el número #03H al acumulador; y quedando un valor de #0FH en éste, al final del programa:

```

MOV A,#0FFH
ADD A,#01H
JC FIN
MOV A,#03H
FIN ADD A,#0FH
END

```

JMP @A + DPTR.- (Salto incondicional) Suma los ocho bits no asignados 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:

JMP @A + DPTR (PC) ← (A) + (DPTR)

Ejemplo:

```

DIRECCIÓN 010H    MOV DPTR,#20H
DIRECCIÓN 012H    MOV A,#05H
DIRECCIÓN 014H    JMP @A+DPTR
                  .
                  .
                  .
DIRECCIÓN 025H    MOV A,#0FFH
                  END

```

JNB.- (Salto) Si el bit indicado es cero, salta a la dirección suma del byte indicando más el PC, de otra manera, sigue su curso normal:

JNB bit, relativa (PC) ← (PC) + 3 FALSO
(PC) ← (PC) + relativa VERDADERO

En el siguiente ejemplo se ejecutarán las dos últimas instrucciones antes de llegar al final debido a que el resultado de la operación JNB resulta falso, es importante tomar en cuenta que al utilizar CLR F0 en lugar de SETB F0 se llevará a cabo el brinco a FIN omitiendo la instrucción que incrementa el acumulador.

```

SETB F0
JNB F0,FIN
INC A
FIN MOV A,#02H
END

```

JNC.- (Salto) Si el bit de carry es un cero, salta a la dirección formada por la suma algebraica del PC incrementado y del byte de desplazamiento relativo; de otra manera procede con la siguiente instrucción. La bandera de carry no es modificada (ver ejemplo de JC).

| | | |
|--------------|------------------------|-----------|
| JNC relativa | (PC) ← (PC) + 2 | FALSO |
| | (PC) ← (PC) + relativa | VERDADERO |

JNZ relativa.- (Salto) 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 carry no es modificada.

| | | |
|--------------|------------------------|-----------|
| JNZ relativa | (PC) ← (PC) + 2 | FALSO |
| | (PC) ← (PC) + relativa | VERDADERO |

En el siguiente ejemplo al ejecutar la segunda instrucción brincaré a la instrucción precedida por la etiqueta FIN omitiendo la suma de A y #02H.

```

MOV A,#04H
JNZ FIN
ADD A,#02H
FIN NOP
END
    
```

JZ relativa.- (Salto) Salta si el acumulador tiene todos sus bits en cero, el acumulador no se modifica y las banderas no son afectadas.

| | | |
|-------------|------------------------|-----------|
| JZ relativa | (PC) ← (PC) + 2 | FALSO |
| | (PC) ← (PC) + relativa | VERDADERO |

En el siguiente ejemplo se forzara un salto a la etiqueta FIN:

```

MOV A,#0H
JZ FIN
MOV R0#04H
FIN NOP
END
    
```

LCALL addr16.- (Llamada) Llama a una subrutina mediante el uso de 2 bytes. La instrucción suma 3 al PC 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 esta dirección, hasta encontrar 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.

LCALL addr (15-8) addr (7-0) (PC) ← (PC) + 3
 (SP) ← (SP) + 1
 ((SP)) ← (PC 7-0)
 (SP) ← (SP) + 1
 ((SP)) ← (PC 15-8)
 (PC) ← (addr 15-0)

Ejemplo:

En este ejemplo se supone que la subrutina TIME se encuentra en localidades altas de memoria, con esto se justifica el uso de la instrucción LCALL, se observa que el procedimiento es exactamente el mismo que con las instrucciones ACALL Y SCALL. Así al encontrarse con la instrucción LCALL ejecutará la subrutina TIME y regresará al flujo normal para dar por terminado el programa (END).

```

LCALL      TIME
              END

TIME      MOV   R1,#0FFH
              DJNZ  R1,$
              RET
    
```

LJMP addr16.- (Salto) Ramificación incondicional del programa mediante dos bytes. El PC se carga con los dos últimos bytes de la instrucción y salta para continuar con la ejecución del programa a partir de esa dirección. Las banderas no son afectadas.

LJMP addr (15-8) addr (7-0) (PC) ← addr (15-0)

Ejemplo:

Al igual que la instrucción anterior, es importante tener en cuenta que si el salto se realiza a localidades de memoria altas, es necesario el empleo de esta instrucción, por lo demás, el uso de ésta es el mismo que con las instrucciones SJMP y AJMP.

```

LJMP FIN
    MOV  R1,#09H
FIN   NOP
        END
    
```

MOV.- (Mover) Esta instrucción permite la modificación de un registro o dirección mediante los modos de direccionamiento usados pudiendo ser inmediato, directo, indirecto, de registro índice y de registro.

| | | | | |
|---|---------------|-----|---|-----------|
| 1 | MOV A, #dato | (A) | ← | #dato |
| 2 | MOV A, Rn | (A) | ← | (Rn) |
| 3 | MOV A,directo | (A) | ← | (directo) |
| 4 | MOV A,@Ri | (A) | ← | ((Ri)) |

Comment [DdCA1]:

| | | | | |
|----|---------------------|-----------|---|-----------|
| 5 | MOV Rn,A | (Rn) | ← | (A) |
| 6 | MOV Rn,directo | (Rn) | ← | directo |
| 7 | MOV Rn, #dato | (Rn) | ← | #dato |
| 8 | MOV directo,A | (directo) | ← | (A) |
| 9 | MOV directo,Rn | (directo) | ← | (Rn) |
| 10 | MOV directo,directo | (directo) | ← | (directo) |
| 11 | MOV directo, @Ri | (directo) | ← | ((Ri)) |
| 12 | MOV directo, #dato | (directo) | ← | #dato |
| 13 | MOV @Ri,A | ((Ri)) | ← | (A) |
| 14 | MOV @Ri,directo | ((Ri)) | ← | directo |
| 15 | MOV @Ri, #dato | ((Ri)) | ← | #dato |

Nota. Destino, origen. Quiere decir Byte origen, donde inicialmente tomará el valor a mover y Byte destino la dirección donde lo colocará. Es importante comprender los siguientes conceptos:

@Ri significa que Ri contiene la dirección de donde se encuentra el dato con el cual se está trabajando.

#dato el número que se escriba en el espacio que dice **dato** será directamente el valor con el que se quiere trabajar.

directo éste será una dirección de memoria.

Es importante no perder de vista que el operando de la derecha es el origen de la transferencia de información, y a su vez, el operando de la izquierda es el destino de esta información, así, el siguiente segmento de programa tiene el objetivo de mostrar todas las combinaciones de direccionamiento que se pueden utilizar con la instrucción MOV, en el espacio designado a comentarios (esto es después del símbolo ;) de cada instrucción se explicará el efecto de la misma (**DIRECCIÓN DE MEMORIA=DATO**):

```

MOV A,#06CH ;transfiere al acc un 6CH. A=#06CH
MOV R1,#03BH ;transfiere a R1 un 03BH. R1=#03BH
MOV 0FCH,R1 ;transfiere el contenido de R1 a la dirección de
;memoria 0FCH. 0FCH=#03BH

MOV A,R1 ;transfiere al acc el contenido de R1. A=#03BH
MOV 03BH,#0AFH ;transfiere un 0AFH a la dirección de memoria 03BH.
;3BH=#0AFH

MOV 0FCH,#0B5H ;transfiere un 0B5H a la dirección de memoria 0FCH.
;FCH=#0B5H

MOV A,03BH ;transfiere al acc el dato contenido en la dirección de
;memoria 03BH o sea A=#0AFH

MOV A,@R1 ;R1 contiene la dirección de donde se encuentra el
;dato que será transferido al acc A=#0AFH

MOV 03BH,0FCH ;transfiere el contenido de la dirección de memoria
;FCH, a la también dirección de memoria 3BH.
;3BH=#0B5H
    
```

```

MOV 0FCH,@R1    ;R1 contiene la dirección en la cual se encuentra el
                 ;dato que será transferido a la dirección de memoria
                 ;FCH;FCH=#0B5H
MOV R1,A        ;transfiere a R1 el contenido del acc R1=#0AFH
MOV R1,03BH     ;transfiere a R1 el contenido de la dirección de
                 ;memoria 03BH. R1=#0B5H
MOV 0FCH,A      ;transfiere el dato contenido en el acc a la dirección ;de
                 ;memoria FCH. FCH=#0AFH
MOV R1,#03BH    ;transfiere un 3BH a R1. R1=#03BH
MOV @R1,A       ;transfiere el contenido del acc a la dirección 03BH
                 ;03BH=#0AFH
MOV @R1,0FCH    ;transfiere el contenido de la dirección de memoria
                 ;0FCH a la también dirección de memoria 03BH.
                 ;03BH=#0B5H
MOV @R1,#021H   ;transfiere un 21H a la dirección de memoria 03BH
                 ;03BH=#021H
    
```

MOVC.- (Mover) Esta instrucción se utiliza para la lectura de tablas contenidas en la memoria de programa (move constant) y cabe señalar que estas tablas sólo pueden ser leídas más no modificadas:

```

MOVC A,@A+DPTR    (A) ←—— ((A+DPTR))
MOVC A,@A+DPTR    (A) ←—— ((A+DPTR))
    
```

Ejemplo:

Al finalizar el siguiente fragmento de programa el acumulador contendrá el número 5, obtenido de una tabla que comienza en la dirección 05BH:

```

MOV DPTR,#05BH
MOV A,#04H
MOVC A,@A+DPTR
END
    
```

```

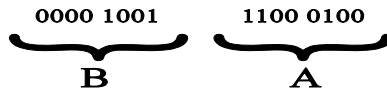
DIRECCIÓN 05BH    DB    01H
                  DB    02H
                  DB    03H
                  DB    04H
DIRECCIÓN 05BH+04H DB    05H
    
```

MOV C,bit MOV bit,C.- (Mover bit) En forma análoga a la instrucción anterior pero operando con el carry del acumulador y cualquier otro bit.

```

MOV C,bit        (C) ←—— (bit)
MOV bit,C        (bit) ←—— (C)
    
```

En el siguiente ejemplo se transferirá el bit 0 del puerto 1 al carry del acumulador:



```
MOV A,#050D
MOV B,#050D
MUL AB
END
```

ORL destino,origen.- (Or lógica) Realiza la operación lógica OR entre los dos bytes dejando el resultado en el byte destino.

| | | | |
|--------------------|-----------|---|-------------------|
| ORL A,Rn | (A) | ← | (A) o (Rn) |
| ORL A,directo | (A) | ← | (A) o (directo) |
| ORL A, @Ri | (A) | ← | (A) o ((Ri)) |
| ORL A, #dato | (A) | ← | (A) o #dato |
| ORL directo.A | (directo) | ← | (directo) o (A) |
| ORL directo, #dato | (directo) | ← | (directo) o #dato |

Ejemplo:

El resultado de las siguientes operaciones será igual a #067H, quedando éste en el acumulador. Tomar en cuenta que son fragmentos de programas diferentes, es decir que no existe relación entre ellos.

```
MOV A,#042H
MOV R1,#025H
ORL A,R1
END
```

```
MOV A,#042H
MOV 03BH,#025H ;SIENDO 03BH UNA DIRECCIÓN DE
ORL A,03BH ;MEMORIA
END
```

```
MOV A,#042H
MOV R1#03BH
MOV 03BH,#025H
ORL A,@R1
END
```

```
MOV A,#042H
ORL A,#025H
END
```

```
MOV A,#042H
MOV 03BH,#025H
```

```

ORL 03BH,A
END

MOV 03BH,#042H
ORL 03BH,#025H
END

```

ORL C,bitdestino.- (Or lógica) Operación lógica OR realizada con el bit de prueba y el del carry, ejemplo:

```

ORL C,bit      (C) ← (C) o (bit)
ORL C,/bit     (C) ← (C) o (-bit)

```

POP directo.- (Sacar) Operación de byte de la pila (stack). El contenido de la localidad de la RAM interna direccionada por el SP es leído, y posteriormente el SP es decrementado por 1. El valor leído es transferido al byte indicado por el direccionamiento directo. Las banderas no son afectadas:

```

POP directo    (directo) ← ((SP))
               (SP)      ← (SP) - 1

```

Ejemplo:
(ver instrucción PUSH).

PUSH directo.- (Guardar) Operación contraria a la anterior, pone un byte a la pila (stack):

```

PUSH directo   (SP)      ← (SP) + 1
               ((SP))   ← (directo)

```

Ejemplo:

El siguiente fragmento de programa aplica las dos últimas instrucciones. Primero se moverá el SP a la dirección 028H, después se cargará en la dirección 010H el dato #045H se guarda en la pila, quedando éste en la dirección 029H e inmediatamente después, se recupera de la pila para transferirlo a la dirección 011H.

```

MOV SP,#028H
MOV 010H,#045H
PUSH 010H
POP 011H
END

```

RET.- (Regreso) Instrucción encargada de regresar el control de una subrutina llamada previamente:


```

SETB C           ;A= 1 1000 0010
RLC A           ;A= 1 0000 0101
END
    
```

RR A.- (Rotación) 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.

```

RR A           (An)  ← (An + 1)   n = 0 - 6
              (A7)  ← (A0)
    
```

Ejemplo:

En este ejemplo se transfiere el número 05H al acumulador y al finalizar el programa se tiene el número 082H.

```

MOV A,#05H     ;A=0000 0101
RR A          ;A=1000 0010
    
```

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

```

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

Ejemplo:

```

MOV A,#081H    ;A=0 1000 0001
RRC A         ;A=1 0100 0000
END
    
```

SETB bit.- (Coloca el bit) Coloca el bit indicado en uno. SETB puede operar sobre la bandera de carry ó cualquier bit direccionable directamente. Las otras banderas no son afectadas, ejemplo:

```

SETB C        (C)  ← 1
              ( bit ) ← 1
    
```

Ejemplo:

Supóngase que el estado en el que se encuentra el carry es 0.

```

MOV A,#00H     ;A= 0 0000 0000
SETB C        ;A= 1 0000 0000
END
    
```


Ejemplo:

```
MOV A,#045H      ;A=#045H
MOV R1,#09H     ;R1=#09H
XCH A,R1        ;A=#09H y R1=#045H
MOV 03BH,#067H ;03BH=#067H
XCH A,03BH     ;A=#067H y 03BH=#09H
MOV 045H,#0FCH ;045H=#0FCH
XCH A,@R1      ;A=0FCH y 045H=#067H
END
```

XCHD A, @Ri.- (Intercambio) Intercambia únicamente el nivel menos significativo entre el acumulador y el contenido de la dirección indicada en Ri, las banderas no son afectadas.

XCHD A, @Ri (A₃₋₀) ←→ ((Ri₃₋₀))

Ejemplo:

```
MOV R1,#03BH    ;R1=#03BH
MOV 03BH,#63H  ;03BH=#063H
MOV A,#81H     ;A=#081H
XCHD A,@R1     ;A=#083H y 03BH=#061H
END
```

XRL destino,origen.- (Or exclusiva) XRL ejecuta la operación lógica OR-Exclusiva entre las variables 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 ó un dato inmediato.

Ejemplo:

| | | | |
|-------------------|-----------|---|---------------------|
| XRL A,Rn | (A) | ← | (A) XOR (Rn) |
| XRL A,directo | (A) | ← | (A) XOR (directo) |
| XRL A, @Ri | (A) | ← | (A) XOR ((Ri)) |
| XRL A,#dato | (A) | ← | (A) XOR #dato |
| XRL directo,A | (directo) | ← | (directo) XOR (A) |
| XRL directo,#dato | (directo) | ← | (directo) XOR #dato |