# 1

# INTRODUCCIÓN A LOS MICROPROCESADORES Y MICROCONTROLADORES

Se ha dicho muchas veces: la electrónica es la ciencia de los componentes. El desarrollo de cada nuevo dispositivo de estado sólido trae consigo, técnicas de diseño diferentes, por lo general, más simple.

La forma de presentarles a ustedes los diferentes aspectos de estos dispositivos será mediante la comparación.

## 1.1 COMPUTADORAS DIGITALES: PRINCIPIOS BÁSICOS

### 1.1.1 Bits y Bytes.

Antes de iniciar con el desarrollo de lo que es una computadora digital, es conveniente explicar lo que es “bits” y “bytes”.

Cuando se pulsa en el teclado de un microcomputador o un terminal de tiempo compartido, es fácil de olvidar que los computadores son máquinas simples de “on” y “off”. Los circuitos lógicos que controlan las operaciones de las computadoras típicamente producen +5V o 0V. Debido a esto el sistema numérico binario es invocado para describir estas señales digitales. Asumiendo la convención de lógica positiva, el nivel +5V se torna en un 1 lógico y el nivel de 0V en un 0 lógico.

Una línea digital individual (bit) por consiguiente puede llevar solo dos fragmentos de información – la línea es un 1 o es un 0. Resolvemos esta escasez aparente de capacidad de manejo de información mediante la combinación de varias líneas digitales juntas (8 bits ≡ bytes). Tal colección de líneas se les refiere como “bus”.

### 1.1.2 Computador de Programa Almacenado.

La figura 1.1 es un diagrama de bloques de un computador digital típico. La unidad central de proceso o CPU, mostrado en la parte izquierda, se asemeja al cerebro humano debido a que aquí se toman todas las decisiones y se genera la temporización del sistema. La unidad lógica aritmética o ALU es contenida en la CPU y todas las operaciones matemáticas se desarrollan allí.

La unidad de memoria mostrada en la figura 1.1 es utilizada para almacenar la secuencia específica de comandos que serán utilizados para instruir a la CPU de desarrollar algunas tareas. A estas instrucciones se les llama programa de computadora (debido a esto es el nombre de computador de programa almacenado).

Finalmente, ninguna tarea útil puede ser desarrollada por el computador sin los dispositivos de entrada/salida (I/O).

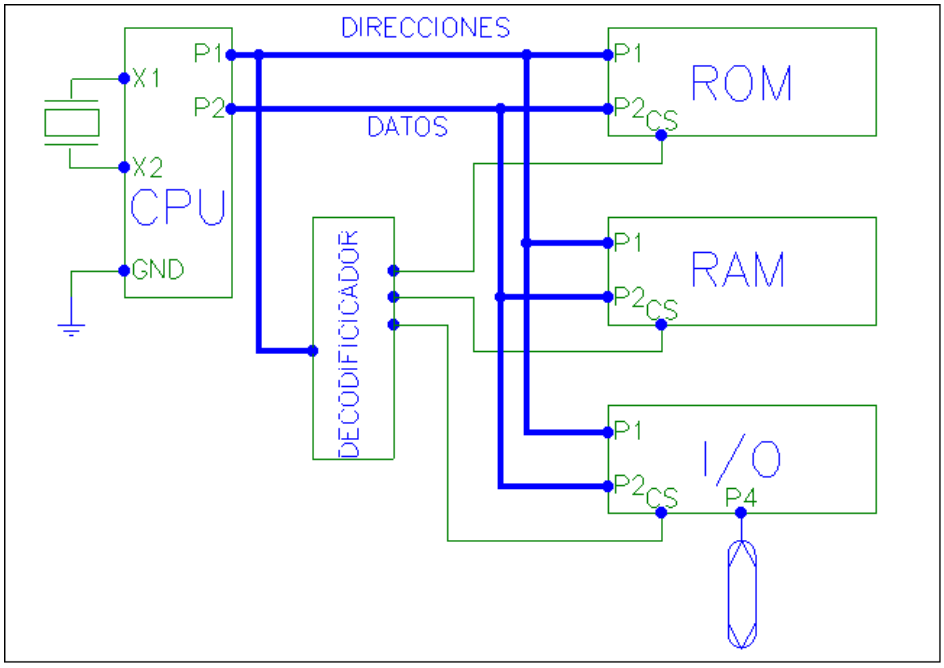


**Figura 1.1 Idea general de un microcomputador con microprocesador**

Definiéndolo desde ahora, un microcontrolador es un circuito integrado que contiene toda la estructura (arquitectura) de un microcomputador, o sea, CPU, RAM, ROM y circuitos de I/O. La idea de un microcomputador, para el diseñador de controles, se limita, ahora, a algo similar a lo que se ilustra en la figura 1.2. Se necesita solamente alimentación de corriente continua (Vcc), un cristal o red RC externa para definir la frecuencia de operación, "quemar" el programa de control en la EPROM, y ya tenemos los puertos, de entrada y salida, listos para hacer conexión con el mundo exterior.

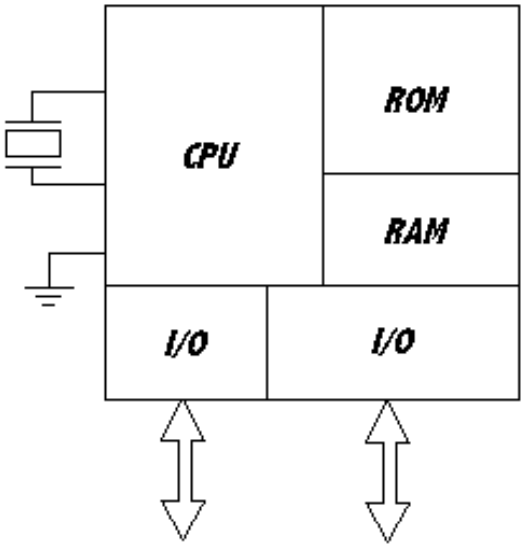


**Figura 1. 2 Idea general de un microcomputador con microcontrolador**

¿Cuáles son las diferencias más notables entre un microcomputador realizado con microprocesadores y un microcontrolador? Hay varias y las ventajas existen para las dos partes. Es mejor decir que cada uno tiene su ubicación especial dentro de las aplicaciones electrónicas.

1. La CPU del microcontrolador es más simple, y sus instrucciones están orientadas, más que todo, a la operación de cada uno de los bits de entrada y salida. Sin embargo, la estructura de la mayoría de los microprocesadores importantes tiene su correspondiente versión de microcontrolador en el mercado.
2. La memoria RAM (de datos) que ofrecen estos elementos es mínima. La razón es simple: las aplicaciones de control e instrumentación primitivas no necesitan almacenar grandes cantidades de información temporal.
3. La memoria ROM (de programa) es limitada. Por lo general, no mayor de 4Kilobytes. Como es lógico hay de dos tipos: EPROM y OTP.
4. No es necesario diseñar complejos circuitos decodificadores porque el mapa de memoria y de puertos I/O, está implícito en el controlador. Por la razón anterior, el circuito impreso de las aplicaciones es muy simple y, en algunas ocasiones, puede ser de una sola capa.
5. La mayoría de los microcontroladores tienen dificultad para entregar al usuario los buses de direcciones, de datos y de control de la CPU, como lo hace fácilmente los sistemas con microprocesador. Algunos controladores lo hacen a través de los puertos de entrada/salida (I/O), utilizando señales especiales de sincronización. Estos buses y señales se pueden emplear para implementar expansión de memoria RAM y ROM por fuera del microcontrolador.
6. La velocidad de operación de los microcontroladores es más lenta que la que se puede lograr con los sistemas microprocesadores. Sin embargo, hay anuncios del desarrollo de circuitos controladores que funcionarán por encima de los 50MHz.
7. De manera similar a los sistemas utilizados con los microprocesadores para escribir, ensamblar y depurar programas en lenguaje de máquina, se requiere un sistema de desarrollo para cada familia de microcontroladores. Está compuesto por un paquete de software con editor, ensamblador y simulador de programas y, al mismo tiempo, se necesita un hardware para "quemar" la memoria EPROM del microcontrolador.

### 1.1.3 Características Arquitectónicas.

- **Estructura Von Newman**

En la figura 1.3 se muestra la estructura tradicional de un computador. Al igual que los microprocesadores, los microcontroladores basados en la arquitectura Von Newman tienen "un solo bus de datos" el cual es utilizado para buscar tanto a las instrucciones como a la data. Las instrucciones de programa y datos son almacenadas en una memoria principal común. Cuando un controlador direcciona la memoria principal, primeramente busca la instrucción, y después busca el dato que soporta a la instrucción. Las dos búsquedas por separado aminoran la operación del controlador.

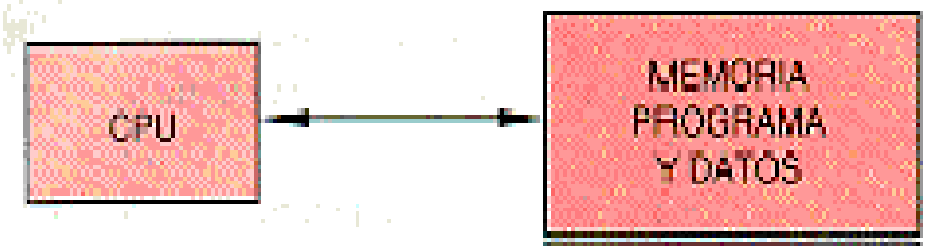


**Figura 1. 3 Estructura Von Newman**

- **Arquitectura Harvard**

Los microcontroladores basados en la arquitectura Harvard tienen el bus de datos y el de instrucciones separados. Esto permite que la ejecución ocurra en paralelo. Como una instrucción puede ser "pre-buscada", la instrucción actual es ejecutada sobre el bus de datos. Una vez la instrucción actual es completada, la siguiente instrucción está lista para ir. Este pre-buscado teóricamente permite una más rápida ejecución que la estructura Von Newman, pero existe alguna adición compleja de Silicio. (véase figura 1.4)

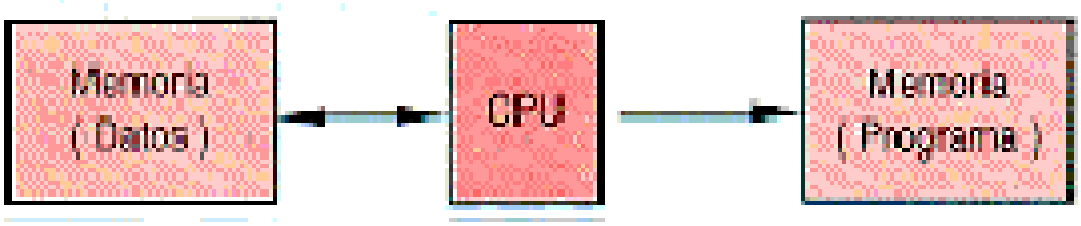


**Figura 1. 4 Arquitectura Harvard**

### 1.1.4 Ir Por (Fetch) y Ejecuta (Execute).

La CPU ha sido diseñada para seguir repetidamente cuatro sencillos pasos.

1. Ir por la data desde la celda de memoria cuya dirección se encuentra en el registro de contador de programa.
2. Adicione 1 a la dirección en el contador de programa.
3. Decodifique el comando presente en el registro de instrucciones y haga lo que se dice.
4. Vaya al paso 1.

Estos cuatro pasos constituye el principio de operación de todos los computadores digitales de programa almacenado. Esto incluye desde el más grande mainframe IBM al más diminuto microcomputador. El principio es llamado "Ir Por y Ejecuta" y es la llave para el entendimiento de las actividades de un microprocesador o microcontrolador.

### 1.1.5 Arquitectura de Tres Buses.

La CPU, unidad de memoria, y los dispositivos I/O deben ser capaz de comunicarse unos con otros. Por ejemplo, la CPU debe ser capaz de especificar cuál celda de memoria va hacer seleccionada, y si el contenido de esta celda deberá ser leída o nueva data escrita hacia la celda.

Este es el propósito de los buses de dirección, data y control mostrado en la figura 1.1. Cuando la CPU requiere leer el contenido de una celda de memoria particular, primeramente presenta la dirección correcta sobre su bus de direcciones. Esto es el contenido actual del contador de programa. Seguidamente, el bus de control causa la lectura a memoria. El bus de dato transfiere el contenido de la celda de memoria seleccionada de vuelta a la CPU y hacia el registro de instrucción.

En resumen, la CPU inicia cada ciclo de comando con una instrucción de "Ir Por" desde la unidad de memoria. El contador de programa es incrementado en preparación para el siguiente ciclo de "Ir Por". Finalmente el código de operación de la instrucción es decodificada y ejecutada durante la fase de ejecución del ciclo.

Desde el punto de vista de la arquitectura de tres buses, solo hay cuatro ciclos de instrucción únicas posibles. Estas se listan en la Tabla 1.1 con el contenido de los tres buses especificados para cada caso.

Algunos microcontroladores se les dota de un bus típico de microprocesador para resolver una amplia gama de problemas y permitir la ampliación de uso de productos ya desarrollados anteriormente.

**Tabla 1.1**

**Ciclos de Instrucción de un Computador Digital**

| Tipo de Instrucción | Bus de Direcciones | Bus de Control | Bus de Dato |
|---|---|---|---|
| Lectura a Memoria | Dirección de la Celda de Memoria | Seleccione la Memoria y Lea | Contenido de la Celda de Memoria Seleccionada |
| Escritura a Memoria | Dirección de la Celda de Memoria | Seleccione la Memoria y Escriba | Data a hacer Escrita a memoria |
| Lectura de I/O | Dirección del Dispositivo I/O | Seleccione el I/O y Lea | Data desde el Dispositivo I/O Seleccionado |
| Escritura a I/O | Dirección del Dispositivo I/O | Seleccione el I/O y Escriba | Data a hacer Escrito hacia el Dispositivo de I/O |

## 1.2 CÓDIGOS DE COMPUTADORAS

Una de las complicaciones que el ser humano ha tenido cuando trabaja con una computadora es que no podemos decirle de una forma sencilla lo que queremos que ella haga. Por ejemplo, suponga que queremos sumar dos números. Como ser humano podemos preguntar los dos números, sumarlos en la mente, y anunciar el resultado. Compare esto con los pasos requeridos para que la computadora haga esto.

1. Lea el primer número de entrada desde el teclado.
2. Almacene este número en la unidad de memoria.
3. Lea el segundo número desde el teclado y almacénelo en el acumulador.
4. Recupere el primer número de la unidad de memoria y adiciónelo con el número que se encuentra en el acumulador, dejando el resultado en el acumulador.
5. Presente el contenido del acumulador por el TRC (tubo de rayos catódicos).

Se requiere de muchas operaciones que nos hace pensar acerca de la ¡utilidad de la computadora en primer lugar! Claro está, la computadora puede hacer cada operación bastante rápido – usualmente en unos pocos microsegundos. Una vez que el programa para sumar dos números ha sido escrito, la hará a la velocidad del rayo y nunca se equivocará. Claro está, esta es la ventaja que tiene la computadora sobre nosotros – una vez que se ha programado para desarrollar una tarea, ella desarrollará esa tarea a una razón que se acerca a millones de operaciones por segundo.

El problema se torna en que uno tiene que programar a la computadora, esto es, diciéndole qué queremos que haga. Desgraciadamente, la computadora no habla nuestro lenguaje y se ve declinada a aprender. Por consiguiente, también está en nosotros el aprender las instrucciones que programarán al computador para desarrollar una tarea particular.

### 1.2.1 Códigos.

En la figura 1.1 imagínese que el contador de programa “apunta hacia” la celda de memoria 7. En esa celda está la palabra dato 76. Sabemos que esa palabra dato será actualmente almacenada en forma binaria como 01001100. ¿Pero este número qué representa? Debemos interpretarlo literalmente como el decimal 76, o ¿éste representa un código para alguna operación del computador?

La respuesta a esta pregunta es que sólo el computador sabe con seguridad. Si el computador está buscando un código de operación, interpretará este byte como un código de operación (incluso si pensamos que es un byte de dato). Si el computador está buscando un byte de dato, lo interpretará como dato. Está en el programador el asegurarse que el código apropiado sea presentado al computador. De esto es lo que se trata la programación de computadoras.

El punto principal que se tiene que ver es que un byte puede tener muchas interpretaciones aparte de su valor decimal. Por ejemplo, el byte 01001100 es interpretado como el decimal 76, el hexadecimal 4C, por un microprocesador 8085 como el código de operación MOV C,H (copia el contenido del registro H en el registro C), el código de operación JMP por el microprocesador

6502, y el código de operación ORL A,R4 (operación lógica OR entre el registro R4 y el acumulador) por el microcontrolador 8749 y 8051.

Existe aún otra interpretación para el byte. En 1968 la ANSI (American National Standards Institute) estableció un código de 7 bits para todas las letras del alfabeto, los números del 0-9, los símbolos comunes de puntuación encontrados en la mayoría de las máquinas de escribir, y varios códigos especiales para propósito de control. Ellos llamaron a su código ASCII (American Standard Code for Information Interchange). Una copia de este código se muestra en la tabla 1.2.

**Tabla 1.2**

**Código Estándar Americano para el Intercambio de Información (ASCII)**

| | Bits menos Significativos | Bits más significativos 0 0000 | 1 0001 | 2 0010 | 3 0011 | 4 0100 | 5 0101 | 6 0110 | 7 0111 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | NUL | DEL | SP | 0 | @ | P | ` | p |
| 1 | 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | 0111 | BEL | ETB | ´ | 7 | G | W | g | w |
| 8 | 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | 1001 | HT | EM | ) | 9 | I | Y | i | y |
| A | 1010 | LF | SUB | * | : | J | Z | j | z |
| B | 1011 | VT | ESC | + | ; | K | [ | k | { |
| C | 1100 | FF | FS | , | < | L | \ | l | \| |
| D | 1101 | CR | GS | - | = | M | ] | m | } |
| E | 1110 | SO | RS | . | > | N | ^ | n | ~ |
| F | 1111 | SI | US | / | ? | O | _ | o | DEL |

Aunque el ASCII es un código de 7 bits, a menudo es escrito en la forma de byte (8 bits) con el bit 8 ignorado (se asume 0) o usado para paridad. Examine la Tabla 1.2, vemos que el byte 01001100 (4C hex) es el código para la letra mayúscula L. Casi todo texto de información esta codificado en el formato ASCII, haciendo posible la transferencia de datos entre dos sistemas de computadoras. Cada vez que usted presiona una tecla sobre un terminal de computadora el código ASCII para este teclado es generado y enviado al computador para su procesamiento.

Actualmente, hay varios códigos en adición al ASCII. Está el código BCD (Decimal Codificado en Binario), útil para aritmética decimal; el código GRAY, en el cual solo cambia un bit entre entradas sucesivas en el código (útil para probar circuitos digitales); el código de complemento a dos mostrado en la Tabla 1.3, en el cual el bit más significativo de cada palabra representa el signo (positivo o negativo) de la palabra, el Baudot o código de cinco niveles utilizados en las primeras máquinas de teletipo y ahora obsoleto, el EBCDIC desarrollado por la IBM para su línea de máquinas de escribir Selectric, y así sucesivamente.

Así que puede ver que hay mucho más para un grupo de dígitos que para su valor decimal. Está en manos del usuario el seleccionar y utilizar el código apropiado para la aplicación entre manos.

**Tabla 1. 3**

**Complemento a dos de los Números Binarios con Signo**

| Decimal | Binario | Hexadecimal |
|---|---|---|
| -128 | 10000000 | 80 |
| -127 | 10000001 | 81 |
| -126 | 10000010 | 82 |
| -125 | 10000011 | 83 |
| -124 | 10000100 | 84 |
| * | * | * |
| * | * | * |
| * | * | * |
| -3 | 1111101 | FD |
| -2 | 11111110 | FE |
| -1 | 11111111 | FF |
| 0 | 00000000 | 0 |
| +1 | 00000001 | 1 |
| +2 | 00000010 | 2 |
| * | * | * |
| * | * | * |
| * | * | * |
| +125 | 01111101 | 7D |
| +126 | 01111110 | 7E |
| +127 | 01111111 | 7F |

## 1.3 LENGUAJES DE COMPUTADORAS

### 1.3.1 Programación en Lenguaje de Máquina y Ensamblado.

El lenguaje de máquina es la representación del programa como el microprocesador o microcontrolador la entiende. No es fácil para el ser humano leerlo y es una causa común de dolores de cabeza. El lenguaje Ensamblado es un formato legible para el ser humano del lenguaje de máquina, la cual hace más fácil para nosotros el tratarlo. Cada proposición en lenguaje Ensamblado corresponde a cada proposición del lenguaje de máquina (sin contar los macros).

Un programa de lenguaje Ensamblado/Máquina es rápido y pequeño. Esto es debido a que usted esta en completo cargo de que es lo que sucede en el programa. Claro esta, si usted escribe un programa lento, largo, tonto, entonces el correrá lentamente, será muy grande, y será tonto. El lenguaje Ensamblado no puede corregir estupideces.

Si usted se inicia en el estudio de los microprocesadores o microcontroladores, le será de gran valor si primero aprende ensamblado. Por la programación en Ensamblado, usted será el amo de

la arquitectura subalterna del chip, el cual es importante si usted intenta hacer cualquier cosa significativa con su microprocesador o microcontrolador.

## 1.4 EL CPU COMO UN RELOJ COMPLEJO

Una manera de visualizar al microprocesador o microcontrolador es como una unidad de temporización compleja. El presenta en su bus de direcciones, direcciones de memoria o de I/O, lee y escribe data sobre su bus de datos, y sincroniza estas actividades con su bus de control.

El microprocesador o microcontrolador controla la temporización de todo el sistema microcomputador. Una vez puesto en movimiento, la CPU interminablemente ejecuta su secuencia de búsqueda y ejecución. Dependerá de los dispositivos de I/O y memoria estar listos cuando les toque su turno. Al microprocesador o microcontrolador no le importa, y no esperará. Leerá tan pronto pueda tanto basura como data con algún significado.

Podemos pensar de las instrucciones almacenadas en la unidad de memoria como códigos que causarán una secuencia particular de eventos en la CPU de tres buses. Manteniendo esta perspectiva en mente, dibujemos un diagrama de tiempo para un programa en lenguaje ensamblado sencillo.

### 1.4.1 Diagramas de Tiempo del Ciclo de Máquina.

La figura 1.5 es un listado de una rutina en lenguaje ensamblado la cual introducirá información desde un dispositivo de I/O por el puerto 3, almacenará esta información en la dirección de memoria 0700H (la letra H mayúscula significa que la dirección se debe interpretar como hexadecimal), y detenerse. Los códigos de instrucción hex se muestran en el centro del listado sobre la misma línea de cada mnemónico. Las direcciones de memoria corresponden a la localidad de memoria en donde los códigos de instrucción son almacenados.

| DIRECCIONES | CÓDIGO DE OPERACIÓN | MNEMÓNICO | COMENTARIOS |
|---|---|---|---|
| 0000 | DB 03 | IN 3 | INSTRODUZCA LA DATA DESDE EL PUERTO 3 AL ACUMULADOR |
| 0002 | 32 00 07 | STA 0700 | ALMACENE EL CONTENIDO DEL ACUMULADOR EN LA PÁGINA 7 LINEA 0 |
| 0005 | 76 | HLT | ALTO |

**Figura 1. 5 Rutina en lenguaje ensamblado utilizado por el Diagrama de Tiempo del Ciclo de Máquina de la Figura 1.6**

Note que las instrucciones IN y STA requieren de códigos de instrucción de dos y tres bytes, respectivamente. El primer byte siempre representa el código de operación para la instrucción. Por consiguiente DB es el código de operación para IN y 32 el código de operación para STA.

Veremos que los microprocesadores y microcontroladores estudiados en este folleto solo utilizan 8 bits para la dirección del I/O. Por esta razón el segundo byte de la instrucción IN corresponden a la dirección de 8 bits del I/O – puerto 3 en este caso.

La instrucción STA requiere una dirección de 16 bits para identificar la celda de memoria en la cual se almacenará la data. Note que el microprocesador o microcontrolador requiere que estos bytes se den en el orden inverso, con la dirección de orden baja primero (00) y la dirección de orden alto al final (07).

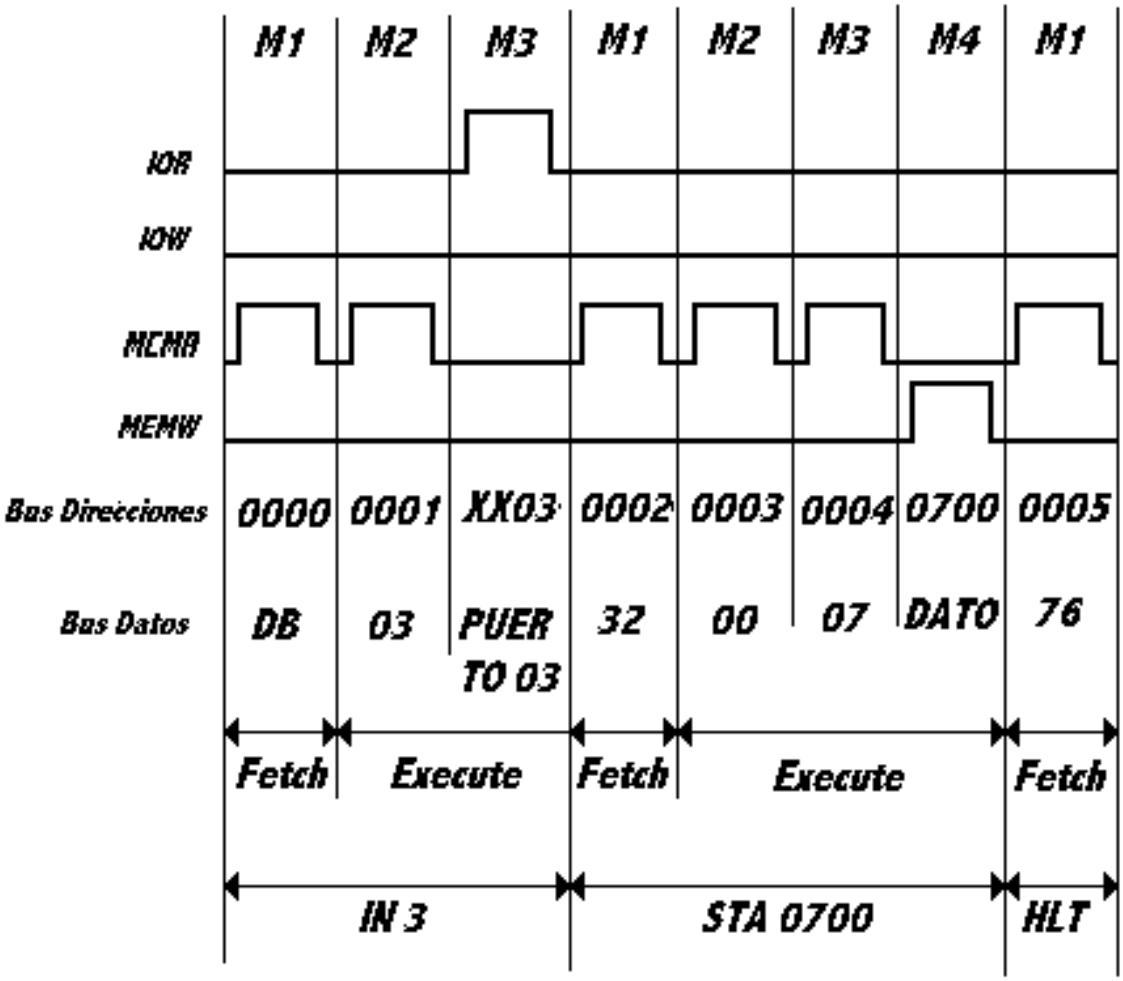


**Figura 1. 6 Diagrama de Tiempo del Ciclo de Máquina. Cada Instrucción produce un patrón único sobre los tres buses del microprocesador. El diagrama es para el programa de la figura 1.5**

La figura 1.6 es el diagrama de tiempo del ciclo de máquina correspondiente a este programa. La presencia de una señal de control es mostrada por un pulso de nivel 1 lógico. Los contenidos de los buses de dirección y datos son dados en hex para una mejor lectura. El diagrama se divide en tres grandes períodos de tiempo correspondientes a las tres instrucciones separadas.

Cada instrucción inicia con un código de operación de búsqueda (fetch – Ciclo de máquina de lectura a memoria). Este ciclo es numerado M1 en la figura 1.6. Siguiendo la instrucción IN 3, el segundo ciclo de máquina (M2) también es un ciclo de lectura a memoria. Esto se debe a que durante el ciclo M1 solo el código de operación fue leído. Este le dice a la CPU que debe introducir data desde el puerto. El ciclo de lectura a memoria M2 identifica la dirección del puerto. Además note que el bus de direcciones se incrementa a 0001 para el ciclo M2.

Una vez que el código de operación y la dirección del puerto son conocidos, la CPU puede ejecutar la instrucción mediante la introducción de la data desde el puerto 3. El ciclo M3 muestra esta fase de ejecución. La línea de control IOR (I/O read) es pulsada, el bus de direcciones muestra 03 (la dirección del puerto) sobre A0 hasta A7, y el bus de datos bidireccional es configurado para entrada de dato. Le corresponde al dispositivo de I/O de colocar su data sobre el bus.

El ciclo de máquina M3 finaliza la instrucción IN y el siguiente código de operación es buscado desde la memoria. Vemos 0002 sobre el bus de direcciones y sobre el bus de dato el código hex 32. Dos ciclos de lectura a memoria siguen en donde se lee primeramente la dirección de memoria de orden bajo y después el orden alto para la instrucción STA. Ahora que la CPU conoce la dirección de memoria apropiada para almacenar la información, él ejecuta un ciclo de escritura a memoria durante M4. Esto se observa por el pulso de escritura a memoria sobre el bus de control, la dirección 0700H sobre el bus de direcciones, y la aparición del byte de dato sobre el bus de datos.

Finalmente, la instrucción HLT puede ser buscada. Esta instrucción no necesita acceso posterior a memoria o I/O y por consiguiente tiene una longitud de solo un ciclo de máquina. La CPU ahora se detiene en un bucle de ocio, esperando por una inicialización para reiniciar la secuencia de búsqueda-y-ejecución.

Los puntos principales de aprender de este diagrama son:

1. Cada instrucción tiene un código de operación con fase de búsqueda-y-ejecución.
2. El ciclo de máquina M1 siempre es de lectura a memoria.
3. La CPU puede desarrollar solo una actividad a la vez y solo una línea de control puede por consiguiente estar activa en cualquier instante dado.
4. Una instrucción sencilla de un microprocesador o microcontrolador puede requerir varios bytes y varios ciclos de máquina; el microprocesador Intel estudiado en este folleto tiene un máximo de cuatro bytes por instrucción y seis ciclos de máquina por instrucción, mientras que el microcontrolador tiene un máximo de tres bytes por instrucción y cuatro ciclos de máquina por instrucción.

### 1.4.2 Tiempo de Instrucción.

La figura 1.7 es un resumen del juego de instrucción para el microprocesador 8085 de Intel. Los mnemónicos son listados en orden alfabético. El equivalente hexadecimal del código de operación es dado. De esta carta usted puede ver que la instrucción IN tiene el código de operación DBH (nuevamente H indica hexadecimal).

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | ACI v | CE v | 2 | 7 |
| | ADC A | 8F | 1 | 4 |
| | ADC B | 88 | 1 | 4 |
| | ADC C | 89 | 1 | 4 |
| | ADC D | 8A | 1 | 4 |
| | ADC E | 8B | 1 | 4 |
| | ADC H | 8C | 1 | 4 |
| | ADC L | 8D | 1 | 4 |
| | ADC M | 8E | 1 | 7 |
| | ADD A | 87 | 1 | 4 |
| | ADD B | 80 | 1 | 4 |
| | ADD C | 81 | 1 | 4 |
| A | ADD D | 82 | 1 | 4 |
| | ADD E | 83 | 1 | 4 |
| | ADD H | 84 | 1 | 4 |
| | ADD L | 85 | 1 | 4 |
| | ADD M | 86 | 1 | 7 |
| | ADI v | C6 v | 2 | 7 |
| | ANA A | A7 | 1 | 4 |
| | ANA B | A0 | 1 | 4 |
| | ANA C | A1 | 1 | 4 |
| | ANA D | A2 | 1 | 4 |
| | ANA E | A3 | 1 | 4 |
| | ANA H | A4 | 1 | 4 |
| | ANA L | A5 | 1 | 4 |
| | ANA M | A6 | 1 | 7 |
| | ANI V | E6 v | 2 | 7 |
| | CALL a | CD a | 3 | 18 |
| | CC a | DC a | 3 | 9/18 |
| | CM a | FC a | 3 | 9/18 |
| | CMA | 2F | 1 | 4 |
| | CMC | 3F | 1 | 4 |
| | CMP A | BF | 1 | 4 |
| | CMP B | B8 | 1 | 4 |
| | CMP C | B9 | 1 | 4 |
| | CMP D | BA | 1 | 4 |
| C | CMP E | BB | 1 | 4 |
| | CMP H | BC | 1 | 4 |
| | CMP L | BD | 1 | 4 |
| | CMP M | BE | 1 | 7 |
| | CNC a | D4 a | 3 | 9/18 |
| | CNZ a | C4 a | 3 | 9/18 |
| | CP a | F4 a | 3 | 9/18 |
| | CPE a | EC a | 3 | 9/18 |
| | CPI v | FE v | 2 | 7 |
| | CPO a | E4 a | 3 | 9/18 |
| | CZ a | CC a | 3 | 9/18 |
| | DAA | 27 | 1 | 4 |
| | DAD B | 09 | 1 | 10 |
| | DAD D | 19 | 1 | 10 |
| D | DAD H | 29 | 1 | 10 |
| | DAD SP | 39 | 1 | 10 |
| | DCR A | 3D | 1 | 4 |
| | DCR B | 05 | 1 | 4 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | DCR C | 0D | 1 | 4 |
| | DCR D | 15 | 1 | 4 |
| | DCR E | 1D | 1 | 4 |
| | DCR H | 25 | 1 | 4 |
| D | DCR L | 2D | 1 | 4 |
| | DCR M | 35 | 1 | 10 |
| | DCX B | 0B | 1 | 6 |
| | DCX D | 1B | 1 | 6 |
| | DCX H | 2B | 1 | 6 |
| | DCX SP | 3B | 1 | 6 |
| | DI | F3 | 1 | 4 |
| E | EI | FB | 1 | 4 |
| H | HLT | 76 | 1 | 4 |
| | IN v | DB v | 2 | 10 |
| | INR A | 3C | 1 | 4 |
| | INR B | 04 | 1 | 4 |
| | INR C | 0C | 1 | 4 |
| | INR D | 14 | 1 | 4 |
| | INR E | 1C | 1 | 4 |
| I | INR H | 24 | 1 | 4 |
| | INR L | 2C | 1 | 4 |
| | INR M | 34 | 1 | 10 |
| | INX B | 03 | 1 | 6 |
| | INX D | 13 | 1 | 6 |
| | INX H | 23 | 1 | 6 |
| | INX SP | 33 | 1 | 6 |
| | JC a | DA a | 3 | 7/10 |
| | JM a | FA a | 3 | 7/10 |
| | JMP a | C3 a | 3 | 10 |
| | JNC a | D2 a | 3 | 7/10 |
| J | JNZ a | C2 a | 3 | 7/10 |
| | JP a | F2 a | 3 | 7/10 |
| | JPE a | EA a | 3 | 7/10 |
| | JPO a | E2 a | 3 | 7/10 |
| | JZ a | CA a | 3 | 7/10 |
| | LDA a | 3A a | 3 | 13 |
| | LDAX B | 0A | 1 | 7 |
| | LDAX D | 1A | 1 | 7 |
| L | LHLD a | 2A | 3 | 16 |
| | LXI B,vv | 01 | 3 | 10 |
| | LXI D,vv | 11 | 3 | 10 |
| | LXI H,vv | 21 | 3 | 10 |
| | LXI SP,vv | 31 | 3 | 10 |
| | MOV A,A | 7F | 1 | 4 |
| | MOV A,B | 78 | 1 | 4 |
| | MOV A,C | 79 | 1 | 4 |
| | MOV A,D | 7A | 1 | 4 |
| | MOV A,E | 7B | 1 | 4 |
| M | MOV A,H | 7C | 1 | 4 |
| | MOV A.L | 7D | 1 | 4 |
| | MOV A,M | 7E | 1 | 4 |
| | MOV B.A | 47 | 1 | 4 |
| | MOV B,B | 40 | 1 | 4 |
| | MOV B,C | 41 | 1 | 4 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| M | MOV B,D | 42 | 1 | 4 |
| | MOV B,E | 43 | 1 | 4 |
| | MOV B,H | 44 | 1 | 4 |
| | MOV B,L | 45 | 1 | 4 |
| | MOV B,M | 46 | 1 | 7 |
| | MOV C,A | 4F | 1 | 4 |
| | MOV C,B | 48 | 1 | 4 |
| | MOV C,C | 49 | 1 | 4 |
| | MOV C,D | 4A | 1 | 4 |
| | MOV C,E | 4B | 1 | 4 |
| | MOV C,H | 4C | 1 | 4 |
| | MOV C,L | 4D | 1 | 4 |
| | MOV C,M | 4E | 1 | 7 |
| | MOV D,A | 57 | 1 | 4 |
| | MOV D,B | 50 | 1 | 4 |
| | MOV D,C | 51 | 1 | 4 |
| | MOV D,D | 52 | 1 | 4 |
| | MOV D,E | 53 | 1 | 4 |
| | MOV D,H | 54 | 1 | 4 |
| | MOV D,L | 55 | 1 | 4 |
| | MOV D,M | 56 | 1 | 7 |
| | MOV E,A | 5F | 1 | 4 |
| | MOV E,B | 58 | 1 | 4 |
| | MOV E,C | 59 | 1 | 4 |
| | MOV E,D | 5A | 1 | 4 |
| | MOV E,E | 5B | 1 | 4 |
| | MOV E,H | 5C | 1 | 4 |
| | MOV E,L | 5D | 1 | 4 |
| | MOV E,M | 5E | 1 | 7 |
| | MOV H,A | 67 | 1 | 4 |
| | MOV H,B | 60 | 1 | 4 |
| | MOV H,C | 61 | 1 | 4 |
| | MOV H,D | 62 | 1 | 4 |
| | MOV H,E | 63 | 1 | 4 |
| | MOV H,H | 64 | 1 | 4 |
| | MOV H,L | 65 | 1 | 4 |
| | MOV H,M | 66 | 1 | 7 |
| | MOV L,A | 6F | 1 | 4 |
| | MOV L,B | 68 | 1 | 4 |
| | MOV L,C | 69 | 1 | 4 |
| | MOV L,D | 6A | 1 | 4 |
| | MOV L,E | 6B | 1 | 4 |
| | MOV L,H | 6C | 1 | 4 |
| | MOV L,L | 6D | 1 | 4 |
| | MOV L,M | 6E | 1 | 7 |
| | MOV M,A | 77 | 1 | 7 |
| | MOV M,B | 70 | 1 | 7 |
| | MOV M,C | 71 | 1 | 7 |
| | MOV M,D | 72 | 1 | 7 |
| | MOV M,E | 73 | 1 | 7 |
| | MOV M,H | 74 | 1 | 7 |
| | MOV M,L | 75 | 1 | 7 |
| | MVI A,v | 3E | 2 | 7 |
| | MVI B,v | 06 | 2 | 7 |
| | MVI C,v | 0E | 2 | 7 |
| | MVI D,v | 16 | 2 | 7 |
| | MVI E,v | 1E | 2 | 7 |
| | MVI H,v | 26 | 2 | 7 |
| | MVI L,v | 2E | 2 | 7 |
| | MVI M,v | 36 v | 2 | 10 |
| N | NOP | 00 | 1 | 4 |
| O | ORA A | B7 | 1 | 4 |
| | ORA B | B0 | 1 | 4 |
| | ORA C | B1 | 1 | 4 |
| | ORA D | B2 | 1 | 4 |
| | ORA E | B3 | 1 | 4 |
| | ORA H | B4 | 1 | 4 |
| | ORA L | B5 | 1 | 4 |
| | ORA M | B6 | 1 | 7 |
| | ORI v | F6 v | 2 | 7 |
| | OUT v | D3 v | 2 | 10 |
| P | PCHL | E9 | 1 | 6 |
| | POP B | C1 | 1 | 10 |
| | POP D | D1 | 1 | 10 |
| | POP H | E1 | 1 | 10 |
| | POP SP | F1 | 1 | 10 |
| | PUSH B | C5 | 1 | 12 |
| | PUSH D | D5 | 1 | 12 |
| | PUSH H | E5 | 1 | 12 |
| | PUSH SP | F5 | 1 | 12 |
| R | RAL | 17 | 1 | 4 |
| | RAR | 1F | 1 | 4 |
| | RC | D8 | 1 | 6/12 |
| | RET | C9 | 1 | 10 |
| | RIM | 20 | 1 | 4 |
| | RLC | 07 | 1 | 4 |
| | RM | F8 | 1 | 6/12 |
| | RNC | D0 | 1 | 6/12 |
| | RNZ | C0 | 1 | 6/12 |
| | RP | F0 | 1 | 6/12 |
| | RPE | E8 | 1 | 6/12 |
| | RPO | E0 | 1 | 6/12 |
| | RRC | 0F | 1 | 4 |
| | RST 0 | C7 | 1 | 12 |
| | RST 1 | CF | 1 | 12 |
| | RST 2 | D7 | 1 | 12 |
| | RST 3 | DF | 1 | 12 |
| | RST 4 | E7 | 1 | 12 |
| | RST 5 | EF | 1 | 12 |
| | RST 6 | F7 | 1 | 12 |
| | RST 7 | FF | 1 | 12 |
| | RZ | C8 | 1 | 6/12 |
| S | SBB A | 9F | 1 | 4 |
| | SBB B | 98 | 1 | 4 |
| | SBB C | 99 | 1 | 4 |
| | SBB D | 9A | 1 | 4 |
| | SBB E | 9B | 1 | 4 |
| | SBB H | 9C | 1 | 4 |
| | SBB L | 9D | 1 | 4 |
| | SBB M | 9E | 1 | 7 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | SBI v | DE v | 2 | 7 |
| | SHLD a | 22 a | 3 | 16 |
| | SIM | 30 | 1 | 4 |
| | SPHL | F9 | 1 | 6 |
| | STA a | 32 a | 3 | 13 |
| | STAX B | 02 | 1 | 7 |
| S | STAX D | 12 | 1 | 7 |
| | STC | 37 | 1 | 4 |
| | SUB A | 97 | 1 | 4 |
| | SUB B | 90 | 1 | 4 |
| | SUB C | 91 | 1 | 4 |
| | SUB D | 92 | 1 | 4 |
| | SUB E | 93 | 1 | 4 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | SUB L | 95 | 1 | 4 |
| S | SUB M | 96 | 1 | 7 |
| | SUI a | D6 | 2 | 7 |
| | XCHG | EB | 1 | 4 |
| | XRA A | AF | 1 | 4 |
| | XRA B | A8 | 1 | 4 |
| | XRA C | A9 | 1 | 4 |
| X | XRA D | AA | 1 | 4 |
| | XRA E | AB | 1 | 4 |
| | XRA H | AC | 1 | 4 |
| | XRA L | AD | 1 | 4 |
| | XRA M | AE | 1 | 7 |
| | XRI a | EE a | 2 | 7 |

**Figura 1. 7 Juego de Instrucciones del microprocesador 8085**

La carta también incluye el número de ciclos de reloj – algunas veces llamados Estados T – requeridos por cada instrucción. La instrucción IN requiere 10 ciclos de reloj, la instrucción STA requiere 13, y la instrucción HLT 5. De esta información podemos concluir que el programa dado en la figura 1.5 requerirá 28 ciclos de reloj. Si el reloj del sistema computador es 2MHz, el tiempo requerido para ejecutar este programa es

$$28\_ciclos\_de\_reloj \times \frac{1}{2MHz} = 14\mu s$$

La carta en la figura 1.7 no indica como estos ciclos de reloj son distribuidos sobre los muchos ciclos de máquina envueltos. Esta información puede ser encontrada (si le interesa) en el manual del usuario del microprocesador. El punto importante de recordar es que *no todos los ciclos de máquina son de igual longitud* aunque se pueda pensar que la figura 1.6 da esa impresión.

Se muestra en la figura 1.8 el juego de instrucciones para el microcontrolador de la familia MCS-51.

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | ACALL a11 | | 2 | 24 |
| | ADD A,R0 | 28 | 1 | 12 |
| | ADD A,R1 | 29 | 1 | 12 |
| | ADD A,R2 | 2A | 1 | 12 |
| | ADD A,R3 | 2B | 1 | 12 |
| | ADD A,R4 | 2C | 1 | 12 |
| A | ADD A,R5 | 2D | 1 | 12 |
| | ADD A,R6 | 2E | 1 | 12 |
| | ADD A,R7 | 2F | 1 | 12 |
| | ADD A,direct | 25 direct | 2 | 12 |
| | ADD A,@R0 | 26 | 1 | 12 |
| | ADD A,@R1 | 27 | 1 | 12 |
| | ADD A,#v | 24 #v | 2 | 12 |
| | ADDC A,R0 | 38 | 1 | 12 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | ADDC A,R1 | 39 | 1 | 12 |
| | ADDC A,R2 | 3A | 1 | 12 |
| | ADDC A,R3 | 3B | 1 | 12 |
| | ADDC A,R4 | 3C | 1 | 12 |
| | ADDC A,R5 | 3D | 1 | 12 |
| | ADDC A,R6 | 3E | 1 | 12 |
| A | ADDC A,R7 | 3F | 1 | 12 |
| | ADDC A,direct | 35 direct | 2 | 12 |
| | ADDC A,@R0 | 36 | 1 | 12 |
| | ADDC A,@R1 | 37 | 1 | 12 |
| | ADDC A,#v | 34 #v | 2 | 12 |
| | ANL A,R0 | 58 | 1 | 12 |
| | ANL A,R1 | 59 | 1 | 12 |
| | ANL A,R2 | 5A | 1 | 12 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| A | ANL A,R3 | 5B | 1 | 12 |
| | ANL A,R4 | 5C | 1 | 12 |
| | ANL A,R5 | 5D | 1 | 12 |
| | ANL A,R6 | 5E | 1 | 12 |
| | ANL A,R7 | 5F | 1 | 12 |
| | ANL A,direct | 55 direct | 2 | 12 |
| | ANL A,@R0 | 56 | 1 | 12 |
| | ANL A,@R1 | 57 | 1 | 12 |
| | ANL A,#v | 54 #v | 2 | 12 |
| | ANL direct,A | 52 direct | 2 | 12 |
| | ANL direct,#v | 53 direct #v | 3 | 24 |
| | ANL C,bit | 82 bit_a | 2 | 24 |
| | ANL C,/bit | B0 bit_a | 2 | 24 |
| | AJMP a11 | | 2 | 24 |
| C | CJNE A,direct,rel | B5 direct,rel | 3 | 24 |
| | CJNE A,#v,rel | B4 #v,rel | 3 | 24 |
| | CJNE R0,#v,rel | B8 #v,rel | 3 | 24 |
| | CJNE R1,#v,rel | B9 #v,rel | 3 | 24 |
| | CJNE R2,#v,rel | BA #v,rel | 3 | 24 |
| | CJNE R3,#v,rel | BB #v,rel | 3 | 24 |
| | CJNE R4,#v,rel | BC #v,rel | 3 | 24 |
| | CJNE R5,#v,rel | BD #v,rel | 3 | 24 |
| | CJNE R6,#v,rel | BE #v,rel | 3 | 24 |
| | CJNE R7,#v,rel | BF #v,rel | 3 | 24 |
| | CJNE @R0,#v,rel | B6 #v,rel | 3 | 24 |
| | CJNE @R1,#v,rel | B7 #v,rel | 3 | 24 |
| | CLR A | F4 | 1 | 12 |
| | CLR C | C3 | 1 | 12 |
| | CLR bit | C2 bit | 2 | 12 |
| | CPL A | F4 | 1 | 12 |
| | CPL C | B3 | 1 | 12 |
| | CPL bit | B2 bit | 2 | 12 |
| D | DA A | D4 | 1 | 12 |
| | DEC A | 14 | 1 | 12 |
| | DEC R0 | 18 | 1 | 12 |
| | DEC R1 | 19 | 1 | 12 |
| | DEC R2 | 1A | 1 | 12 |
| | DEC R3 | 1B | 1 | 12 |
| | DEC R4 | 1C | 1 | 12 |
| | DEC R5 | 1D | 1 | 12 |
| | DEC R6 | 1E | 1 | 12 |
| | DEC R7 | 1F | 1 | 12 |
| | DEC direct | 15 direct | 2 | 12 |
| | DEC @R0 | 16 | 1 | 12 |
| | DEC @R1 | 17 | 1 | 12 |
| | DIV AB | 86 | 1 | 48 |
| | DJNZ R0,rel | D8 rel | 2 | 24 |
| | DJNZ R1,rel | D9 rel | 2 | 24 |
| | DJNZ R2,rel | DA rel | 2 | 24 |
| | DJNZ R3,rel | DB rel | 2 | 24 |
| | DJNZ R4,rel | DC rel | 2 | 24 |
| | DJNZ R5,rel | DD rel | 2 | 24 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| D | DJNZ R6,rel | DE rel | 2 | 24 |
| | DJNZ R7,rel | DF rel | 2 | 24 |
| | DJNZ direct,rel | D5 direct,rel | 3 | 24 |
| I | INC A | 04 | 1 | 12 |
| | INC R0 | 08 | 1 | 12 |
| | INC R1 | 09 | 1 | 12 |
| | INC R2 | 0A | 1 | 12 |
| | INC R3 | 0B | 1 | 12 |
| | INC R4 | 0C | 1 | 12 |
| | INC R5 | 0D | 1 | 12 |
| | INC R6 | 0E | 1 | 12 |
| | INC R7 | 0F | 1 | 12 |
| | INC direct | 05 direct | 2 | 12 |
| | INC @R0 | 06 | 1 | 12 |
| | INC @R1 | 07 | 1 | 12 |
| | INC DPTR | A3 | 1 | 24 |
| J | JB bit,rel | 20 bit rel | 3 | 24 |
| | JBC bit,rel | 10 bit rel | 3 | 24 |
| | JC rel | 40 rel | 2 | 24 |
| | JMP @A+DPTR | 73 | 1 | 24 |
| | JNB bit,rel | 30 bit rel | 3 | 24 |
| | JNC rel | 50 rel | 2 | 24 |
| | JNZ rel | 70 rel | 2 | 24 |
| | JZ rel | 60 rel | 2 | 24 |
| L | LCALL a | | 3 | 24 |
| | LJMP a | | 3 | 24 |
| M | MOV A,R0 | E8 | 1 | 12 |
| | MOV A,R1 | E9 | 1 | 12 |
| | MOV A,R2 | EA | 1 | 12 |
| | MOV A,R3 | EB | 1 | 12 |
| | MOV A,R4 | EC | 1 | 12 |
| | MOV A,R5 | ED | 1 | 12 |
| | MOV A,R6 | EE | 1 | 12 |
| | MOV A,R7 | EF | 1 | 12 |
| | MOV A,direct | E5 direct | 2 | 12 |
| | MOV A,@R0 | E6 | 1 | 12 |
| | MOV A,@R1 | E7 | 1 | 12 |
| | MOV A,#v | 74 #v | 2 | 12 |
| | MOV R0,A | F8 | 1 | 12 |
| | MOV R1,A | F9 | 1 | 12 |
| | MOV R2,A | FA | 1 | 12 |
| | MOV R3,A | FB | 1 | 12 |
| | MOV R4,A | FC | 1 | 12 |
| | MOV R5,A | FD | 1 | 12 |
| | MOV R6,A | FE | 1 | 12 |
| | MOV R7,A | FF | 1 | 12 |
| | MOV R0,direct | A8 direct | 2 | 24 |
| | MOV R1,direct | A9 direct | 2 | 24 |
| | MOV R2,direct | AA direct | 2 | 24 |
| | MOV R3,direct | AB direct | 2 | 24 |
| | MOV R4,direct | AC direct | 2 | 24 |
| | MOV R5,direc | AD direct | 2 | 24 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| M | MOV R6,direct | AE direct | 2 | 24 |
| | MOV R7,direct | AF direct | 2 | 24 |
| | MOV R0,#v | 78 #v | 2 | 12 |
| | MOV R1,#v | 79 #v | 2 | 12 |
| | MOV R2,#v | 7A #v | 2 | 12 |
| | MOV R3,#v | 7B #v | 2 | 12 |
| | MOV R4,#v | 7C #v | 2 | 12 |
| | MOV R5,#v | 7D #v | 2 | 12 |
| | MOV R6,#v | 7E #v | 2 | 12 |
| | MOV R7,#v | 7F #v | 2 | 12 |
| | MOV direct,A | F5 direct | 2 | 12 |
| | MOV direct,R0 | 88 direct | 2 | 24 |
| | MOV direct,R1 | 89 direct | 2 | 24 |
| | MOV direct,R2 | 8A direct | 2 | 24 |
| | MOV direct,R3 | 8B direct | 2 | 24 |
| | MOV direct,R4 | 8C direct | 2 | 24 |
| | MOV direct,R5 | 8D direct | 2 | 24 |
| | MOV direct,R6 | 8E direct | 2 | 24 |
| | MOV direct,R7 | 8F direct | 2 | 24 |
| | MOV direct,direct | 85 d1,d2 | 3 | 24 |
| | MOV direct,@R0 | 86 direct | 2 | 24 |
| | MOV direct,@R1 | 87 direct | 2 | 24 |
| | MOV direct,#v | 75 direct,#v | 3 | 24 |
| | MOV @R0,A | F6 | 1 | 12 |
| | MOV @R1,A | F7 | 1 | 12 |
| | MOV @R0,direct | A6 direct | 2 | 24 |
| | MOV @R1,direct | A7 direct | 2 | 24 |
| | MOV @R0,#v | 76 #v | 2 | 12 |
| | MOV @R1,#v | 77 #v | 2 | 12 |
| | MOV C,bit | A2 bit_a | 2 | 24 |
| | MOV bit,C | 92 bit_a | 2 | 24 |
| | MOV DPTR,#vv | 90 #vv | 3 | 24 |
| | MOVC A,@A+DPTR | 93 | 1 | 24 |
| | MOVC A,@A+PC | 83 | 1 | 24 |
| | MOVX A,@R0 | E2 | 1 | 24 |
| | MOVX A,@R1 | E3 | 1 | 24 |
| | MOVX A,@DPTR | E0 | 1 | 24 |
| | MOVX @R0,A | F2 | 1 | 24 |
| | MOVX @R1,A | F3 | 1 | 24 |
| | MOVX @DPTR,A | F0 | 1 | 24 |
| | MUL AB | A4 | 1 | 48 |
| N | NOP | 00 | 1 | 12 |
| O | ORL A,R0 | 48 | 1 | 12 |
| | ORL A,R1 | 49 | 1 | 12 |
| | ORL A,R2 | 4A | 1 | 12 |
| | ORL A,R3 | 4B | 1 | 12 |
| | ORL A,R4 | 4C | 1 | 12 |
| | ORL A,R5 | 4D | 1 | 12 |
| | ORL A,R6 | 4E | 1 | 12 |
| | ORL A,R7 | 4F | 1 | 12 |
| | ORL A,direct | 45 direct | 2 | 12 |
| | ORL A,@R0 | 56 | 1 | 12 |
| | ORL A,@R1 | 57 | 1 | 12 |

| | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | ORL A,#v | 44 #v | 2 | 12 |
| O | ORL direct,A | 42 direct | 2 | 12 |
| | ORL direct,#v | 43 direct #v | 3 | 24 |
| | ORL C,bit | 72 bit_a | 2 | 24 |
| | ORL C,/bit | A0 bit_a | 2 | 24 |
| | POP direct | D0 direct | 2 | 24 |
| P | PUSH direct | C0 direct | 2 | 24 |
| | RET | 22 | 1 | 24 |
| | RETI | 32 | 1 | 24 |
| R | RL A | 23 | 1 | 12 |
| | RLC A | 33 | 1 | 12 |
| | RR A | 03 | 1 | 12 |
| | RRC A | 13 | 1 | 12 |
| | SETB C | D3 | 1 | 12 |
| | SETB bit | D2 bit_a | 2 | 12 |
| | SJMP rel | 80 rel | 2 | 24 |
| | SUBB A,R0 | 98 | 1 | 12 |
| | SUBB A,R1 | 99 | 1 | 12 |
| | SUBB A,R2 | 9A | 1 | 12 |
| | SUBB A,R3 | 9B | 1 | 12 |
| S | SUBB A,R4 | 9C | 1 | 12 |
| | SUBB A,R5 | 9D | 1 | 12 |
| | SUBB A,R6 | 9E | 1 | 12 |
| | SUBB A,R7 | 9F | 1 | 12 |
| | SUBB A,direct | 95 direct | 2 | 12 |
| | SUBB A,@R0 | 96 | 1 | 12 |
| | SUBB A,@R1 | 97 | 1 | 12 |
| | SUBB A,#v | 94 | 2 | 12 |
| | SWAP A | C4 | 1 | 12 |
| | XCH A,R0 | C8 | 1 | 12 |
| | XCH A,R1 | C9 | 1 | 12 |
| | XCH A,R2 | CA | 1 | 12 |
| | XCH A,R3 | CB | 1 | 12 |
| | XCH A,R4 | CC | 1 | 12 |
| | XCH A,R5 | CD | 1 | 12 |
| | XCH A,R6 | CE | 1 | 12 |
| | XCH A,R7 | CF | 1 | 12 |
| | XCH A,direct | C5 direct | 2 | 12 |
| | XCH A,@R0 | C6 | 1 | 12 |
| | XCH A,@R1 | C7 | 1 | 12 |
| | XCHD A,@R0 | D6 | 1 | 12 |
| X | XCHD A,@R1 | D7 | 1 | 12 |
| | XRL A,R0 | 68 | 1 | 12 |
| | XRL A,R1 | 69 | 1 | 12 |
| | XRL A,R2 | 6A | 1 | 12 |
| | XRL A,R3 | 6B | 1 | 12 |
| | XRL A,R4 | 6C | 1 | 12 |
| | XRL A,R5 | 6D | 1 | 12 |
| | XRL A,R6 | 6E | 1 | 12 |
| | XRL A,R7 | 6F | 1 | 12 |
| | XRL A,direct | 65 direct | 2 | 12 |
| | XRL A,@R0 | 66 | 1 | 12 |
| | XRL A,@R1 | 67 | 1 | 12 |

| O | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | XRL A,#v | 64 #v | 2 | 12 |

| O | INSTRUCCIÓN | CÓDIGO | BYTES | T |
|---|---|---|---|---|
| | XRL direct,A | 62 direct | 2 | 12 |
| | XRL direct,#v | 63 direct,#v | 3 | 24 |

**Figura 1. 8 Juego de Instrucciones de la Familia MCS-51**

### 1.4.3 Tiempo del Procesador.

La figura 1.9 ilustra la temporización básica del procesador para los ciclos de máquina de lectura y escritura a memoria. Cada estado T se identifica como un pulso del reloj del sistema. En esta instancia ambos ciclos de máquina se muestran con longitud de tres estados T, pero esto puede no ser cierto siempre.

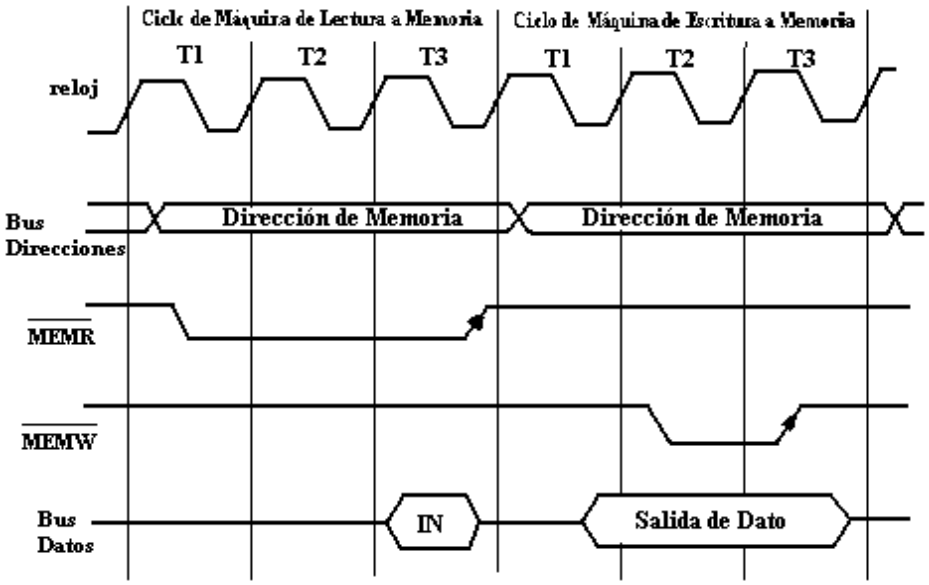


**Figura 1. 9 Temporización del Procesador para los ciclos de máquina de lectura y escritura a memoria**

A cambio de tratar de mostrar las 16 líneas por separado del bus de direcciones, se muestran como dos líneas paralelas simbolizando que algunas de las líneas tienen un nivel lógico de 1 y otras un nivel lógico de 0. Cuando estas líneas se cruzan, una nueva dirección es mostrada por la CPU. Similarmente, las ocho líneas del bus de datos son definidas solamente cuando un dato válido está presente y se deja desconocido bajo otra condición.

Examinando el ciclo de máquina de lectura a memoria de la figura 1.9, el contenido del bus de direcciones se torna válido durante el ciclo T1. Cerca del final de este ciclo de máquina la línea de $\overline{MEMR}$ (lectura a memoria) se vuelve activa (la barra dibujada por arriba de MEMR significa que esta línea toma un nivel bajo cuando se activa). La unidad de memoria tiene tiempo hasta el flanco descendente del reloj durante el ciclo T3 para encontrar la data pedida por la CPU

y colocarla sobre el bus de datos. A este tiempo se le refiere como el tiempo de acceso de la memoria.

Note que el bus de datos esta a la expectativa para recibir data válida desde la memoria cuando el flanco descendente de T3 ocurra, y en ningún otro tiempo durante este ciclo de máquina hay dato válido sobre el bus de datos.

El ciclo de escritura a memoria es similar al ciclo de lectura a memoria excepto que el procesador es el que presenta la data en vez de la unidad de memoria. Para darle a la memoria el suficiente tiempo para fijar la data, el bus de datos, temprano, en el ciclo de máquina contiene data válida. Cuando la línea $\overline{MEMW}$ toma nivel bajo la unidad de memoria almacena el byte sobre el bus de datos en la dirección especificada por las líneas de dirección A0 hasta A15.

La figura 1.10 ilustra la temporización del procesador para las operaciones de lectura y escritura a I/O. Usted puede ser capaz de ver que estas son idénticas a los ciclos de lectura y escritura a memoria con la excepción que las líneas $\overline{IOR}$ e $\overline{IOW}$ serán activas en vez de $\overline{MEMR}$ y $\overline{MEMW}$ .

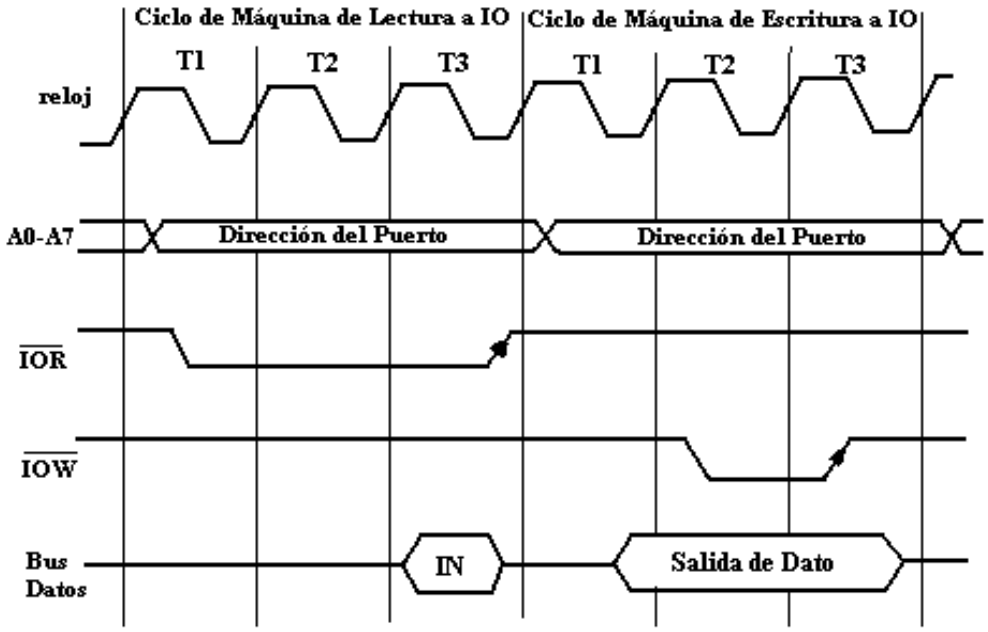


**Figura 1. 10 Temporización del Procesador para los ciclos de máquina de lectura y escritura de I/O**