



Universidad
de Huelva

Escuela Politécnica Superior
Universidad de Huelva

TERCER CURSO. ELECTRÓNICA DIGITAL

INTRODUCCIÓN AL 8051



Manuel Sánchez Raya
Versión 1.0
20 de Marzo de 2000

ÍNDICE

1.- Introducción.....	3
2.- Software.....	4
3.- Arquitectura de la familia MCS-51.....	6
3.1.- Introducción.....	6
3.2.- Almacenamiento de código en el 8051.....	10
3.2.1.- RAM interna en el 8051.....	10
3.2.2.- Memoria fuera del chip.....	11
3.4.- Entrada y Salida.....	11
3.5.- Secuencias de ejecución de instrucciones en el 8051.....	12
3.5.1.- Señales de control del bus.....	13
3.5.2.- El oscilador.....	15
3.6.- Banco de Registros.....	15
3.7.- Registros de Función Especial.....	16
4.- Decodificación de direcciones.....	16
4.1.- Decodificador de una sola dirección.....	17
4.2.- Decodificador de direcciones múltiple.....	17
5.- Montaje de Microcontroladores.....	18
5.1.- Independiente.....	18
5.2.- Expansión de memoria para el 8051.....	19
5.2.1.- Código externo.....	19
5.2.2.- Código Externo, Memoria RAM Externa y Puertos Externos.....	20

BIBLIOGRAFÍA

C and the 8051, Mc Graw-Hill

Apuntes de la asignatura Procesadores de Propósito General

Introducción a los Microcontroladores, Jose Adolfo González Vázquez Mc Graw-Hill

Apuntes de prácticas: Manual del compilador C51 de KEIL

ARQUITECTURA Y PROGRAMACIÓN DEL 8051

1.- Introducción.

Intel diseñó allá por el año 1982, el miembro más importante de la familia de microcontroladores MCS-51 para aplicaciones industriales, el 8051. Este microcontrolador contaba con 4K de EPROM y 128 bytes de memoria RAM interna. Posteriormente se fueron añadiendo más miembros a esta familia que aumentaron la cantidad de EPROM y RAM internas, posibilitando la creación de programas aún más complejos.

Un microcontrolador consta de los siguientes elementos básicos:

- ?? CPU.
- ?? Memoria dedicada a contener programas (ROM).
- ?? Memoria para soporte de datos temporales (RAM).
- ?? Módulos de Entrada/Salida (Puertos paralelo y serie, temporizadores, etc).

Con carácter general, los microcontroladores se emplean en los sistemas de control que reúnan las siguientes propiedades:

1. El precio es un factor crítico.
2. La capacidad de memoria necesaria es pequeña.
3. El procesamiento se realiza en tiempo real.
4. Se opera en decimal y se manipulan bits específicos.
5. No se gobiernan muchos periféricos.
6. Son sistemas dedicados y siempre ejecutan el mismo programa de aplicación.

Estas características diferencian a los microcontroladores de los sistemas que se desarrollan alrededor de un microprocesador, en los cuales se precisa mucha memoria, acceso directo a memoria (DMA), diversas interrupciones y posibilidad de controlar numerosos periféricos. Entre las áreas de aplicación más importantes de los microcontroladores de 8 bits podemos destacar las siguientes:

- ✂✂ Electrodomésticos.
- ✂✂ Automoción.
- ✂✂ Teclados, displays, impresoras, modems y calculadoras.
- ✂✂ Instrumentos de medida y sistemas de alarma.
- ✂✂ Electromedicina.
- ✂✂ Equipos de audio y video.
- ✂✂ Paneles de control y comunicaciones móviles.
- ✂✂ Máquinas de coser y bordar.
- ✂✂ Juegos electrónicos y tragaperras.

Aunque nos concentramos en la familia del 8051, esto no limita el campo de acción, puesto que si encontramos el compilador de C apropiado, podemos transferir los ejemplos, a pesar de las extensiones del lenguaje, a otras familias de microcontroladores. Puesto que la mayoría de microcontroladores tienen temporizadores internos similares y puertos, los ejemplos que emplean características internas también se pueden adaptar. Aunque los comandos para arrancar los temporizadores e interrupciones pueden comportarse de forma diferente, los principios básicos se mantienen.

2.- Software.

Para escribir un programa en lenguaje de alto nivel no es necesario tener idea de cómo funciona el hardware por dentro, como sucede con muchos programadores novatos. Como vamos a usar el lenguaje de programación C bastante, vamos a comenzar viendo un ejemplo para aclarar el objetivo del tema. Ahora veremos como se transforman las líneas de un programa en C a las instrucciones máquina específicas que se ejecutan en el microcontrolador.

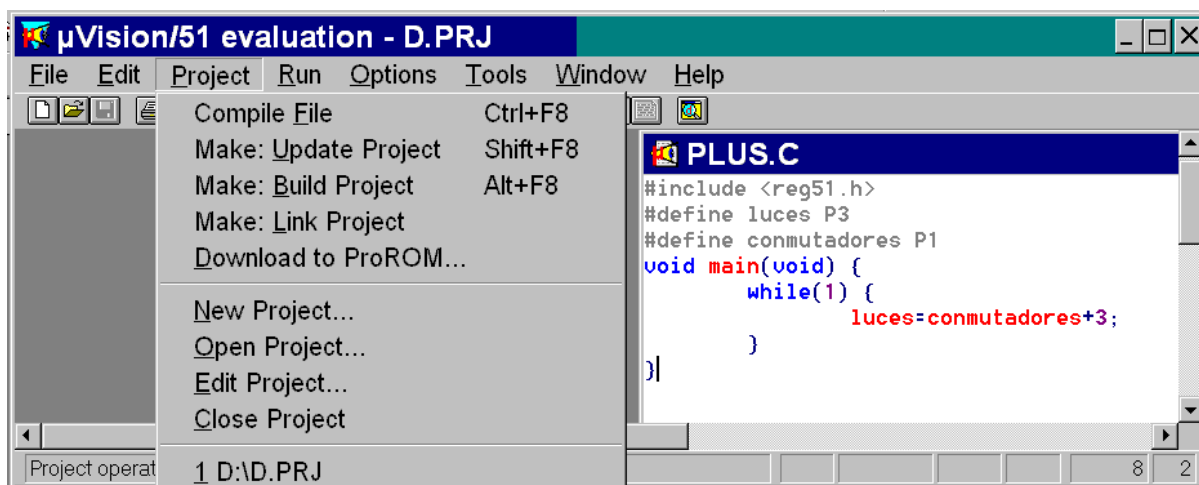
Para comenzar, usaremos un programa simple que para su funcionamiento necesita como entradas ocho conmutadores conectados a un puerto de entrada/salida (E/S) y ocho LEDs a un segundo puerto. El programa lee el patrón de conmutadores, suma tres al valor que ha leído, y manda el resultado a los LEDs. En otros apartados hablaremos del hardware, pero por ahora observemos el software que necesitamos y como funciona.

```
/* Programa para sumar tres a las entradas de los conmutadores */
#include <reg51.h>
#define luces P3
#define conmutadores P1

void main(void) {
    while(1) {
        luces = conmutadores + 3;
    }
}
```

Este es el programa en C que hace esto. Como pasa con la mayoría de programas en C, hay varias líneas antes de las instrucciones que hacen la función. Realmente, la única línea que funciona es la séptima donde se les suma tres a las entradas de los conmutadores antes de enviar el dato a los LEDs. La segunda línea incluye un fichero que le indica al compilador todos los detalles hardware específicos a la familia 8051. Las dos líneas siguientes asignan el nombre *luces* al puerto 3 (P3) y el nombre conmutadores al puerto 1 (P1). La línea siguiente es necesaria porque en todo programa C completo necesita una función main, después analizaremos las subrutinas y funciones. El while(1) es una forma de decir que queremos realizar la operación una y otra vez sin fin. Las llaves { y }, son como delimitadores para mantener las operaciones en su lugar y señalar cuando esta comienza y termina.

De esta forma, empleando el entorno de desarrollo suministrado por KEIL, que será el que empleemos en el laboratorio, y habiendo escrito el programa como muestra la figura. Seleccionamos del menú adecuado que se compile el fichero haciendo clic en *Project, Compile File*.



Aparece una ventana, informando que la ha comenzado la compilación, y unos segundos después, informa que la compilación ha sido un éxito (*compilation successful*). Si no, el entorno señala las líneas del programa donde se haya encontrado problemas y habrá que corregirlas y volverlo a intentar.

Ahora, hacemos clic en File, Nuevo y encontramos el fichero de listado que ha generado el compilador. Aquí están copiadas las partes esenciales.

INSTRUCCIONES RESULTANTES DE LA COMPILACIÓN

0000	E590	MOV	A, P1
0002	2403	ADD	A, #03H
0004	F5B0	MOV	P3, A
0006	80F8	SJMP	?C0001

A la derecha tenemos los equivalentes alfanuméricos de las instrucciones denominadas nemónicos de ensamblador (lenguaje ensamblador), que son más fáciles de entender para los humanos. Recordemos que lo que queríamos hacer era sumar tres a los conmutadores. El compilador hizo todo lo demás. El compilador escogió copiar primero la entrada de P1 (los conmutadores) a A (el acumulador, un almacenamiento temporal). Luego, sumo el número tres a A. Luego envió el valor de A hacia P3 (los LEDs). Finalmente, salta hacia tras para leer de nuevo la entrada en un bucle sin fin. Podríamos haber escrito uno mismo este programa de lenguaje ensamblador.

A la izquierda tenemos la información que tiene sentido para el microprocesador, el código máquina o instrucciones máquina. Los primeros cuatro dígitos son las direcciones donde se localizan las instrucciones. En la dirección 0000H tenemos E5H, el código máquina para la instrucción de copia en el acumulador desde una posición de memoria específica. En la dirección 0001H tenemos 90H, la dirección para el puerto P1. Estos dos bytes juntos constituyen una instrucción. Como realiza el hardware esta copia de datos se discute en el capítulo 7 (La UCP).

La segunda instrucción está en la dirección 0002H. El código 24H le indica al computador que sume un valor fijo al acumulador. El segundo byte de la instrucción es el valor a sumar, aquí 03H. La tercera instrucción, F5H en la dirección 0004H, copia el acumulador a la dirección que se encuentra en el siguiente byte, aquí B0H, la dirección del puerto P3.

La última instrucción es la que hace que este programa tenga un bucle sin fin. El salto corto, código 80H, emplea el segundo byte, con la dirección donde se encuentra la siguiente instrucción (0008H en este caso) para generar la siguiente dirección ($0008H + F8H = 0000H$). El salto corto solo puede llevar a una dirección no más de 128 posiciones en cualquier sentido a partir de la dirección actual. Emplea menos códigos de instrucción que un salto largo que puede avanzar o retroceder a cualquier parte en un rango de 16 bits en la familia del 8051. Comentaremos brevemente la forma en que los códigos de instrucción se componen en el ordenador y de que forma cambia el puntero de instrucción.

¿ C o Ensamblador ?

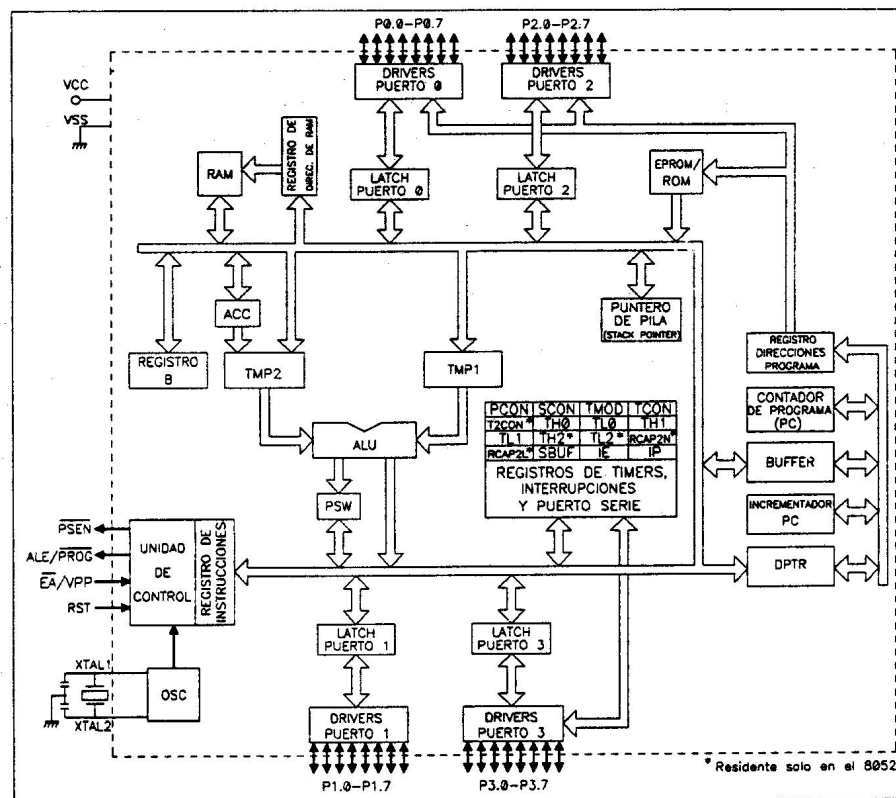
Aunque la mejor elección para programar el 8051 es en C, si queremos entender la operación que realiza internamente tendremos que usar ensamblador. Si escribimos en C, un compilador realizará el paso de lenguaje de alto nivel a código máquina. Si escribimos en lenguaje ensamblador, un ensamblador trasladará los nemónicos a los códigos numéricos equivalentes. Podríamos realizar el ensamblado a mano buscando los códigos en una tabla, pero esto no se hace en la práctica por lo tedioso que resulta.

3.- Arquitectura de la familia MCS-51.

3.1.- Introducción.

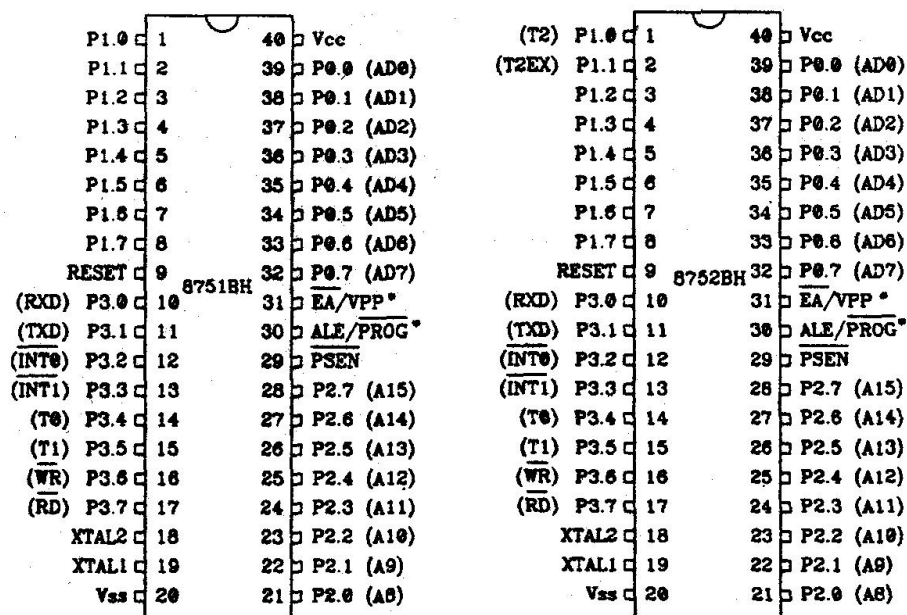
Los tres componentes básicos de esta familia son los microcontroladores 8051, 8751 y 8031, los cuales están configurados de manera muy similar, formados por los siguientes elementos:

- UCP de 8 bits
- Memoria de datos (RAM) de 128 bytes.
- Memoria de programa (ROM) de 4KB (8051).
- 4 puertos de E/S con 8 líneas cada uno.
- 1 puerto de E/S serie.
- 2 contadores-temporizadores de 16 bits programables.
- 1 oscilador para las señales de reloj.



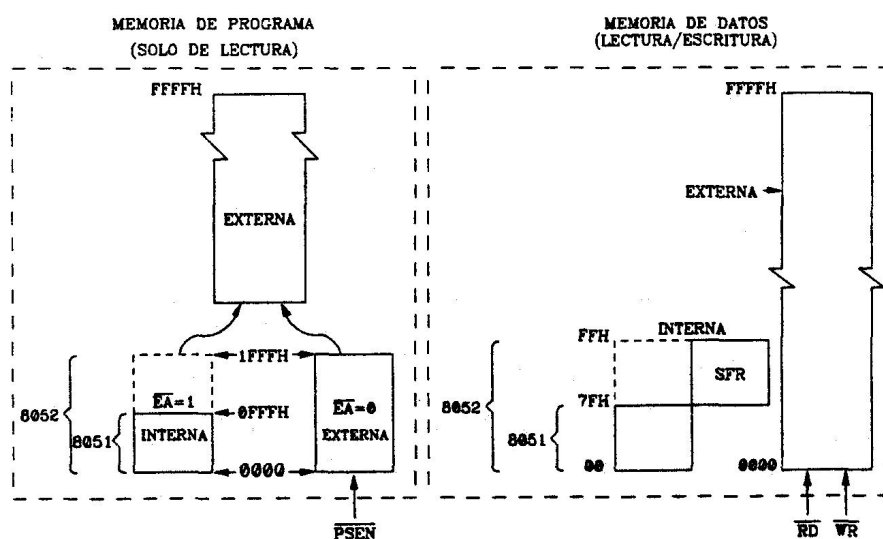
El encapsulado del 8051 posee solamente 40 patillas, debiendo soportar numerosas funciones. Este hecho obliga a que cada terminal posea dos funciones diferentes multiplexadas en el tiempo. Las funciones generales y las señales de dicho microcontrolador son:

- ✖✖32 líneas bidireccionales (4 puertos de E/S).
- ✖✖Un canal serie full-duplex.
- ✖✖Capacidad de control de 64 KB de memoria de código.
- ✖✖Capacidad de control de 64 KB de memoria de datos.
- ✖✖2 contadores.
- ✖✖2 líneas de interrupción.



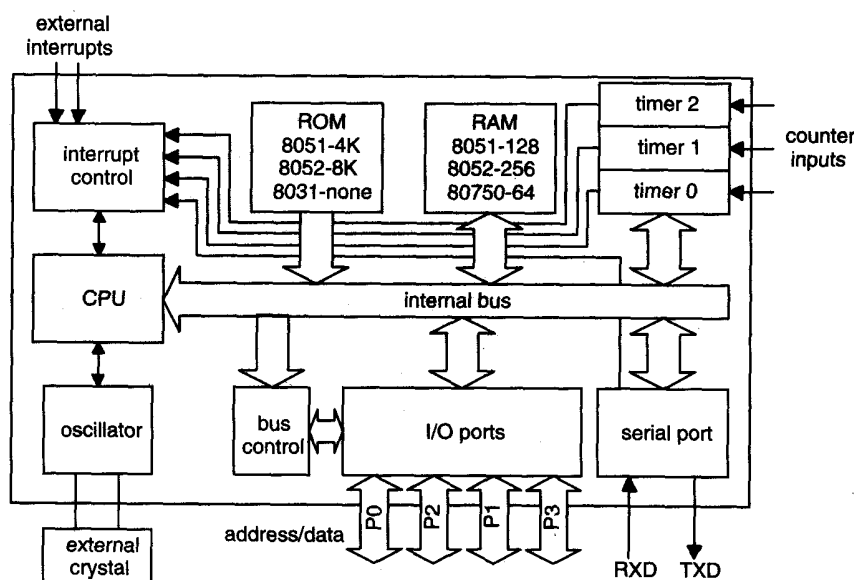
El espacio de memoria gobernado por el 8051 se divide en tres zonas:

1. Memoria interna de datos: posee 256 bytes, de los cuales 128 se dedican a registros específicos de la CPU y los restantes a datos, el 8752 posee otros 128 bytes más accesibles solo mediante direccionamiento indirecto.
2. Memoria interna + externa de código: de los 64 KB totales de esta zona, los 4 KB inferiores son los que se hallan implementados en el propio chip como ROM programable en fábrica (el 8752 posee 8 KB). Los 60 KB restantes hay que conectarlos externamente.
3. Memoria externa de datos: el 8051 puede direccionar hasta 64 KB de memoria RAM externa.



Este tipo de memoria se accede en base al tipo de direccionamiento empleado. El direccionamiento puede ser:

- Direccionamiento directo e indirecto: se puede emplear para la zona de RAM interna comprendida entre 00H y 7FH, que suele ser donde se coloca también la pila.
- Direccionamiento solo directo: se emplea con la zona de registros de función especial (SFR) situada entre la dirección 80H y FFH de la memoria interna.
- Direccionamiento solo indirecto: cuando escribimos en la zona de SFR usando este direccionamiento, en realidad escribimos en una zona de RAM interna a la que solo se accede mediante direccionamiento indirecto.
- Direccionamiento a nivel de bit: esta área tiene una longitud de 16 bytes (segmento 20H a 2FH). Cada uno de los 128 bits de este segmento se puede direccionar directamente (00H a 7FH). Los bits pueden referirse de dos forma diferentes, bien por sus direcciones (bits 00H a bits 7FH), o por los bytes que los contienen (20H a 2FH) seguidos por el número de bit: 20.0 a 20.7 serían los 8 bits de la posición 20H.



La figura anterior muestra una vista más detallada, específica a la familia 8051. Junto con la UCP, tenemos también memoria ROM y RAM, así como E/S. Se muestra un bus sencillo aunque direcciones y datos viajan internamente sobre líneas separadas. Por ahora, ignoramos las interrupciones y los temporizadores.

Volvamos a la primera instrucción del programa de ejemplo software. Esta instrucción provoca que la UCP emita la dirección del puerto (desde el que se va a recoger el dato sobre el bus de direcciones), emite la señal para que los datos del puerto se coloquen en el bus de datos, y luego activa los latches para almacenar el contenido del bus de datos en un registro interno. Estos son los pasos para ejecutar MOV A,P1.

La siguiente instrucción, ADD A,#03H, tiene lugar enteramente dentro de la UCP tras haber realizado un ciclo FETCH para leer de memoria el código de la instrucción. El proceso de

la suma hace uso de la unidad lógica aritmética del interior de la UCP. Además de la suma, existen instrucciones para la resta, multiplicación y división, así como operaciones lógicas como AND, OR, or exclusiva y complemento.

Además de leer datos de E/S, los bytes de instrucción (código) provienen de la memoria, por lo que también describiremos esta secuencia.

3.2.- Almacenamiento de código en el 8051.

El código se almacena en ROM puesto que un programa no debe modificar nunca su propio código y el código debe estar presente en cuanto pongamos el dispositivo en marcha. El almacenamiento de programa de la familia 8051 es una de las cosas que diferencia los miembros. Unos pocos miembros mantienen el almacenamiento original que era 2K bytes, mientras que otros solo disponen de 1K y otros tanto como 32K. Algunos no disponen de código en chip y deben tener almacenamiento de código externo.

3.2.1.- RAM interna en el 8051.

Dentro de la familia 8051 podemos tener desde 64 a 256 bytes de RAM interna para almacenaje de datos, esta es la memoria estática más rápida y que tiene los más variados modos de direccionamiento.

Los registros de función especial (*SFR*, *Special function register*) son direcciones de memoria interna que controlan los “periféricos” disponibles en el chip, como los puertos, temporizadores e interrupciones, así como otras características del procesador. En el ejemplo del comienzo con luces y conmutadores, ambos puertos son registros de función especial (P1 en la dirección 90H y P3 en la B0H). Otros SFRs de interés particular son el acumulador (ACC, en dirección E0H), el registro B (en 0FH) y el puntero de datos (DPH en 83H y DPL en 82H). La siguiente tabla muestra todos los SFR del 8051 estándar. Estos SFR figuran principalmente en las instrucciones.

Símbolo	Descripción	Direccionamiento Directo
P0	Puerto 0	80H
SP	Puntero de Pila	81H
DPTR	Puntero de datos (2 bytes)	
DPL	Parte Baja Puntero de Datos	82H
DPH	Parte Alta Puntero de Datos	83H
PCON	Control de Alimentación	87H
TCON	Control del Temporizador	88H
TMOD	Modo del Temporizador	89H
TL0	Temporizador bajo 0	8AH
TL1	Temporizador bajo 1	8BH
TH0	Temporizador alto 0	8CH
TH1	Temporizador alto 1	8DH
P1	Puerto 1	90H

3.2.2.- Memoria fuera del chip.

Con la excepción de unos pocos miembros, los más pequeños, toda la familia del 8051 pueden acceder a un espacio de memoria externa al chip. Para hacer esto, los 8 bits más significativos de direcciones aparecen en el Puerto 2. Las direcciones más bajas y los datos aparecen sobre el puerto 0. Para ahorrar patillas (el 8051 original solo tiene cuarenta) el puerto P0 primero se emplea como los 8 bits menos significativos del bus de direcciones y luego como bus de datos. Esto se denomina como sabemos multiplexado. Empleamos menos patillas pero se reduce la velocidad de los accesos a memoria y necesitamos un latch externo de direcciones.

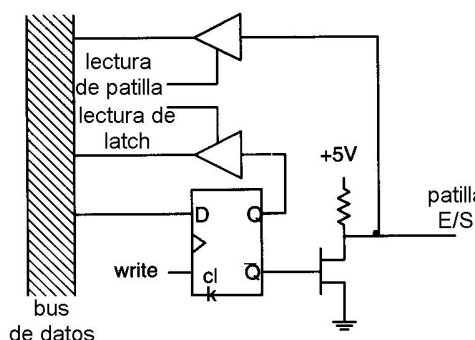
El acceso externo a memoria con el 8051 siempre es más lento tanto por el multiplexado como porque necesitan una instrucción adicional para acceder a la posición de memoria externa. Solo disponemos de tres instrucciones para realizar este acceso.

3.4.- Entrada y Salida.

Si no tenemos la manera de intercambiar información con el mundo exterior el ordenador sería inservible. Mientras que con los PC (Ordenadores Personales) hay formas estandarizadas de realizar las conexiones de E/S (COM1, LPT1, teclado, etc), los microcontroladores destacan proporcionando entradas y salidas mucho más adaptables. El término **puerto** se usa para referirse a un bloque de E/S. Hay dos tipos comunes de puertos, serie y paralelo.

Puertos Paralelo. Los puertos paralelos son grupos de (normalmente ocho) bits sobre patillas individuales. Un puerto paralelo latchea o retiene su valor hasta que volvemos a mandar otro valor. Podemos realizar puertos de salida externos con latches de 8 bits (como el 74374). Un puerto paralelo de entrada pasa los estados de las patillas al microprocesador en el instante que se ejecuta la instrucción de copia para esa dirección. Podemos realizar puertos de entrada mediante buffers triestado octales (como el 74244).

La conexión del 8051 al mundo exterior se realiza normalmente a través de puertos. Siempre tenemos una serie de patillas disponible directamente en el chip y podemos tener también puertos adicionales añadidos sobre el bus de expansión. Los puertos internos del 8051 son inusuales porque solo son parcialmente bidireccionales (el término correcto es cuasi-bidireccional). Podemos ver el hardware en el esquema para una patilla.



Patilla de puerto interno básico

Podemos colocar un cero en cualquier momento pero, para realizar una entrada, la circuitería de salida debe estar desconectada. En otro caso leeríamos nuestra propia salida a través del latch en lugar de la señal proveniente del exterior. El aspecto software clave es la necesidad de que en **cualquier puerto que usemos como entrada deben tener 1 escrito en él**. Algunas de las

patillas de los puertos pueden tener otras funciones especiales, en caso de que las usemos de esta forma, no debemos usar estas patillas como puertos de entrada o salida.

No hay forma de desactivar o fijar la dirección del puerto. Recordemos que las patillas del puerto pueden absorber una corriente máxima de 1.6 mA pero solo ofrecer decenas de microamperios. También, P0 cuando se usa como puerto de E/S (opuesto al bus de datos para expansión de memoria externa) no dispone de resistencias de pull-up, el dispositivo externo tendrá que proporcionar el nivel lógico alto.

Puertos Serie. Los puertos serie transfieren bits simples de datos uno tras otro, tomando al menos ocho transferencias para intercambiar un byte. La forma más común de puerto serie emplea solo una patilla para cada dirección de la transferencia. Normalmente el hardware interno maneja los detalles de temporización e incluye los registros de desplazamiento paralelo/serie. En conjunto, el hardware para esta función recibe el nombre de *transmisor/receptor asíncrono universal* (UART). Este elemento descarga al procesador del trabajo de gestionar la temporización y organizar los bits de forma que el procesador pueda hacer otras tareas. Es posible mandar información serie directamente sobre patillas de puertos ordinarios usando software y temporizadores, particularmente cuando la tasa de transmisión es baja, pero resulta mejor usar el hardware de la UART cuando sea posible.

3.5.- Secuencias de ejecución de instrucciones en el 8051

Antes de entrar con las instrucciones máquina en el próximo capítulo resulta necesario conocer el proceso de ejecución de una instrucción.

La lectura de un código de instrucción (fetch) requiere primero mandar la dirección de la instrucción (desde el contador de programa) habilitando un buffer triestado en la CPU para enviar el PC al bus de direcciones. En ese momento, la parte baja del bus de direcciones permanece solo hasta que la señal ALE desaparece puesto que estas mismas patillas llevan la información del byte de código, cuando la memoria lo haya enviado (recordemos que en el 8051 el bus de datos y la parte menos significativa del bus de direcciones se encuentran multiplexado). Durante el tiempo desde que la dirección está en el bus, el dispositivo de memoria externa que contiene la instrucción puede decodificar la dirección y reconocer que se le está pidiendo una respuesta.

Para un ciclo fetch, tras un retardo controlado por el reloj, el procesador coloca una señal, habilitación del almacenamiento de programa (PSEN). Esto provoca que el dispositivo de almacenamiento de código habilite sus salidas triestado y dirija el bus de datos con el contenido de la dirección particular de la instrucción que se está pidiendo. Al final de la señal PSEN, una vez que el byte de código está en el bus, la CPU almacena el código desde el bus de datos.

A continuación, la CPU decodifica la instrucción de forma que se lleve a cabo la secuencia correcta de operaciones para ejecutar la instrucción, copiar un byte de datos a una dirección específica, comparar dos valores, o incluso cambiar la dirección de la próxima búsqueda de instrucción (un salto).

Tomemos una instrucción concreta, un comando para copiar el contenido de la dirección de memoria 2045H en el registro puntero de datos (*MOV DPTR,#2045H*). Cuando el primer byte de la instrucción haya sido decodificado, implicará que son necesarios dos bytes más de instrucción, los dos bytes que van al registro. El próximo paso para ejecutar esta instrucción, es incrementar el contador de programa en uno y buscar el segundo byte de instrucción. Finalmente, se recoge el tercer byte de instrucción. Los bytes entrantes en este caso van directamente a DPTR. Una mirada a la tabla de instrucciones nos dice que este proceso emplea 24 ciclos de reloj para la instrucción MOV DPTR.

Consideremos una instrucción que transfiere datos a dispositivos externos. En este caso, las líneas de control RD o WR entran en juego. Tomemos la instrucción *copiar a dato externo* (MOVX). Necesita que el operando se encuentre ya en el acumulador (quizás con MOV A,#dato) y la dirección de memoria externa que se encuentre en el puntero de datos (DPTR). Los pasos para ejecutar la instrucción (MOVSX @DPTR,A) involucran buscar el byte de instrucción, decodificarlo, poner el valor de DPTR en el bus de direcciones, poner el valor del acumulador en el bus de datos, y luego activar la señal WR (escritura en memoria). Esta es la finalidad de la CPU. Todos lo que hacen los dispositivos de memoria es reconocer la dirección con su circuitería de decodificación y guardar los datos cuando la señal WR termine.

3.5.1.- Señales de control del bus.

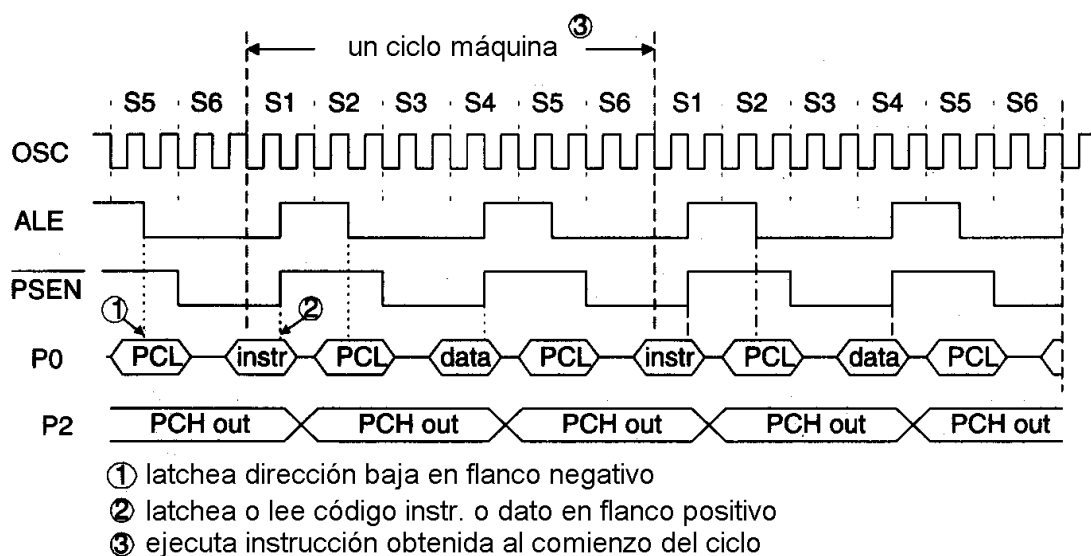
Habiendo visto el proceso de ejecución de instrucciones, ahora veremos las señales involucradas. Las señales dentro del 8051 no están descritas en el libro del fabricante y no nos interesan cuando usemos el dispositivo en la práctica. Más importantes son las señales externas para expandir la memoria fuera del chip, añadir puertos, o realizar interfaces con otros dispositivos. Todas estas señales pertenecen a la expansión fuera del chip.

ALE

Esta señal, habilitación del latch de direcciones (*ALE, address latch enable*), se usa para activar un latch externo que capture la parte baja de direcciones colocado sobre P0 antes de que el dato sea transferido desde o hacia memoria externa.

PSEN

Habilitación del almacenamiento del programa (*/PSEN, program store enable*) indica que la ROM debe colocar el código en el bus de datos. /PSEN solo se activa si se produce acceso a memoria externa. Si colocamos la patilla /EA a masa, todos los accesos se realizan a la ROM externa. Si dejamos /EA flotante alto, los accesos se realizarán en el rango dentro del chip que no provocan que la señal /PSEN pase a nivel bajo.



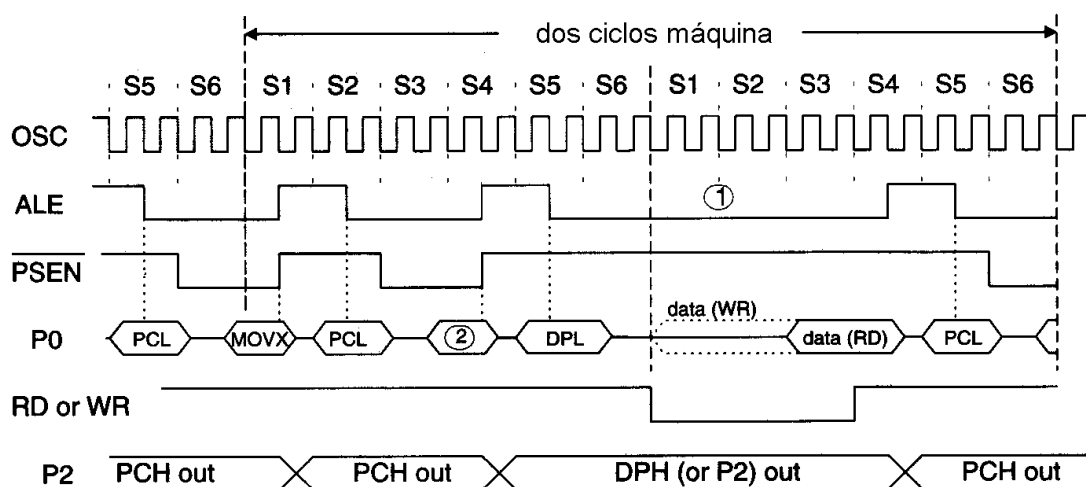
Temporización del ciclo FETCH

Las formas de onda de la figura muestran un ciclo FETCH externo. Los bordes sugieren que existen tiempos de subida y bajada asociados con las señales mientras que las señales en medio (ni 1 ni 0) para P0 muestran el bus en triestado en los intermedios entre la colocación de los datos. Si enganchamos un osciloscopio a las líneas del bus, podremos observar algo mucho más indefinido, especialmente durante los tiempos de triestado.

Hay que tener en cuenta que un microprocesador concreto tiene una velocidad máxima debido a estas transiciones no instantáneas. Mientras más pongamos el chip a su límite superior de velocidad, más redondeadas aparecerán las señales y más probable será que el microprocesador no funcione.

RD y WR

Estas señales solo se emplean para los *dispositivos de almacenamiento externos* y puertos externos. El siguiente cronograma muestra la temporización de una instrucción MOBS. Si usamos la instrucción MOBS con el acumulador como destino, generaremos una señal de *lectura* y con el acumulador como fuente, una señal de *escritura*.



- ① notar que ALE solo desaparece para la instrucción MOVX
 ② las instrucciones de un solo byte tienen una segunda fase de FETCH

Cronograma de Lectura/Escritura externa(MOVX)

Mostrar una instrucción de tres bytes nos permite observar el hecho de que todos los ciclos fetch involucran dos bytes, por lo que los ciclos fetch de uno o tres bytes incluyen un ciclo fetch oculto.

3.5.2.- El oscilador

Todos los miembros de la familia del 8051 usan un cristal externo para el oscilador. Podemos escoger, según el miembro de la familia, una frecuencia del cristal que varía desde 500 KHz a 33 o 40 MHz. Habrá que hojear el manual del fabricante de cada dispositivo en concreto.

Cuando observamos como el procesador lleva a cabo realmente las instrucciones, podemos constatar que el ciclo máquina básico no es un periodo de reloj (cada instrucción no ocurre en 83 ns si tenemos un cristal de 12 MHz). Para un 8051 *cada instrucción toma unos 12 ciclos de reloj* para completarse.

3.6.- Banco de Registros

Parte de la memoria interna del 8051 puede ser direccionada como cuatro bancos de registros, en grupos de 8 bytes. La designación *R0...R7* se refiere a esos ocho bytes. Las direcciones reales en RAM interna de estos registros pueden ser una de cuatro, dependiendo de la configuración de dos bits en la palabra de estado de programa (*PSW, program status word*). Los bancos de registros permiten moverse rápidamente de una actividad a otra.

En lenguaje ensamblador, el control del banco de registros se lleva a cabo programando 2 bits del PSW. En C, la elección del banco de registros depende de directivas del compilador específicas. En KEIL C podemos usar la directiva *registerbank* para escoger un banco para el módulo completo, o usar la directiva *using* para hacer lo mismo en una función sencilla.

3.7.- Registros de Función Especial.

Los registros de función especial (SFRs) controlan los periféricos del chip. La tabla siguiente lista todos los registros de función especial del 8051. Los detalles de cada registro se encuentran con el periférico interno relacionado.

Los puertos y otros SFRs también son direccionables a bit, por lo que por ejemplo, P3 puede direccionarse como byte B0H o puede direccionarse con instrucciones de bit desde B0H a B7H. Los registros que también se pueden direccionar a bit se muestran en cursiva en la tabla. Todas las direcciones de byte y bits de los SFR están por encima de 7FH para evitar conflictos con la RAM interna (direccionable de forma directa) y la memoria de bit, direccionable bit a bit.

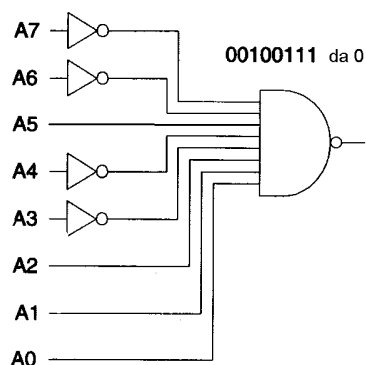
Símbolo	Descripción	Direccionamiento Directo
<i>P0</i>	Puerto 0	80H
<i>SP</i>	Puntero de Pila	81H
<i>DPTR</i>	Puntero de datos (2 bytes)	
<i>DPL</i>	Parte Baja Puntero de Datos	82H
<i>DPH</i>	Parte Alta Puntero de Datos	83H
<i>PCON</i>	Control de Alimentación	87H
<i>TCON</i>	Control del Temporizador	88H
<i>TMOD</i>	Modo del Temporizador	89H
<i>TL0</i>	Temporizador bajo 0	8AH
<i>TL1</i>	Temporizador bajo 1	8BH
<i>TH0</i>	Temporizador alto 0	8CH
<i>TH1</i>	Temporizador alto 1	8DH
<i>P1</i>	Puerto 1	90H
<i>SCON</i>	Controlador Serie	98H
<i>SBUF</i>	Buffer de datos Serie	99H
<i>P2</i>	Puerto 2	A0H
<i>IE</i>	Habilitación Interrupciones	A8H
<i>P3</i>	Puerto 3	B0H
<i>IP</i>	Prioridad de Interrupciones	B8H
<i>PSW</i>	Palabra de Estado de Programa	D0H
<i>ACC</i>	Acumulador	E0H
<i>B</i>	Registro B	F0H

4.- Decodificación de direcciones

Uno de los pasos para conseguir un código de instrucción o byte de datos de la memoria (o una lectura de un conmutador conectado a un puerto) es que el dispositivo adecuado responda a la petición. Cuando la UCP coloca una dirección en el bus de direcciones, algún hardware tiene que reconocer la dirección y devolver los datos de una posición de memoria particular a la UCP. Para esto es necesario realizar una decodificación de direcciones. La posición de memoria

debe responder solo cuando se de una combinación específica de bits en el bus de direcciones. Esto se realiza mediante circuitería digital combinacional sencilla.

4.1.- Decodificador de una sola dirección.



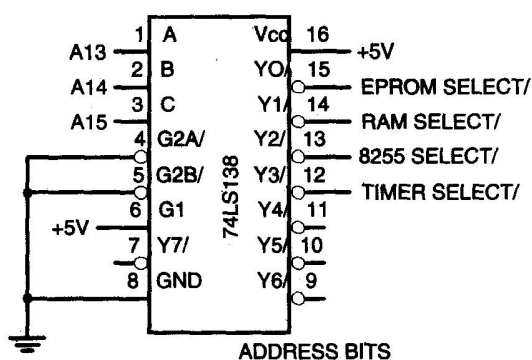
Decodificador de una sola dirección

Tomando una puerta NAND de múltiples entradas y colocando inversores delante de entradas específicas, podemos realizar un decodificador de direcciones que devuelve nivel bajo solo para una combinación de entradas. La circuitería de la figura habilita la salida (lógica 0) cuando la dirección específica 27H aparece en el bus de direcciones. Puesto que una puerta NAND da cero solo cuando todas las entradas son "1", la salida a "0" indica que la dirección particular está seleccionada. La señal a la entrada de la puerta NAND debe ser 11111111 de forma que, con los inversores, 00100111 debe tener lugar en la circuitería, o sea una dirección 27H.

Para entender un decodificador hay que ir hacia atrás a partir de la salida y realizar un análisis. Si la salida está a nivel bajo para activar el sistema, como deben estar las entradas? Además, si las entradas están de esta forma, que señal genera el decodificador de direcciones?

4.2.- Decodificador de direcciones múltiple.

Otro dispositivo común para realizar decodificador de direcciones es el 74138. Este dispositivo dispone de tres líneas de entrada, que seleccionan una de las ocho posibles salidas. La tabla de verdad y el esquema se muestran en la siguiente figura.



	A15	A8	A7	A0
EPROM	0 0 0	X X X X X	X X X X X X X X	
RAM	0 0 1	X X X X X	X X X X X X X X	
8255	0 1 0	X X X X X	X X X X X X X X	
TIMER	0 1 1	X X X X X	X X X X X X X X	

Con este dispositivo, usando las seis líneas de direcciones más significativas, podemos seleccionar entre varios dispositivos diferentes, varios chips de RAM o EPROM, latches para puertos adicionales, o varios dispositivos periféricos programables como contadores o displays LCD.

En el esquema, el 74138 se ha conectado para seleccionar una EPROM en 0000H, un chip de RAM en 2000H, un chip de puerto paralelo (8255) en 4000H, y un chip temporizador (8253) en 6000H. Como podemos ver de las indicaciones binarias de bit bajo el esquema, el circuito decodifica

solo las tres líneas superiores de direcciones. Para dispositivos de 8K de RAM o EPROM, todo el resto de líneas se dirigen al dispositivo en sí. Para el 8255, sucede que solo tiene dos líneas usadas en el chip (A0 y A1) por lo que el resto son bits *no importa*.

Si pensamos en la dirección binaria del chip, son 010xxxxxxxxxxx00 hasta 010xxxxxxxxxxx11. Esto significa que las cuatro direcciones pueden ser de 4000H a 4003H, o puede ser 4004H a 4007H, o cualquier grupo similar de cuatro direcciones hasta 5FFCH a 5FFFH. Cuando algunas de las líneas de dirección no están involucradas en la selección de un dispositivo, el dispositivo *no es decodificado completamente*, y las direcciones repetidas son *direcciones solapadas*.

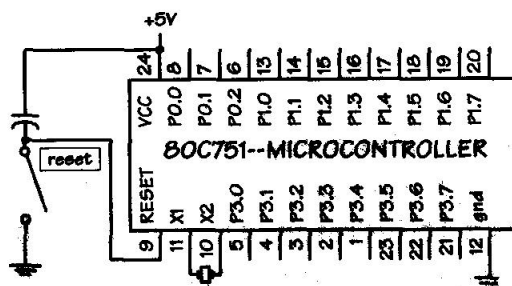
Tabla de verdad del 74138

Entradas												
habilitación		selección			salidas							
G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

5.- Montaje de Microcontroladores.

5.1.- Independiente

El 8051 fue probablemente el primer ordenador en un chip del mercado y todas las versiones con código en el chip (ROM, EPROM o EEPROM) pueden funcionar con un chip simple. Es bastante común tener el procesador entero como un chip y un cristal de cuarzo, como con el AT89C2052, un derivado del 8051 pero con 2K bytes de memoria FLASH y por tanto reprogramable.



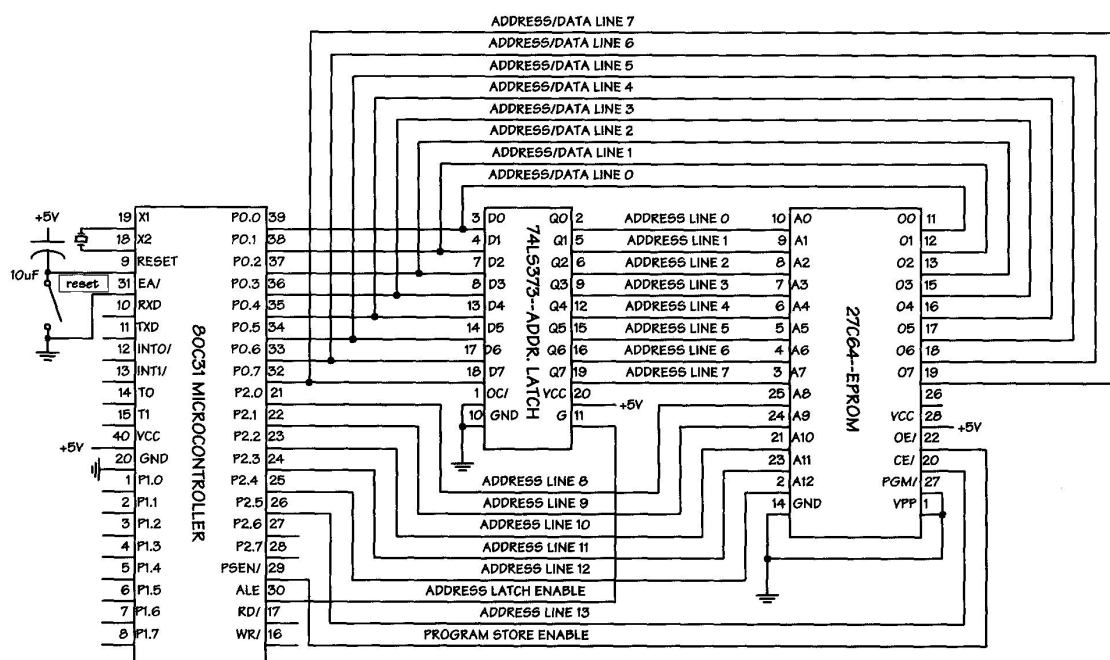
Este diseño *puede reemplazar varios chips de lógica, varias PLDs y a veces varios dispositivos analógicos*. Por ejemplo, en lugar de usar un integrador con una gran constante de tiempo para tratar una señal analógica, podemos sustituirlo por un conversor voltaje a frecuencia que alimente el contador de un micro de un solo chip. También se pueden más prestaciones extra sin consto, modificando tan solo el programa. Puede auto calibrarse, hacerse adaptativo, o tener cualquier número de otras posibilidades para hacer el sistema más eficiente, más fácil de calibrar y en definitiva más fácil de usar.

5.2.- Expansión de memoria para el 8051.

Probablemente la mayoría de las aplicaciones del 8051 hoy día usan chips adicionales para guardar el código y a veces también los datos. La expansión es directa, *siempre se hace de la misma manera*. El esquema siguiente muestra una expansión de chip simple del espacio de código que es obligada para cualquier aplicación que use el 8031, que no tiene ROM o EPROM en el chip. El mayor problema de la expansión de memoria es la *pérdida de al menos dos puertos* de 8 bits en funciones de bus externo.

5.2.1.- Código externo.

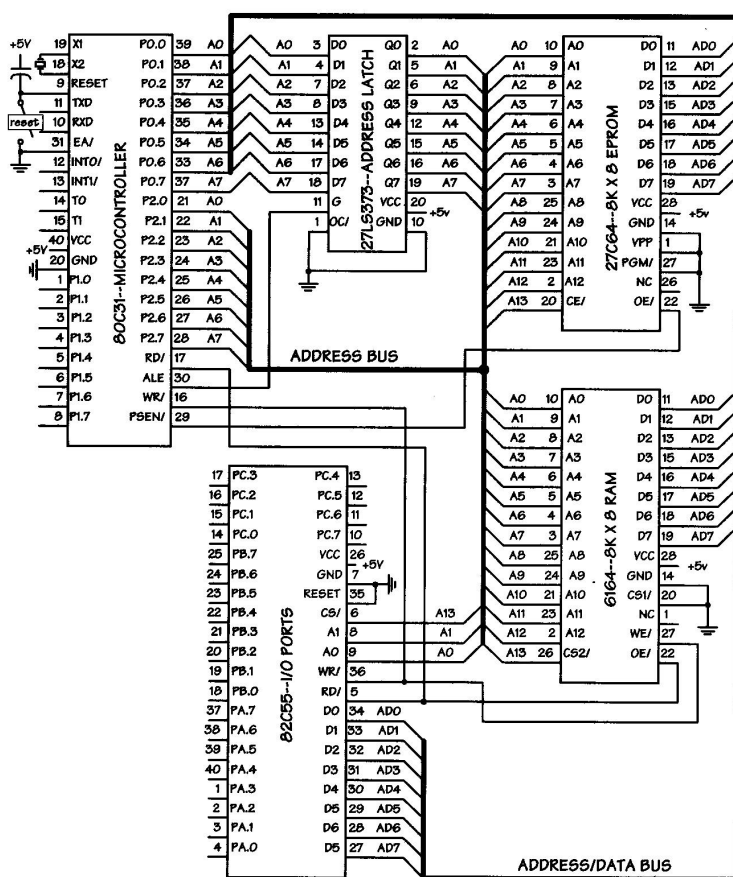
Al expandir externamente el microcontrolador, estamos limitados por la información multiplexada direcciones/datos en los ocho bits bajos (P0). Para mantener el byte bajo de direcciones hasta que la CPU haya transferido los datos, debemos almacenarlo en el flanco de baja del pulso ALE. El 74LS373 es la elección común. El *byte alto de direcciones* permanece sin cambios en P2 durante el ciclo completo y podemos usarlo directamente para decodificación de direcciones.



La *señal /PSEN* proporciona la señal de lectura o habilitación de salida (OE) a la EPROM cuando la CPU dirige los datos en el bus. En el primer ejemplo, la decodificación de direcciones es innecesaria. Con solo una EPROM, la CPU nunca manda la señal /PSEN excepto para acceder al código en este chip. El uso de A13 unido con la señal /CS es un tipo de decodificación puesto que el rango de direcciones donde A13 se encuentra a nivel lógico alto no direcciona la EPROM (o cualquier otra cosa). Así, el rango de direcciones de código va desde 0000H a 1FFFFH.

5.2.2.- Código Externo, Memoria RAM Externa y Puertos Externos

Una vez que hemos perdido los dos puertos, resulta sencillo añadir una RAM externa, dando dos bits para RD y WR. El esquema siguiente muestra este tipo de sistema con RAM extra y un *dispositivo controlador de puertos 8255*.



Para *ahorrarnos un decodificador de direcciones*, la RAM se ha mapeado en la dirección base (0000H) y el 8255 se ha colocado en lo alto del espacio de direcciones. Observando el 6164, /CS1 se ha colocado a masa, por lo que se encuentra siempre habilitada. CS2 se ha conectado a A13. Puesto que esto es una entrada activa a nivel alto, A13 debe estar a nivel alto para seleccionar este chip. En binario, esto significa que el sistema tiene RAM desde 00100000 00000000 hasta 00111111 11111111 (2000H-3FFFFH). La patilla de selección del 8255 es activa a nivel bajo (/CS), por lo que A13 debe estar bajo para seleccionar el chip. Para cualquier

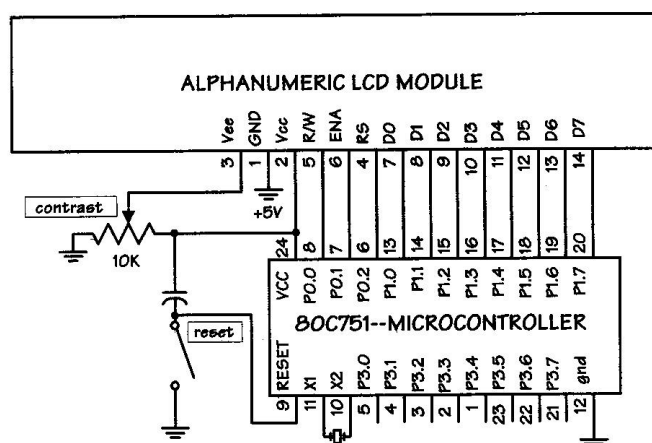
dirección donde algunos bits de direcciones no están conectados (dan igual) los otros bits se consideran a nivel bajo normalmente, por lo que el 8255, usando solo los bits A0 y A1 internamente, tendrán direcciones desde (2000H a 3FFFH).

La **expansión de la RAM** emplea 2 bits de P3 para las señales /RD y /WR. Los controles para la RAM externa (/RD y /WR) son diferentes de los controles para la EPROM (PSEN) por lo que el código y los datos externos se pueden encontrar en las mismas direcciones numéricas.

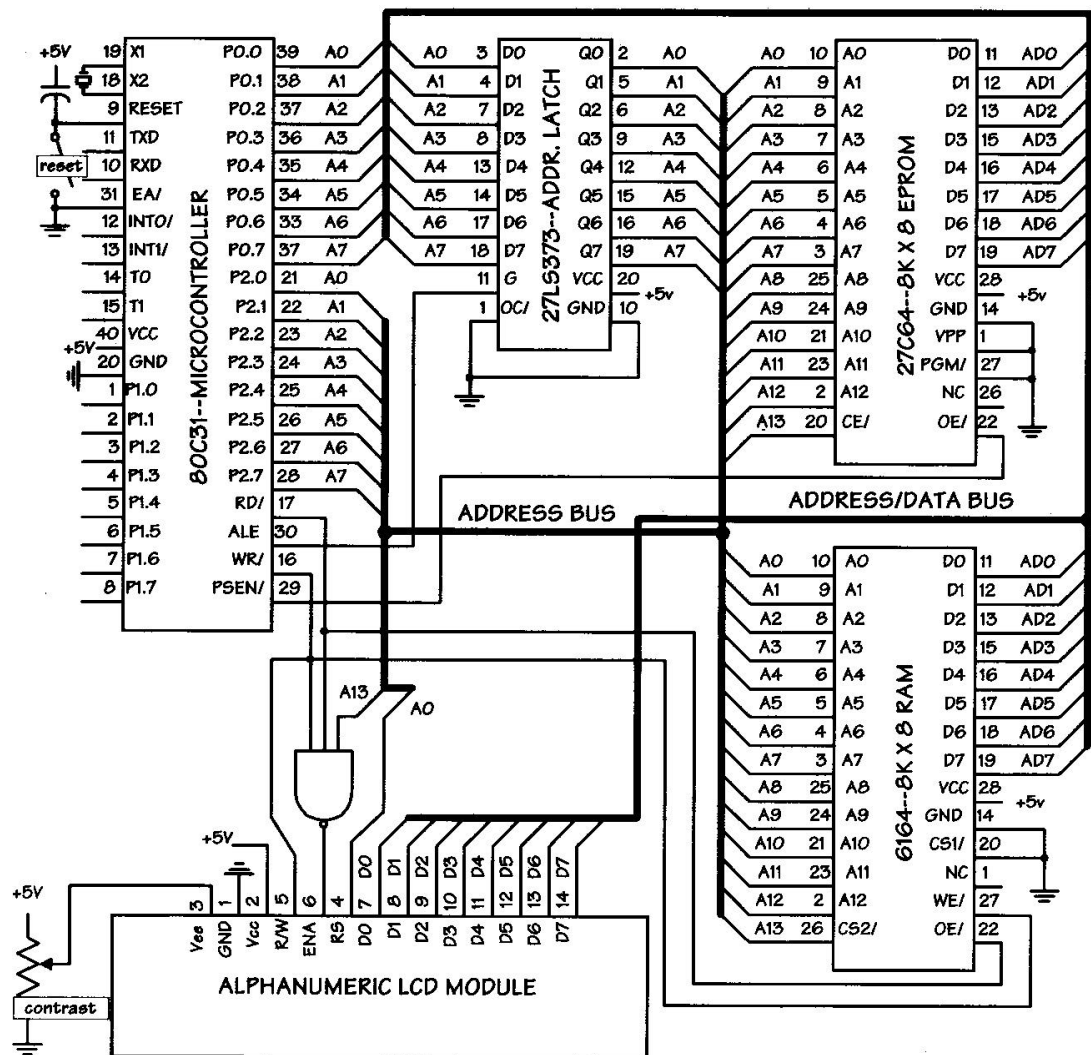
5.2.4.- Control de un display LCD.

Un **display LCD inteligente** se maneja como si se tratase de una zona de memoria accesible por el microcontrolador, de forma que cualquier escritura de caracteres ASCII en este espacio de memoria provoca una actualización del carácter correspondiente sobre las líneas del visualizador.

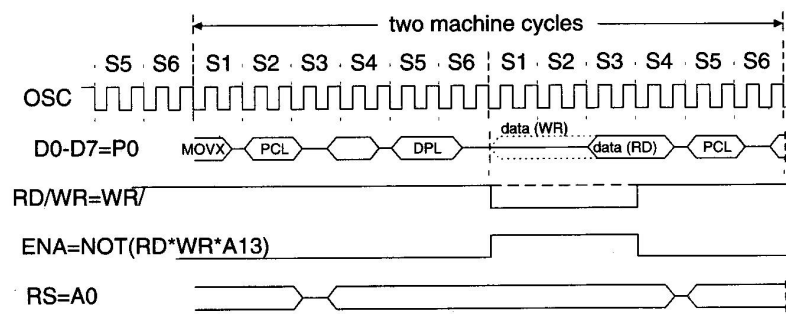
La conexión más simple consiste en su **conexión directa** con las patillas del microcontrolador, éste accede al dispositivo escribiendo y leyendo como si se tratase de un chip de memoria.



También se puede conectar el display a un sistema completo y utilizar las señales /RD, /WR y A13 para efectuar su decodificación en la parte más alta de direcciones como si se tratase de una zona más de RAM. El display dispone de **dos registros de escritura/lectura**, en uno se escribe la **posición del cursor** donde se va a escribir el carácter y en el otro registro se escribe el **carácter** en cuestión.



En la siguiente figura podemos observar el ***cronograma necesario*** que tiene que realizar el microcontrolador para realizar la escritura de una determinada posición de la memoria del visualizador.



Mediante un decodificador de direcciones realizado mediante un 74LS138 podemos direccionar los dispositivos de un sistema completo con RAM y ROM externas, además de un **conversor A/D de 10 bits** realizado mediante un AD573. Como el conversor dispone de una interfaz estándar de 8 bits es necesario transferir la información en dos partes, byte bajo y byte alto de la conversión. Esto se realiza mediante las patillas /LBE (Low Byte Enable) y /HBE (High Byte Enable). La **conversión se inicia con la patilla /CNV** al escribir en un puerto determinado, realizando la lectura de este mismo puerto se leen los dos bits más significativos, al activar la señal /HBE y después se lee de otra dirección el byte menos significativo activando la señal /LBE.

