

10 Introducción a los Microprocesadores

Objetivo del capítulo. Se pretende dar una descripción general del funcionamiento interno de un microprocesador, su arquitectura y los elementos externos indispensables para su utilización en diversas aplicaciones que sustituyen a la electrónica digital basada dispositivos MSI y SSI.

Bibliografía.

1. Introducción a las microcomputadoras, Vol I.
Adam Osborne, Mc Graw Hill.

- 1.1. Introducción. Breve historia de los microprocesadores, microcomputadoras, microcontroladores, entrenadores y sistemas mínimos, controladores lógicos programables (PLC), lógica alambrada y lógica programada.
- 1.2. Un sistema de microcomputadora típico.
- 1.3. Arquitectura de una CPU.
- 1.4. Ejecución de instrucciones.
- 1.5. Comunicación de la CPU con la lógica externa.
- 1.6. Decodificación de memoria RAM y ROM.
- 1.7. Decodificación de puertos de entrada/salida. Mapeo a memoria y mapeo a puerto.
 - 1.8.1. Entrada/salida programada.
 - 1.8.2. Entrada/salida por interrupciones.
 - 1.8.3. Entrada/salida por D.M.A.
- 1.9. Criterios para seleccionar un microprocesador
- 1.10. Ejemplos de microprocesadores de 8 bits
- 1.12. Ejemplos de memorias ROM, EPROM y RAM estáticas comerciales
- 1.13. Ejemplos de puertos
- 1.14. Codificación de teclados y displays

1.1 Introducción.

El objetivo de estas notas es presentar un curso introductorio a los sistemas digitales basados en microprocesador, cubriéndose los aspectos básicos de hardware, software involucrados en el diseño de sistemas con microprocesador enfocados a instrumentación y control de procesos, se dedica una sección especial al uso de la computadora personal en algunas aplicaciones como un sistema más accesible y fácil de programar.

La aparición del microprocesador (1971) ha marcado una revolución en el

¡Error

Introducción a los microprocesadores

campo del diseño de controladores industriales y de sistemas lógicos en general, teniendo impacto principalmente en sistemas complejos en lo referente a costo, flexibilidad y minimización de espacio físico ocupado.

Los problemas de instrumentación y control lógico relativamente complejos deben tomar en cuenta dos alternativas de solución:

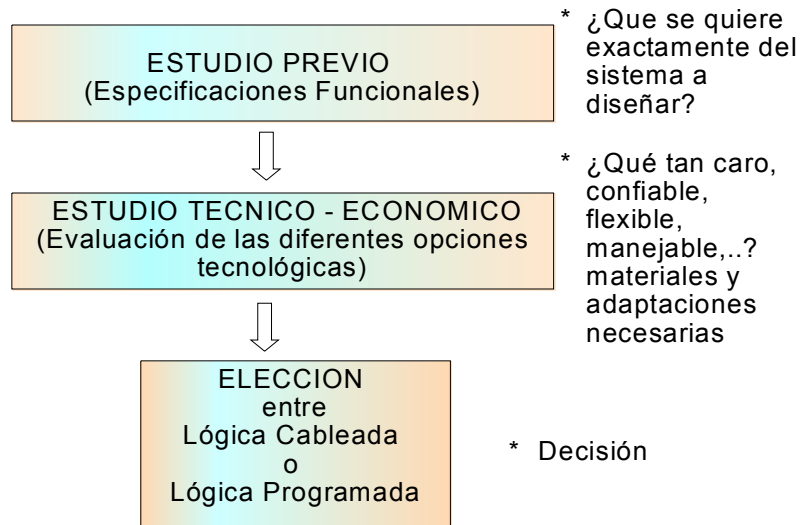
- Lógica Cableada.
- Lógica Programada.

Estas soluciones no son necesariamente electrónicas; en la siguiente tabla se muestran algunas *Opciones Tecnológicas* que pueden ser solución de problemas digitales

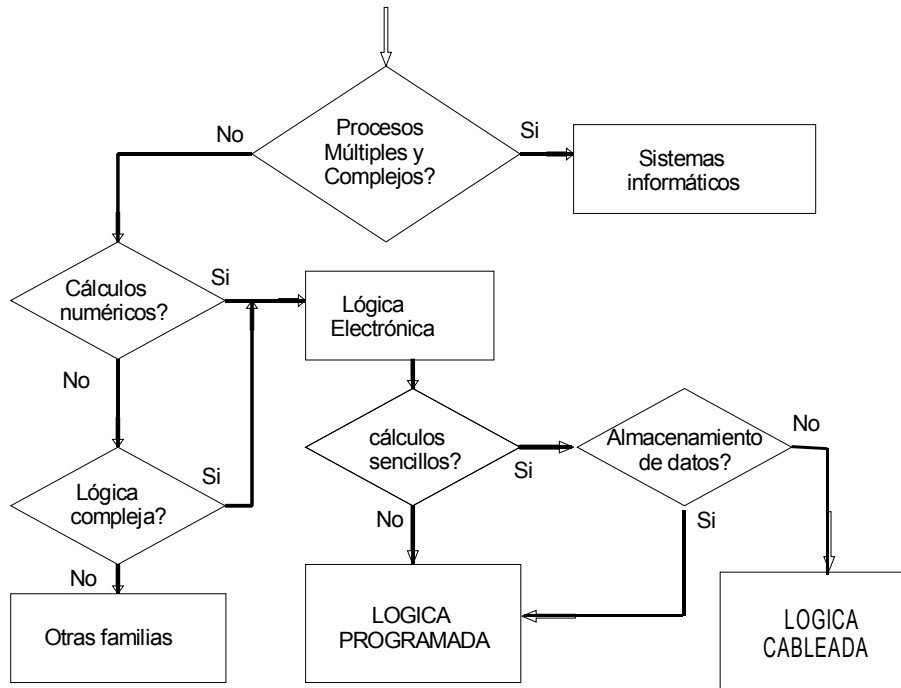
TIPO DE LÓGICA	FAMILIA TECNOLÓGICA	SUBFAMILIA TECNOLÓGICA
LÓGICA CABLEADA	Eléctrica	Relés electromagnéticos Electroneumática Electrohidráulica
LÓGICA CABLEADA	Electrónica	Electrónica discreta
LÓGICA PROGRAMADA	Electrónica	<u>Sistemas informáticos</u> <u>Sistemas Mínimos</u> Controladores Lógicos Programables

Tabla 1 Opciones tecnológicas para sistemas digitales.

Este curso se avoca solamente a los aspectos subrallados en la tabla anterior . En la figura 1 se muestran los lineamientos generales recomendados a la hora de buscar una solución adecuada a un problema digital



Como una guía en la elección entre lógica cableada o programada en la figura 2 se muestran algunas consideraciones generales a tomar en cuenta

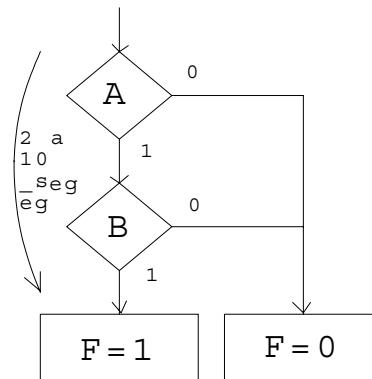
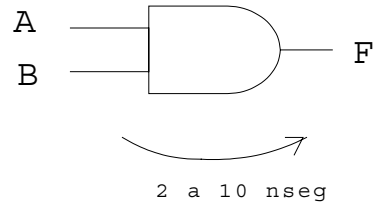


Otras consideraciones que puede ser útil tener presente a la hora de decidir, son mostradas en la tabla 2

TIPO DE LOGICA	DESVENTAJAS	VENTAJAS
CABLEADA	Solución rígida. Mayor trabajo de alambrado.	Puede resultar económica. Muy veloz.
PROGRAMADA	Necesidad de programación. Costo inicial. Relativa lentitud.	Solución flexible. Poco alambrado.

Tabla 2 Tabla comparativa entre lógica cableada y programada

En la figura 3 se ilustra la diferencia en velocidad de los dos tipos de lógica



Lógica Alambrada | Lógica Programada

Las opciones para la lógica programada como se puede ver en la tabla 1, son:

- Sistemas mínimos (basados en microprocesador o microcontrolador)
- Automátas o Controladores Lógico Programables (P.L.C.)
- Microcomputadoras
- Minicomputadoras

1.2. Un sistema de Microcomputadora Típico

Para establecer la diferencia entre una microcomputadora y algunos términos relacionados, a continuación se presentan las acepciones que se les dará a lo largo del curso.

Microprocesador (μp).- Es un controlador complejo síncrono y programable en un solo circuito integrado.

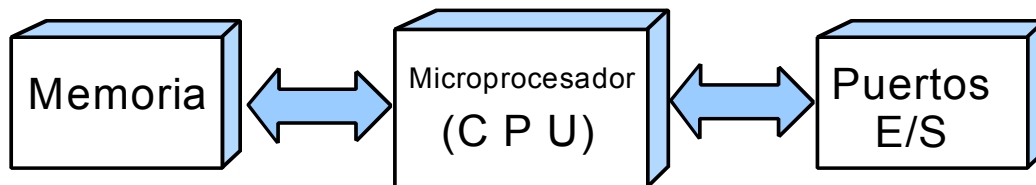
Microcomputadora (μc).- Es una computadora cuya unidad central de proceso (CPU) es un microprocesador. En la figura 4 se muestra la estructura básica de una microcomputadora que consiste en la CPU, memoria y puertos de entrada y salida.

A los puertos de entrada y salida se conectan los dispositivos conocidos como *periféricos*, estos últimos son los que entran en contacto con el usuario humano, o con

Introducción a los microprocesadores

los procesos físicos con los que la computadora interactúa. Los dispositivos periféricos son de naturaleza muy diversa, pueden ser tan sencillos como un joystick o un teclado o tan complejos como una impresora láser o un procesador de audio y de video, o un controlador de disco duro, o un CD-ROM.

Microcontrolador.- Se le llama microcontrolador a una microcomputadora construida en un sólo circuito integrado, es decir, cuando en un sólo circuito están implementados el microprocesador, algunos puertos y memoria.



Sistema Mínimo.- Un sistema mínimo es una microcomputadora diseñada con el mínimo de componentes para resolver un problema específico, ya sea de instrumentación, control, entrenamiento, etc.

Entrenador.- Un entrenador es un sistema mínimo diseñado por el fabricante del μ p para que el usuario se familiarice con el uso y operación de dicho μ p, normalmente es un sistema que cuenta con un teclado y un display sencillos, software en ROM, poca memoria y un medio de almacenamiento magnético, o bien conexión serie a un computadora personal.

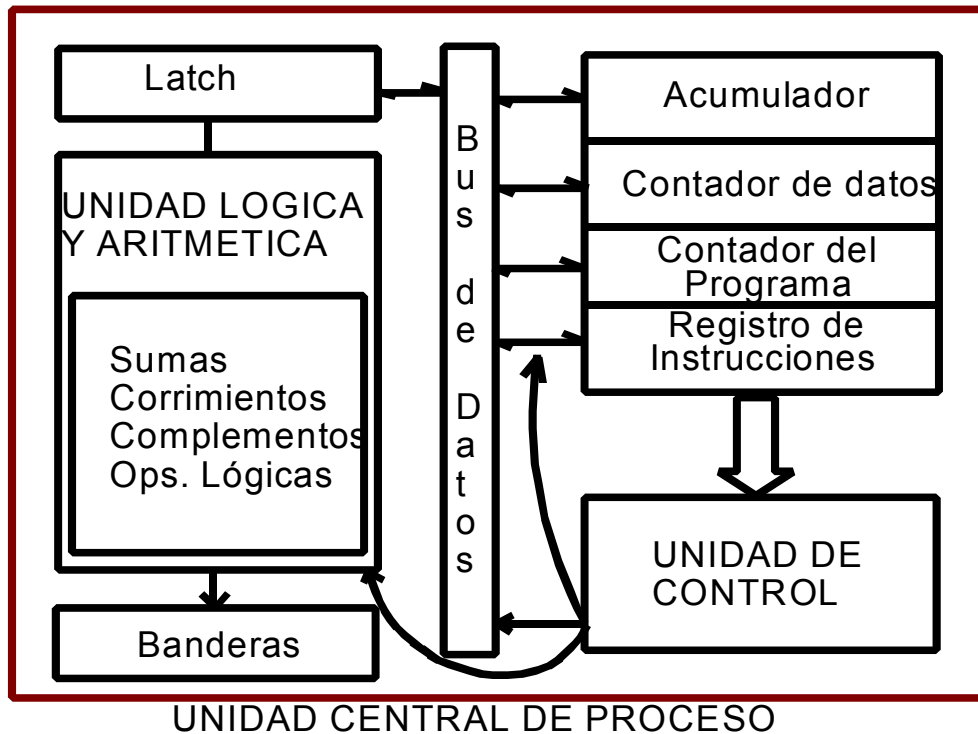
Algunos criterios para elección de un microprocesador.- A continuación se enlistan algunas características que pueden servir como punto de comparación a la hora de elegir entre un procesador u otro de los existentes comercialmente:

- Longitud de palabra (Número de líneas del bus de datos, además es importante el número de bits implicados en las operaciones aritméticas y por lo tanto la velocidad con que se ejecutan éstas)
- Número y longitud de los registros internos.
- Capacidad de conexión a puertos
- Manejo de interrupciones
- Capacidad de acceso directo a memoria (DMA)
- Velocidad (frecuencia del reloj del sistema)
- Capacidad de conexión de memoria

Además hay que considerar el costo del sistema total, incluyendo μ p, puertos, memoria, etc. el costo y disponibilidad de software de apoyo (ensambladores, depuradores, lenguajes de alto nivel), así como la existencia de entrenadores.

1.3. Arquitectura de una CPU.

Las funciones fundamentales de un μp son: Interpretación de códigos y ejecución de instrucciones, para poder realizar estas funciones requiere por lo menos los componentes que se muestran en la figura 5. A continuación se describe el funcionamiento de estos componentes.



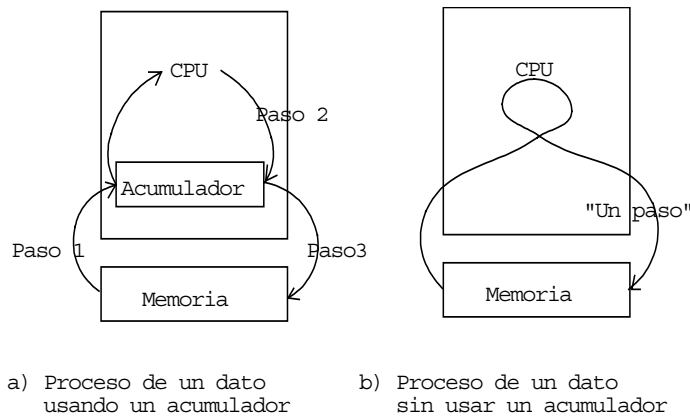
Registros de la CPU.

Los registros son memoria RAM interna del μp que es usada para fines específicos.

Registro Acumulador (A). La CPU debe poseer por lo menos un acumulador. El acumulador se encarga de guardar datos provisionalmente traídos de memoria, además guarda el resultado de las operaciones realizadas.

El tamaño del acumulador también es un indicador de la velocidad con que el μp puede realizar operaciones aritméticas y lógicas, debido a que la CPU usa en estas operaciones el número de bits del acumulador. Al número de bits del acumulador se le llama *longitud de palabra* del μp y normalmente coincide con el número de bits del bus de datos.

¿Porqué usar un acumulador, en vez de operar directamente con la memoria? En la figura 6 (b) se muestra que al operar con un dato de memoria directamente (sin un acumulador) se realiza aparentemente un solo paso, a diferencia del inciso (a) de la misma figura, en donde se requieren tres pasos, sin embargo, al usar un acumulador no se está obligado a regresar el resultado a la memoria, pudiendo solo realizar alguno de los tres pasos en un momento dado, en cambio en el esquema sin acumulador el resultado siempre se tendrá que regresar a la memoria, lo cual en realidad son dos pasos obligatorios.



Apuntador de Datos (DP). Para que la CPU pueda acceder a un dato de la memoria, debe poseer algún medio para guardar su "dirección", es decir, para indicar la localidad específica de memoria en donde está el dato. El registro contador o apuntador de datos (no es único) es el que se encarga de guardar estas direcciones.

La "dirección" es una combinación de 1's y 0's, por ésto, un contador de datos de n bits podrá direccionar hasta 2^n direcciones diferentes. Casi todos los μp de 8 bits usan un contador de datos de 16 bits, así que con el pueden direccionar $2^{16}=65536$ (64k) localidades de memoria

En la memoria se puede tener dos tipos de información: Datos y Códigos de instrucción (programa). Así, la CPU debe tener forma de manejar la memoria como Memoria de Datos y Memoria de Programa. Tanto el acumulador como el contador de datos están destinados a manejar preferentemente memoria de datos. La CPU posee entonces otros dos registros para manejar exclusivamente memoria de programa.

Registro de Instrucción (IR). En este registro se almacena el código de instrucción traído de memoria. (cualquier dato almacenado en el IR será interpretado como una instrucción por el μp).

Registro apuntador de instrucciones (IP). En este registro se almacena la dirección del siguiente código de instrucción que será cargado en el IR.

De esta manera, DP apunta siempre a memoria de datos, mientras que IP apunta siempre a memoria de programa.

1.4. Ejecución de instrucciones

Para aclarar el uso de los registros dentro del μ p a continuación se describe paso a paso la ejecución de un programa para leer un dato de memoria (de datos).

Dirección de Memoria	Contenido	Significado
400	9C	Apunta a la dirección que está guardada a continuación
401	0A	Parte LS de la dirección
402	30	Parte MS de la dirección
403	40	Trae el dato apuntado al acumulador
...
300A	7A	dato
300B	0F	dato
...

Tabla 3 Memoria de datos y programa a ejecutar

La siguiente es la condición inicial de los registros del μ p antes de la ejecución del programa:

A	X X		
DP	X X	X X	
IR	X X		
IP	0 4	0 0	

Donde: X = valor desconocido

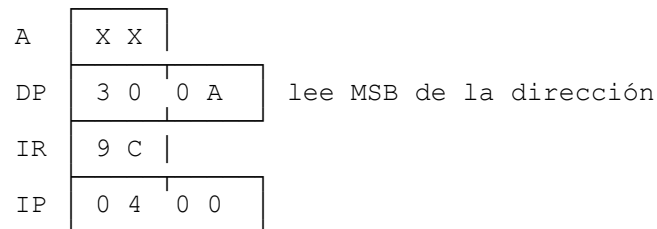
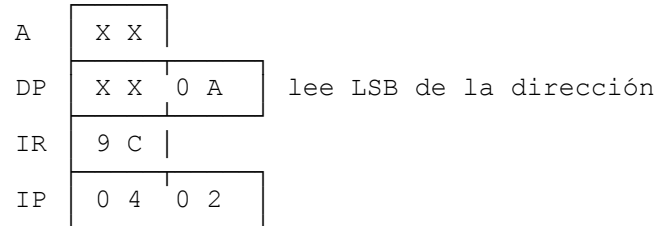
Al poner un 0400 en el registro IP la CPU busca la siguiente instrucción en esa localidad, esto inicia la ejecución de nuestro programa.

1er paso: La CPU lee la instrucción guardada en 0400 e incrementa el IP

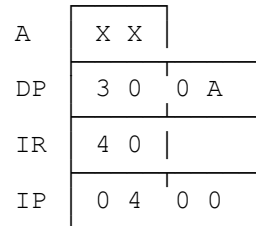
A	X X		
DP	X X	X X	
IR	9 C		
IP	0 4	0 1	

Introducción a los microprocesadores

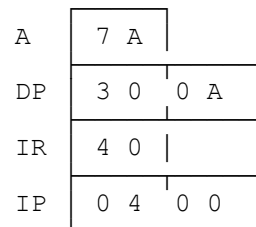
2o. paso: El código 9C es interpretado por la CPU y ésta inicia su ejecución (aquí la CPU sabe que hay un dato de 16 bits a continuación y que será interpretado como una dirección de memoria de datos) esto se hace en dos pasos: primero se carga el LSB y luego el MSB de la dirección



3er paso: Aquí la CPU sabe que terminó de ejecutar una instrucción y que el siguiente byte apuntado por IP es una nueva instrucción a ejecutar, la lee e incrementa el IP



4o paso: El código en IR es interpretado por la CPU y se inicia su ejecución (aquí la CPU sabe que deberá leer memoria de datos usando la dirección en DP)



Obsérvese que cada instrucción requiere tres etapas básicas:

```
|<----- CICLO DE INSTRUCCION ----->|
|Traída de la instruc. | Decodificación de instruc. |Ejecución|
|<----- CICLO "FETCH" ----->|
```

Unidad Lógica y Aritmética (ALU)

Introducción a los microprocesadores

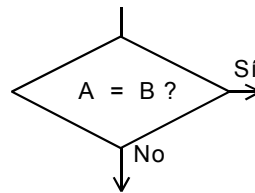
Esta es la unidad encargada de realizar las operaciones aritméticas y lógicas sobre los datos manejados por la CPU, estas operaciones son básicamente:

- Suma binaria
- Complementos a uno y a dos
- Operaciones booleanas (AND, OR, EXOR)
- Corrimientos y rotaciones

Estas operaciones pueden ser realizadas con 8 o con 16 bits en la mayoría de los μ p. Algunos μ p pueden realizar división y multiplicación de datos enteros (8 y 16 bits).

Registro de Estado del Programa (PSR) o de Banderas.

En este registro se almacena información de un bit acerca del resultado de las operaciones realizadas en la ALU. Esta información es consultada por la CPU cuando en el programa hay un *salto condicional*, por ejemplo, para realizar la siguiente instrucción



En el programa se haría la resta $A - B$ y después se consultaría la bandera de cero para decidir si **A** es igual a **B** o no lo es.

Bandera de cero (Z). Se activa si el resultado de la última operación fue cero.

Bandera de acarreo (C). Se activa si se generó un acarreo en la última operación.

Bandera de signo (S). Refleja el contenido del MSB del resultado (útil en aritmética de complemento a 2, de lo contrario puede ignorarse).

Bandera de sobreflujo (O). Indica que el resultado no cabe en el número de bits destinados a guardarlo. Este indicador es útil cuando se trabaja con aritmética de complemento a 2 (si se está usando aritmética positiva, este bit deberá ignorarse, así como SF) e indica que el resultado de la operación no deberá tomarse como correcto, pues se ha desbordado el espacio para almacenarlo.

Bandera de paridad (P). Se activa para indicar la paridad del resultado (útil en detección de errores en transmisión de datos)

Existen otras banderas que tienen relación más con el estado del sistema de la CPU que con las operaciones de la ALU y que serán explicados posteriormente, tales pueden ser: la bandera de interrupciones habilitadas, bandera de dirección de operaciones con cadenas, etc.

Unidad de Control.

Esta unidad se encarga de decodificar el significado de las instrucciones del programa, y de acuerdo a ello secuenciar los elementos lógicos de la ALU para realizar la operación requerida por el programa. Esta función la realiza consultando una unidad interna donde el diseñador de la CPU ha colocado las instrucciones elementales denominadas *microcódigo* para decodificar las instrucciones de lenguaje de máquina. Esta unidad es controlada por el contenido del registro IR.

1.5. Comunicación de la CPU con la lógica externa

Líneas Básicas de un Microprocesador.

Para que el μP pueda comunicarse con dispositivos externos, tales como memorias y puertos, éste utiliza líneas de comunicación. Estas líneas de acuerdo a su función pueden ser clasificadas en:

- El Bus de Datos
- El Bus de Direcciones
- El bus de Control

Además el μP debe poseer líneas de polarización (voltajes y tierra) y líneas de sincronización (reloj). En la siguiente figura se muestra un diagrama de patitas típico de un μP de 8 bits.

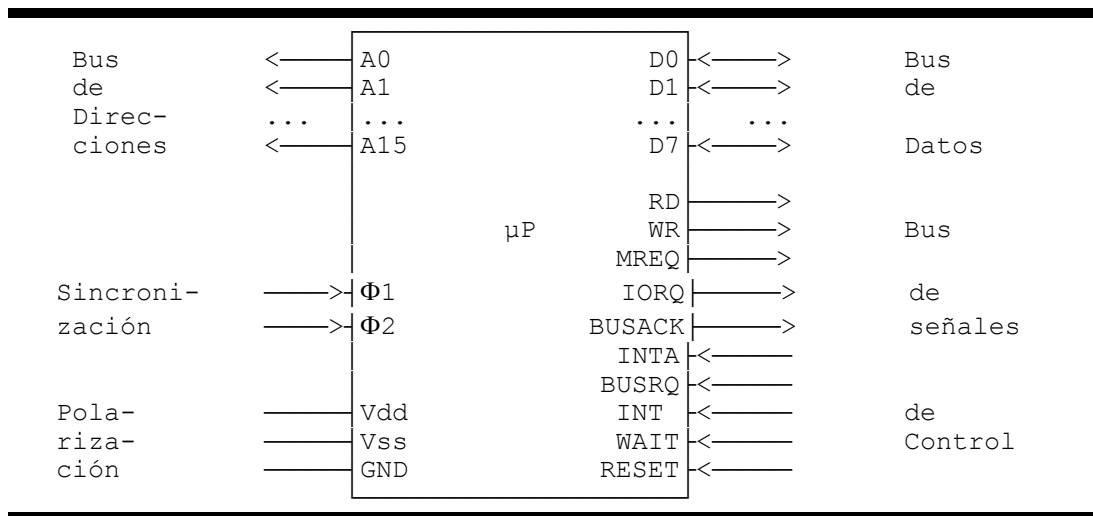


Fig. 8 Diagrama de patitas de un μP típico de 8 bits

A continuación se describe en términos generales el funcionamiento de las señales principales del μP .

Bus de datos D_0, \dots, D_7 . Este conjunto de líneas bidireccionales es el que se encarga de transmitir la información (de 8 bits) hacia o desde los dispositivos externos al μP tales como memorias o puertos.

El tamaño del bus de datos (longitud de palabra) es el que define la capacidad del μP , así si este bus es de 8 líneas se dice que el μP es de 8 bits.

Bus de Direcciones A_0, \dots, A_{15} . En este conjunto de líneas de salida el μP establece la dirección (de 16 bits) de la localidad de memoria o puerto de E/S que esta accesorando. El número de líneas de este bus determina la cantidad de localidades de memoria que se pueden accesar, así para 16 líneas, se pueden accesar $2^{16} = 65,536$ localidades, es decir desde 0000h hasta FFFFh.

La mayoría de los μP de 8 bits usan sólo la mitad el bus de direcciones para accesar puertos, es decir solamente 8 líneas A_0, \dots, A_7 , con lo cual se tienen disponibles solamente $2^8 = 256$ puertos de entrada o salida, es decir, desde 00h hasta FFh.

Señal de lectura READ (RD). Esta línea de salida se activa cada vez que el μp está realizando una lectura de memoria o puerto. Es decir, cuando RD está activa el bus de datos está funcionando en dirección hacia el μp y por lo tanto está esperando que algún dispositivo coloque un dato en él.

Señal de escritura WRITE (WR). Esta línea de salida se activa cada vez que el μp está realizando una operación de escritura a puerto o memoria. Es decir, cuando WR está activa el bus de datos está funcionando en dirección hacia los dispositivos externos y por lo tanto contiene un dato válido que puede ser tomado por algún dispositivo.

Señal de acceso a memoria MEMORY REQUEST (MREQ). Esta línea de salida se activa cada vez que el μp está realizando una operación con memoria (lectura o escritura). Es decir, indica que el bus de direcciones tiene una dirección válida de memoria.

Señal de acceso a puerto INPUT OUTPUT REQUEST (IORQ). Esta línea de salida se activa cada vez que el μp está realizando una operación con puertos (lectura o escritura). Es decir, indica que la mitad menos significativa del bus de direcciones tiene una dirección válida de puerto.

Señal de solicitud de los buses BUS REQUEST (BUSRQ). Esta línea de entrada es activada por algún dispositivo externo como una solicitud al μp para que se "desconecte" (puesto en tercer estado) de los buses de direcciones y de μp . Una vez que el μp se ha desconectado de estas líneas, el dispositivo externo toma el mando de ellas para realizar *acceso directo a memoria (DMA)*.

Señal de reconocimiento a la solicitud de los buses BUS ACKNOWLEDGE (BUSACK). Esta línea es activada por el μp en respuesta a una solicitud de los buses mediante BUSRQ, para indicar que ya se ha "desconectado" de los buses de direcciones y de μp . Así le indica al dispositivo externo que puede tomarse el mando de ellas para realizar *acceso directo a memoria (DMA)*.

Señal de solicitud de interrupción INT. Esta línea es activada por dispositivos externos para *solicitar una interrupción* al programa que está ejecutando el μp .

Señal de reconocimiento de interrupción INTA. Cuando el μp está habilitado para aceptar interrupciones y algún dispositivo externo solicitó una a través de INT, el μp le indicará que acepta la interrupción activando INTA.

Una vez es activada INTA el μp interrumpe el programa que está ejecutando y ejecuta una *rutina de atención a la interrupción*. Dicha rutina debe haber sido elaborada para atender al dispositivo solicitante. Al terminar de ejecutarse esta rutina, la ejecución continuará en donde fue interrumpida.

Señal de restablecimiento RESET. Cuando esta línea es activada, el μp ejecuta un salto en el programa a una localidad fija (normalmente la 0000h).

El diseñador del sistema deberá aprovechar esto para ubicar en esa localidad (que deberá quedar en memoria ROM) un programa de inicialización de todos los dispositivos y los programas en ejecución.

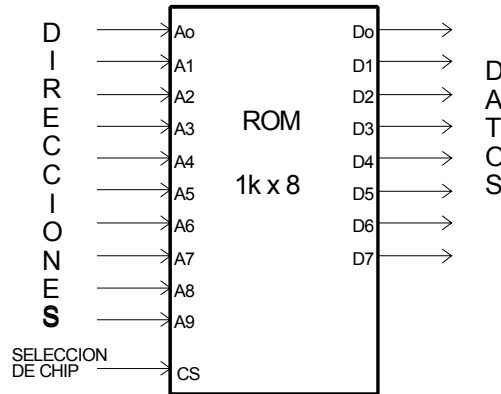
1.6 Decodificación de Memoria RAM y ROM

En esta sección se describirá la manera como se conectan dispositivos externos a la CPU, tales como Memoria RAM y ROM y puertos de entrada y salida

Conexión con memoria ROM.

Introducción a los microprocesadores

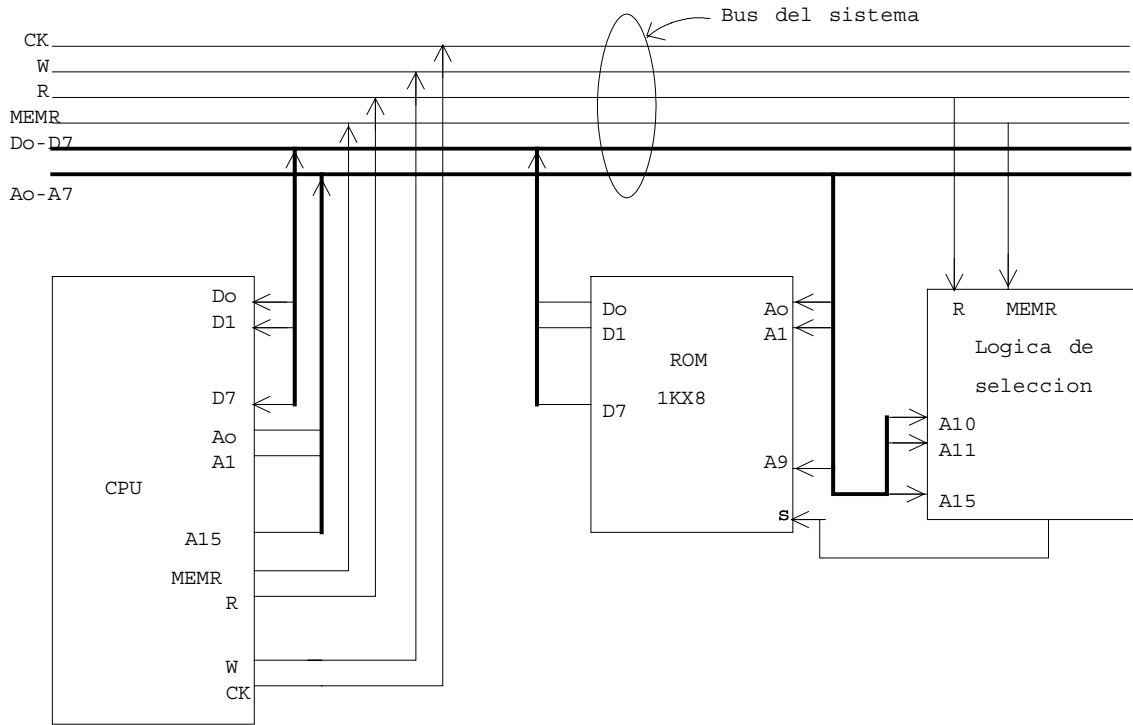
Una memoria ROM típica está organizada por localidades de memoria de 8 bits cada una, a diferencia de las memorias RAM que pueden normalmente contener un bit, 4 u 8 bits por localidad. En la figura 9 se muestra el diagrama funcional de una ROM típica de 1 Kilobyte, es decir, de 1K x 8 bits.



Obsérvese que la ROM es un dispositivo asíncrono, es decir, no requiere señal de reloj.

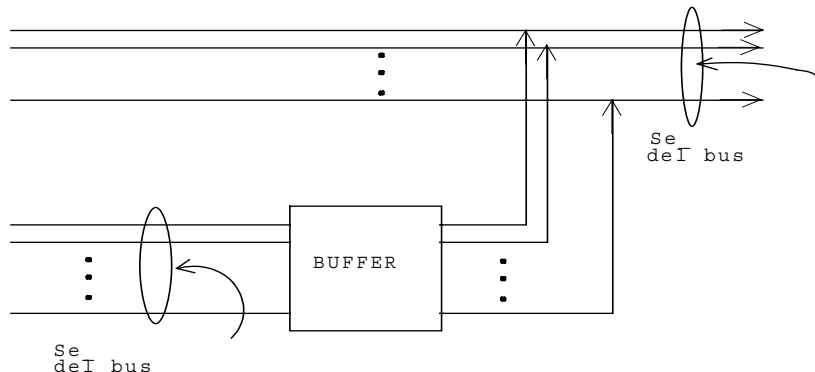
- Las k líneas de dirección de la ROM seleccionan alguna de las 2^k localidades internas de memoria. Además, un circuito de ROM puede poseer una o varias líneas S que seleccionan el chip y normalmente se usan para que la ROM responda a partir de una dirección llamada *dirección base* proporcionada por una *lógica de selección* especialmente diseñada, también llamada *decodificador*.

En la figura 10 se ilustra como se conectaría una ROM de 1Kb con la CPU.



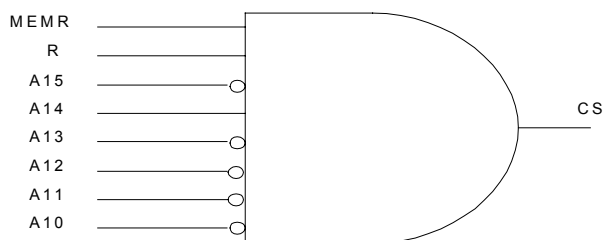
Introducción a los microprocesadores

*OBSERVACION.- Aunque aquí se supone que los dispositivos se conectan al bus del sistema directamente, en una conexión real, el bus se debe proteger mediante un BUFFER como se muestra en la figura 11



Para seleccionar el buffer se deben observar las REGLAS DE CARGA (FAN-OUT) de la circuitería usada, es decir, los buffer deben ser capaces de proporcionar corriente a todos los dispositivos conectados al bus.

La lógica de selección para la memoria ROM de la figura 10 puede ser tan sencilla como se muestra en la figura 12.



De acuerdo a la lógica de la figura 12, la señal de selección de chip (CS) se activará cuando la

$$\bar{A}15.A14.\bar{A}13.\bar{A}12.\bar{A}11.\bar{A}10 = 1$$

CPU lea las localidades de memoria en las cuales, es decir, cuando A15=0, A14=1, A13=0, A12=0, A11=0 y A10=0, es decir, cuando el bus de direcciones contenga una dirección en el rango de 4000h a 43ffh como se detalla en la tabla 4.

Lo anterior quiere decir que la ROM del ejemplo responderá en el rango siguiente

desde 4000H hasta $4000H + 1K-1 = 4000H + 3FFh = 43ffh$

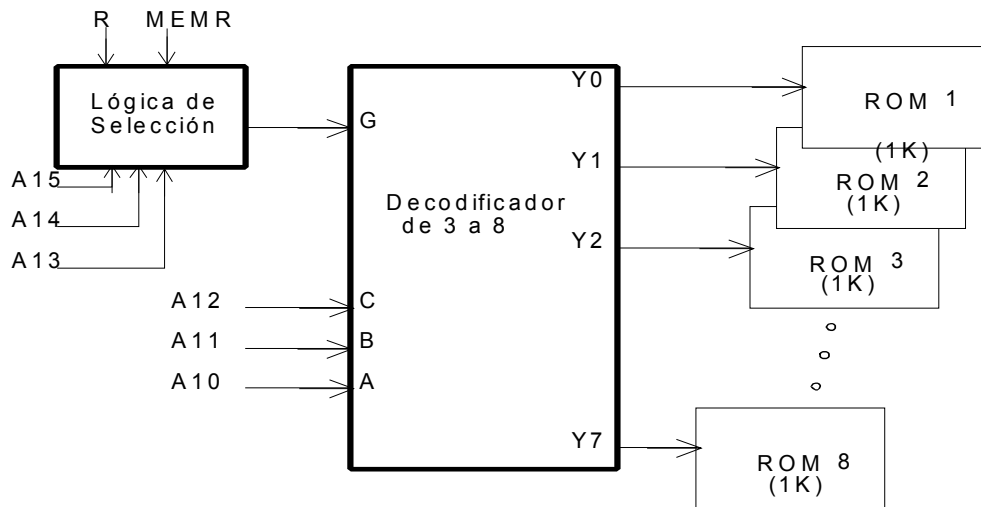
Así, cuando el programa que esté ejecutando la CPU lea una localidad de memoria en este rango, la información será tomada del chip de ROM conectado como se ha mostrado aquí.

Introducción a los microprocesadores

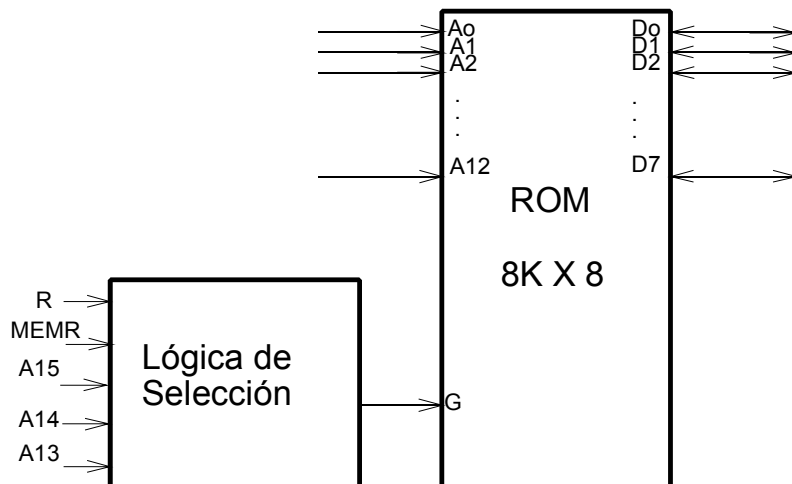
Dirección	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	CS
4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
de 4000 a 43ff	0	1	0	0	0	0	*	*	*	*	*	*	*	*	*	*	1
43ff	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
otra	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0

Tabla 4 Mapa de memoria para la figura 10

Cuando se van a seleccionar varios chips de memoria (lo cual es el caso más común) puede usarse también un decodificador de circuito integrado, por ejemplo, en la figura 13 se muestra la selección de 8 kb de ROM usando un conjunto de 8 chips de memoria de 1kb cada uno.

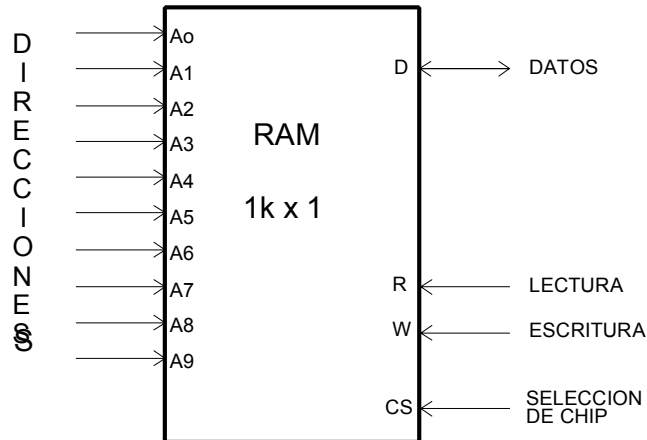


El diagrama de la figura 13 puede interpretarse como si fuera un sólo bloque de 8K de ROM como se muestra en la figura 14



Conexión con Memoria RAM

Lo discutido anteriormente es válido también para una RAM, salvo que a diferencia de la ROM, ésta también puede recibir datos, para ésto, requiere de líneas R (read) y W (write), además de las otras líneas descritas para la ROM. como se muestra en la figura 15.



1.7. Decodificación de Puertos de Entrada/Salida

Existen tres métodos mediante los cuales la CPU puede realizar transferencia de datos con el exterior.

- 1.- Entrada / salida programada. En este caso, toda transferencia de datos con el exterior es directamente controlada por el programa que está ejecutando el microprocesador.
- 2.- Entrada / salida por interrupciones. La lógica externa interrumpe el programa que esté ejecutando el microprocesador para ser atendida.
- 3.- Acceso Directo a Memoria (DMA). En este caso la lógica externa "desconecta" al microprocesador, para operar directamente sobre la memoria del del sistema. Es decir, la lógica externa se "adueña" por momentos de la memoria del sistema, para lo cual dicha lógica debe poseer características similares a las del μp , para poder manejar memoria.

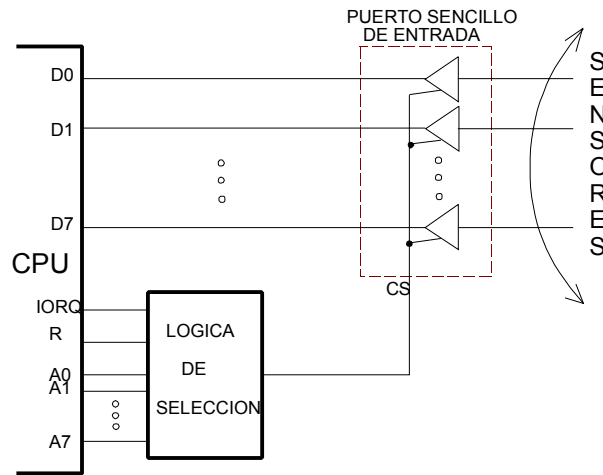
1.7.1. Entrada/Salida Programada.

En este caso la transferencia de datos se realiza a través de un puerto de E/S. La mayoría de los μp s no usan todas sus líneas de dirección para seleccionar puertos, en general, para μp s de 8 bits supondremos que se usarán sólo 8 líneas (de A0 a A7) para tal fin.

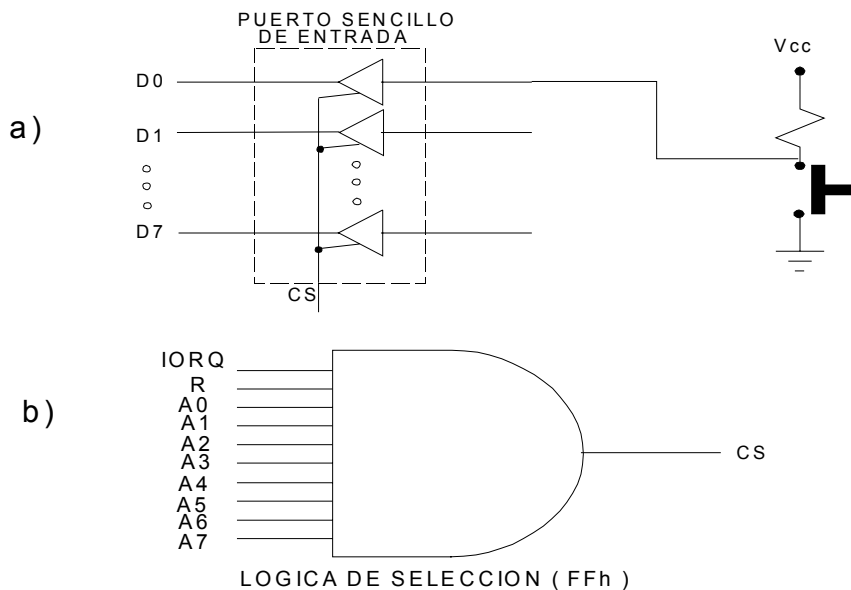
*Los puertos de E/S mas complejos se tratarán en el siguiente capítulo, aquí se mencionará como construir algunos puertos sencillos

Un puerto de entrada sencillo.

Es posible construir un puerto de entrada usando solamente un buffer con capacidad de tercer estado como conexión con el exterior. Dicho buffer sólo se activará en el momento en que el bus de datos esté listo para recibir un dato de entrada, para esto se deberá diseñar la lógica de selección adecuada. En la figura 16 se muestra la conexión de un puerto de este tipo.



De acuerdo a la figura 16, el buffer dejará pasar la información del exterior al bus de datos sólo cuando se active IOR, R y la dirección elegida con la lógica de selección. De esta manera la forma de obtener en el programa la información del exterior, es mediante una lectura al puerto con la dirección seleccionada.



Introducción a los microprocesadores

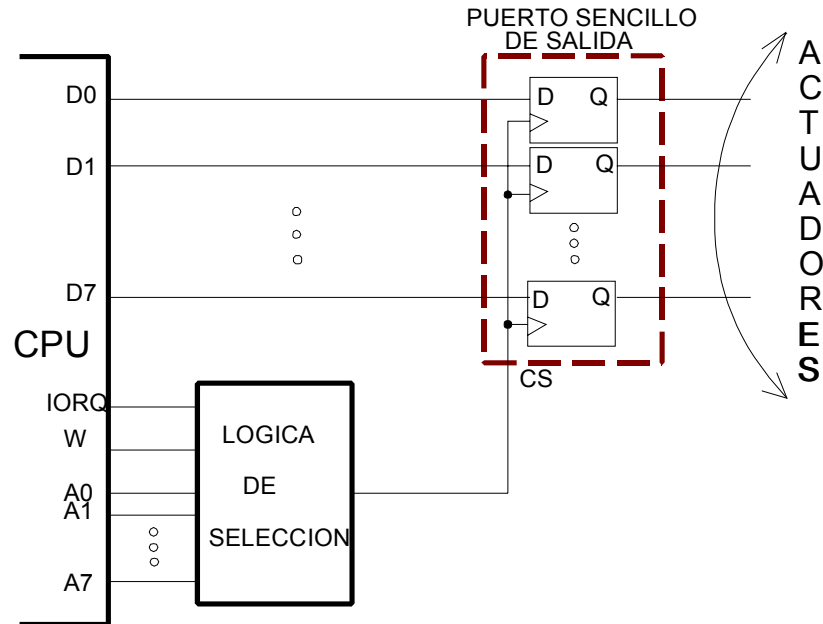
Ejemplo: En la figura 17(a) se muestra la conexión de un botón pulsador normalmente abierto al bit LSB del puerto sencillo de entrada cuya lógica de selección se muestra en la figura 17(b), la cual establece la dirección ffh.

Supongamos que se quiere leer el estado del pulsador mediante un programa BASIC sería como sigue

```
D% = INP(255)
IF (D% AND 1)=0 THEN PRINT "Botón presionado"
ELSE PRINT "Botón suelto"
```

* La operación (D% AND 1) es necesaria para *enmascarar* los bits que no nos interesan, es decir, para que sea cual sea su valor, éstos no afecten la comparación.

Un puerto de salida sencillo.

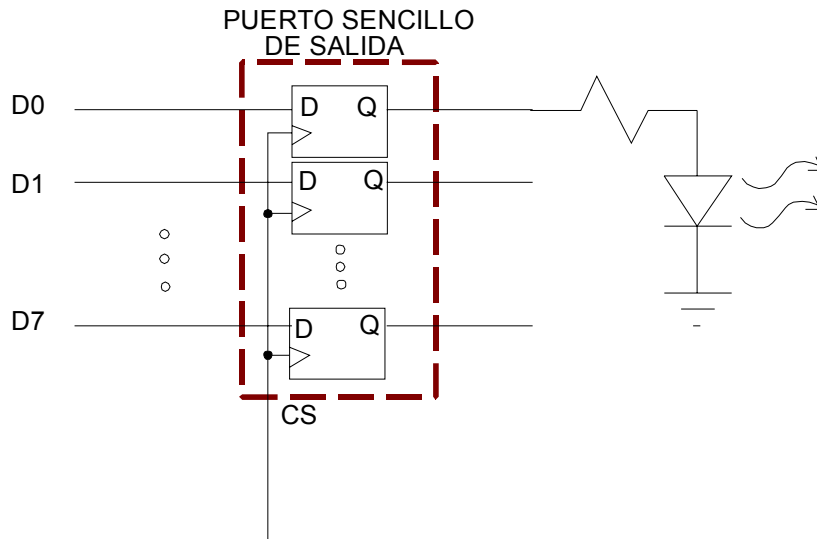


Un puerto sencillo de salida puede construirse mediante un Flip-Flop tipo Latch por cada bit de salida. De esta manera, cuando se desea "sacar" la información del bus de datos a través del puerto, bastará con activar la señal de reloj del latch para que el dato que en ese momento se encuentre en el bus de datos se quede "congelado" a la salida del latch. Para que sólo pueda ocurrir en una operación de salida a puerto, deberá diseñarse la lógica de selección adecuada como se muestra en la figura 18

Introducción a los microprocesadores

Ejemplo: En la figura 19 se muestra la conexión de un led al bit LSB de un puerto de salida sencillo. Suponiendo la misma dirección base que en el ejemplo anterior (FFh). El siguiente programa BASIC encendería el LED un momento y después lo apagaría.

```
OUT 255,1 : REM Enciende el Led
FOR i=1 TO 1000 : NEXT i : REM pausa
OUT 255,0 : REM Apaga el Led
```



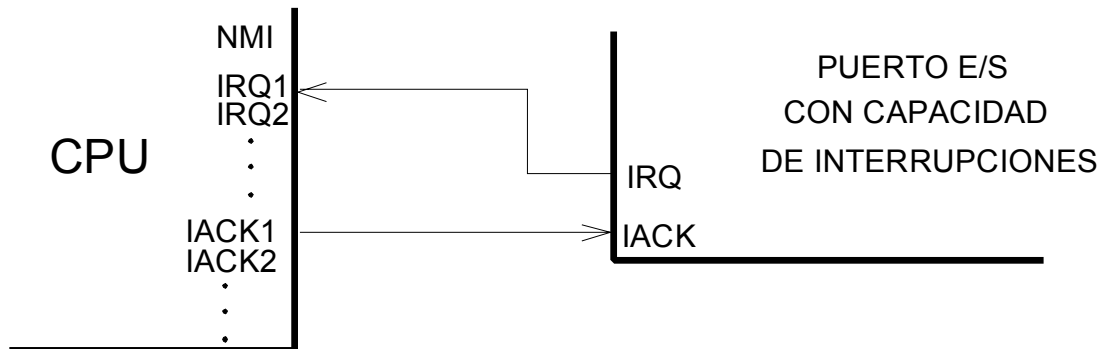
Mapeo a puerto y a memoria

Obsérvese que en el lugar de la línea IOR, podría usarse la línea MEMR, de esta manera el puerto (de entrada o de salida), será visto por la CPU como si fuera una localidad de memoria, ya que responderá a operaciones con memoria y no con puerto.

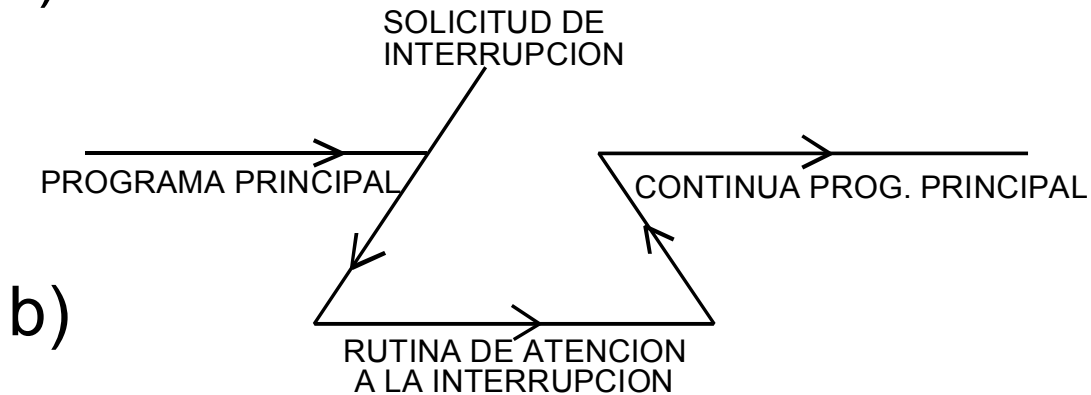
Dependiendo de si se usa IOR ó MEMR, se dice que el dispositivo de E/S está mapeado a puerto ó mapeado a memoria, en este último caso en lugar de INP, OUT, se deberá usar PEEK, POKE (En Basic), para accederlo.

1.7.2 Entrada/Salida por Interrupciones.

La mayoría de los μ ps poseen una o varias líneas mediante las cuales la lógica externa puede solicitar interrupciones. Por ejemplo, en la figura 20(a) se muestra un μ p que posee varias líneas llamadas IRQ1, IRQ2,... a través de las cuales puede recibir una solicitud de interrupción, de la misma manera, posee líneas IACK1, IACK2,... para responder si ha reconocido o no la solicitud correspondiente.



a)

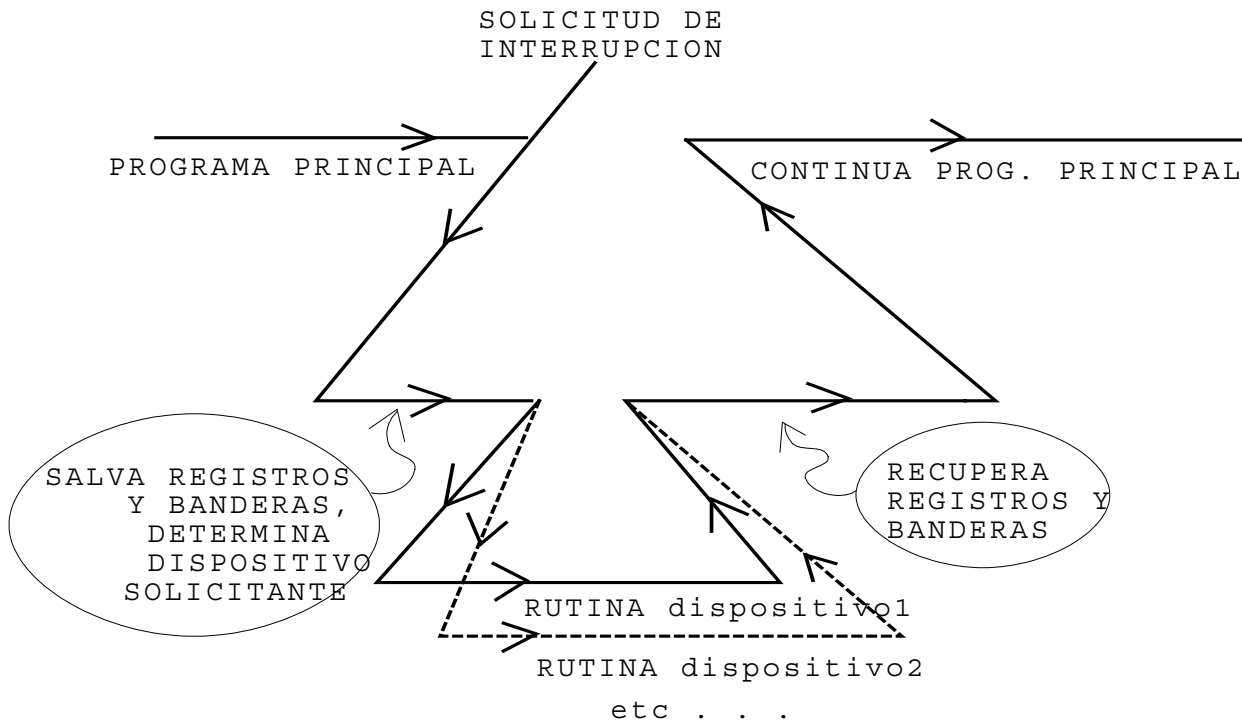


b)

Respuesta de un μp a una Interrupción.

Cuando la CPU reconoce una solicitud de interrupción (a través de IRQ), lo indica activando IACK, a continuación realiza un salto a una dirección de memoria en donde estará la *rutina de atención a interrupción*, es decir, transfiere la ejecución del programa a una rutina especial para tratar la interrupción, en forma simbólica esto se ilustra en la figura 20 (b).

Debido a que una interrupción puede ocurrir en cualquier punto de la ejecución del programa principal, la rutina de atención a la interrupción en su inicio deberá salvar el contenido de todos los registros que vaya a alterar, para antes de retornar al programa principal dejarlos como estaban, de lo contrario, se alterará la información que está manejando dicho programa principal, lo cual tendría consecuencias fatales en el funcionamiento del programa. Por otro lado, en un sistema de microcomputadora existen varios dispositivos capaces de solicitar interrupción, de hecho, varios pudieran solicitarla al mismo tiempo, por ésto, la rutina de atención deberá identificar al principio cual es la fuente de interrupción y así ejecutar la subrutina adecuada dentro de la rutina de atención a la interrupción. Esto se ilustra en la figura 21.



Por otro lado, ¿Cómo sabe la CPU en dónde encontrar la rutina de atención a la interrupción?. Cualquier otra rutina se ejecuta normalmente mediante un salto a la localidad especificada en el programa, pero en una interrupción **no** es el programa el que indica el salto, de hecho, el programa es interrumpido en un momento impredecible. Así que la única manera de que la CPU localice la rutina de atención es que el mismo dispositivo solicitante se identifique de alguna manera para que de acuerdo a esta identificación la CPU lo atienda. Esto se ha resuelto de varias maneras:

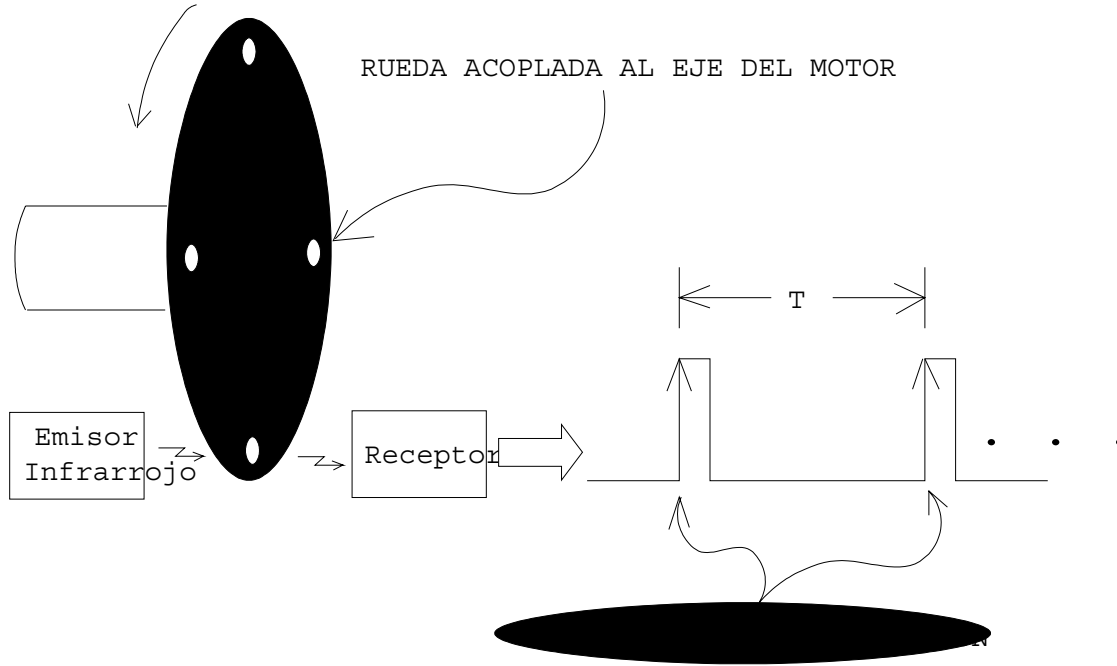
- El dispositivo solicitante envía al μp un código de identificación mediante el cual el μp forma un *vector de interrupción*.
- El dispositivo solicitante está conectado a un línea específica de solicitud de interrupción (IRQ1, IRQ2, ... etc) ya sea directamente a la CPU, o a través de un dispositivo *controlador de interrupciones* y la CPU tiene asignada a cada una de estas líneas un vector de interrupción fijo.

El **vector de interrupción** es un dato a través del cual se calcula la dirección de la rutina de atención a la interrupción solicitada.

* Las solicitudes de interrupción a través de las líneas IRQ pueden ser ignoradas por el μp si el programador así lo desea, *enmascarando* un bit de control, por ésto a estas líneas se les llama enmascarables. Casi todos los microprocesadores poseen una línea NMI (solicitud de *interrupción no enmascarable*), la cual no puede ser deshabilitada mediante el programa, a esta línea se conectan las interrupciones más importantes para el funcionamiento del sistema.

Introducción a los microprocesadores

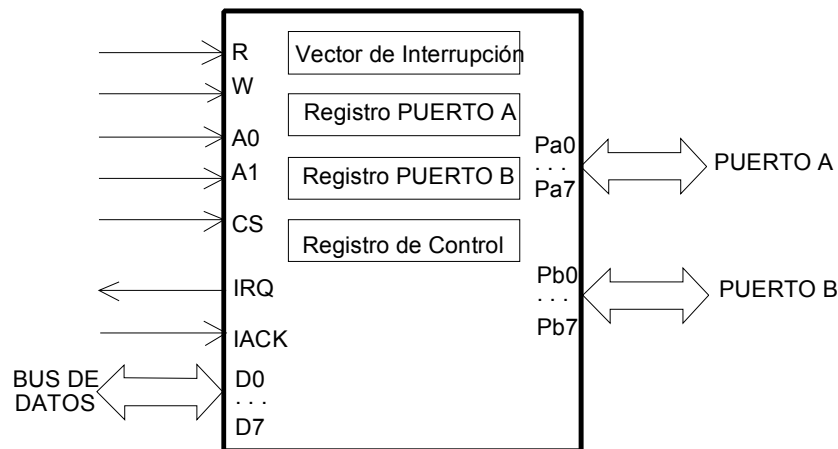
Ejemplo: Un ejemplo de aplicación típico de un proceso manejado mediante interrupciones, es la medición de velocidad angular usando un sensor acoplado a la flecha de un motor como se muestra en la figura 22.



La rutina de atención para este caso mediría el tiempo T , a partir del cual se puede calcular la velocidad en el programa principal. La ventaja de esta manera de medir velocidad angular es que el programa principal no necesita detenerse a esperar el siguiente pulso del sensor, es decir, el programa puede estar realizando otras labores (cálculos, despliegue de pantalla, impresión, acceso a disco, lectura de teclado, etc.) mientras el sensor no envíe un nuevo pulso.

En la figura 23 se muestra un dispositivo típico de Entrada/Salida con capacidad de interrupciones. Dicho dispositivo posee 2 puertos de entrada/salida programables, cada línea del puerto puede programarse para solicitar o no interrupción bajo alguna condición, así como su dirección (entrada o salida). Para programar estas condiciones el dispositivo posee un *registro de control*. De esta manera el dispositivo es visto por la CPU como cuatro puertos seleccionables mediante $A1$ y $A0$ como se muestra en la siguiente tabla.

A1 A0	Selecciona	Tipo de puerto
0 0	Puerto A	Entrada/salida
0 1	Puerto B	Entrada/salida
1 0	Vector de Interr.	Entrada
1 1	Reg. de Control	Salida



DISPOSITIVO CON DOS PUERTOS E/S
Y CON CAPACIDAD DE INTERRUPCIONES

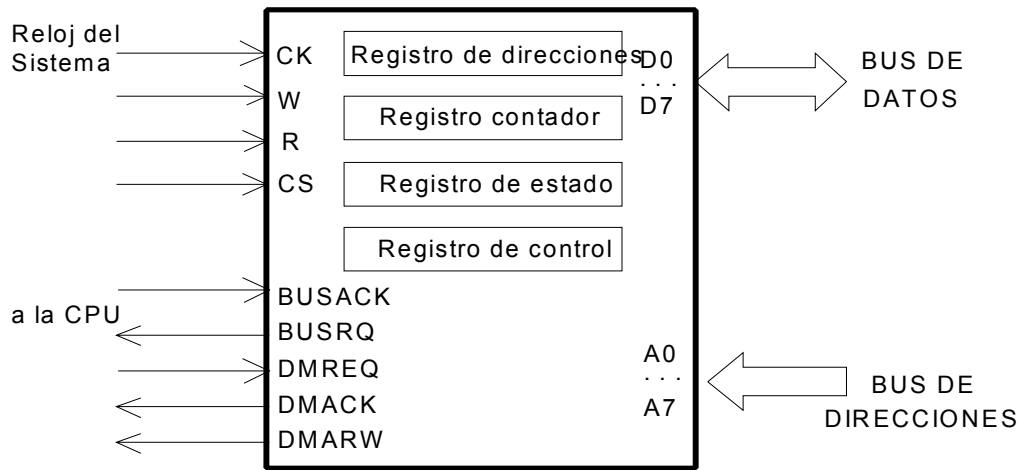
1.7.3. Entrada / Salida por D.M.A. (Acceso Directo a la Memoria).

Un dispositivo con capacidad de DMA proporciona la forma más rápida de transferencia de datos de E/S con la memoria, ya que en una operación DMA la CPU queda "desconectada" de sus buses y por lo tanto, es parada toda actividad de transferencia de datos entre la CPU y la memoria, dejando esta última libre para ser operada por el dispositivo con capacidad de DMA.

Un dispositivo DMA debe tener lógica tipo CPU capaz de leer y escribir en la memoria usando los buses de la CPU, esta última deberá poner sus conexiones a los buses en 3er. estado en operaciones de DMA.

Un dispositivo controlador típico de DMA debe poseer por lo menos los siguientes registros internos, además de las líneas mostradas en la figura 24

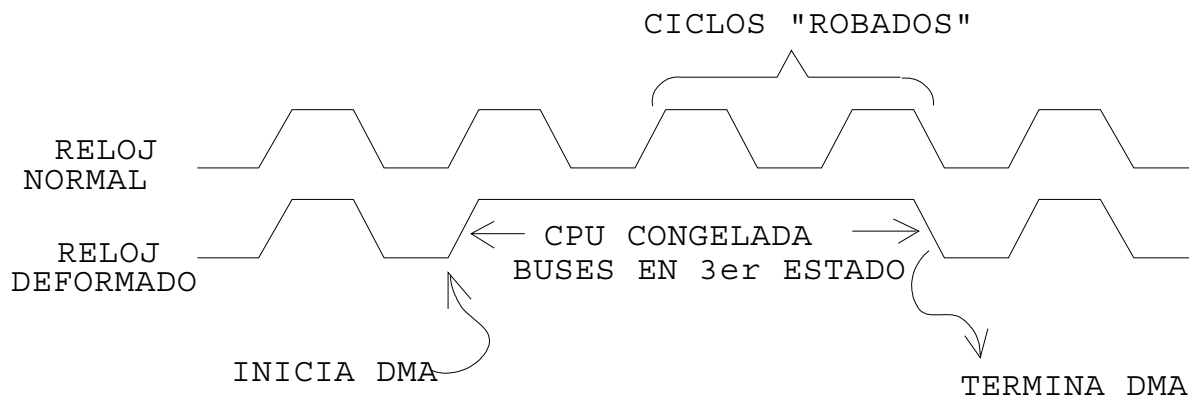
- Un registro de direcciones, el cual contiene la dirección de la siguiente localidad de memoria a acceder.
- Un registro contador, el cual contiene el número de bytes que restan por leerse O escribirse del bloque programado al principio.
- Un registro de control, mediante el cual la CPU puede programar al dispositivo DMA para arrancar, parar, establecer la dirección del flujo de datos (lectura o escritura).
- Un registro de estado, a través del cual la CPU puede darse cuenta del estado de la operación que el dispositivo está efectuando.



DISPOSITIVO CON CAPACIDAD DE ACCESO DIRECTO A LA MEMORIA (DMA)

Una operación DMA se pone en marcha cuando un dispositivo externo lo solicita al dispositivo DMA a través de la línea DMAREQ. Si el dispositivo DMA ha sido programado adecuadamente, solicitará a la CPU el control de los buses a través de la línea BUSRQ, si la CPU está lista reconocerá la solicitud a través de la línea BUSACK, lo cual hará que el dispositivo de DMA active también DMACK e inmediatamente proceda a imitar lo que la CPU (ya desconectada) hubiera hecho en una operación de acceso a memoria (lectura o escritura).

DMA con Robo de Ciclo. Esta técnica es usada por algunos μ ps y consiste en "alargar" el ciclo de reloj de la CPU para "congelarla" mientras el dispositivo DMA toma control de los buses como se muestra en la figura 25.



Otra técnica consiste en el uso de una señal INHIBIT mediante la cual el dispositivo DMA puede desconectar todas las señales del bus de la CPU mientras el primero toma el control de los buses. Una

Introducción a los microprocesadores

modificación de esta última técnica sustituye la señal INHIBIT por dos señales BUSRQ y BUSACK para darle oportunidad a la CPU de ignorar por un tiempo la solicitud de DMA (a través de BUSRQ) y contestarla (a través de BUSACK) sólo hasta que esté lista.

De esta manera el programador de operaciones DMA sólo se debe encargar de programar al inicio (inicializar) al dispositivo DMA y después el propio dispositivo se encargará de las transferencias sin interrumpir la lógica posterior del programa.

*El único efecto sobre el programa en ejecución es que éste verá retardada su ejecución por los ciclos DMA insertados por el dispositivo, sin que éste sea advertido en ningún punto por la lógica del programa (salvo en la inicialización). Esto se ilustra en la siguiente figura:

Ejemplos de microprocesadores de 8 bits

1.8. Criterios para la Selección de μ 's.

Además de los criterios ya mencionados en el capítulo anterior (longitud de palabra, velocidad, compatibilidad, costo, sistemas de desarrollo disponibles, etc.), es conveniente considerar algunas cuestiones adicionales:

μ de 4 bits.- Son μ 's muy económicos para aplicaciones en control de hornos de microondas, lavadoras y otros electrodomésticos o dispositivos de poca complejidad.

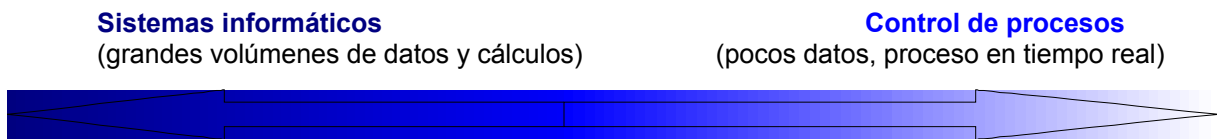
μ de 8 bits.- La mayoría de los μ 's de 8 bits poseen arquitectura muy similar a la descrita en el capítulo anterior varían un poco en velocidad (desde 1 a 0.2 μ seg). El aspecto más variable es la disponibilidad de circuitería de apoyo para diferentes μ 's y la capacidad de interrupciones.

μ de 16 bits.- La variación en la arquitectura de los μ de 16 bits es mucho mayor que la de 8 bits dependiendo del fabricante, por ejemplo:

- El bus de datos puede ser de 8 ó 16 bits, si el μ de 16 bits posee un bus de datos de 8 bits, ésto le da compatibilidad con la circuitería desarrollada para μ de 8 bits. (pudiendo adoptar así desarrollos previos para 8 bits).
- Todos los μ de 16 bits incorporan instrucciones de multiplicación, división y manipulación de bloques, ésto reduce la programación necesaria para aplicaciones que requieren este tipo de instrucciones, además, la velocidad de ejecución aumenta.

μ de 32 bits.- Estos μ 's han sido diseñados para aplicaciones de alta tecnología (minicomputadoras). Debido a ésto, no serán considerados en este curso, ya que no son recomendables para implementar sistemas mínimos.

Microcontroladores.- Una corriente cada vez más fuerte es el uso de los μ c para aplicaciones de control de procesos o bien en las cuales no se requiere manejo de grandes volúmenes de datos. Esta opción es casi siempre la más adecuada para la implementación de sistemas mínimos, ya que los microcontroladores ya tienen integrados en un sólo módulo los componentes necesarios para múltiples aplicaciones de adquisición de datos, monitoreo y control de procesos. El siguiente diagrama muestra esquemáticamente la naturaleza preferente de las aplicaciones de los microprocesadores y de los microcontroladores



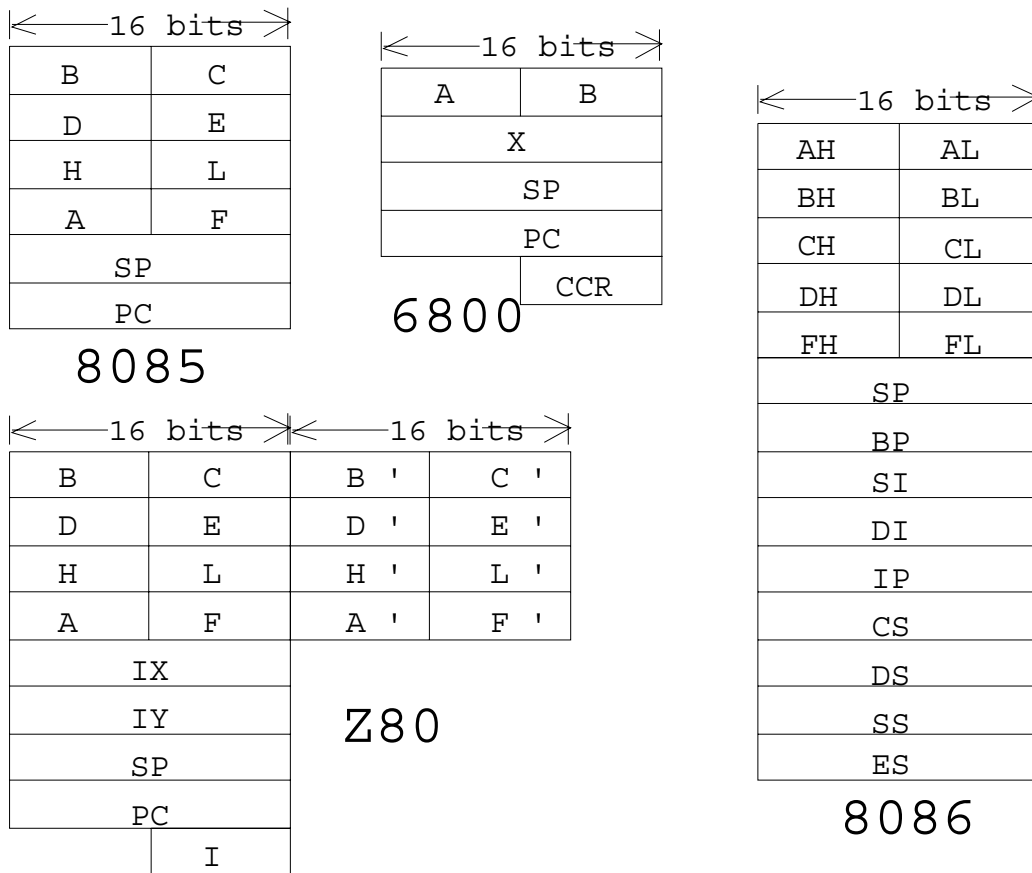
Como una muestra de la gran diversidad de opciones para elegir un microprocesador comercial en la siguiente tabla se muestran sólo algunos pocos de los μ 's más comunes.

Matric	Fabricante	Tipo	Longitud de palabra	Bus de direcc.	Mhz	reloj Año
Z80	zilog	μ	8	16	2.5	1976
Z8000	zilog	μ	16	16	4	1981
6800	motorola	μ	8	16	1	1974
68000	motorola	μ	16	24	8	1980
6502	commodore	μ	8	16	1	1975
8085	intel	μ	8	16	2.5	1976
8086/8088	Intel	μ	16	20	5	1979

Introducción a los microprocesadores

80286	intel	μp	16	32	5	1982
80386	intel	μp	32	32	10	1984
8048	intel	μc	8	12	2.5	1977
8096	Intel	μc	16	16	6	----
80196	intel	μc	16	16	12	1990
6805	motorola	μc	8	8	4	1979
6811	motorola	μc	8	16	2	1990

Otro punto de comparación son los registros internos de un μp (cuántos y de que tamaño son). En la figura 26 se muestran los registros de algunos μp's de 8 bits.

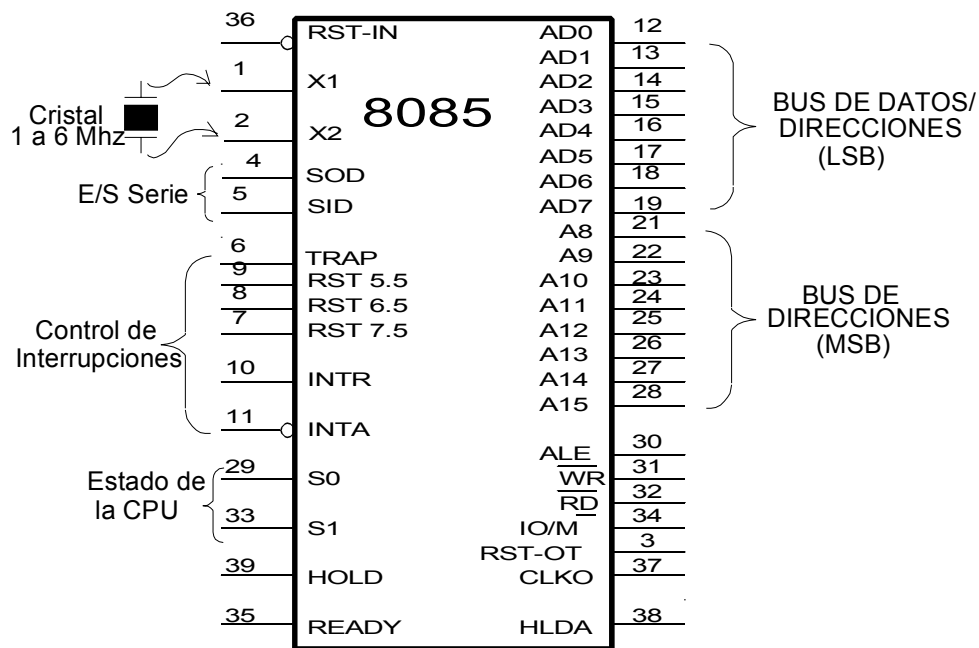


El Microprocesador 8085 de INTEL. A continuación se describen algunas de las características generales del 8085, así como el funcionamiento de sus principales líneas, con el fin de proporcionar la información básica para su conexión con dispositivos externos. Se ha elegido como ejemplo este μp por ser el antecesor del 8088/8086, al cual esta dedicada la segunda mitad del curso.

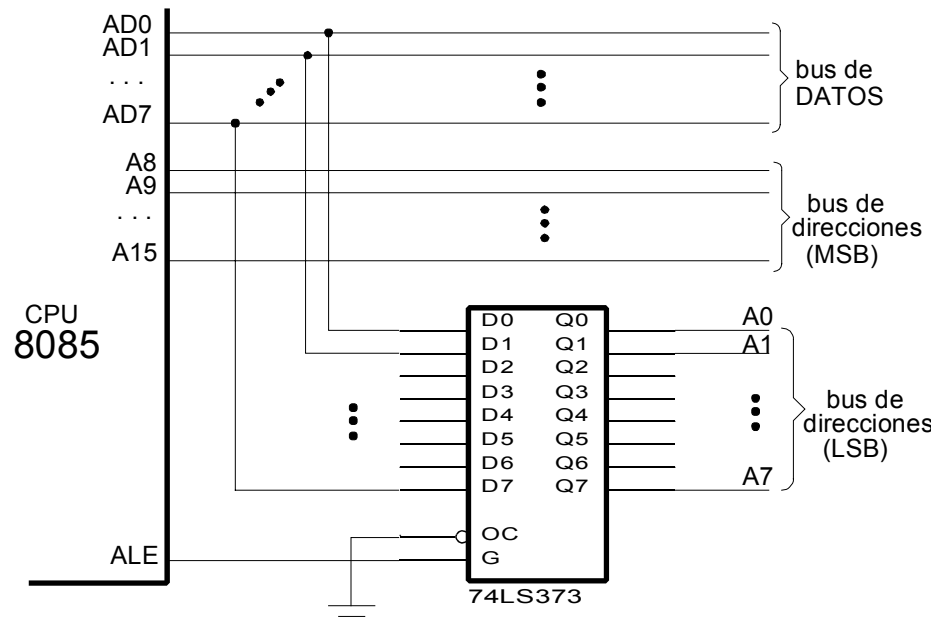
Características generales del 8085:

- Bus de datos de 8 bits
- Circuitería de reloj integrado
- Una sola fuente (+5V)
- 80 instrucciones
- velocidad de 770 000 sumas/seg.
- puerto de E/S serie integrada
- 16 bits de dirección
- Bus de datos/direcciones multiplexada

En la figura 27 se muestra el diagrama de patitas del 8085 y a continuación se describe la función de las patitas más importantes del mismo. Se describirán aquí solamente las que funcionan de manera diferente a lo descrito de manera general en el capítulo anterior de acuerdo al nombre de la línea



Bus de Datos/Direcciones (AD0,...,AD7,A8,...,A15).- En el 8085, el bus de datos se encuentra compartido (multiplexado) con la mitad menos significativa del bus de direcciones. De esta manera, para poder usar los datos de manera separada se requiere de circuitería externa adicional como se muestra en la figura 28



Selección de memoria o puerto IO/\overline{M} . Esta línea se pone en alto para indicar operaciones con puertos y se pone en bajo en operaciones con memoria.

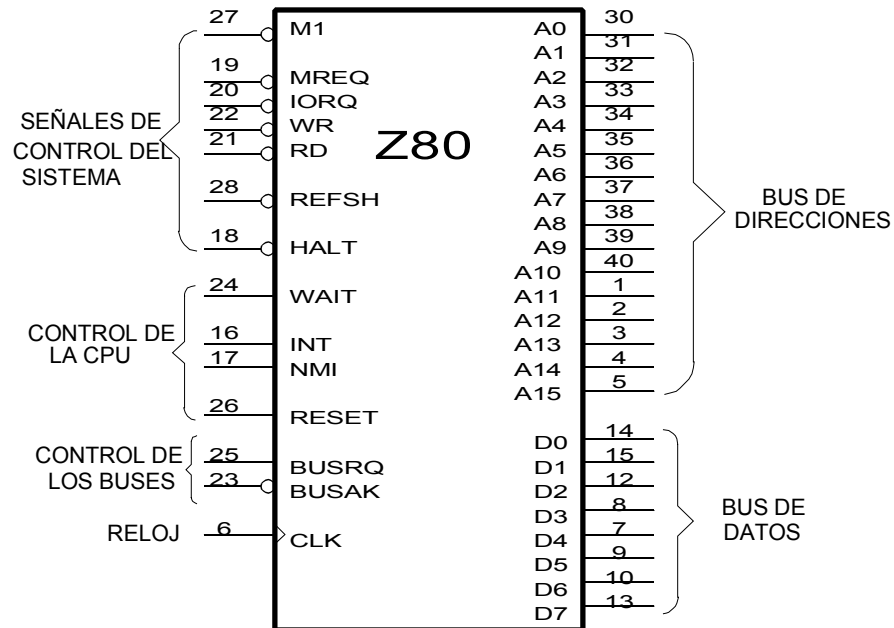
RESET IN. Cuando esta línea es activada por algún dispositivo o evento externo ocasiona un salto a la localidad 0000H. El diseñador por lo tanto deberá colocar en esa localidad el inicio del programa de arranque o reinicialización de todo el sistema.

Señal de espera \overline{RDY} . Cuando esta línea es activada (en bajo) por algún dispositivo externo pone al 8085 en un estado de espera (para dispositivos lentos). El dispositivo lento deberá poner esta línea en bajo cuando no está listo para responder.

Señales del puerto serie SID Y SOD. Estas líneas son la entrada y salida respectivamente del puerto serie interno del 8085.

Estado de la CPU S0 Y S1. Este par de líneas junto con IO/M indican el estado de la CPU (estado "halt", lectura/escritura a memoria o a puerto, reconocimiento del interrupción)

El Microprocesador Z80 de ZILOG. En forma similar a como se hizo con el 8085, se describen aquí las características más sobresalientes del Z80 que lo hacen diferente de los otros μp de 8 bits. Se ha elegido el Z80 como ejemplo por ser uno de los μp de 8 bits más poderosos.



Características generales del Z80:

- Bus de datos de 8 bits
- Bus de direcciones de 16 bits
- 158 instrucciones
- 16 registros internos
- Refrescamiento de memorias dinámicas
- Operaciones de 16 bits
- Una sola fuente (+5V.)
- Una sola señal de reloj.

En la figura 29 se muestra el diagrama de patitas del Z80, a continuación se describen las patitas que realizan funciones algo diferentes a las descritas hasta aquí.

Indicador de ciclo FETCH $\overline{M1}$. Ciclo de Máquina 1 (indica que la CPU está ejecutando el ciclo FETCH). Otra función que tiene esta línea es en conjunto con IOREQ, cuando ambas están activas indican reconocimiento de interrupción.

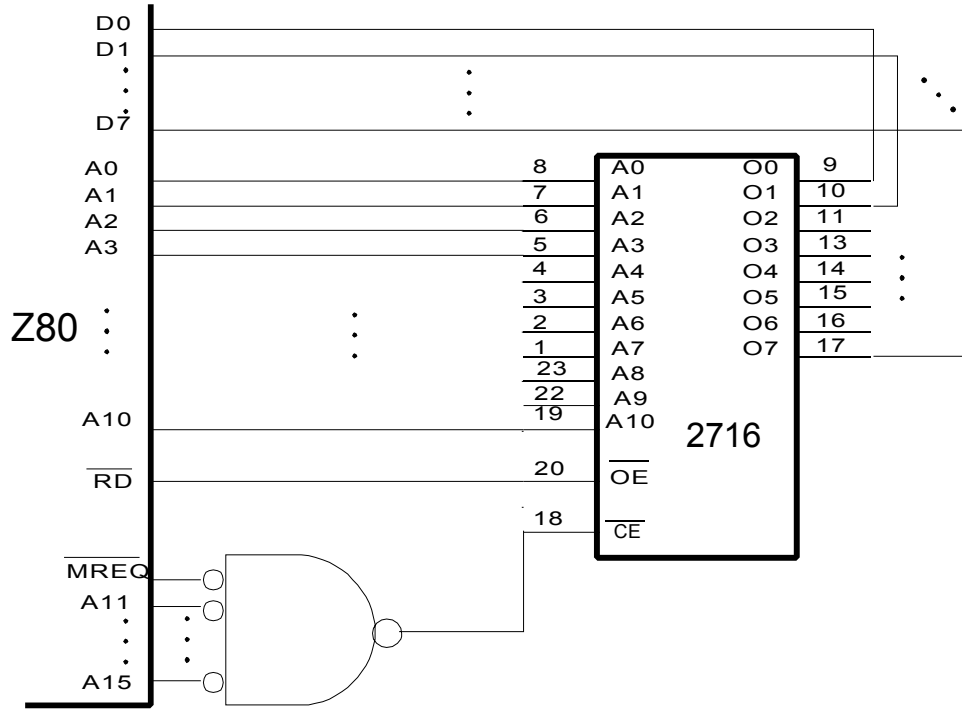
Refresco de memoria \overline{REFSH} . Junto con MREQ indican que la dirección activa en el bus de direcciones es para refrescamiento de memoria.

\overline{HALT} . - Indica que la CPU está ejecutando instrucciones NOP o una instrucción HALT.

Estado de espera \overline{WAIT} . Por medio de esta línea los dispositivos lentos pueden solicitar a la CPU que se ponga en un estado de espera hasta que los dispositivos estén listos para responder.

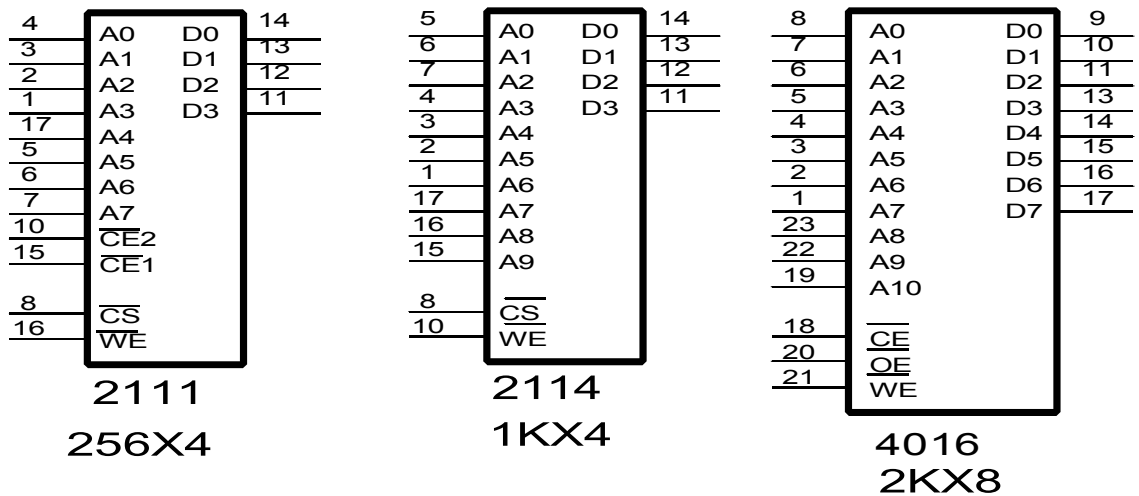
Introducción a los microprocesadores

Ejemplo: Conectar 2K de ROM al Z80 usando el 2716. En la figura 31 se muestra la conexión de estos dispositivos tomando como dirección base la 0000h.



Ejemplos de memorias RAM

En la figura 32 se muestran los diagramas de patitas de 3 dispositivos coerciales de memoria RAM estática, nuevamente es recomendable consultar las hojas de los fabricantes para ver la gran cantidad de dispositivos existentes en el mercado.

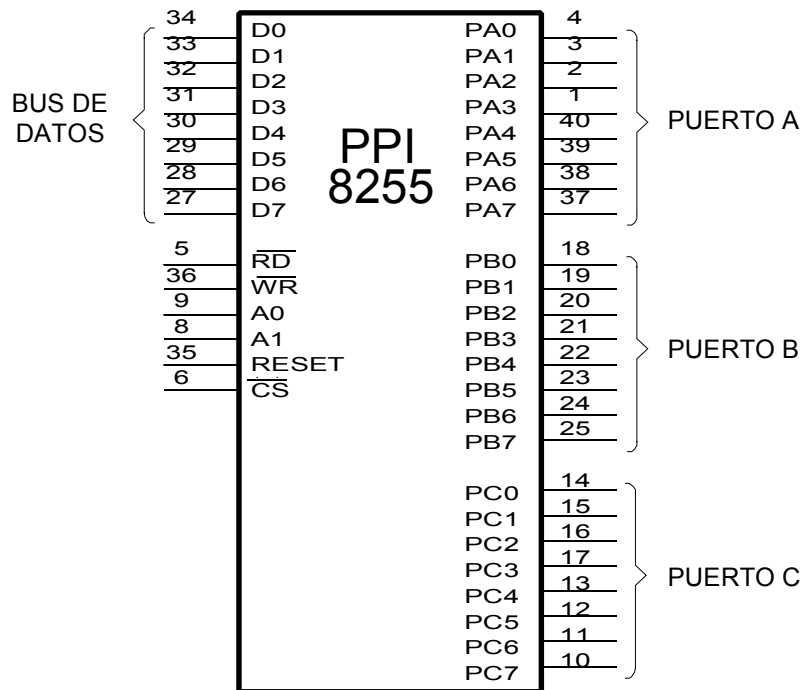


Obsérvese en la figura 32 que el 2111 y el 2114 usan sólo una línea para control de lectura/escritura, esta línea \overline{WE} debe ponerse en alto en operaciones de lectura y en bajo en operaciones de escritura.

En cambio para el 4016 se dispone de dos líneas, \overline{OE} que se activa para seleccionar lectura y \overline{WE} que se activa para seleccionar escritura.

1.11. Ejemplos de Puertos

Uno de los puertos típicos, directamente compatibles con el 8085, 8088 y 8086, es el PPI 8255 (Programmable Peripheral Interface), su diagrama de patitas se muestra en la figura 33.



El 8255 posee internamente cuatro registros, tres de los cuales corresponden directamente a 3 puertos programables A, B y C de 8 bits y el cuarto registro (de control) es usado precisamente para programar el funcionamiento de los puertos A, B, y C. Estos puertos pueden ser programados para trabajar como puertos de entrada y/o de salida. Para programar los puertos de 8255 simplemente se

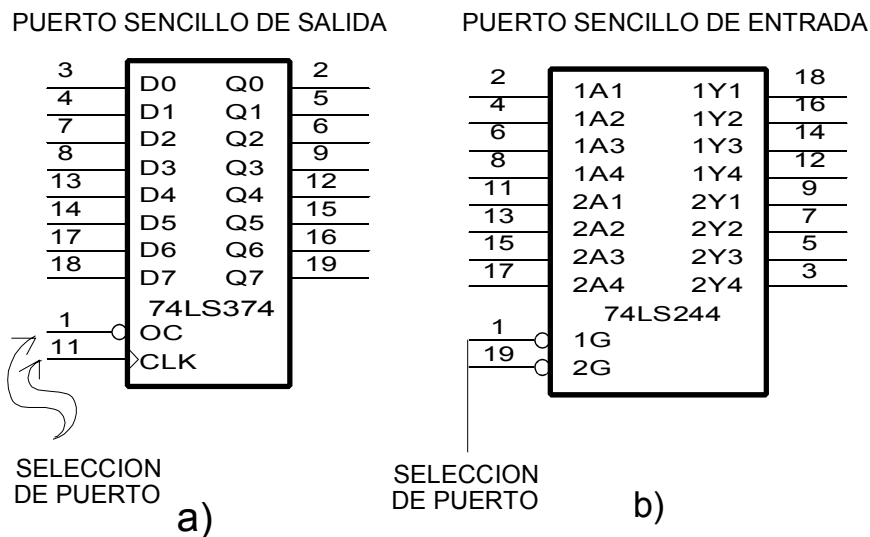
Introducción a los microprocesadores

escribe un código al registro de control, la forma de escoger el código se basa en la información proporcionada por el fabricante y se verá posteriormente.

En la siguiente tabla se muestra la manera de seleccionar los diferentes registros internos de 8255 mediante las líneas A0, A1 y CS. sta información se debe tener presente para diseñar la lógica de selección del 8255.

CS	A1	A0	Selecciona
0	0	0	Puerto A
0	0	1	Puerto B
0	1	0	Puerto C
0	1	1	Registro de Control
1	*	*	Tercer estado

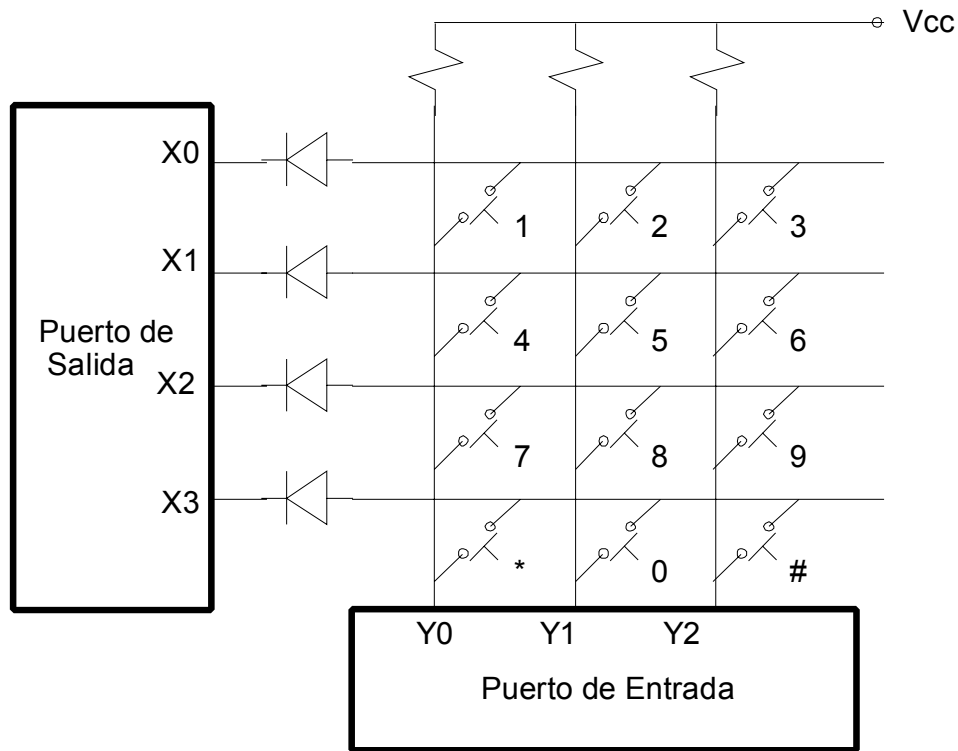
Como puerto sencillo de salida también pueden usarse el 74373, 74374, 74377 que contienen 8 Flip-Flops tipo D. De la manera como se describió en el capítulo anterior.



Como puerto sencillo de entrada, se puede utilizar el 74244, que contiene 8 puertas buffer con capacidad de tercer estado.

1.12 Decodificación de Teclados.

La idea básica para la decodificación de teclados hace uso de un arreglo matricial de líneas conectadas a dos puertos, un puerto de salida para activar los renglones y un puerto de entrada para recoger la información de las columnas. Este esquema se muestra en la figura 35.



Ejemplo: El siguiente es un programa básico que detecta si fue o no presionada la tecla 7,8 ó 9. de acuerdo al diagrama de la figura 35.

```
10 P1 = &h300 : P2 = &h301
20 OUT P1,&h0B : REM Activa el renglón Y2
30 D = INP (P2) : REM Lee información de columnas
40 IF (D AND 1) = 0 THEN PRINT "Tecla 9"
50 IF (D AND 2) = 0 THEN PRINT "Tecla 8"
60 IF (D AND 4) = 0 THEN PRINT "Tecla 7".
```

*OBSERVACION: Cuando se presiona una tecla se producen oscilaciones que duran aproximadamente 20 mseg. Debido a ésto, el programa anterior requiere realizar un filtrado previo, antes de la decodificación de la tecla presionada. Además, el programa del ejemplo no detecta si la tecla ya se ha soltado. En la figura 36 se muestra un diagrama de flujo de más completo para la decodificación de teclados.

