UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

AUTOR: Jaime Ayala Pérez

Arquitectura de computadoras		Clave: 1364
Plan: 2005		Créditos: 8
Licenciatura: Informática		Semestre: 3º
Área: Informática (Redes y telecomunicaciones)		Hrs. Asesoría: 2
Requisitos: Ninguno		Hrs. Por semana: 4
Tipo de asignatura:	Obligatoria (x)	Optativa ( )

Objetivo general de la asignatura

Al finalizar el curso, el alumno conocerá el fundamento teórico para comprender el funcionamiento de las computadoras digitales y contará con los elementos prácticos para analizar y diseñar los subsistemas lógicos que componen a éstas.

Temario oficial (horas sugeridas 64)

1. Introducción	(6 h)
2. Sistemas de numeración	(8 h)
3. Códigos	(8 h)
4. Álgebra de Boole	(8 h)
5. Circuitos combinatorios	(10 h)
6. Circuitos secuenciales	(10 h)
7. Memorias	(8 h)
8. Unidades funcionales	(6 h)

Introducción

La finalidad de este material es presentar de una manera clara y sencilla el análisis, diseño y funcionamiento de los diferentes módulos que componen una Computadora Digital. Este material está organizado de la siguiente manera:

En el tema 1 presentamos una clasificación de computadoras, las unidades básicas que constituyen una computadora digital, así como el funcionamiento de cada una de ellas. En el tema 2 estudiaremos los sistemas de numeración base 2, 8, 10 y 16, realizaremos conversiones entre diferentes bases, además de realizar las operaciones de suma, resta, multiplicación y división en base 2, 8, 10 y 16, con números con signo y sin signo. En el tema 3 explicaremos las diferentes formas de representar los caracteres en una Computadora Digital utilizando diversos códigos. En el tema 4, presentamos las bases de la electrónica digital como son: los tipos y uso de las compuertas lógicas básicas, así como también las técnicas de reducción de funciones lógicas como son: el Álgebra de Boole y los mapas de Karnaugh. En el tema 5 se analizan, diseñan y construyen diferentes circuitos combinacionales utilizando compuertas básicas (AND, OR y NOT) y que forman parte de una Computadora Digital como lo son: los sumadores, multiplexores y demultiplexores, codificadores y decodificadores, comparadores, etc. En el tema 6 se explica el procedimiento para diseñar circuitos secuenciales (utilizando flip-flops tipo RS, JK, D y T) y que son la base para la construcción de una Unidad de Control o para diseñar y construir módulos especializados como son: contadores, registros de desplazamiento, etc. En el tema 7 se presentan los diferentes tipos de memoria, su estructura interna y el funcionamiento de cada uno de ellos durante un proceso de lectura y/o de escritura en una Computadora Digital. En el tema 8 se presentan las unidades especializadas que forman parte de una Computadora Digital como lo son: el puerto paralelo, puerto serie y el controlador de interrupciones entre otros.

TEMA 1. INTRODUCCIÓN

Objetivo particular

El objetivo de este tema es presentar al alumno los conceptos de computadora, microcomputadora, microprocesador, las unidades básicas de una computadora, los tipos de computadoras, una clasificación de computadoras digitales y los elementos que forman parte de un microprocesador con la finalidad de que el alumno entienda qué es y qué no es una computadora digital y cuáles son los elementos básicos para el funcionamiento de la misma.

Al finalizar este tema, el alumno identificará los conceptos de computadora, microcomputadora, microprocesador, las unidades básicas de una computadora, los tipos de computadoras, una clasificación de computadoras digitales y los elementos que forman parte de un microprocesador.

Temario detallado

1.1. Estructura básica de las computadoras
1.2. Organización de un microcomputador (Estructura de von Newman)
1.3. El Microprocesador
    1.3.1. Bus de direcciones
    1.3.2. Bus de datos
    1.3.3. Bus de control
    1.3.4. Unidad de Control
    1.3.5. Unidad lógica aritmética
    1.3.6. Registros

Introducción

En la actualidad, la arquitectura (organización interna) de las Computadoras Digitales constituye un importante tema de estudio y, sin duda alguna, su utilización y su importancia aumentarán en el futuro. En este tema explicaremos la arquitectura básica de una computadora digital, el

funcionamiento de cada uno de sus componentes y la interrelación entre sí de dichos componentes.

1.1 Estructura básica de las computadoras

Al hablar de computadoras hoy en día se está haciendo referencia a máquinas electrónicas-mecánicas, esto es, máquinas cuyas funciones se efectúan utilizando circuitos electrónicos analógicos y/o digitales.

Definición de computadora

Una computadora es una máquina electro-mecánica que procesa información.

Un modelo básico de una computadora consta de los siguientes módulos: Unidad de Entrada, una Unidad de Salida, Unidad de Central de Proceso, Unidad de Memoria y de la Unidad de Control interconectadas por buses unidireccionales y bidireccionales como se muestra en la figura 1.1

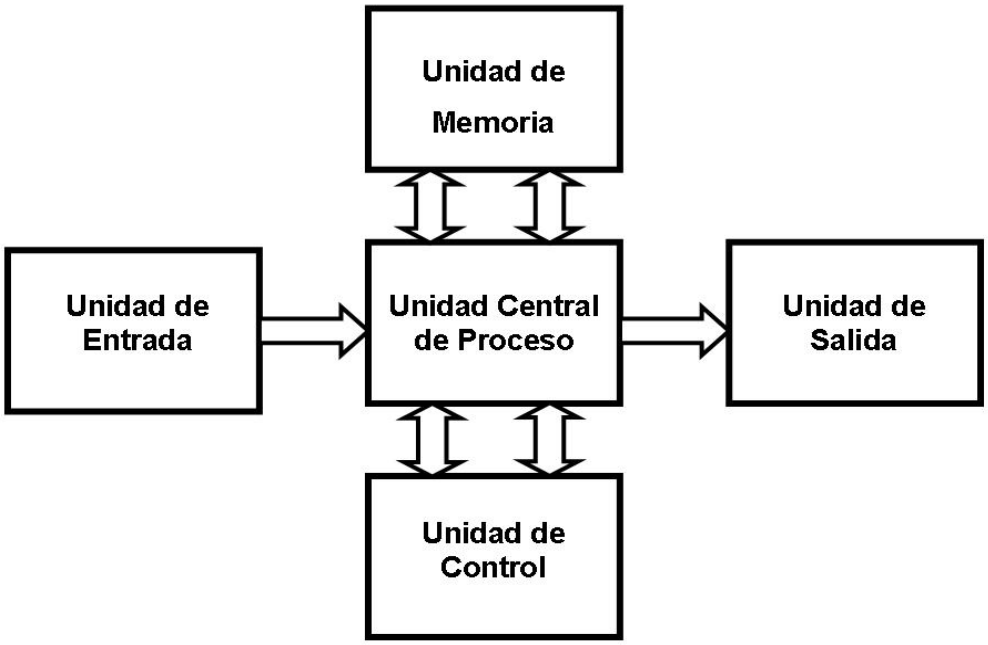


Figura 1.1 Modelo básico de una Computadora

La Unidad de Entrada

La unidad de entrada suministra los datos y las instrucciones a la computadora. Para cada tipo de problema que se requiera resolver, solamente se necesita alimentar a la computadora con un nuevo conjunto de datos e instrucciones.

La Unidad Central de Proceso

Una vez introducidos en la computadora los datos y las instrucciones, esta unidad realiza con ellos diferentes cálculos, por ejemplo: operaciones aritméticas (adición, sustracción, multiplicación y división), operaciones lógicas (OR, AND y NOT) y operaciones de desplazamiento a la izquierda o a la derecha, así como también operaciones de comparación entre otras.

La unidad Central de Proceso puede funcionar a velocidades más altas que la unidad de memoria, a pesar de que tiene que realizar varios pasos para completar una adición. Los resultados de las operaciones aritméticas se introducen de nuevo en la unidad de memoria para su almacenamiento, o estos resultados pueden transferirse posteriormente a una unidad de salida, que puede ser una cinta magnética, disco magnético, impresora o monitor, etc.

Unidad de Memoria

Generalmente, las instrucciones (programa) y los datos se almacenan en la memoria interna del computador (la Unidad de Memoria), la cual se diferencia del dispositivo de entrada (que también puede ser una memoria) por su velocidad de operación. La memoria interna se diseña para almacenar y manipular cantidades relativamente pequeñas de datos, pero a velocidades muy rápidas. La velocidad de la computadora básica está limitada, por la velocidad de la memoria interna.

Unidad de control

Esta unidad controla todas las operaciones de la computadora. Interpreta las instrucciones contenidas en la memoria y dice a las otras unidades dónde han de obtener los datos, dónde han de suministrarlos y qué operaciones han de realizar.

Unidad de salida

La unidad de salida muestra los datos en un Tubo de Rayos Catódicos (TRC) como dispositivo de salida. Conforme el (programador) usuario introduce algún programa vía el teclado, cada carácter se visualiza simultáneamente en el TRC. Además del TRC la computadora cuenta con otros dispositivos de salida como son el diskette, disco duro, memoria USB, cinta magnética, disco duro o imprimirlos en papel.

Tipos de Computadoras

A partir de este modelo básico de computadora se han diseñado y construido dos tipos de computadoras:

- Computadoras analógicas, y
- Computadoras digitales.

Una ***computadora analógica*** trabaja internamente con señales electrónicas continuas o analógicas de tal manera que mide las magnitudes de las cantidades en un circuito eléctrico que se coloca para imitar (en paralelo con, o análogo a) la ecuación del fenómeno físico investigado, la señal de entrada (a procesar) es una señal analógica y la señal de salida es una señal analógica representada en forma gráfica en un Tubo de Rayos Catódicos, ver Fig.1.2.a.

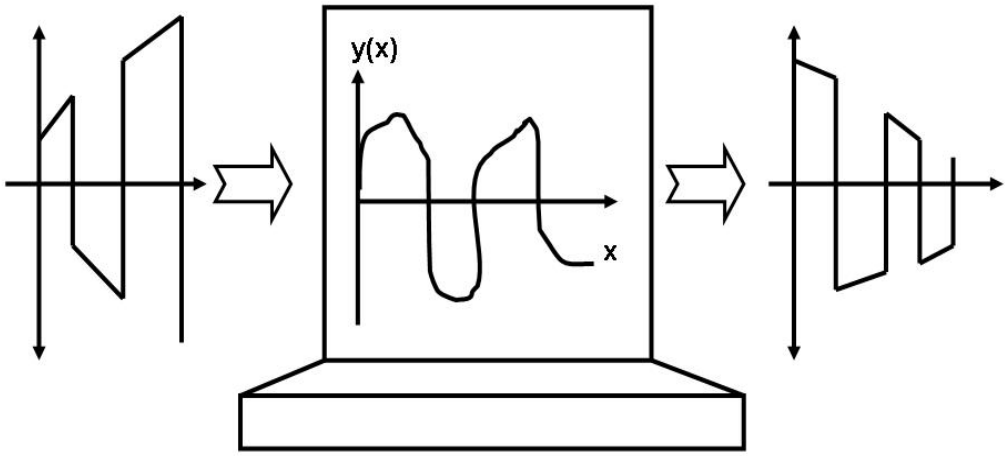


Figura 1.2a Esquema de una Computadora Analógica

El principal inconveniente de las computadoras analógicas es que pueden alcanzar velocidades de resolución superiores a algunos centenares de ciclos por segundo o Hertz. No obstante, esta reducida velocidad no significa, necesariamente, que la computadora analógica sea un dispositivo ineficaz. Además, este tipo de computadoras ocupan un espacio demasiado grande y su costo de mantenimiento es muy alto. La ventaja principal es su exactitud que es mucho mejor que la de una computadora digital.

Una ***computadora digital*** trabaja internamente con señales electrónicas digitales o discretas, la señal de entrada (a procesar) puede ser: una señal analógica, una señal digital o un mensaje de texto, y la señal de salida puede ser una señal analógica, una señal digital y/o un mensaje de texto modificada por un programa específico previamente almacenado en la unidad de Memoria, ver Fig.1.2.b.

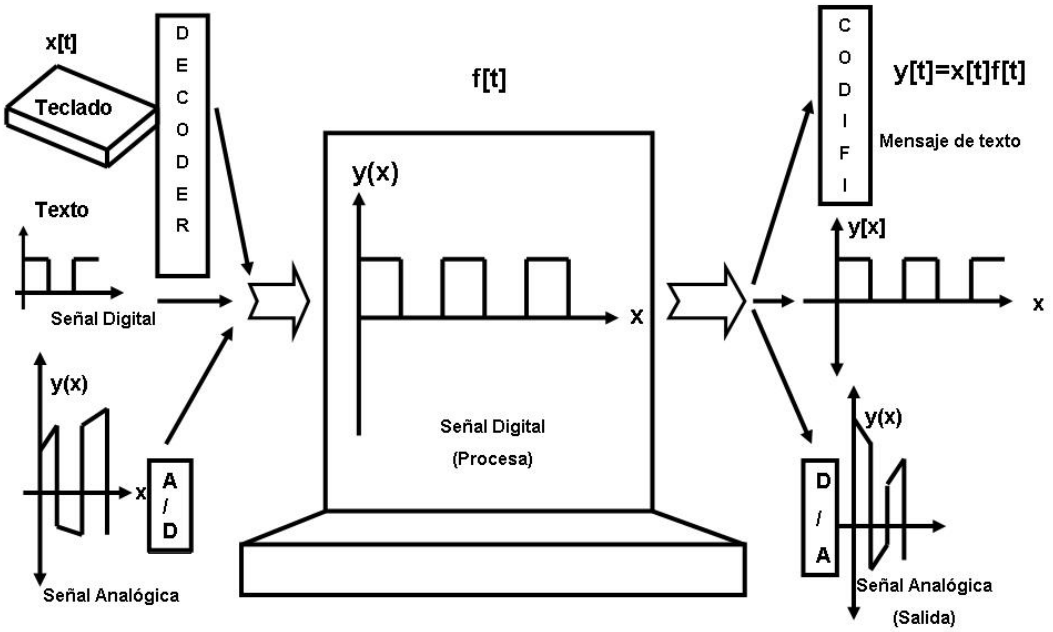


Figura 1.2b Esquema de una Computadora Digital

Las computadoras digitales son más compactas, más baratas, ocupan menos espacio, su costo de mantenimiento es muy bajo y su característica principal es su velocidad. Esta velocidad se logra debido a que están construidas con componentes electrónicos de alta velocidad. Estos componentes se utilizan para formar circuitos que realizan funciones más complejas y que operan con señales discretas (de dos niveles: nivel lógico “1” y el nivel lógico “0”). Dichas computadoras realizan a altas velocidades las operaciones aritméticas y lógicas basadas en el sistema de numeración de base 2 (ver tema 2). Esta característica binaria, permite la utilización del álgebra de Boole en el diseño y construcción de algunos componentes que constituyen una computadora digital.

Las necesidades modernas de computación han llevado a incrementar el uso de combinaciones de computadoras analógicas y digitales, conocidas como **computadoras híbridas.** En este trabajo no se considerarán las computadoras analógicas ni las híbridas, sino únicamente las digitales.

Clasificación de Computadoras digitales

Las computadoras digitales pueden clasificarse en dos categorías:

- Computadoras para propósitos generales, y
- Computadoras para propósitos específicos.

Una ***computadora para propósitos generales*** se diseña de modo que pueda ser programada para resolver una amplia variedad de problemas administrativos, de medicina, de ingeniería, etc. Las computadoras de propósito general pueden estudiar, en unos cuantos minutos, un problema de medicina, hacer una contabilidad financiera, estudiar el diseño de un proyecto de ingeniería o de jugar ajedrez con el operador (el usuario).

Una ***computadora de propósitos específicos*** está diseñada en torno a un problema específico y es optimizada para tratar solamente con ese tipo de problema. Generalmente este tipo de computadora es de menor tamaño, más barata y más eficaz para la realización de esa función específica. Dos ejemplos típicos pueden ser el control de un avión y el control de una videocámara.

Ambos tipos de computadoras (de propósito general y de propósito específico) tienen básicamente la misma estructura. La diferencia reside en las unidades específicas empleadas para introducir los datos en la computadora y para proporcionar la información al exterior.

En este tema y en los temas siguientes haremos referencia a las computadoras digitales de propósito general.

1.2 Organización de un microcomputador (estructura von Newman)

La estructura **von Newman** es el modelo básico de arquitectura usado en la gran mayoría de las computadoras digitales actuales. Las dos principales características de la estructura de von Newman son: el uso del sistema de **numeración binario** y el concepto de "**programa almacenado**". La estructura von Newman está formada por (ver Figura 1.3):

- Unidad Central de Proceso

- Unidad de Memoria
- Unidad de Control
- Unidad de E/S

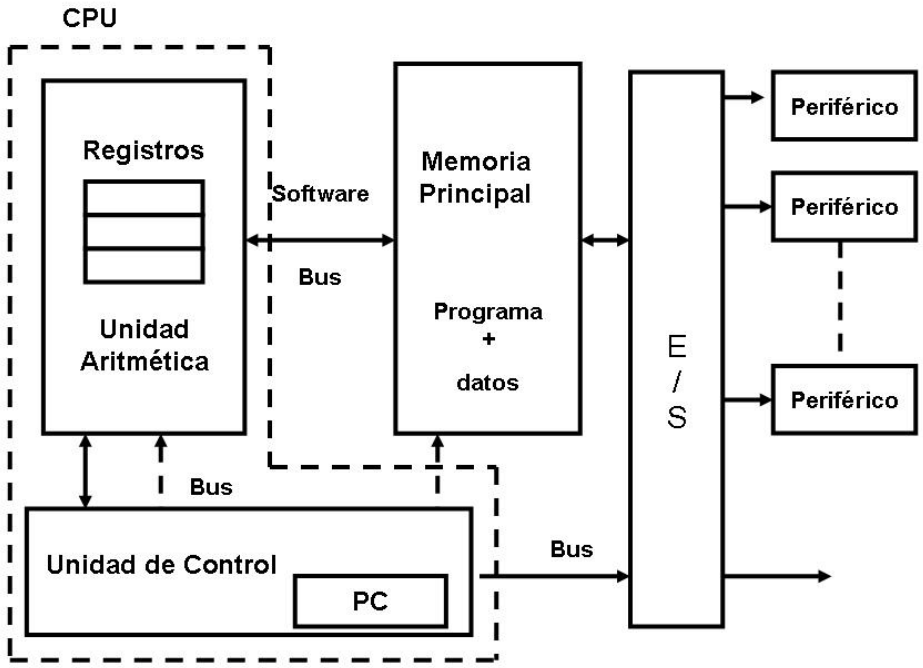


Figura 1.3 Arquitectura Von Newman

La **Unidad Central de Proceso (CPU)** está formada por una unidad Aritmética y un banco de registros y se encarga de realizar operaciones elementales tales como suma, resta, multiplicación, etc.

La **Unidad de Memoria** se encuentra dividida en celdas, las cuales se identifican mediante una dirección. Todas las celdas son de tamaño fijo. Dicha unidad se encarga de almacenar datos e instrucciones (programa).

La **Unidad de Control** se encarga de leer una tras otra las instrucciones máquina almacenadas en memoria principal. Además, genera las señales de control para que la computadora ejecute las instrucciones. Esta unidad contiene un elemento llamado **Contador de Programa** el cual indica la posición de memoria de la siguiente instrucción.

La **Unidad de Entrada y Salida** realiza la transferencia de información con los periféricos. Los periféricos permiten cargar datos y programas en la Memoria Principal y sacar los resultados.

Todas las unidades están conectadas por medio de un **bus** (conjunto de líneas y/o alambres por las cuales se transfiere información de cualquier dispositivo a otro) **unidireccionales** (un sólo sentido) o **bidireccionales** (en ambos sentidos) cuyo objetivo es hacer que las instrucciones, datos y señales de control circulen entre las distintas unidades de la computadora.

La estructura de von Newman utiliza el modelo de "**programa almacenado**" y dicho modelo presenta las siguientes ventajas:

- Se pueden ejecutar diversos programas.
- Tiene gran velocidad de ejecución.
- Se pueden construir programas automodificables, intérpretes, compiladores, etc.

Como se mencionó anteriormente, una de las principales aportaciones de la estructura de von Newman es el concepto de "**programa almacenado**" el cual se explicará a continuación.

Las computadoras con este tipo de estructura resuelven un problema en una operación de dos fases: **compilación** y **ejecución**. Durante la fase de **compilación** se lee una serie de instrucciones introducidas (programa fuente), se traducen a lenguaje de máquina y se almacenan en la memoria principal. Cada instrucción se almacena en una palabra (o varias palabras, según se requiera), como una instrucción única. Durante la fase de **ejecución**, cada instrucción se llama en secuencia desde la unidad de almacenamiento y se retiene temporalmente en el **registro de instrucción** mientras se ejecuta. Esta operación de dos fases, en la cual el programa fuente se traduce y se almacena (compilación) y luego se ejecuta (ejecución) de manera automática y secuencial, se conoce como concepto de **programa almacenado**.

El concepto de programa almacenado permitió la lectura (almacenamiento) de un programa dentro de la memoria de la computadora, y después la ejecución de las instrucciones del mismo sin tener que volverlas a escribir. Una computadora con la capacidad de "*programa almacenado*" podría ser utilizada para varias aplicaciones tan solo cargando y ejecutando el programa apropiado.

1.3 El Microprocesador

El ***microprocesador*** es una Unidad Central de procesos (CPU) implementada en uno o más circuitos integrados utilizando diversas tecnologías (MOS, Bipolar, etc.). La mayoría de los microprocesadores vienen implementados en un solo circuito integrado (C.I.) o "chip" de 40 o más terminales.

Se define como ***microcomputadora*** a la computadora digital en la que la CPU está formada por un microprocesador y sus componentes auxiliares por circuitos integrados que forman la familia del microprocesador (memoria, interfaces, decodificadores, *buffers*, puertos de entrada/salida, etc.). La Figura 1.4 ilustra un diagrama de una computadora digital basada en el microprocesador 8085A.

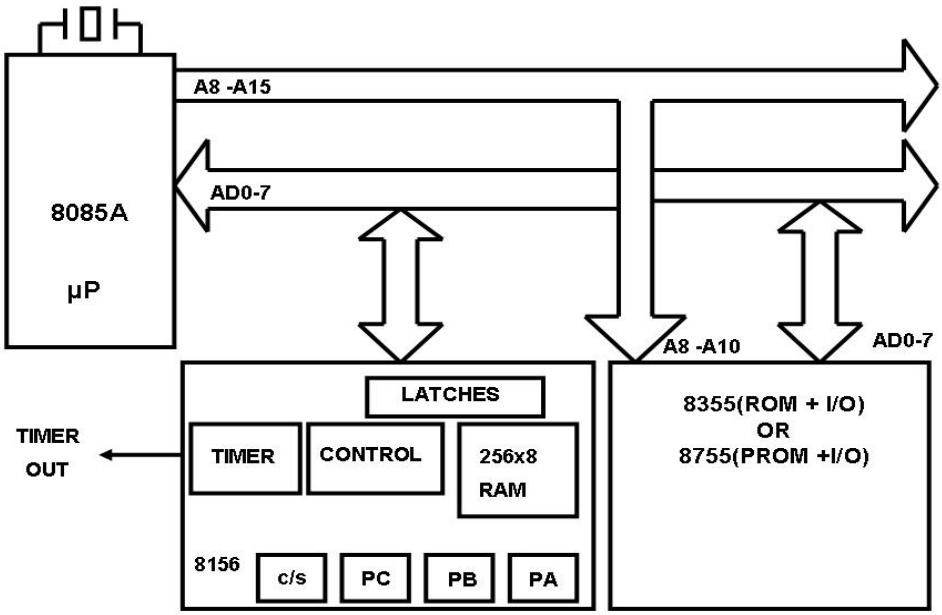


Figura 1.4 Sistema mínimo en base a el microprocesador 8085A

El microprocesador o CPU es la parte principal de la computadora digital y está compuesta por: Buses de Direcciones, Datos y de Control, la Unidad de Control, Unidad Lógica Aritmética (ALU, Arithmetic Logic Unit), un banco de registros, registro de banderas, etc., como lo muestra la figura 1.5, los cuales se explicarán a continuación.

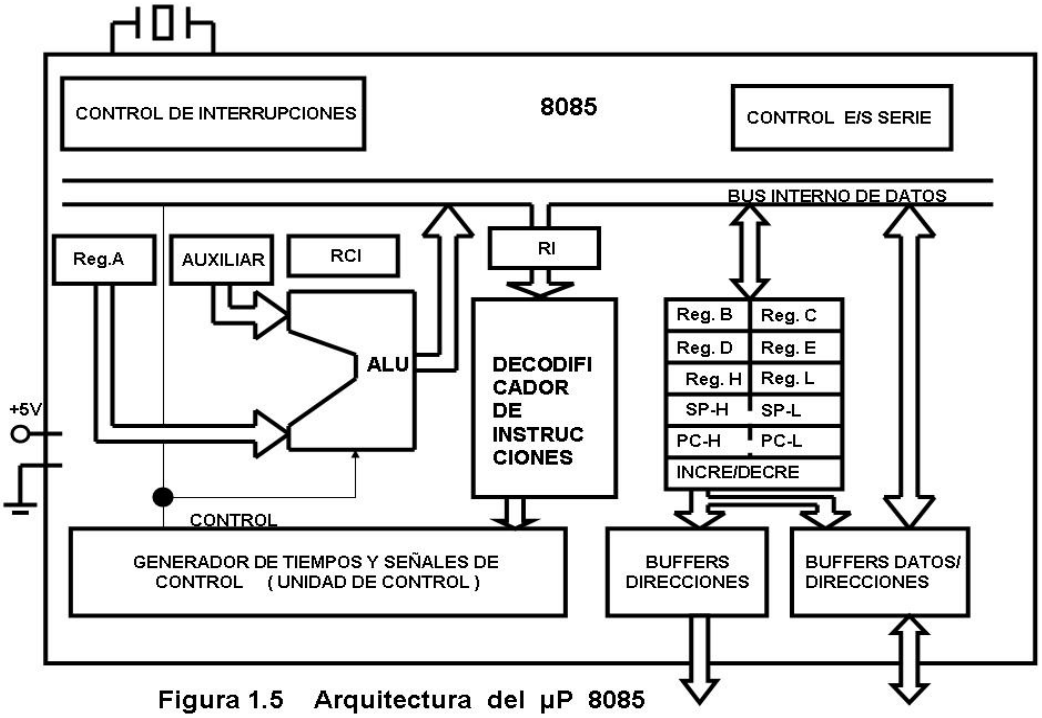


Figura 1.5 Arquitectura del µP 8085

1.3.1. Bus de direcciones

El bus de direcciones es utilizado por el microprocesador para direccionar o llamar a las diversas posiciones de memoria. A través de estas líneas, el microprocesador puede acceder a la información sólo con llevar a las líneas citadas la posición que contiene la palabra a extraer. A continuación se dará la orden correspondiente de lectura, y la palabra en cuestión pasará al interior del microprocesador a través del Bus de Datos.

1.3.2 Bus de datos

La función del Bus de datos es mover o enviar los operandos o los resultados de los cálculos hacia la ALU para ser procesados desde o hacia la unidad de Memoria o hacia un dispositivo de entrada salida (puerto).

1.3.3 Bus de control

El bus de control controla las operaciones de entrada/salida tales como leer una localidad de memoria, escribir hacia una localidad de memoria, leer a partir de un periférico (puerto), escribir hacia un periférico.

1.3.4 Unidad de Control

Todas las acciones dentro de una computadora deben estar sincronizadas y seguir las instrucciones de un programa. La Unidad de Control recibe las instrucciones codificadas en binario desde la memoria, y decide cuándo, cómo y qué operaciones se deben ejecutar para realizar cada instrucción e indica cuál es la que se debe ejecutar a continuación. Se considera a la Unidad de Control como el cerebro de la computadora.

1.3.5 Unidad lógica aritmética

Esta unidad es la que ejecuta realmente el trabajo de procesamiento aritmético y lógico. La ALU puede recibir datos y efectuar con ellos operaciones aritméticas, lógicas, de comparación y desplazamiento, entre otras. La ALU tiene dos registros asociados a ella para almacenar los datos y sobre los que va a realizar las operaciones: El ***registro acumulador*** y el ***registro auxiliar***, cada uno de ellos de 8 bits. El registro principal de las ALU es el ***registro Acumulador***, debido que al comienzo de una operación, el acumulador contiene uno de los operandos y al final de la operación, contiene el resultado.

1.3.6 Registros (Banco de registros)

La CPU o el microprocesador cuenta con dos tipos de registros: ***registros de propósito general*** y los ***registros de propósito especial***. Los registros de propósito general se utilizan para el manejo de los datos, y los registros de propósito especial tienen funciones ya definidas.

Registros de propósito general

El banco de registros de propósito general está formado por los registros: **B, C, D, E, H** y **L** los cuales son de longitud de 8 bits. Estos registros tienen la característica que se pueden unir por pares para formar registros de 16 bits llamados: **BC, DE** y **HL**. El registro **W-Z** (o INCRE/DECRE) no es accesible por programa y tan sólo se utiliza por la unidad de control para la ejecución interna de instrucciones.

Registros de propósito especial

Los registros de propósito especial son: El **Stack Pointer**, el **Registro de Banderas**, **Contador de Instrucción**, **Registro de Instrucción**, etc. y sus funciones son:

- El **registro Stack Pointer** permite la gestión de interrupciones y subrutinas.

- El **registro de banderas** tiene información del resultado de la última operación. A la información contenida se le da el nombre de bandera y entre las cuales se pueden mencionar las banderas de Cero, Paridad, Signo, Acarreo, etc.

- El **contador de Instrucciones** apunta a la próxima instrucción que se debe ejecutar.

- El **registro de Instrucción** almacena cada instrucción conforme se llama a partir de la Unidad de Memoria e inicia su ejecución.

Bibliografía del Tema 1

Intel, *Microsystem Components Handbook*, EE.UU., Intel Corporation. Literature Department, Vol. 1 y 2, 1985.

Motorola, *MC68340 Integrated processor User´s Manual*, EE.UU., Motorola Incorporation. Literature Department, Vol. 1, 1990.

García Narcia, O., *8085A e Interfaces,* México, IPN, 1982.

McGlynnn, R., Daniel, *Modern Microprocessor System Design, EE.UU.,* John Wiley & Sons, 1980.

Torres Portero Manuel, *Micro y Microcontroladores Aplicados a la Industria*, Madrid, Paraninfo, 1989.

Hayes P. John, *Diseño de Sistemas Digitales y Microprocesadores*, México, McGraw-Hill, 1986.

Lenk, D. John, *Handbook of Microprocessors, Microcomputers, and Minicomputers*, EE.UU., Prentice-Hall, INC., Englewood Cliffs, N.J., 1979.

Actividades de Aprendizaje

A.1.1 Realizar una tabla comparativa de diversos microcomputadores de 8, 16, 32, y 64 bits, tomando en cuenta, fabricante, velocidad de reloj, número de registros, tipos de direccionamiento y número de instrucciones.

A.1.2 Realizar una tabla comparativa de diversos tipos memorias RAM tomando en cuenta, fabricante, número de bits, número máximo de localidades y tiempo de acceso.

A.1.3 Realizar una tabla comparativa de diversos tipos de memorias ROM tomando en cuenta, fabricante, número de bits, número máximo de calidades y tiempo de acceso.

A.1.4 Realizar una tabla comparativa de diversos puertos paralelos y puertos serie considerando: fabricante, número de puertos, número de bits por puerto y tiempo de acceso.

A.1.5 Diseña en bloques una microcomputadora básica de 32 bits.

A.1.6 Diseña en bloques una microcomputadora básica de 64 bits.

Cuestionario de autoevaluación

1. ¿Qué es una computadora?
2. ¿Cuáles son los tipos de computadoras?
3. ¿Cómo se clasifican las computadoras digitales?
4. ¿Qué es un registro?
5 ¿Qué operaciones realiza la Unidad Lógica Aritmética?
6. ¿Cuáles son las unidades de una computadora digital?
7. ¿Cuáles son las aportaciones de la arquitectura von Newman?
8. ¿Qué es un Bus?
9. ¿Cuál es la función del Stack Pointer?
10. ¿Cuál es la función del registro de Banderas?

Examen de autoevaluación

Relaciona las columnas escribiendo dentro del paréntesis el número que una los conceptos con sus respectivas definiciones.

1.) Estructura von Newman		( ) Trabaja internamente con señales electrónicas digitales ó discretas
2.) Computadora Digital		( ) Permite la gestión de interrupciones y subrutinas.
3.) Microcomputadora		( ) Programa Almacenado
4.) Computadora Analógica		( ) Stack Pointer, Registro de Banderas, Contador de Instrucción, Registro de Instrucción.
5.) Registro de Banderas		( ) Indica la posición de memoria de la siguiente instrucción.
8.) Contador de Programa		( ) Unidad que ejecuta el trabajo de procesamiento aritmético y lógico
6.) Registro Stack Pointer		( ) Computadora digital en la que el CPU está formada por un microprocesador y sus componentes auxiliares por circuitos integrados
7.) Microprocesador		( ) Informa del resultado de la última operación
9.) Unidad lógica-Aritmética		( ) Trabaja internamente con señales electrónicas continuas o analógicas
10.) Registros de propósito especial		( ) Unidad Central de procesos (CPU) implementada en uno o más circuitos integrados

TEMA 2. SISTEMAS DE NUMERACIÓN

Objetivo Particular

El alumno identificará los diferentes sistemas de numeración posicional, las conversiones entre diferentes sistemas, así como las operaciones aritméticas básicas en cada uno de los sistemas de numeración.

Temario detallado

2.1. Conversión entre bases

- 2.1.1. Sistema decimal
- 2.1.2. Sistema binario
- 2.1.3. Sistema octal
- 2.1.4. Sistema hexadecimal
- 2.1.5. Sistema de base “n”

2.2. Aritmética binaria

- 2.2.1. Operaciones aritméticas con números en diferentes bases
- 2.2.2. Complemento a la base y a la base disminuida
- 2.2.3. Representación de números con signo
- 2.2.4. Operaciones aritméticas con números asignados

Introducción

La característica fundamental de una computadora digital es su habilidad para representar, almacenar y procesar información. La información se puede representar en diferentes sistemas numéricos. El sistema decimal es el más ampliamente usado en nuestra civilización. No obstante, no es razonable esperar que este sistema sea el más eficiente para la construcción y el funcionamiento de las computadoras. El sistema binario ha probado ser el más eficiente para utilizarlo en las computadoras digitales.

En el mundo digital la información se representa mediante “unos” y “ceros” (binario), los cuales pueden ser procesados fácilmente mediante dispositivos bi-estables, tales como flip-flops, registros, contadores, etc. Por esto es fundamental conocer el sistema binario y su relación con otros sistemas de numeración como el decimal, el octal y el hexadecimal.

2.1. Conversión entre bases

La forma más comúnmente usada para realizar la conversión entre diferentes bases es utilizando el sistema posicional. En el **sistema posicional**, el valor significativo asignado a cada dígito es una cantidad que está en función a su posición.

En el sistema posicional, un número **N** se representa en cualquier base **n** por la ecuación (2.1)

$$N = d_{p-1}n^{p-1} + d_{p-2}n^{p-2} + \cdots + d_0 n^0 + d_{-(q-1)}n^{-(q-1)} + d_{-(q-2)}n^{-(q-2)}$$

o en su forma compacta

$$N = \sum_{i=0}^{p-1} d_i n^i + \sum_{j=1}^{q} d_j n^{-(i)}$$

donde

d son los dígitos (coeficientes) del número

n la base del sistema

p el número de dígitos enteros

q el número de dígitos fraccionarios

En un número cualquiera, al dígito entero que se encuentre más a la derecha se le da el nombre de “***menos significativo***” y el que se encuentra más a la izquierda el de “***más significativo”***. En los dígitos fraccionales esta consideración sigue siendo válida.

Finalmente, la tabla 2.1 nos presenta una forma de relacionar el sistema posicional en cualquier base “n”, donde n = 2, 8, 10 y 16.

Decimal	Binario	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Tabla 2.1 Equivalencias entre diferentes sistemas de Numeración

Entre los sistemas de numeración más utilizados se encuentran los sistemas de numeración Decimal, Binario, Octal y Hexadecimal los cuales explicaremos a continuación.

2.1.1 Sistema Decimal

El sistema decimal emplea diez diferentes dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9). Por esto se dice que la “base” del sistema decimal es diez. Para representar números mayores a 9, se combinan dos o más dígitos base, y cada uno de éstos tendrá un valor según la posición que ocupe. El sistema decimal se

representa en forma posicional por medio de la ecuación (2.1), con **n** = 10 y donde **d** puede representar cualquier dígito entre 0 y 9.

Ejemplo Representar el número $(425)_{10}$ en forma posicional.

Solución Utilizando la ecuación (2.1) con 3 dígitos enteros (p = 3) y 0 dígitos fraccionarios (q = 0).

$$425 = \sum_{i=0}^{3-1} d_i 10^i + \sum_{j=1}^{0} d_j 10^{-(i)}$$
$$= [d_0 10^0 + d_1 10^1 + d_2 10^2]$$
$$= [5x10^0 + 2x10^1 + 4x10^2]$$
$$= [5x1 + 2x10 + 4x100] = [5 + 20 + 400] = 425$$

Ejemplo Representar el número $(3637.25)_{10}$ en forma posicional

Solución Utilizando la ecuación (2.1) con 4 dígitos enteros (p = 4) y 2 dígitos fraccionarios (q = 2).

$$3637.25 = \sum_{i=0}^{4-1} d_i 10^i + \sum_{j=1}^{2} d_j 10^{-(i)}$$
$$= [d_0 10^0 + d_1 10^1 + d_2 10^2 + d_3 10^3] + [d_{-1} 10^{-1} + d_{-2} 10^{-2}]$$
$$= [7x10^0 + 3x10^1 + 6x10^2 + 3x10^3] + [2x10^{-1} + 5x10^{-2}]$$
$$= [7x1 + 3x10 + 6x100 + 3x1000] + [2/10 + 5/100]$$
$$= [7 + 30 + 600 + 3000] + [0.2 + 0.05] = 3637.25$$

Conversión de decimal a binario

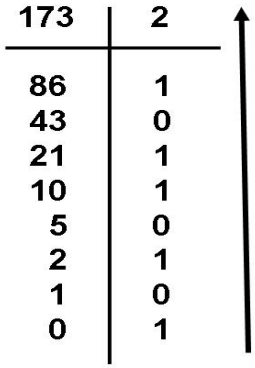
El método utilizado para convertir un número decimal a binario es el método de divisiones sucesivas. Este método consiste en los pasos siguientes:

1. Dividir el número decimal entre 2
2. El residuo (uno o cero) es el dígito menos significativo, el cual se almacena en un arreglo unidimensional.
3. Dividir entre 2 el cociente de la división anterior, pero ahora el residuo se coloca en la siguiente posición de más significación.

4. Repetir el paso anterior y el residuo se coloca en la siguiente posición de más significativo (valor posicional).
5. Repetir el paso anterior hasta obtener un cociente de cero.
6. Los números en el arreglo unidimensional se muestran de abajo hacia arriba.

Ejemplo Convertir a binario el número $(173)_{10}$ a base 2

Solución



finalmente el número $(173)_{10} = (10101101)_{2.}$

Ejemplo Convertir a binario el número $(3129)_{10}$ a base 2

Solución

3129	2
1564	1
782	0
391	0
195	1
97	1
48	1
24	0
12	0
6	0
3	0
1	1
0	1

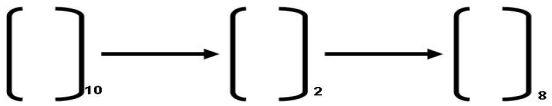
Por lo tanto $(3129)_{10} = (110000111001)_2$

Conversión de decimal a octal

Para realizar la conversión de base 10 a base 8 se tienen dos métodos.

Primer Método

Este método consiste en convertir el número decimal a número binario y luego de binario a base octal. La conversión de base 10 a base 2 se realiza por el método de divisiones sucesivas y luego el resultado lo convertimos a base 8, es decir:



Ejemplo Convertir el número $(153)_{10}$ a base $()_8$

Solución

Para este ejemplo, convertimos el número $(153)_{10}$ a base 2 utilizando el método de divisiones sucesivas y posteriormente realizamos la conversión de base 2 a base 8 utilizando la tabla 2.1.

$(153)_{10}$ ---------- $(010 \quad 011 \quad 001)_2$ ----------$(2\ 3\ 1)_8$

Segundo Método: Método de las divisiones sucesivas

Este método consiste en dividir el número decimal entre 8 hasta que el cociente sea igual a cero.

Ejemplo Convertir el número $(75658)_{10}$ a base $()_8$

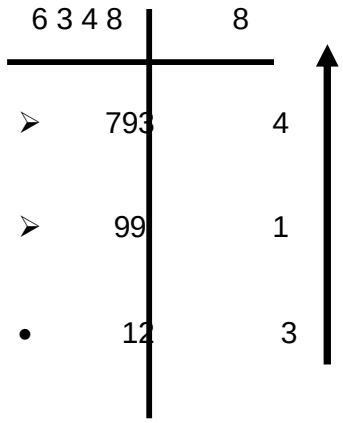
Solución

75658	8
9457	2
1182	1
147	6
18	3

por lo tanto $(75658)_{10} = (223612)_8$

Ejemplo Convertir el número $(6348)_{10}$ a $()_8$

Solución

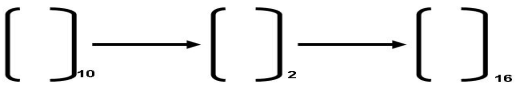


Finalmente obtenemos la conversión deseada $(6348)_{10} = (14314)_8$.

Conversión de base decimal a base hexadecimal

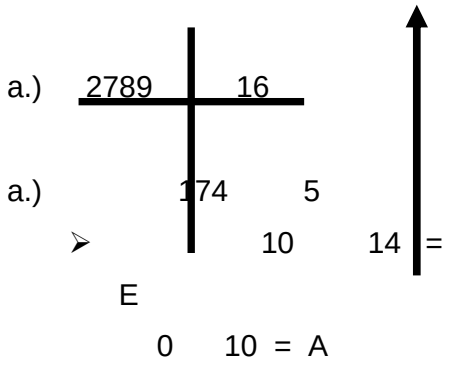
Para realizar la conversión de base 10 a base 16 se tienen los mismos métodos que el inciso anterior.

El primer método consiste en convertir el número en base 10 a base 2 y luego de base 2 a base 16, es decir:



Ejemplo Convertir el número $(2789)_{10}$ a base $()_{16}$

Solución

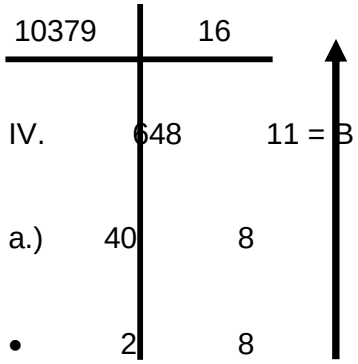


Por lo tanto $(2789)_{10} = (AE5)_{16}$.

El segundo método, es el método de las divisiones que se utilizó en la conversión decimal a binario, pero dividiendo entre 16.

Ejemplo Convertir el número $(10379)_{10}$ a base $(\ )_{16}$

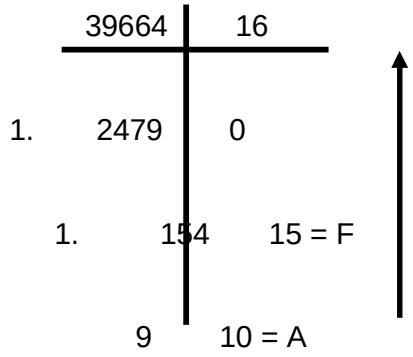
Solución



Por lo tanto $(10379)_{10}$ = (288B) 16

Ejemplo Convertir el número $(39664)_{10}$ a base $(\ )_{16}$

Solución



Por lo tanto $(39664)_{10} = (9AF0)_{16}$

2.1.2 Sistema Binario

El sistema binario emplea sólo dos dígitos base (0 y 1) para representar un número, su base es 2. Para representar números mayores a 1, se combinan dos o más dígitos base, y cada uno de éstos tendrá un valor según la posición que ocupe. El sistema binario se representa en forma posicional por medio de la ecuación (2.1), con **n** = 2 y **d** puede representar solo los números 0 y 1.

Ejemplo Representar el número $(1010)_2$ en forma posicional

Solución El $(1010)_2$ tiene 4 dígitos enteros (p = 4) y 0 dígitos fraccionarios (q = 0) y a partir de la ecuación (2.1) la forma posicional de dicho número es la siguiente:

$$(1010)_2 = \sum_{i=0}^{4-1} d_i 2^i + \sum_{j=1}^{0} d_{-j} 2^{-(i)}$$
$$= d_0 2^0 + d_1 2^1 + d_2 2^2 + d_3 2^3 + d_4 2^4$$
$$= 0x2^0 + 1x2^1 + 0x2^2 + 1x2^3$$
$$= 0x1 + 1x2 + 0x4 + 1x8 = 0 + 2 + 0 + 8 = 10$$

el cual es equivalente en el sistema decimal a $(10)_{10}$.

Ejemplo Representar el número $(10111.101)_2$ en forma posicional.

Solución Utilizando la ecuación (2.1) la forma posicional de dicho número con 5 dígitos enteros (p = 5) y 3 dígitos decimales (q=3) es la siguiente:

$$(10111.101)_2 = \sum_{i=0}^{5-1} d_i 2^i + \sum_{j=1}^{3} d_{-j} 2^{-(i)}$$

$$= d_0 2^0 + d_1 2^1 + d_2 2^2 + d_3 2^3 + d_4 2^4 + d_{-1} 2^{-1} + d_{-2} 2^{-2} + d_{-3} 2^{-3}$$

$$= 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$$

$$= 1x1 + 1x2 + 1x4 + 0x8 + 1x16 + 1/2 + 0/4 + 1/8$$

$$= 1 + 2 + 4 + 0 + 16 + 0.5 + 0 + 0.125$$

El cual es equivalente en el sistema decimal a el número $(23.625)_{10}$.

Conversión de binario a decimal

Para convertir un número binario (base 2) a decimal (base 10) se utiliza la ecuación general (2.1).

Ejemplo Convertir el número $(11011)_2$ a decimal

Solución

$$11011 = 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$$

$$= 16 + 8 + 0 + 2 + 1 = 27$$

Por lo tanto el número $(11011)_2 = (27)_{10}$.

Conversión de binario a octal

Para convertir un número binario (base 2) a octal (base 8) se forman bloques de bits de 3 bits cada uno, tanto a la derecha como a la izquierda del punto decimal y a continuación se sustituye cada bloque por su equivalente en base 8 utilizando la tabla 2.1. En caso de que falten bits para formar cada bloque, se le agregan tanto ceros como sea necesario.

Ejemplo Convertir el número $(1110101)_2$ a octal

Solución

Se forman los bloques de 3 bits cada uno, a partir del punto decimal

$$\underbrace{1}\quad\underbrace{110}\quad\underbrace{101}$$

En el tercer bloque faltan 2 bits se completa con ceros.

$$\underbrace{001}\quad\underbrace{110}\quad\underbrace{101}$$

Se sustituye cada uno de los bloques por su equivalente en base 8 utilizando la tabla 2.1

$$\underbrace{001}\quad\underbrace{110}\quad\underbrace{101}$$

$$1\qquad 6\qquad 5$$

Por lo tanto el número $(1110101)_2 = (165)_8$.

Conversión de binario a hexadecimal

Para convertir un número binario (base 2) a hexadecimal (base 16) se forman bloques de bits de 4 bits cada uno, tanto a la derecha como a la izquierda del punto decimal y a continuación se sustituye cada bloque por su equivalente en base 16 utilizando la tabla 2.1. En caso de que falten bits para formar cada bloque, se le agregan tanto ceros como sea necesario.

Ejemplo Convertir el número $(1110101011)_2$ a hexadecimal

Solución

Se forman los bloques de 4 bits cada a partir del punto decimal

$$\underbrace{11}\quad\underbrace{1010}\quad\underbrace{1011}$$

En el tercer bloque faltan 2 bits se completa con ceros.

$$\underbrace{0011}\quad\underbrace{1010}\quad\underbrace{1011}$$

Se sustituye cada uno de los bloques por su equivalente en base 16 utilizando la tabla 2.1

$$\underbrace{0011}_{\mathbf{3}}\ \underbrace{1010}_{\mathbf{A}}\ \underbrace{1011}_{\mathbf{B}}$$

Por lo tanto el número $(1110101011)_2 = (3AB)_{16}$.

2.1.3 Sistema Octal

El sistema octal emplea 8 dígitos base (0, 1, 2, 3, 4, 5, 6 y 7) para representar un número, su base es 8 lo cual es potencia de 2 por lo que la conversión a la base binaria es directa. Para representar números mayores a 7, se combinan dos o más dígitos base, y cada uno de éstos tendrá un valor según la posición que ocupe. El sistema octal también se puede representar en forma posicional por medio de la ecuación (2.1), con **n** = 8 y **d** puede representar los dígitos del 0 al 7.

Ejemplo Representar el número $(7410)_8$ en forma posicional

Solución Utilizando la ecuación (2.1) la forma posicional de dicho número es la siguiente:

$$(7410)_8 = \sum_{i=0}^{4-1} d_i 8^i + \sum_{j=1}^{0} d_{-j} 8^{-(i)}$$
$$= d_0 8^0 + d_1 8^1 + d_2 8^2 + d_3 8^3$$
$$= 0x8^0 + 1x8^1 + 4x8^2 + 7x8^3$$
$$= 0x1 + 1x8 + 4x64 + 7x512 = 0 + 8 + 256 + 3584 = 3848$$

que equivale al número decimal $(3848)_{10}$, y

al número binario = $(111\ \ 100\ \ 001\ \ 000)_2$ = (7 4 1 0)$_8$

Ejemplo Represente el número octal 4725.451 en forma posicional

Solución Utilizando la ecuación (2.1) con p = 4 dígitos enteros y q = 3 dígitos fraccionarios, la forma posicional de dicho número es la siguiente:

$$
\begin{aligned}
4725.451 &= \sum_{i=0}^{3} d_i 8^i + \sum_{j=1}^{3} d_j 8^{-(i)} \\
&= [d_0 8^0 + d_1 8^1 + d_2 8^2 + d_3 8^3] + [d_{-1} 8^{-1} + d_{-2} 8^{-2} + d_{-3} 8^{-3}] \\
&= [5x8^0 + 2x8^1 + 7x8^2 + 4x8^3] + [4x8^{-1} + 5x8^{-2} + 1x8^{-3}] \\
&= [5x1 + 2x8 + 7x64 + 4x512] + [4/8 + 5/64 + 1/512] \\
&= [5 + 16 + 448 + 2048] + [0.5 + 0.78125 + 0.001953125] \\
&= 2517 + 1.283203125 \\
&= 2518.283293125
\end{aligned}
$$

que equivale al número decimal $(2518.283293125)_{10}$.

Conversión de octal a decimal

Para convertir un número octal a base 10 se puede realizar utilizando la ecuación 2.1

Ejemplo Convertir el número $(254)_8$ a base 10

Solución

$$(254)_8 = 2x8^2 + 5x8^1 + 4x8^0 = 128 + 40 + 4$$

por lo tanto $(254)_8$ = $(172)_{10}$

Conversión de octal a binario

Debido a que la base 8 y la base 2 están relacionadas ($8 = 2^3$), la conversión al sistema binario es directa. El procedimiento es reemplazar cada dígito octal por sus tres dígitos binarios equivalentes utilizando la tabla 2.1.

Ejemplo Convertir el número $(567)_8$ a binario (base 2)

Solución

A partir de la tabla 2.1 vemos que el número $(567)_8$ está compuesto por

$(5)_8 = (101)_2$ $(6)_8 = (110)_2$ $(7)_8 = (111)_2$

que al realizar la conversión tenemos lo siguiente:

$$(567)_8 = (101\ \ 110\ \ 111)_2$$

Conversión de octal a hexadecimal

La conversión de octal a hexadecimal consiste en

- Pasar cada uno de los dígitos que forman el número a base 2.
- Formar bloques de 4 bits cada uno, tanto a la derecha como a la izquierda del punto decimal.
- Sustituir cada uno de los bloques por su equivalente en base 16 utilizando la tabla 2.1.

Ejemplo Convertir el número $(557)_8$ a hexadecimal (base 16)

Solución

A partir de la tabla 2.1 vemos que el número $(557)_8$ está compuesto por

$(5)_8 = (101)_2$ $(5)_8 = (101)_2$ $(7)_8 = (111)_2$

A continuación se divide el número en bloques de 4 bits cada uno

1 0110 1111

Sustituyendo cada uno de los bloques por su equivalente en base 16 utilizando la tabla 2.1.

$$\underbrace{0001}_{\mathbf{1}}\ \underbrace{0110}_{\mathbf{6}}\ \underbrace{1111}_{\mathbf{F}}$$

$$(557)_8 = (16F)_{16}$$

2.1.4 Sistema Hexadecimal

El sistema hexadecimal emplea 16 dígitos y como solo se dispone de 10 dígitos decimales se recurre al empleo de letras para representar los seis dígitos faltantes. Los dígitos empleados por el sistema hexadecimal son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. La base de este sistema es 16 (**n** = 16) y este sistema también es potencia de 2 por lo que la conversión a la base binaria es directa.

Ejemplo Representar el número hexadecimal $(C2AB)_{16}$ en forma posicional.

Solución A partir de la ecuación (2.1) la forma posicional de dicho número es la siguiente:

$$(C2AB)_{16} = \sum_{i=0}^{4-1} d_i 16^i + \sum_{j=1}^{0} d_{-j} 16^{-(i)}$$

$$= d_0 16^0 + d_1 16^1 + d_2 16^2 + d_3 16^3$$

$$= Bx16^0 + Ax16^1 + 2x16^2 + Cx16^3$$

$$= 11x1 + 10x16 + 2x256 + 12x4096 = 11 + 160 + 512 + 49152 = 49835$$

que equivale al número decimal = 12x4096 + 2x256 + 10x16 + 11 = $(49835)_{10}$

y en el sistema de numeración binario = $(1100\ \ 0010\ \ 1010\ \ 1011)_2$

Ejemplo Representar el número hexadecimal $(AF145.AB1)_{16}$ en forma posicional.

Solución Utilizando la ecuación (2.1), se observa que el número tiene 5 dígitos enteros (p=5) y 3 dígitos fraccionarios (q=3). Por lo tanto tenemos que:

$$AF145.AB1 = \sum_{i=0}^{5-1} d_i 16^i + \sum_{j=1}^{3} d_j 16^{-(i)}$$

$$= [d_0 16^0 + d_1 16^1 + d_2 16^2 + d_3 16^3 + d_4 16^4] + [d_{-1} 16^{-1} + d_{-2} 16^{-2} + d_{-3} 16^{-3}]$$

$$= [5x16^0 + 4x16^1 + 1x16^2 + Fx16^3 + Ax16^4] + [Ax16^{-1} + Bx16^{-2} + 1x16^{-3}]$$

$$= [5x16^0 + 4x16^1 + 1x16^2 + 15x16^3 + 10x16^4] + [10x16^{-1} + 11x16^{-2} + 1x16^{-3}]$$

$$= [5x1 + 4x16 + 1x256 + 15x4096 + 10x65536] + [10/16 + 11/256 + 1/4096] =$$

$$= [5 + 64 + 256 + 61440 + 655360] + [0.625 + 0.04296875 + 0.000244140625]$$

$$= (717126.814140625)_{10}$$

Conversión de hexadecimal a decimal

Para convertir un número de base hexadecimal a base 10 podemos utilizar la ecuación 2.1

Ejemplo Convertir a el número $(A2E4)_{16}$ a base 10

Solución

$$(A2E4)_{16} = Ax16^3 + 2x16^2 + Ex16^1 + 4x16^0 =$$

$$= 10x16^3 + 2x16^2 + 14x16^1 + 4x16^0$$

$$= 40960 + 512 + 224 + 4 = (41700)_{10}$$

finalmente tenemos que el número $(A2E4)_{16}$ = $(41700)_{10}$

Conversión de hexadecimal a binario

La conversión de hexadecimal a binario es directa, debido a que ambas bases están relacionadas ($16 = 2^4$). El procedimiento es reemplazar cada dígito hexadecimal por sus cuatro dígitos binarios equivalentes con el apoyo de la tabla 2.1

Ejemplo Convertir el número $(48A)_{16}$ a base $()_2$

Solución

A partir de la tabla 2.1 tenemos lo siguiente

$(4)_{16} = (0100)_2$ $(8)_{16} = (1000)_2$ $(A)_{16} = (1010)_2$

finalmente obtenemos la conversión deseada

$$(48A)_{16} = (010010001010)_2$$

Conversión de hexadecimal a octal

La conversión de hexadecimal a octal consiste de los pasos siguientes:

- Cada dígito en hexadecimal se sustituye por sus equivalentes 4 bits binarios, utilizando la tabla 2.1.
- Se divide el número en bloques de 3 dígitos hacia la derecha como a la izquierda a partir del punto decimal.
- Se sustituye cada uno de los bloques por su equivalente en base 8 utilizando la tabla 2.1.

Ejemplo Convertir el número $(48A)_{16}$ a base $()_8$

Solución

A partir de la tabla 2.1 tenemos lo siguiente

$(4)_{16} = (0100)_2$ $(8)_{16} = (1000)_2$ $(A)_{16} = (1010)_2$

Se divide el número en bloques de 3 dígitos a partir del punto decimal

$$010 \quad 010 \quad 001 \quad 010$$

Se sustituye cada uno de los bloques formados por su equivalente en base 8 utilizando la tabla 2.1

$$\underbrace{010}_{2} \quad \underbrace{010}_{2} \quad \underbrace{001}_{1} \quad \underbrace{010}_{2}$$

Finalmente, el resultado de la conversión es

$$(48A)_{16} = (2212)_8$$

Algoritmo para la conversión de números decimales a otra base (2,8 y 16)

La conversión de números decimales a otra base (por ejemplo, base 2, 8 ó 16) se puede realizar por el método de multiplicaciones sucesivas por la base. Esto método consiste en los pasos siguientes:

1. Multiplicar el número decimal por la base a la que de desea convertir.
2. Dividir el resultado en su parte fraccionaria ($\mathbf{f_i}$) y en su parte entera ($\mathbf{d_{-i}}$),
3. Multiplicar la parte $\mathbf{f_i}$ por la base a convertir.
   La parte fraccionaria del resultado es $\mathbf{f_2}$ y la parte entera es $\mathbf{d_{-2}}$.
4. Repetir el proceso hasta que $\mathbf{f_m}$ es cero o hasta que se considere que la conversión es lo suficientemente exacta.

Esto se podrá entender con una serie de ejemplos que a continuación se presentan.

Ejemplo Convertir el número $(0.4375)_{10}$ a binario

Solución

$0.4375 \times 2 = 0.8750 \quad = 0.0875 + 0 \quad d_{-1} = 0$

$0.8750 \times 2 = 1.7500 \quad = 0.7500 + 1 \quad d_{-2} = 1$

$0.7500 \times 2 = 1.5000 \quad = 0.5000 + 1 \quad d_{-3} = 1$

$0.5000 \times 2 = 1.0000 \quad = 0.0000 + 1 \quad d_{-4} = 1$

finalmente el número $(0.4375)_{10} = (0.0111)_2$

Ejemplo Convertir el número $(0.6328125)_{10}$ a base 8

Solución

$0.6328125 \times 8 = 5.0625 = 0.0625 + 5 \qquad d_{-1} = 5$

$0.0625 \times 8 = 0.5000 = 0.5000 + 0 \qquad d_{-2} = 0$

$0.5000 \times 8 = 4.0000 = 0.0000 + 4 \qquad d_{-3} = 4$

finalmente el número $(06328125)_{10} = (504)_8$

Ejemplo Convertir el número $(0.6328125)_{10}$ a base 16

Solución

$0.6328125 \times 16 = 10.1250 = 0.1250 + 10 \qquad d_{-1} = 10 = A$ (en base 16)

$0.1250 \times 16 = 2.0000 = 0.0000 + 2 \qquad d_{-2} = 2$

finalmente $(06328125)_{10} = (A2)_{16}$

Algoritmo para la conversión de fracciones de cualquier base (2,8 y 16) a base decimal

La conversión de fracciones de una base **b** a decimal se puede realizar por el método de división. Este método se puede describir como sigue:

1. Dividir el dígito menos significativo por la base **b**. El cociente es $\mathbf{M_1}$.
2. Sumar el cociente $\mathbf{M_1}$ con el dígito que sigue en significación y dividir por la base. El cociente es $\mathbf{M_2}$.
3. Continuar el proceso hasta que se suma el dígito fraccional más significativo y se divide por la base. El último cociente es $\mathbf{M_n}$.

Esto se podrá entender con una serie de ejemplos que a continuación se presentan.

Ejemplo Convertir a decimal el número $(0.10101)_2$

Solución

El dígito menos significativo es 1

M1 = 1/2 = 0.5
M2 = (0.5 + 0)/2 = 0.25
M3 = (0.25 + 1)/2 = 0.625
M4 = (0.625 + 0)/2 = 0.3125
M5 = (0.3125 + 1)/2 = 0.65625

finalmente $(0.10101)_2 = (0.65625)_{10}$

Ejemplo Convertir a decimal el número $(0.F6B)_{16}$

Solución

El dígito menos significativo es 1

M1 = 11/16 = 0.6875
M2 = (0.6875 + 2)/16 = 0.16796875
M3 = (0.16796875 + 6)/16 = 0.3855
M4 = (0.3855 + 15)/16 = 0.96159

Finalmente $(0.F62b)_{16} = (0.96159)_{10}$.

2.1.5 Sistema de base "n"

El sistema de base ***n*** más ampliamente usado para el diseño y construcción de pequeños sistemas digitales hasta una computadora digital (con $\mathbf{2^n}$ procesadores) es el sistema binario es por su facilidad de trabajar únicamente entre dos estados ("0" y "1"), pero para la programación de dichos sistemas digitales o computadoras digitales se utilizan los sistemas binario, octal, decimal y hexadecimal. Pero existen otros sistemas de base **n**, donde **n** puede ser un número entero positivo mayor que 1, y que cumplen con las mismas

características de los sistemas de base 2, 8, 10 y 16 como son: presentarse como un sistema de numeración posicional y cumplir con las reglas de la aritmética decimal.

El principal inconveniente de estos sistemas de base **n** (por ejemplo **n** = 5 ó 7) es que no tienen una aplicación práctica para el diseño de circuitos digitales ni mucho menos para computadora digital.

Como se mencionó anteriormente existen otros sistemas de base **n** = 3, 5, 6, etc., que se pueden representar en un sistema posicional, además de que permiten realizar las operaciones aritméticas de suma, resta, multiplicación y división. En esta sección vamos a presentar el caso del sistema de base **n** = 5, pero se puede extender a cualquier otra base.

El sistema de base 5 emplea cinco diferentes dígitos (0, 1, 2, 3 y 4). Para representar números mayores a 5, se combinan dos o más dígitos base y cada uno de éstos tendrá un valor según la posición que ocupe. El sistema de base 5 se representa en forma posicional por medio de la ecuación (2.1), con **n** = 5 y donde **d** puede representar cualquier dígito entre 0 y 4.

Ejemplo Representar el número $(14432)_5$ en forma posicional.

Solución Utilizando la ecuación (2.1) con 5 dígitos enteros (**p** = 5) y 0 dígitos fraccionarios (**q** = 0).

$$(14432)_5 = \sum_{i=0}^{5-1} d_i 5^i + \sum_{j=1}^{0} d_j 5^{-(i)}$$
$$= [d_0 5^0 + d_1 5^1 + d_2 5^2 + d_3 5^3 + d_4 5^4]$$
$$= [2x5^0 + 3x5^1 + 4x5^2 + 4x5^3 + 1x5^4]$$
$$= [2 + 15 + 100 + 500 + 625] = (1242)_{10}$$

La conversión de base 10 a base 5 también se puede realizar utilizando el algoritmo de divisiones sucesivas como se muestra a continuación con un ejemplo.

Ejemplo Convertir a binario el número $(1242)_{10}$ a base 5

Solución

1242	5
248	2
49	3
9	4
1	4
0	1

Además, con este sistema de base 5, también se pueden realizar las operaciones aritméticas básicas como se muestra a continuación:

Suma en base 5

Comprobación

$$\begin{array}{r} 1\ \ 1 \\ \hline (4\ 2\ 3)_5 \\ + \\ (2\ \ 4\ 0)_5 \\ \hline (1\ \ 2\ 1\ \ 3)_5 \end{array} \qquad \begin{array}{r} (1\ 1\ 3\ )_{10} \\ + \\ (\ 7\ 0\ \ )_{10} \\ \hline (\ 1\ 8\ 3\ )_{10} \end{array}$$

Multiplicación en base 5

Comprobación

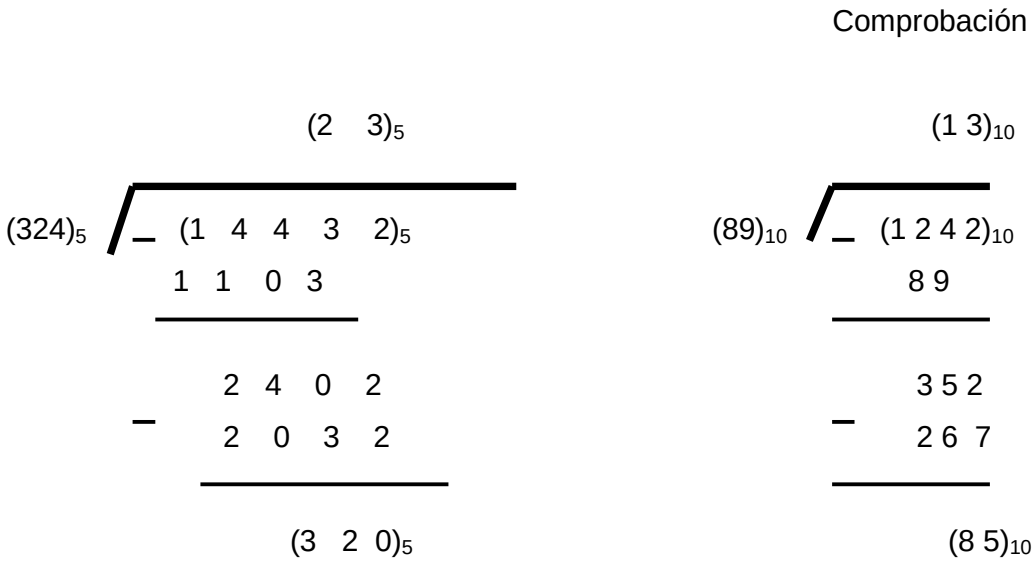
$$\begin{array}{r} 1\ \ 1 \\ \hline (4\ \ 3)_5 \\ \mathbf{X}\quad (\ 2)_5 \\ \hline (1\ \ 4\ \ 1\ \ )_5 \end{array} \qquad \begin{array}{r} (\ 2\ 3\ )_{10} \\ \mathbf{x}\ \ (\ \ 2\ \ )_{10} \\ \hline (\ 4\ \ 6\ )_{10} \end{array}$$

División en base 5

En la división en base 5 los únicos cinco dígitos posibles tanto en el cociente como en el residuo son 0, 1, 2, 3 y 4. La división en base 5 se puede efectuar utilizando el mismo procedimiento que se utiliza en la división decimal.

Ejemplo Realizar la operación siguiente $(1242)_{10} / (89)_{10}$ en base 5.

Solución Convertimos los números de base 10 a base 5, con lo cual tenemos



2.2 Aritmética binaria

En una computadora digital, las operaciones aritméticas se realizan en el sistema binario porque el diseño y construcción de circuitos lógicos (ver tema 5.5 y 5.6) para realizar aritmética binaria es mucho más sencilla que para la aritmética decimal.

2.2.1 Operaciones Aritméticas con números en diferentes bases

Las operaciones aritméticas básicas que se efectúan con los números en base decimal, también se pueden llevar a cabo en los sistemas de numeración de base **n**. En esta sección explicaremos cómo se realiza la suma, resta, multiplicación y división en los sistemas de numeración binaria, octal y hexadecimal.

- **Suma**

La suma en cualquier sistema de numeración (2, 8, 10, ó 16) se reduce a los cuatro casos siguientes:

positivo	positivo	negativo	negativo
+	+	+	+
positivo	negativo	positivo	negativo

En esta sección explicaremos el caso de sumar dos números positivos y en la sección **2.2.4** explicaremos los otros casos.

- **Suma en base 2**

Para realizar la suma de dos números binarios se usan las siguientes cuatro reglas fundamentales

0	1	0	1
+ 0	+ 0	+ 1	+ 1
---	---	---	---
0 0	**0** 1	**0** 1	**1** 0
C S	**C** S	**C** S	**C** S

donde

S es el resultado de sumar el Sumando (Augendo) y el Adendo, y
C es el acarreo que se produce al realizar la suma.

Suma de dos números binarios positivos

Ejemplo Realice la operación siguiente: $(0101)_2 + (1011)_2$

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente:

Comprobación

```
   1  1 1 1
 ----------------
     0 1 0 1          ( 5 )10
+                   +
     1 0 1 1          ( 11 )10
 ================    ==========
   1 0 0 0 0          ( 16 )10
```

Ejemplo Realice la operación siguiente $(10111)_2 + (11001)_2 + (10011)_2$

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente:

Comprobación

```
     1
  1  1  1   1   1   1
 ------------------------
     1  0  1  1  1          ( 23 )10
+    1  1  0  0  1      +   ( 25 )10
     1  0  0  1  1          (19 )10
 ==================     ==========
  1  0  0   0  0  1  1      ( 67 )10
```

Ejemplo Realice la operación indicada

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente

Comprobación

```
    1  1   1  1  1
  1  1  1   1  1  1
```

	1 1 0 0 1		$(25)_{10}$
	0 0 1 1 1		$(7)_{10}$
+	1 0 0 1 1	+	$(19)_{10}$
	1 1 1 0 1		$(29)_{10}$
	1 0 1 0 0 0 0		$(80)_{10}$

Para el caso de la suma de números fraccionarios se utilizan las mismas 4 reglas anteriores.

Suma de dos números fraccionarios positivos

Ejemplo Realice la suma siguiente: $(0.84375)_{10} + (0.28125)_{10}$

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente

			Comprobación
	0 1 1 1 1		
	0 . 1 1 0 1 1		$(0.84375)_{10}$
+		+	
	0 . 0 1 0 0 1		$(0.28125)_{10}$
	0 1. 0 0 1 0 0		$(1.125)_{10}$

Ejemplo Realice la suma siguiente

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente

Comprobación

```
   1 1 1 1 1 1 1
  ----------------
    1 1 0 1. 1 0 1 1          ( 13.6875   )10
+                           +
    1 0 1 1. 0 1 1 0          ( 11.3750  )10
  ================          ==========
  1 1 0 0 1. 0 0 0 1          ( 25.0625  )10
```

Ejemplo Realice la siguiente suma

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente

Comprobación

```
     1
  1 1 1  1  1  1
 ------------------------
      1  0  1 . 1  1          ( 5.75  )10
+     1  1  0 . 0  1      +   ( 6.25  )10
      1  0  0 . 1  1          ( 4.75 )10
 ========================   ==========
  1  0  0   0  0 . 1  1       ( 16.75 )10
```

- **Suma en base 8**

Ejemplo Realice la suma siguiente $(17)_{10} + (10)_{10}$ en base 8.

Solución Pasamos los números de base 10 a base 8 con lo cual tenemos

$$\begin{array}{r} (2\ \ 1)_8 \\ + \\ (1\ \ 2)_8 \\ \hline (3\ \ 3)_8 \end{array} \qquad \begin{array}{r} (1\ \ 7)_{10} \\ + \\ (1\ \ 0)_{10} \\ \hline (2\ \ 7)_{10} \end{array}$$

Ejemplo Realice la suma siguiente $(187)_{10} + (117)_{10}$

Solución Pasamos los números de base 10 a base 8 con lo cual tenemos

$$\begin{array}{r} \ \ 1\ \ 1 \\ \hline 2\ \ 7\ \ 3 \\ + \\ 1\ \ 6\ \ 5 \\ \hline 4\ \ 6\ \ 0 \end{array} \qquad \begin{array}{r} (1\ 8\ 7)_{10} \\ + \\ (1\ 1\ 7)_{10} \\ \hline (3\ 0\ 4)_{10} \end{array}$$

Ejemplo Realice la suma siguiente $(71.718750)_{10} + (115.234375)_{10}$

Solución Pasamos los números de base 10 a base 8 con lo cual tenemos

$$\begin{array}{r} \ \ \ \ 1\ \ 1 \\ \hline 1\ 0\ 7.\ 5\ 6 \\ + \\ 1\ 6\ 3.\ 1\ 7 \\ \hline 2\ 7\ 2\ .7\ \ 5 \end{array} \qquad \begin{array}{r} (\ \ 7\ \ 1.7\ 1\ 8\ 7\ 5\ 0)_{10} \\ + \\ (1\ \ 1\ 5\ .\ 2\ 3\ 4\ 3\ 7\ 5)_{10} \\ \hline (1\ 8\ 6\ .\ 9\ 5\ 3\ 1\ 2\ 5)_{10} \end{array}$$

- **Suma en base 16**

Ejemplo Realice la suma siguiente $(717)_{10} + (110)_{10}$

Solución Pasamos los números de base 10 a base 16 con lo cual tenemos

$$\begin{array}{r} (2\ \ C\ \ D)_{16} \\ + \\ (0\ \ 6\ \ E)_{16} \\ \hline (3\ \ 3\ \ B)_{16} \end{array} \qquad \begin{array}{r} (7\ \ 1\ \ 7)_{10} \\ + \\ (1\ \ 1\ \ 0)_{10} \\ \hline (8\ \ 2\ \ 7)_{10} \end{array}$$

Ejemplo Realice la suma siguiente $(1870)_{10} + (1107)_{10}$

Solución Pasamos los números de base 10 a base 16 con lo cual tenemos

$$\begin{array}{r} \ \ 1 \\ \hline 7\ \ 4\ \ E \\ + \\ 4\ \ 5\ \ 3 \\ \hline (B\ \ A\ \ 1)_{16} \end{array} \qquad \begin{array}{r} (1\ 8\ 7\ 0)_{10} \\ + \\ (1\ 1\ 0\ \ 7)_{10} \\ \hline (2\ \ 9\ \ 7\ \ 7)_{10} \end{array}$$

Ejemplo Realice la suma siguiente $(9901.75)_{10} + (987117.625)_{10}$

Solución Convertimos los números de base 10 a base 16 con lo cual tenemos

$$\begin{array}{r} \ \ 1\ \ 1 \\ \hline 2\ 6\ A\ D.\ 6 \\ + \\ F\ 3\ F\ E\ D.\ A \\ \hline (F\ 6\ 6\ 9\ \ B.\ 6)_{16} \end{array} \qquad \begin{array}{r} (\ \ 9\ 9\ 0\ 1\ .7\ 5\ 0)_{10} \\ + \\ (\ 9\ 8\ 7\ 1\ 1\ 7.6\ 2\ 5)_{10} \\ \hline (9\ 9\ 7\ 0\ 1\ 9.\ 3\ 7\ \ 5)_{10} \end{array}$$

- **Multiplicación**

➢ **Multiplicación en base 2**

Para efectuar la multiplicación binaria se utilizan las 4 reglas siguientes:

0	1	0	1
+ 0	+ 0	+ 1	+ 1
-------	-------	-------	--------
0 0	**0 0**	**0 0**	**0 1**
C S	**C S**	**C S**	**C** S

donde

S es el resultado de multiplicar los dos operandos, y

C es el acarreo que se produce al realizar la multiplicación.

Multiplicar dos números binarios positivos

Ejemplo Realice la multiplicación siguiente $(27)_{10}$ x $(3)_{10}$

Solución Utilizando las cuatro reglas anteriores y convirtiendo los números a base 2 tenemos lo siguiente

Comprobación

1 1 0 1 1	(27)$_{10}$
x 1 1	**x** (3)$_{10}$
----------	----------
	(81)$_{10}$

1 1 1 1

1 1 0 1 1

1 1 0 1 1

---

1 0 1 0 0 0 1

Ejemplo Realice la multiplicación siguiente

Solución Utilizando las cuatro reglas anteriores, tenemos lo siguiente

```
        1 1 1 0
      x 1 0 1 1
    -------------
      1 1 1 0
    1 1 1 0
  0 0 0 0
1 1 1 0
-------------------
1 0 0 1 1 0 1 0
```

Comprobación

```
  ( 14 )10
x ( 11 )10
-----------
   1 4
   1 4
-----------
(1 5 4) 10
```

También una multiplicación se puede realizar por sumas sucesivas, como se mostrará a continuación

Ejemplo Realice la multiplicación siguiente $(13)_{10}$ x $(3)_{10}$

Solución Primero convertimos los números a base 2 y luego aplicamos las 4 reglas de la suma binaria.

```
      1  1  0  1
+
      1  1  0  1
-------------------
   1  1  0  1  0
+
      1  1  0  1
-------------------
1  0  0  1  1  1
```

Comprobación

```
    ( 1  3 )10
+
    ( 1  3 )10
---------------
      2  6
+
      1  3
---------------
    ( 3  9 )10
```

Ejemplo Multiplicar en binario los números $(11)_{10}$ x $(4)_{10}$.

Solución

```
                    1   1
                ----------------
                1   0   1   1
          +
                1   0   1   1
            ---------------------
              1 0   1   1   0
          +
                1   0   1   1
            ---------------------
            1 0   0   0   0   1
          +
                1   0   1   1
            ---------------------
            1   0 1   1   0   0
```

Comprobación

```
      1  1
    +
      1  1
    ---------
      2  2
    +
      1  1
    ---------
      3  3
    +
      1  1
    ---------
      4  4
```

Si queremos realizar la multiplicación de dos números formados por parte entera y parte fraccionaria se emplean las mismas reglas anteriores.

Ejemplo Realice la multiplicación de $(13.375)_{10}$ x $(3.5)_{10}$

Solución

Comprobación

$$\begin{array}{r} 1\ 1\ 0\ 1\ ,\ 0\ 1\ 1 \\ \times\qquad\quad 1\ 1\ .\ 1 \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 1\ \ \\ \hline 1\ 1\ 0\ 1\ 0\ 1\ 1 \\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ \ \\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ \ \\ \hline 1\ 0\ 1\ 1\ 1\ 0\ .\ 1\ 1\ 0\ 1 \end{array}$$

$$\begin{array}{r} (13.375)_{10} \\ \times\quad (3.5)_{10} \\ \hline 6\ 6\ 8\ 7\ 5 \\ 4\ 0\ 1\ 2\ 5 \\ \hline (4\ 6\ .\ 8\ 1\ 2\ 5)_{10} \end{array}$$

- **Multiplicación en base 8**

Ejemplo Realice la multiplicación siguiente $(25)_{10}$ x $(10)_{10}$

Solución Convertimos los números de base 10 a base 8 con lo cual tenemos

$$\begin{array}{r} (3\quad 1)_8 \\ \times\ (1\quad 2)_8 \\ \hline 6\quad 2 \\ 3\quad 1\ \ \\ \hline (3\quad 7\quad 2)_8 \end{array}$$

$$\begin{array}{r} (2\ 5\ )_{10} \\ \times\quad (1\ \ 0)_{10} \\ \hline 0\ \ 0 \\ 2\ \ 5\ \ \\ \hline (2\ \ 5\ \ 0)_{10} \end{array}$$

Ejemplo Realice la suma siguiente $(251)_{10}$ x $(117)_{10}$

Solución Pasamos los números de base 10 a base 8 con lo cual tenemos

Comprobación

3 7 3

x 1 6 5

2 3 4 7
2 7 4 2
3 7 3

$(7\ 1\ 2\ 6\ 7)_8$

$(2\ 5\ 1)_{10}$

x $(1\ 1\ 7)_{10}$

1 7 5 7
2 5 1
2 5 1

$(2\ 9\ 3\ 6\ 7)_{10}$

Ejemplo Realice la multiplicación $(71.8750)_{10}$ x $(15.125)_{10}$

Solución Pasamos los números de base 10 a base 8 con lo cual tenemos

Comprobación

1 0 7 . 7

x 1 7 . 1

1 0 7 7
7 6 7 1
1 0 7 7

$(2\ 0\ 7\ 7\ .0\ 7)_8$

$(7\ 1.8\ 7\ 5)_{10}$

x $(1\ 5.1\ 2\ 5)_{10}$

3 5 9 3 7 5
1 4 3 7 5 0
7 1 8 7 5
3 5 9 3 7 5
7 1 8 7 5

$(1\,0\,8\ 7\ .1\ 0\ 9\ 3\ 7\ 5)_{10}$

- **Multiplicación en base 16**

Ejemplo Realice la multiplicación siguiente $(717)_{10} + (101)_{10}$

Solución Pasamos los números de base 10 a base 16, con lo cual tenemos

$$\begin{array}{r} 2\ C\ D \\ \times\quad 6\ 5 \\ \hline E\ 0\ 1 \\ 1\ 0\ C\ E \\ \hline 1\ 1\ A\ E\ 1 \end{array}$$

Comprobación

$$\begin{array}{r} (7\ 1\ 7)_{10} \\ \times\ (1\ 0\ 1)_{10} \\ \hline 7\ 1\ 7 \\ 7\ 1\ 7 \\ \hline (7\ 2\ 4\ 1\ 7)_{10} \end{array}$$

Ejemplo Realice la multiplicación siguiente $(1870)_{10} \times (1107)_{10}$

Solución Pasamos los números de base 10 a base 16, con lo cual tenemos

$$\begin{array}{r} 7\ 4\ E \\ \times\ 4\ 5\ 3 \\ \hline 1\ 5\ E\ A \\ 2\ 4\ 8\ 6 \\ 1\ D\ 3\ 8 \\ \hline 1\ F\ 9\ 6\ 4\ A \end{array}$$

$$\begin{array}{r} (1\ 8\ 7\ 0)_{10} \\ \times\ (1\ 1\ 0\ 7)_{10} \\ \hline 1\ 3\ 0\ 9\ 0 \\ 1\ 8\ 7\ 0 \\ 1\ 8\ 7\ 0 \\ \hline (2\ 0\ 7\ 0\ 0\ 9\ 0)_{10} \end{array}$$

Ejemplo Realice la multiplicación siguiente $(91.0625)_{10} \times (97.625)_{10}$

Solución Convertimos los números de base 10 a base 16, con lo cual tenemos

$$\begin{array}{r} 5\ B\ .\ 1 \\ \times\quad 6\ 1\ .\ A \\ \hline 3\ 8\ E\ A \\ 5\ B\ 1 \\ 2\ 2\ 2\ 6 \\ \hline 2\ 2\ B\ 9\ .\ F\ A \end{array}$$

Comprobación

$$\begin{array}{r} (9\ 1\ .\ 0\ 6\ 2\ 5)_{10} \\ \times\quad (9\ 7\ .\ 6\ 2\ 5\ )_{10} \\ \hline 4\ 5\ 5\ 3\ 1\ 2\ 5 \\ 1\ 8\ 2\ 1\ 2\ 5\ 0 \\ 5\ 4\ 6\ 3\ 7\ 5\ 0 \\ 6\ 3\ 7\ 4\ 3\ 7\ 5 \\ 8\ 1\ 9\ 5\ 6\ 2\ 5 \\ \hline (8\ 8\ 8\ 9\ .\ 9\ 7\ 6\ 5\ 6\ 2\ 5)_{10} \end{array}$$

- **Resta**

La resta en cualquier sistema de numeración (2, 8, 10, ó 16) se reduce a los cuatro casos siguientes:

positivo	positivo	negativo	negativo
-	-	-	-
positivo	negativo	positivo	negativo

En esta sección explicaremos el caso de restar dos números positivos y en la sección 2.6 explicaremos los casos restantes.

- **Resta en base 2**

Para realizar la resta de dos números binarios se utilizan las siguientes cuatro reglas fundamentales

0	1	0	1
+ 0	+ 0	+ 1	+ 1
-------	-------	-------	--------
0 0	**0** 1	**1** 1	**0** 0
P R	**P R**	**P R**	**P R**

donde:

R es el resultado de restar el minuendo y el sustraendo, y
P es el préstamo.

Para realizar la operación de resta o sustracción de dos números positivos (Minuendo mayor que el sustraendo) se utiliza el método de complemento a dos, el cual consiste en los pasos siguientes:

1. Verificar que el minuendo sea mayor que el sustraendo.
2. Al sustraendo se le aplica la operación “complemento a 1”, el cual consiste en intercambiar los 1s por 0s y los 0s por 1s.
3. Al resultado anterior se le aplica la operación “complemento a 2” la cual consiste en sumarle una unidad.
4. Sumar el minuendo con el resultado de la operación anterior.
5. En caso de que el bit de acarreo sea igual a 1, este se ignora.

Ejemplo Realice la resta siguiente $(29)_{10} – (15)_{10}$

Solución Representamos los números anteriores en base 2 y posteriormente aplicamos el complemento a 2.

	1	1	1	0	1	← Minuendo
-	0	1	1	1	1	← Sustraendo

Paso 1

Si analizamos el bit más significativo de los dos números, vemos que el bit más significativo vale 1 mientras que el bit más significativo del sustraendo vale 0 por lo tanto el minuendo es mayor que el sustraendo.

Paso 2

Aplicamos el complemento a “1”

```
  0  1  1  1  1 <----------   Sustraendo

  1  0  0  0  0 <----------   Operación Complemento a “1”
```

Paso 3 Aplicamos el complemento a “2”

```
  1  0  0  0  0 <----------   Resultado de la operación
                              Complemento a “1”
+                 1
-------------------------
  1  0  0  0  1 <----------   Resultado de la operación
                              Complemento a “2”
```

Paso 4

```
     1  1  1  0  1 <----------   Minuendo
+
     1  0  0  0  1 <----------   Resultado de la operación
                                 Complemento a “2”
-------------------------
1    0  1  1  1  0
↑
```

Bit de acarreo

Paso 5

Ignorar el bit de acarreo si tiene un valor de 1. Por lo tanto dicho bit se elimina.

Finalmente el resultado de la operación es $(01110)_2$

	Comprobación
1 1 1 0 1	$(29)_{10}$
-	-
0 1 1 1 1	$(15)_{10}$
0 1 1 1 0	$(14)_{10}$

Ejemplo Realice la resta siguiente $(55)_{10} – (45)_{10}$

Solución Representamos los números anteriores en base 2 y posteriormente aplicamos el método.

1 1 0 1 1 1	←	Minuendo
- 1 0 1 1 0 1	←	Sustraendo

Paso 1

Analizando el bit más significativo tanto del minuendo como del sustraendo vemos que el bit más significativo del minuendo y del sustraendo tienen el mismo valor, entonces analizamos la columna anterior al bit más significativo y en ella vemos que el bit del minuendo vale 1 mientras que el bit del sustraendo vale 0 y por lo tanto si se puede realizar la operación indicada.

Paso 2 Aplicamos el complemento a “1”

1 0 1 1 0 1	←	Sustraendo
0 1 0 0 1 0	←	Operación Complemento a “1”

Paso 3 Aplicamos el complemento a “2”

0 1 0 0 1 0 ← Resultado de la operación

Complemento a “1”

\+ 1

0 1 0 0 1 1 ← Resultado de la operación

Complemento a “2”

Paso 4

1 1 0 1 1 1 ← Minuendo

\+

0 1 0 0 1 1 ← Resultado de la operación

Complemento a “2”

1 0 0 1 0 1 0

Bit de acarreo

Paso 5 Ignorar el bit de acarreo si tiene un valor de 1.

Finalmente el resultado de la operación es $(0\ \ 0\ \ 1\ 0\ 1\ \ 0)_2$.

Comprobación

$$\begin{array}{r} 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1 \\ - \\ 1\ \ 0\ \ 1\ \ \ 1\ \ 0\ \ \ 1 \\ \hline (0\ \ \ 0\ \ \ 1\ \ \ 0\ \ \ \ 1\ \ \ 0)_2 \end{array} \qquad \begin{array}{r} (55)_{10} \\ - \\ (45)_{10} \\ \hline (10)_{10} \end{array}$$

- **Resta en base 8**

Ejemplo Realice la resta $(25)_{10} - (10)_{10}$ en base 8.

Solución Convertimos los números de base 10 a base 8, con lo cual tenemos:

11 – Pido prestado una unidad y con eso se forma dicho número

comprobación

$$\begin{array}{r} \text{----------} \\ 3\ \ 1 \\ - \\ 1\ \ 2 \\ \hline (1\ \ \ 7)_8 \end{array} \qquad \begin{array}{r} (2\,5)_{10} \\ - \\ (1\,0)_{10} \\ \hline (1\ \ \ 5)_{10} \end{array}$$

Ejemplo Realice la resta siguiente $(251)_{10} - (117)_{10}$ en base 8.

Solución Convertimos los números de base 10 a base 8, con lo cual tenemos:

13 → Préstamo Comprobación

$$\begin{array}{r} \text{----------} \\ 3\ \ 7\ \ 3 \\ -\ 1\ \ 6\ \ 5 \\ \hline (2\ \ \ 0\ \ \ 6)_8 \end{array} \qquad \begin{array}{r} (2\ 5\ 1)_{10} \\ -\ \ (1\ 1\ 7)_{10} \\ \hline (1\ 3\ 4)_{10} \end{array}$$

Ejemplo Realice la resta $(71.8750)_{10}$ - $(15.125)_{10}$ en base 8.

Solución Pasamos los números de base 10 a base 8, con lo cual tenemos:

		Comprobación	
	1 0 7 . 7		$(7\ 1.8\ 7\ 5)_{10}$
-		-	
	1 7 . 1		$(1\ 5.1\ 2\ 5)_{10}$
	(0 7 0 . 6)		$(5\ 6.7\ 5\ 5)_{10}$

- **Resta en base 16**

Ejemplo Realice la resta $(25)_{10}$ - $(10)_{10}$ en base 16.

Solución Convertimos los números de base 10 a base 16, con lo cual tenemos:

	11 – Pido préstamo y formo el número	Comprobación
	1 9	$(2\ 5)_{10}$
-		-
	A	$(1\ 0)_{10}$
	F	$(1\ 5)_{10}$

Ejemplo Realice la resta siguiente $(251)_{10}$ - $(117)_{10}$ en base hexadecimal.

Solución Convertimos los números de base 10 a base 16; con lo cual tenemos:

	F B	$(2\ 5\ 1)_{10}$
-		-
	7 5	$(1\ 1\ 7)_{10}$
	$(8\ 6)_{16}$	$(1\ 3\ 4)_{10}$

Ejemplo Realice la resta $(71.8750)_{10}$ - $(15.125)_{10}$ en base 16

Solución Convertimos los números de base 10 a base 16 utilizando la tabla 2.1, con lo cual tenemos:

$$\begin{array}{r} 47.E \\ - \\ F.2 \\ \hline (38.C)_{16} \end{array} \qquad \begin{array}{r} (71.875)_{10} \\ - \\ (15.125)_{10} \\ \hline (56.750)_{10} \end{array}$$

- **División**

Al igual que la operación de multiplicación, la división se puede realizar de diferentes formas, las cuales presentamos a continuación:

- **División en base 2**

La división binaria es mucho más fácil, porque los únicos dos posibles dígitos cocientes son 0 y 1. La división binaria se puede efectuar utilizando el mismo procedimiento que se utiliza en la división decimal, es decir:

$$\begin{array}{r} 12 \\ 3\overline{)36} \\ -\,3 \\ \hline 06 \\ -\,6 \\ \hline 0 \end{array}$$

Ejemplo Realizar la operación siguiente $(43)_{10}$ / $(3)_{10}$ en binario

Solución Convertimos los números de base 10 a base 2, con lo cual tenemos

Comprobación

```
        0  0  1  1  1  0
     ___________________
1 1 |  1  0  1  0  1  1
       1  1
     ---------------
        1  0  0
     -     1  1
     ------------
           1  1
        -  1  1
        ---------
           0  1
```

```
        1  4
     ___________
3  |  4  3
      3
     ---------
      1  3
   -  1  2
     ---------
         1
```

Operaciones de apoyo

```
   1  0  1
-
   0  1  1
-----------
   0  0  1
```

```
      1  0  0
+
            1
--------------
      1  0  1
```

```
      1  0  1
+
      1  0  1
--------------
   1  0  1  0
```

```
   1  0  0
-
   0  1  1
-----------
   0  0  1
```

```
      1  0  0
+
            1
--------------
      1  0  1
```

```
         1  0  0
+
         1  0  1
-----------------
   1  0  0  1
```

Ejemplo Realizar la operación siguiente $(217)_{10}/(11)_{10}$ en binario

Solución Convertimos los números de base 10 a base 2

Comprobación

```
          0  0  0  1  0  0  1  1
      ____________________________
1011 / 1  1  0  1  1  0  0  1
       1  0  1  1
       ----------
          0  0  1  0  1  0  0
       -     0  1  0  1  1
             ----------
                1  0  0  1  1
             -  0  1  0  1  1
                ----------
                0  1  0  0  0
```

```
          1  9
      ____________
11 / 2  1  7
     1  1
     ----------
        1  0  7
     -     9  9
     ----------
           8
```

Operaciones de apoyo

```
  1  1  0  1          0  1  0  0          1  1  0  1
-                  +                  +
  1  0  1  1                1             0  0  1  0
-----------        -----------        -----------
  0  0  1  0          0  1  0  1        1  0  0  1  0

  1  0  1  0  0        1  0  1  0  0        1  0  1  0  0
-                  +                  +
  0  1  0  1  1                1             1  0  1  0  1
-----------        -----------        -----------
  0  1  0  0  1        1  0  1  0  1      1  0  1  0  0  1

  1  0  0  1  1        1  0  1  0  0        1  0  0  1  1
-                  +                  +
  0  1  0  1  1                1             1  0  1  0  1
-----------        -----------        -----------
  0  1  0  0  0        1  0  1  0  1      1  0  1  0  0  0
```

- **División en base 8**

Ejemplo Realice la división de los números $(25)_{10}$ / $(10)_{10}$ en base 8.

Solución Convertimos los números de base 10 a base 8, con lo cual tenemos:

```
          2
1 2 ) 3 1
    - 2 4
    ---------
          5
```

Comprobación

```
          2
10 ) 2 5
   - 2 0
   ---------
          5
```

Ejemplo Realice la división siguiente $(2501)_{10}$ - $(117)_{10}$ en base 8

Solución Convertimos los números de base 10 a base 8 con lo cual tenemos

```
            2 5
1 6 5 ) 4 7 0 5
      - 3 5 2
      ---------
        1 1 6 5
      - 1 1 1 1
      ---------
            5 4
```

Comprobación

```
            2 1
1 1 7 ) 2 5 0 1
      - 1 1 7
      ---------
        1 6 1
      - 1 1 7
      ---------
            4 4
```

División en base 16

Ejemplo Realice la división $(2075)_{10}$ / $(13)_{10}$ en base 16.

Solución Convertimos los números de base 10 a base 16 utilizando la tabla 2.1, con lo cual tenemos:

```
          9  F                         1 5 9
      ___________                 ___________
  D  /  8  1  B               1 3 / 2 0 7 5
    -      7  5                  -    1 3
    ------------                 ------------
           C  B                       7 7
       -   C  3                  -    6 5
       ------------              ------------
             8                       1 2 5
                                 -   1 1 7
                                 ------------
                                         8
```

Ejemplo Realice la división siguiente $(2501)_{10}$ - $(17)_{10}$ en base hexadecimal.

Solución Convertimos los números de base 10 a base 16 con lo cual tenemos:

```
            9  3                        1 4  7
      ___________                 ___________
  11 / -  9  C  5             1 7 / - 2 5 0  1
          9  9                          1 7
    -----------                  -------
             3  5                      8  0
       -     3  3                -     6  8
       -----------               -----------
               2                     1  2  1
                                 -   1  1  9
                                 -----------
                                           2
```

2.2.2 Complemento a la base y a la base disminuida

Los complementos se usan en las computadoras digitales para simplificar la operación de sustracción y para manipulaciones lógicas. Los complementos para cada sistema de base **n** son:

* El complemento a la base (**n**), y
* El complemento a la base disminuida (**n-1**)

- **Complemento a la base *n***

Dado un número positivo **N** en base ***n*** con una parte entera de ***p*** dígitos, el complemento de **n** de **N** se define como $\mathbf{n^p - N}$ para $\mathbf{N \neq 0}$ y 0 para $\mathbf{N = 0}$. A continuación presentamos algunos ejemplos.

El complemento de 10 del número $(23)_{10}$ es $10^2 – 23 = 77$ (con $\boldsymbol{p} = 2$).
El complemento de 10 del número $(0.37)_{10}$ es $1 – 0.37 = 0.63$.

El complemento de 2 de (1001) es
$(2^4)_{10}$ - $(1001)_2$ $=(10000)_2 – 1001 =00111$ (con $\boldsymbol{p} = 4$).
El complemento de 2 del número $(0.0101)_2$ es
$(1)_2 – (0.0101)_2 = (0.1011)_2$.

Como se mencionó anteriormente, el complemento a la base se utiliza para facilitar la operación de substracción. Para obtener el complemento a una base **n** de un número se obtiene restando a la “base menos uno” cada uno de los dígitos del número a convertir y sumándole uno al resultado de las restas. El acarreo final se ignora.

Ejemplo Realizar la substracción decimal
Solución

29 -12 = 17

El complemento a diez de 2 es 9-2 =7.
El complemento a diez de 1 es 9-1 =8.

Por lo tanto, el complemento a diez de 12 es 87 + 1 = 88, por lo tanto:

$$\begin{array}{rr} & 29 \\ + & 88 \\ \hline & 117 \end{array}$$

Ignorar el Acarreo 

Finalmente, obtenemos

$$\begin{array}{rr} & 29 \\ - & 12 \\ \hline & 17 \end{array}$$

de esta manera se realiza la resta decimal empleando el "complemento diez" de dos números en base decimal.

- **Complemento a la base disminuida (*n-1*)**

Dado un número positivo **N** en base **n** con una parte entera de ***p*** dígitos y una parte fraccionaria de q dígitos, el complemento de **n-1** de **N** se define como $\mathbf{n^{p}-n^{-q}-N.}$ A continuación presentamos algunos ejemplos.

El complemento de 9 del número $(327)_{10}$ es $10^3 - 1 - 327 = 672$ (con $\boldsymbol{p} = 3$), y

$10^{-m} = 10^0 = 1$ (con $\boldsymbol{q} = 0$)

El complemento de 9 del número $(0.173)_{10}$ es $1 - 10^3 - 0.173 = 0.826$(con $\boldsymbol{q}= 3$), y

$10^{p} = 10^0 = 1$ (con $\boldsymbol{p} = 0$)

El complemento de 1 del número $(101100)_2$ es

$(10^6 - 1)_{10} - (101100)_2 = (111111 - 101100)_2 = (010011)_2$ (con $\boldsymbol{p} = 6$)

El complemento de 1 del número $(0.0110)_2$ es

$(1- 2^{-4})_{10} – (0.0110)_2 - (0.1111 – 0.0110)_2 = (0.1001)_2$

A continuación presentamos más ejemplos del **Complemento a la base n** aplicados a diferentes bases.

Complemento a la base 2

En nuestro caso, un caso muy interesante el complemento a la base (**n** = 2) aplicado a la resta binaria, la cual se realiza utilizando el complemento a dos de un número binario. El complemento a dos de un número binario se obtiene restando a 1 cada uno de los dígitos del número y sumándole 1 al resultado. Para el sistema de numeración binario, esto mismo se logra cambiando directamente los unos por ceros y los ceros por unos y sumando uno al resultado.

Ejemplo Obtener el complemento a 2 del número $(0000\ 0101)_2$

Solución

0000 0101 Complemento a 1 ⟶ 1111 1010

\+ 1

Complemento a 2 ⟶ 1111 1011

Ejemplo Convertir el número 0111 1111 $(+127)_{10}$ a negativo

Solución

Número original = 01111 1111

Complemento a uno = 1000 0000

\+ 1

Complemento a dos = 1000 0001 = -127

Ejemplo Encontrar el valor absoluto del número $(1000\ 0010)_2$

Solución

Número original = 1000 0010

Complemento a uno = 0111 1101

\+ 1

Complemento a dos = 0111 1110 = $(+126)_{10}$

Ejemplo Obtener el complemento a dos del número $(101001)_2$

Solución

Número original = 101001

Complemento a uno = 010110

\+ 1

Complemento a dos = 010111

Complemento a la base 8

El complemento a 8 de un número octal se obtiene restando cada uno de los dígitos del número 7 sumándole 1 al resultado.

Ejemplo Obtener el complemento a 8 del número $(027)_8$, es decir, (obtener su negativo)

Solución

377 350

\- 027 + 1

---

350 –complemento a 7 351 ---------complemento a 8

Luego realizamos los siguiente $(351)_8 = (-122)_{8,}$ es decir, $(351)_8$ es el valor absoluto de $(-122)_8$.

Ejemplo Obtener el complemento a 8 del número $(256)_8$, es decir, (obtener su valor absoluto).

Solución

```
  377                          121
- 256                      +    1
----------                  ___________
  121 –complemento a 7          122   ---------complemento a 8
```

Luego realizamos que $(256)_8 = (-122)_8$, es decir, $(122)_8$ es el valor absoluto de $(256)_8$.

Nota:

A partir de los ejemplos anteriores observamos que del complemento de un número positivo se obtiene su negativo y del complemento de un número negativo se obtiene su valor positivo.

Complemento a la base 16

El complemento a 16 de una cantidad hexadecimal se obtiene restando cada uno de los dígitos de la cantidad a F_{16} y sumándole 1 al resultado.

Ejemplo Obtener el complemento a 16 del número del número $(27)_{16}$ (Obtener su negativo)

Solución

```
   FF                          D8
 - 27                      +   1
 ---------                   --------
  D8   Complemento a 15         D9   - Complemento a 16
```

posteriormente tenemos que $(-27)_{16} = (D9)_{16}$.

Ejemplo Obtener el complemento a 16 del número del número $(D4)_{16}$
(Obtener su negativo)

Solución

FF		2B	
- D4		+ 1	
---		---	
2B	Complemento a 15	2C	- Complemento a 16

finalmente realizamos $(D4)_{16} = (-2C)_{16}$.

2.2.3 Representación de números con signo

Una computadora digital que procesa únicamente números positivos no es muy útil. La mayoría de las computadoras digitales trabajan con números signados (números positivos y números negativos). Las computadoras utilizan el método de complemento a dos para representar los números con signo (números signados).

Se presenta un número problema muy interesante si se desea conocer cuándo un número es negativo y cuándo es positivo. Una manera práctica que se emplea al diseñar una computadora digital es utilizar al bit más significativo del número para representar el signo. Un “1” en esa posición representa al signo negativo y un “0” representa al signo positivo.

0 xxx xxxx	representa un número positivo de 7 bits
1 xxx xxxx	representa un número negativo de 7 bits

La ventaja de usar este esquema para representar números signados es que no se necesitan circuitos digitales especiales para efectuar operaciones aritméticas. Únicamente se requiere una atención especial en la lógica de la programación con el bit de signo.

Para analizar esta representación de números signados, observemos las tres columnas de números de cuatro dígitos binarios de la Tabla 2.2. La columna 1 se forma sumando 1 a partir de 0000. Observemos que cuando se suma 1 al número 1111 se pasa a 0000. Recuerda que los números son de 4 dígitos, por eso se ignora el quinto del bit del resultado. Por lo tanto sin considerar el último bit como signo se tienen 16 números binarios desde $(0000)_2$ a $(1111)_2$.

1111	1 111	0111
1110	1 110	0110
1101	1 101	0101
1100	1 100	0100
1011	1 011	0011
1010	1 010	0010
1001	1 001	0001
1000	1 000	0000
0111	0 111	1111
0110	0 110	1110
0101	0 101	1101
0100	0 100	1100
0011	0 011	1011
0010	0 010	1010
0001	0 001	1001
0000	0 000	1000
Columna 1	Columna 2	Columna 3

Tabla 2.2 Interpretación del signo

En columna 2 de la Tabla 2.2 se forman dos grupos de 8, la diferencia es el bit más significativo. Si utilizamos al bit más significativo para el bit de signo, la parte superior de la columna 2 forma los números negativos y la parte inferior forma los números positivos.

Por otro lado hemos observado que si al número $(1111)_2$ (de la columna 2) se le suma 1 se obtiene $(0000)_2$ ó $(0)_{10}$ por lo que reconocemos al número binario

1111 con el número decimal -1, al número binario 1110 como decimal -2, etc. y al 1000 como -8.

Con este resultado, si al grupo de los números negativos lo colocamos debajo del número 000 se obtiene la columna 3 de la tabla 2.2. Abajo del $(0000)_2$ ó $(0)_{10}$ tendremos ahora el número $(1111)_2$ ó $(-1)_{10}$ y arriba del $(0000)_2$ el número $(0001)_2$ ó $(+1)_{10}$.

Si consideramos al cero como número positivo, tendremos una cantidad de números negativos igual a la cantidad de números positivos más uno, es decir, con cuatro dígitos binarios tendremos del 1_{10} al 7_{10} (8 dígitos positivos) y del -1_{10} al -8_{10} (8 dígitos negativos).

Este método se puede extender a cualquier cantidad de dígitos binarios. La tabla 2.3 está formada por números binarios de 8 bits con signo.

Binario	Octal	Hexadecimal	Decimal
0111 1111	177	7F	+ 127
0111 1110	176	7E	+ 126
0111 1101	175	7D	+ 125
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
0000 0010	002	02	+ 2
0000 0001	001	01	+ 1
0000 0000	000	00	0
1111 1111	377	FF	- 1
1111 1110	376	FE	- 2
1111 1101	375	FD	- 3
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
1000 0010	202	82	- 126
1000 0001	201	81	- 127

1000 0000	200	80	- 128

Tabla 2.3 Representación de números en diferentes bases con signo.

A partir de la tabla 2.3 se observa que los números de 377 a 200 en el sistema octal y de FF a 80 en el sistema hexadecimal son negativos. El tercer dígito de los números en el sistema decimal es únicamente de dos bits, ya que estamos trabajando con ocho bits

2.2.4 Operaciones aritméticas con números asignados

En esta sección presentamos las operaciones aritméticas números principalmente con signo negativo y en base 2 y base 16, ya que dichas bases son las más utilizadas.

- **Operaciones aritméticas en base 2**

Suma binaria

Caso a) Sumando menor que el sustraendo

Ejemplo Realice la suma siguiente 17 + -29

Solución Para resolver esta suma lo primero que tenemos que realizar es calcular el complemento a dos del número -29 y luego aplicar las reglas de la suma y en caso de que exista un bit de acarreo se debe ignorar.

$(29)_{10} = (1\ 1\ 1\ 0\ 1)_2$

Complemento a 1

```
  0 0 0 1 0
+         1
-----------
  0 0 0 1 1
```

$(-29)_{10} = (00011)_2$

Finalmente realizamos la operación de suma

Comprobación

```
      0 0 1 1
  ----------------
      1 0 0 0 1          ( 17 )10
  +                   +
      0 0 0 1 1          ( -29 )10
  ================     ==========
      1 0 1 0 0          ( 12 )10
      ↑
```

Bit de Acarreo

Caso b) Sumando mayor que el sustraendo

Ejemplo Realice la operación $(15)_{10} + (-11)_{10}$

Solución

Comprobación

```
     0000  1111          (+15)10
  +                   +
      1111  0101         (-11)10
  ================     ==========
     10000  0100         (+04)10
```

Para el caso de números fraccionarios

Ejemplo Realice la operación $(5.5)_{10} + (3.25)_{10}$ indicada

Solución

	0101.1000		$(5.5)_{10}$
+		+	
	0011.0100		$(3.25)_{10}$
	1000.1100		$(8.75)_{10}$

Ejemplo Realice la operación $(5.75)_{10} – (3.5)_{10}$ indicada

Solución

$(3.50)_{10}$ = 0011.1000

1100.0111 ---------- Complemento a 1

\+ 1 ---------- Complemento a 2

$(-3.50)_{10}$ = 1100.1000

Luego realizamos la suma

Decimal		Binario
$(5.75)_{10}$		0101.1100
+ $(-3.50)_{10}$	+	1100.0100
---		---
$(2.25)_{10}$		1 0010.0100

Este bit de acarreo se ignora

Nota

Observa cómo se obtiene el complemento a dos para números fraccionarios.

Multiplicación binaria

Ejemplo Realizar la operación 10 por -8

Solución

Decimal	Binario	Octal	Hexadecimal
10	0000 1010	012	0A
x (-8)	x 0000 1000	x 008	x 0C
----------	----------	----------	----------
-80	0000 0000	120	78
	0 0000 000		
	00 0000 00		
	000 0101 0		
	----------		
	000 0101 0000		
	1010 1111	257	AF
	+ 1	+ 1	+ 1
	----------	----------	----------
Complemento a 2	(1011 0000)	260	B0

División Binaria

Ejemplo Realizar la operación de 36/3 en el sistema binario.

Solución

12	00001100	14	C
------	----------	------	------
3 / 36	11/ 001000100	3 / 44	3/ 24
-3	1101	-3	-24
--------	----------	----------	-------
06	00011	14	0
- 6	1101	-14	
----------	----------	----------	
0	00000	0	

Apoyo

$(3)_{10} = (00000011)_2$ $(-3)_{10} = (11111101)_2$

Nota

En el sistema octal y hexadecimal los valores negativos se obtienen con los complementos a 8 y a 16 respectivamente de los valores positivos. El complemento a 8 de un número octal se obtiene restando cada uno de los dígitos del número 7 sumándole 1 al resultado. El complemento a 16 de una cantidad hexadecimal se obtiene restando cada uno de los dígitos de $(F)_{16}$ y sumándole 1 al resultado.

Nota

A partir de los ejemplos anteriores observamos que del complemento de un número positivo se obtiene su negativo y del complemento de un número negativo se obtiene su valor positivo.

- **Operaciones aritméticas en base 16**

Suma en base 16

Ejemplo Realizar la suma de $(–13)_{10} + (-11)_{10}$ en base hexadecimal

Solución

-13_{10}

-11_{10}

-24_{10}

$(+13)_{10}$ => $(0D)_{16}$ $(+11)_{10}$ => $(0B)_{16}$

Sacamos el complemento a 15

```
   F  F
 -
   0  D
 ------
   F  2
 +    1
 ------
   F  3
```

Sacamos el complemento a 15

```
   F  F
 -
   0  B
 ------
   F  4
 +    1
 ------
   F  5
```

Finalmente, realizamos la suma

```
     F  5
   +
     F  3
   -------
   1 E  8
```

Bit de signo

Comprobación

La comprobación se realiza obteniendo el valor absoluto del resultado de la suma de la manera siguiente

```
     F  3
 -
     E  8
 ------
     1  7
   +
        1
 ------
     1  8
```

Con lo que $18_{16} = 24_{10}$

Ejemplo Realizar la suma de $(-19957)_{10} + (10999)_{10}$ en base hexadecimal

Solución

$$\begin{array}{r} (-19957)_{10} \\ + \\ (-10999)_{10} \\ \hline (-30956)_{10} \end{array}$$

$(+19957)_{10}$ => $(4DF5)_{16}$ $(+10999)_{10}$ => $(2AF7)_{16}$

Complemento a 15 del número $(-19957)_{10}$

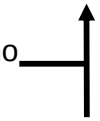
$$\begin{array}{r} F\ F\ F\ F \\ - \\ 4\ D\ F\ 5 \\ \hline B\ 2\ 0\ A \\ + \\ 1 \\ \hline B\ 2\ 0\ B \end{array}$$

Complemento a 15 del número $(-10999)_{10}$

$$\begin{array}{r} F\ F\ F\ F \\ - \\ 2\ A\ F\ 7 \\ \hline D\ 5\ 0\ 8 \\ + \\ 1 \\ \hline D\ 5\ 0\ 9 \end{array}$$

Realizamos la suma de los complementos

```
    B  2  0  B
+
    D  5  0  9
-------------
 1  8  7  1  4
```

Bit de signo 

Comprobación

Para comprobar el resultado obtenemos el valor absoluto del resultado de la operación de la manera siguiente:

```
    F  F  F  F
-
    8  7  1  4
-------------
    7  8  E  B
+
             1
-------------
    7  8  E  C
```

Ejemplo Realizar la suma

Solución

-19957_{10}

-10999_{10}

-30956_{10}

$(+19957)_{10}$ => $(4DF5)_{16}$ $(+10999)_{10}$ => $(2AF7)_{16}$

complemento a 15 del número 4DF5

```
  F F F F
-
  4 D F 5
-------
  B 2 0 A
+
        1
-------
  B 2 0 B
```

complemento a 15 del número 2AF7

```
  F F F F
-
  2 A F 7
-------
  D 5 0 8
+
        1
-------
  D 5 0 9
```

Realizamos la suma de los complementos

```
  B 2 0 B
+
  D 5 0 9
-------
1 8 7 1 4
```

Bit de signo

Finalmente la suma de

-19957_{10}

+

-10999_{10}

-30956_{10}

$B\ 2\ 0\ B_{16}$

+

$D\ 5\ 0\ 9_{16}$

$8\ 7\ 1\ 4_{16}$

Comprobación

Para comprobar el resultado, obtenemos el valor absoluto del resultado de la suma de la manera siguiente

```
  F  F  F  F
-
  8  7  1  4
-------------
  7  8  E  B
+
           1
-------------
  7  8  E  C
```

Bibliografía del tema 2

Ayala Pérez, Jaime *Arquitectura de Computadoras*, México, Comunicación interna, 2005.

Ayala Pérez, Jaime, *Aritmética para Computadoras*, México ENEP. Aragón, UNAM, (colección de Cuadernos de la ENEP ARAGÓN, No. 33,) 1989.

Charles H Roth Jr., *Fundamentals of Logic Design*, 4ª ed., EE.UU., West Publishing Company, 1992.

García, Narcia, O., *8085A e Interfaces*, México, IPN, Agosto 1982.

Hamacher, V. C., *Computer Organization*, EE.UU., McGraw Hill International Book Company, 1982.

Hayes P. John, *Diseño de Sistemas Digitales y Microprocesadores*, McGraw Hill, 1986.

Lee, Samuel C *Digital circuits and logic design,* EE.UU., Prentice Hall, Inc., Englewood Cliffs, N. J., 1976.

Mandado, E., *Sistemas Electrónicos Digitales*, Madrid, Marcombo Boixareu, 1980.

Mano, Morris, M., *Arquitectura de Computadoras*, México, Prentice Hall Hispanoamericana, 1988.

Mano, Morris, M., *Diseño Digital*, México, Prentice Hall Hispanoamericana, 1987.

Peatman, B. John, *Microcomputer-Based Design*, Tokio, McGraw Hill / Kogakusha, Ltd, 1982.

Tocci, Ronald, Digital Design, Principles and Applications, EE.UU., Prentice Hall, 1977.

Actividades de Aprendizaje

A.2.1. Realiza las siguientes conversiones de base 10 a base 2.

a.) $(3789)_{10}$

b.) $(5589)_{10}$

c.) $(5875.125)_{10}$

d.) $(7896.479)_{10}$

e.) $(5589.879)_{10}$

f.) $(51875.9272)_{10}$

A 2.2. Realiza las siguientes conversiones de base 16 a base 2.

a.) $(78A9C)_{16}$

b.) $(8FA9)_{16}$

c.) $(5875.127)_{16}$

d.) $(7ADF6.4D9)_{16}$

e.) $(5A5D8.8FC)_{16}$

f.) $(15C7A5.92FD)_{16}$

A.2.3. Realiza las siguientes conversiones de base 8 a base 2.

a.) $(75157)_{8}$

b.) $(2475)_{8}$

c.) $(5176.127)_{8}$

d.) $(41756.427)_{8}$

e.) $(51275.725)_{8}$

f.) $(157755.5256)_{8}$

A.2.4. Realiza las siguientes conversiones de base 2 a base 10.

a.) $(11100001)_2$

b.) $(1100111)_2$

c.) $(10101.111)_2$

d.) $(11011.1101)_2$

e.) $(110101.10111)_2$

f.) $(1101111.11011)_2$

A.2.5. Realiza las siguientes conversiones de base 16 a base 10.

a.) $(78A9C)_{16}$

b.) $(8FA9)_{16}$

c.) $(5875.127)_{16}$

d.) $(7ADF6.4D9)_{16}$

e.) $(5A5D8.8FC)_{16}$

f.) $(15C7A5.92FD)_{16}$

A.2.6. Realza las siguientes conversiones de base 8 a base 10.

a.) $(76567)_8$

b.) $(21457)_8$

c.) $(5176.125)_8$

d.) $(341766.427)_8$

e.) $(751724.757)_8$

f.) $(5721754.5256)_8$

A.2.7. Realiza las siguientes conversiones de base 10 a base 16.

a.) $(87897)_{10}$

b.) $(68679)_{10}$

c.) $(15875.127)_{10}$

d.) $(578976.459)_{10}$

e.) $(357558.847)_{10}$

f) $(2159775.925)_{10}$

A.2.8. Realiza las siguientes conversiones de base 10 a base 8.

a.) $(657297)_{10}$

b.) $(27899)_{10}$

c.) $(51475.1237)_{10}$

d.) $(45787976.457)_{10}$

e.) $(7356558.847)_{10}$

f.) $(902159775.325)_{10}$

A.2.9. Realiza las siguientes conversiones de base 16 a base 8.

a.) $(27899)_{16}$

b.) $(5FBAD4.1237)_{16}$

c.) $(A6D5FB.ACB7)_{16}$

d.) $(DE65CD.84DA)_{16}$

e.) $(FA2DC75.325F)_{16}$

A.2.10. Realiza las siguientes conversiones de base 8 a base 16.

a.) $(7765667)_{8}$

b.) $(1214547)_{8}$

c.) $(515176.175)_{8}$

d.) $(27341766.423)_{8}$

A.2.11. Diseña e implementa un programa para computadora en cualquier lenguaje de programación que realice la conversión de un número entero (de cualquier longitud) en base 10 a base 2.

A.2.12. Diseña e implementa un programa para computadora en cualquier lenguaje de programación que realice la conversión de un número (T >19 dígitos) en base 10 a base 2.

donde

T = X + Y

X es el número de dígitos enteros.

Y es el número de dígitos decimales,

y comprueba los ejercicios del inciso A.2.1.

A.2.13. Diseña e implementa un programa para computadora en cualquier lenguaje de programación que realice la conversión de un número (de T>19 dígitos) en base N a base M.

donde

T = X + Y, y

X es el número de dígitos enteros.

Y es el número de dígitos decimales,

y compruebe los ejercicios del inciso A.2.2. a A.2.10.

A.2.14. Diseña e implementa un programa para computadora en cualquier lenguaje de programación que muestre las tablas de multiplicación en base 16.

A.2.15. Diseña e implementa un programa para computadora en cualquier lenguaje de programación que muestre las tablas de multiplicación en base 8.

A.2.16. Diseña e implementa un programa para computadora en cualquier lenguaje de programación que sume N números de cualquier longitud en base 2.

Cuestionario de autoevaluación

1. ¿Qué es un sistema de numeración posicional?
2. ¿Cuáles son los sistemas de numeración más utilizados por una computadora digita y por el hombre?
3. ¿Cuáles son las ventajas de utilizar el sistema binario en un computadora digital?
4. ¿Cuál es el procedimiento para convertir un número decimal a base 2?
5. ¿Cuáles son los pasos del método complemento a dos?
6. ¿Cuál es el procedimiento para convertir un número en base dos a base 16?
7. ¿Cuál es el uso principal del complemento a la base (**n**) y a la base disminuida (**n-1**)?
8. ¿Cuáles son las operaciones básicas para realizar una división binaria?

9. ¿Cuál es el procedimiento para convertir un número decimal con parte entera y parte decimal a base 2?
10. Demuestra que al aplicar el complemento del complemento se obtiene el valor original.

Examen de autoevaluación

Elige la opción que conteste correctamente cada una de las siguientes preguntas.

1.-¿Cual es el resultado de la siguiente conversión $(11100111.10011)_2$ a $()_{10}$?

a.) 132.59375
b.) 213.59573
c.) 312.37559
d.) 231.59375

2.- ¿Cuál es el resultado de la siguiente conversión $(543.09375)_{10}$ a $()_2$?

a.) 1000011111.00011
b.) 10001110.100101
c.) 100011110.00011
d.) 10001100.100011

3.- ¿Cuál es el resultado de realizar la siguiente operación?

$$\begin{array}{r} (110101.1110)_2 \\ +\ (1000101.1101)_2 \\ (101001.1101)_2 \\ \underline{(111011.1011)_2} \end{array}$$

a.) $(222.575)_{10}$
b.) $(225.075)_{10}$
c.) $(202.075)_{10}$
d.) $(232.575)_{10}$

4.- ¿Cuál es el resultado de realizar la siguiente operación?

$$\begin{array}{rl} & (F3)_{16} \\ x & (3B)_{16} \\ \hline \end{array}$$

a.) $(34100)_8$

b.) $(34101)_8$

c.) $(34001)_8$

d.) $(30410)_8$

5.- ¿Cuál es el resultado de realizar la siguiente operación?

$$\begin{array}{rl} & (34)_8 \\ x & (65)_8 \\ \hline \end{array}$$

a.) $(3CC)_{16}$

b.) $(5CC)_{16}$

c.) $(C3C)_{16}$

d.) $(CC5)_{16}$

6.- ¿Cuál es el resultado al realizar la siguiente operación?

$$(11)_2 \overline{)\ (1101111)_2}$$

a.) Cociente: $(100010)_2$, Residuo : $(1)_2$

b.) Cociente: $(111001)_2$, Residuo : $(1)_2$

c.) Cociente: $(110011)_2$, Residuo : $(0)_2$

d.) Cociente: $(100101)_2$, Residuo : $(0)_2$

7.- ¿Cuál es el resultado al realizar la siguiente operación?

$(4)_{16}$ ⟌ $(30D4)_{16}$

a.) Cociente: $(35C)_{16}$, Residuo : $(F)_{16}$

b.) Cociente: $(5C3)_{16}$, Residuo : $(2)_{16}$

c.) Cociente: $(C35)_{16}$, Residuo : $(0)_{16}$

d.) Cociente: $(53C)_{16}$, Residuo : $(5)_2$

8.) ¿Cual es el complemento de la base del número $(25.639)_{10}$

a.) $(64.036)_{10}$

b.) $(25.360)_{10}$

c.)$(74.360)_{10}$

d.) $(74.361)_{10}$

9.) ¿Cual es el complemento de la base del número $(101100)_2$

a.) $(10100)_2$

b.) $(010011)_2$

c.)$(011101)_2$

d.) $(010101)_2$

10.) Cual es el complemento de la base disminuida del numero $(25.639)_{10}$

a.) $(74.036)_{10}$

b.) $(54.360)_{10}$

c.)$(74.360)_{10}$

d.) 57.8603 $)_{10}$

11.) ¿Cual es el complemento de la base disminuida del número $(101100)_2$

a.) $(10001)_2$

b.) $(010011)_2$

c.)$(011011)_2$

d.) $(011100)_2$

12.) ¿Cual es el complemento de la base disminuida del número $(0.0110)_2$

a.) $(0.1001)_2$

b.) $(1.1001)_2$

c.)$(0.1101)_2$

d.) $(10.001)_2$

TEMA 3. CÓDIGOS

Objetivo particular

Presentar al alumno diferentes códigos para representar números, caracteres alfanuméricos en una computadora digital, así como también los códigos para la detección de errores durante la transmisión y/o el almacenamiento de la información en forma binaria.

Temario detallado

3.1 Códigos numéricos

- 3.1.1 Binario
- 3.1.2 BCD
- 3.1.3 Exceso-3
- 3.1.4 Gray

3.2 Códigos alfanuméricos

- 3.2.1 ASCII
- 3.2.2 BCDIC
- 3.2.3 EBCDIC

3.3 Códigos por detección de error

- 3.3.1 Paridad par
- 3.3.2 Paridad impar

Introducción

La codificación de la información es una manera de especificar tanto los caracteres numéricos como los alfabéticos empleando otros símbolos. Los códigos en las computadoras se utilizan para proporcionar un modo de expresar los caracteres (numéricos y alfabéticos) empleando solamente los símbolos binarios 1 y 0. La elección de alguno de los códigos binarios existentes depende de la función o uso a que se destina. Algunos códigos son más convenientes cuando han de efectuar operaciones aritméticas y algunos otros son utilizados para la detección de errores cuando se transmite y/o almacena información.

3.1 Códigos numéricos

Una computadora digital trabaja internamente con números discretos, generalmente las unidades de Entrada/Salida (a través de sus periféricos) reciben o envían información en forma decimal. Dado que la mayor parte de los circuitos lógicos solo aceptan señales discretas, los números decimales se pueden codificar en términos de señales binarias mediante diversos códigos como son: Código Binario, BCD, Exceso-3, etc., los cuales explicaremos a continuación.

3.1.1 Binario

El código binario es quizá uno de los códigos más utilizados en una computadora. La razón de esto obedece a la facilidad que representa construir cualquier “sistema” en base a solamente 2 dígitos: 0 y 1, los cuales, dentro de un “sistema”, son interpretados de diversas maneras, tales como: SI o NO, VERDADERO o FALSO, niveles Alto y Bajo de voltaje, OFF y ON, etc. Como se ve, el hecho de manejar tan sólo 2 dígitos hace posible la versatilidad que este código ha cobrado hoy en día. La tabla 3.1 especifica la codificación de caracteres numéricos utilizando el código binario.

Decimal	Código Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tabla 3.1 Código Binario

3.1.2 BCD (Código Decimal Codificado en Binario)

El código **BCD** se utiliza en las computadoras para representar los números decimales 0 a 9 empleando el sistema de numeración binario. Los números representados en código **BCD** se escriben utilizando ceros y unos. La tabla 3.2 especifica la codificación de caracteres numéricos.

Decimal	Decimal Codificado en Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tabla 3.2 Código BCD

A partir de la tabla 3.2, se observa que este código requiere el empleo de un carácter binario de cuatro posiciones (cuatro bits) para especificar el carácter de un dígito decimal. Evidentemente, este código es mucho menos eficiente que el sistema decimal, pero presenta la ventaja de especificar los caracteres mediante las cifras 0 y 1, que constituyen el lenguaje del computador, por lo que el código **BCD** puede ser utilizado en una computadora. Algunos ejemplos de representación de números decimales en este código son:

Decimal	BCD			
22			0001	0010
35			0011	0101
671		0110	0111	0001
2579	0010	0101	0111	1001

Puede verse que cada cifra decimal requiere un equivalente de cuatro bits codificado o “nibble” (palabra de 4 bits) en binario. Para especificar un número, el código **BCD** requiere más posiciones que el sistema decimal. Pero, por estar en notación binaria, resulta extremadamente útil. Otro punto que debe tenerse presente es que la posición de cada bit, dentro de los cuatro bits de cada cifra, es muy importante (como sucede en todo sistema de numeración posicional). Puede especificarse la ponderación de cada una de las posiciones y algunas veces se emplea para indicar la forma de codificación. El peso de la primera posición (situada a la derecha es 2^0=1, el de la segunda, 2^1=2; el de la tercera, 2^2= 4 y el de la cuarta, 2^3=8. Leyendo el número de la izquierda a derecha, la ponderación es 8-4-2-1, por lo que este código se denomina también un código **8421**.

Cabe aclarar, que este código (8421) no es el mismo que los números binarios, consideremos los casos siguientes:

Diez en **Binario** es 1010	Diez en **BCD** es 0001 0000.
Dieciséis en **Binario** es 10000	Dieciséis en **BCD** es 0001 0110.

La confusión entre los códigos **BCD** y **Binario** se origina debido a que son

exactamente iguales las nueve primeras cifras en **BCD** y en **Binario**. Después, los números son completamente diferentes.

La característica principal de la codificación **BCD** es análoga a la de de los números en el sistema octal; puede ser reconocida y leída fácilmente. Por ejemplo, compárense las representaciones **Binaria** y **BCD** leyendo los números en cada una de sus formas.

Decimal	Binario	BCD
141	10001101	0001 0100 0001
2179	100010000011	0010 0001 0111 1001

Sin embargo, cuando se utiliza esta forma de codificación en operaciones aritméticas se presentan dificultades adicionales. Veamos lo que sucede cuando se suman 8 y 7 en ambas formas (**Binario** y **BCD**).

Decimal	Binaria	BCD	
8	1000	1000	
+ 7	+ 0111	+ 0111	
15	1111	1111 →	No es un carácter aceptable en **BCD** (15 es 0001 001)

Para realizar operaciones aritméticas con el código **BCD** se necesitan sumadores especiales. Cuando se desea la propiedad de fácil reconocimiento y la manipulación aritmética, puede utilizarse un código modificado.

Los conceptos anteriores también son aplicables a números decimales con fracciones.

Por ejemplo exprese el número decimal 7324.269 en **BCD.**

7 - 0111
3 - 0011
2 - 0010
4 - 0100
2 - 0010
6 - 0110
9 - 1001

Por tanto

0111 0011 0010 0100 . 0010 0110 1001

7 3 2 4 . 2 6 9

Finalmente, las razonas del empleo de este código son:

a) Ahorra espacio al representar un número decimal,

b) Permite trabajar en forma binaria con un mínimo de espacio.

3.1.3 Exceso-3

Este código se deriva del **BCD**, y se obtiene sumando 3 al mencionado código. Este código es particularmente útil en la ejecución de operaciones aritméticas usando complementos. Al igual que el código **BCD** ponderado, este código sirve para representar números decimales a binarios, por grupos de 4 bits por cada dígito decimal.

La tabla 3.3 muestra las cifras decimales 0-9, el código **BCD** y el código de exceso en tres, que es una forma modificada del código **BCD**.

Decimal	BCD	Exceso en Tres
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Tabla 3.3 Código de Exceso en tres

Como su nombre lo indica, cada carácter codificado en exceso en tres es tres unidades mayor que en **BCD**. Así, seis ó 0110 se escribe 1001, que es nueve en **BCD**. Ahora bien, 1001 solamente es nueve en BCD, en el código de exceso en tres, 1001 es seis.

Ejemplo

Decimal			Exceso en tres	
2				0101
25			0101	1000
629		1001	0101	1100
3271	0110	0101	1010	0100

El código de exceso en tres facilita la operación aritmética, es decir,

```
   3                    0110
+  9                 +  1100
-----                ------
  12                  1  0010
                   +
                       0011  0011
                    ----------
                       0100  0101
```

Se lee 0100 0101 ó 12 (exceso en tres).

Existen algunas reglas especiales aplicables a la suma (como la adición de 3 a cada uno de los números del ejemplo anterior), pero estos pasos se realizan fácilmente, y de modo automático, en la computadora, haciendo del código de exceso en tres muy conveniente para las operaciones aritméticas. En el código de exceso en tres, el reconocimiento de la representación de las cifras no es directo, ya que al leer cada dígito debe restarle mentalmente en tres, si bien ello resulta más fácil que la conversión de números grandes representados en el sistema binario puro.

Ya hemos indicado que el **BCD** es un código ponderado; el de Exceso en Tres no lo es. Un bit de la segunda posición (2) de **BCD** representa un 2. En el código de exceso en tres, un bit situado en una cierta posición no indica la adición de un valor numérico al número. Por ejemplo, en **BCD**, 0100 es 4 y al sumarle el bit 2 se añade un 2, resultando el número 0110, o sea, dos unidades mayor. En el código de Exceso en tres, 0111 representa la cifra 4 y la cifra 6 es 1001, no existiendo un cambio numérico sistemático.

3.1.4 Código Gray

El código **Gray** es uno de los códigos cíclicos más comunes y esto es debido a las siguientes características:

- Cambia solamente uno de sus bits al pasar a la siguiente posición, es decir, el cambio entre dos números progresivos es de un bit.
  Por esta característica este código es empleado con frecuencia en la detección de errores, como veremos en el tema 3.3.
- Facilita la conversión a la forma binaria.

Además, este código suele emplearse en codificadores de desplazamiento angular con el eje óptico o mecánico, es decir, emplea un tipo de rueda codificadora que presenta posiciones sucesivas, cubriendo la superficie de un disco, cada una de las cuales está representada por una nueva palabra; el

código de Gray admite ambigüedad en una posición.

En la tabla 3.4 se muestra la equivalencia para los números decimales 0 a 15, del código Gray, el sistema decimal y del binario puro. A todo número binario le corresponde una representación en el código Gray, por lo que la lista de equivalencias indicadas sólo tiene carácter ilustrativo.

Decimal	Binario	Código Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Tabla 3.4 Código Gray

A partir de la tabla 3.4 se puede observar que entre cada dos palabras cualesquiera sucesivas del código Gray, solamente cambia un bit. Esto no ocurre en el sistema binario; al pasar del decimal 7 al 8, cambian los cuatro bits del código binario, mientras que solamente cambia un bit en el código Gray. Al pasar de decimal 9 al 10, y el 2 de 0 a 1, es decir, se producen dos cambios, mientras que en el código Gray se pasa de 1101 a 1111 con un sólo cambio, el del bit 2^1 de 0 a 1.

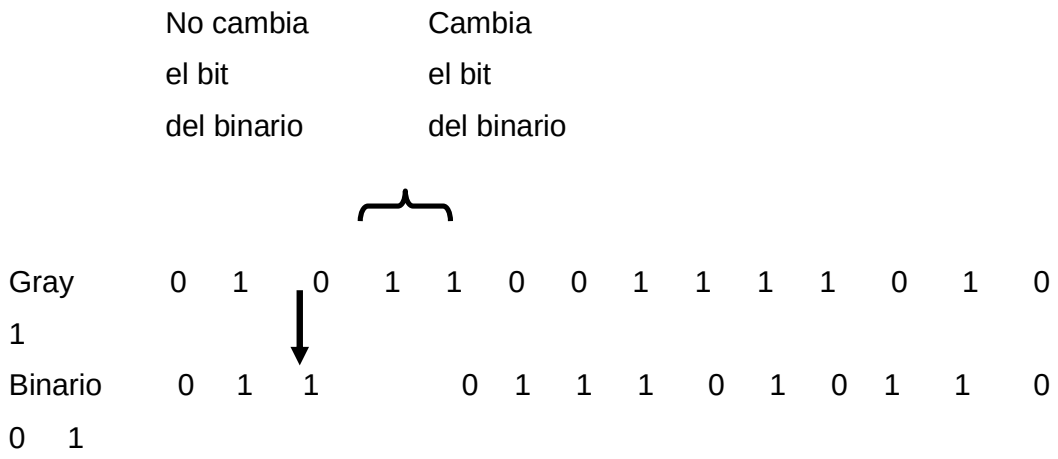
Para convertir la palabra representada en código Gray a su forma binaria, debe empezarse primeramente por la conversión del bit más significativo (**BMS**). En binario, el bit menos significativo es el 2^0 y el bit más significativo es el más alto en la posición ponderada (para cuatro bits, es el 2^3). Un ejemplo de un número binario y de su forma Gray equivalente es estando el **BMS** a la izquierda.

Gray	1	011010111001
Binario	1	101100101110

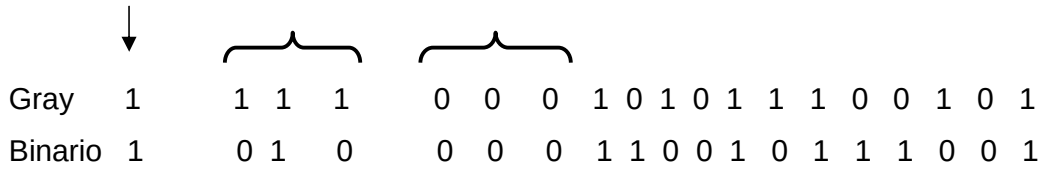
Para hacer la conversión, se repite en la forma binaria el mismo bit que aparece en la forma Gray hasta alcanzar el primer 1, que se repite también. En nuestro ejemplo, la forma Gray empieza por 1, el cual se repite como primer bit (**BMS**) de la forma binaria. Se sigue repitiendo este bit, en el código binario, esperando que los siguientes bits, en la forma Gray, sean 0 (una posición en el ejemplo).

Para cada 1 que aparezca a continuación (después del primero) en la forma de Gray, se cambia el bit correspondiente, de la palabra codificada en binario, respecto del que le precede en la forma binaria. El segundo 1 de la forma Gray indica cambio de bit en la forma binaria; como anteriormente era 1 pasa a ser 0. De acuerdo con esta regla, el siguiente 1 de la forma Gray indica el cambio del bit anterior (0) a 1. El siguiente 0 de la forma Gray significa que se mantiene el bit precedente, de la forma binaria, repitiéndose de nuevo el 1. Este procedimiento se reitera para el resto de la palabra.

Ejemplo



Ejemplo



Cuando aparece un 1 en la forma Gray se produce un cambio del bit precedente de la forma binaria (cualquier que fuese); un 0 mantiene el bit de la forma binaria como estaba.

El procedimiento para convertir una palabra binaria en su forma Gray es sencillo. La regla cosiste en comparar cada par de bits sucesivos (empezando con el Bit Más Significativo). Si son iguales se escribe un 0 en la forma Gray y si son diferentes se escribe un 1. Al empezar la comparación el primer bit se compara con cero.

Ejemplo

Binario	0	1	1	0	1	0	1	1	1	1	0	1	1	0	0	1	0	1	0
Gray	0	1	0	1	1	1	1	0	0	0	1	1	0	1	0	1	1	1	1

Ejemplo

Binario	0	1	1	1	0	1	0	1	0	1	0	1	1	1	1	1
Gray	0	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0

Nota

El resultado se obtiene al sumarse los dos bits, en módulo dos (sin arrastre), y el resultado se coloca debajo, como bit de la forma Gray.

3.2 Códigos alfanuméricos

La información a procesar por una computadora está formada por letras del alfabeto, números decimales, caracteres especiales u órdenes, las cuales han de codificarse en binario. Para representar los dígitos decimales se necesita un código de cuatro bits (como sucede en el código **BCD**). Pero para representar estos dígitos (0 - 9), más las 26 letras del alfabeto, más algunos caracteres especiales, se necesita un código de por lo menos, seis bits (2^6 = 64 combinaciones). Para tal representación se utilizan los códigos **ASCII**, **BCDIC** y **EBCDIC** los cuales explicaremos a continuación.

3.2.1 ASCII

El código **ASCII** (**American Standard Code for Information Interchange**) es un código normalizado que está siendo muy aceptado por los fabricantes de computadoras. Este código ocupa ocho bits con los cuales se permite la representación de los números decimales (0-9), caracteres alfabéticos (letras minúsculas y mayúsculas), signos especiales (por ejemplo, *,+,=, etc.) y más

de treinta órdenes o instrucciones de control (por ejemplo, comienzo de mensaje, final de mensaje, retorno de carro, cambio de línea, etc.)

La tabla 3.5 muestra la representación de los números decimales, caracteres alfabéticos y algunos caracteres especiales en código ASCII. La numeración convenida para el código ASCII establece una secuencia de izquierda a derecha, de tal modo que la posición del bit 7 es la posición del bit de orden más elevado. Esta misma codificación puede emplearse para impresoras de alta velocidad, ciertos teletipos, etc.

Carácter	Código ASCII	Código EBCDIC	Carácter	Código ASCII	Código EBCDIC
Blanco	P010 0000	0100 0000	A	P100 0001	1100 0001
.	P010 1110	0100 1011	B	P100 0010	1100 0010
(	P010 1000	0100 1101	C	P100 0011	1100 0011
+	P010 1011	0100 1110	D	P100 0100	1100 0100
S	P010 0100	0100 1011	E	P100 0101	1100 0101
*	P010 1010	0100 1101	F	P100 0110	1100 0110
)	P010 1001	0110 0000	G	P100 0111	1100 0111
	P010 1101	0110 0001	H	P100 1000	1100 1000
/	P010 1111	0110 1011	I	P100 1001	1100 1001
‘	P010 1100	0111 1101	J	P100 1010	1101 0001

	P010 0111	0111 0001	K	P100 1011	1101 0010
=	P010 1101	0111 0001	L	P100 1100	1101 0011
0	P010 0000	1111 0000	M	P100 1101	1101 0100
1	P010 0001	1111 0001	N	P100 1110	1101 0101
2	P010 0010	1111 0010	O	P100 1111	1101 0110
3	P010 0011	1111 0011	P	P101 0000	1101 0111
4	P010 0100	1111 0100	Q	P100 0001	1101 1000
5	P010 0101	1111 0101	R	P100 0010	1101 1001
6	P010 0110	1111 0110	S	P100 0010	1110 0001
7	P010 0111	1111 0111	T	P100 0100	1110 0010
8	P010 1000	1111 1000	U	P100 0101	1110 0011
9	P010 1001	1111 1001	V	P100 0110	1110 0101
			W	P100 0111	1110 0110
			X	P100 1000	1110 0111
			Y	P100 1001	1110 1000
			Z	P100 1010	1110 1001

Tabla 3.5 Códigos ANSCII y EBCDIC

donde

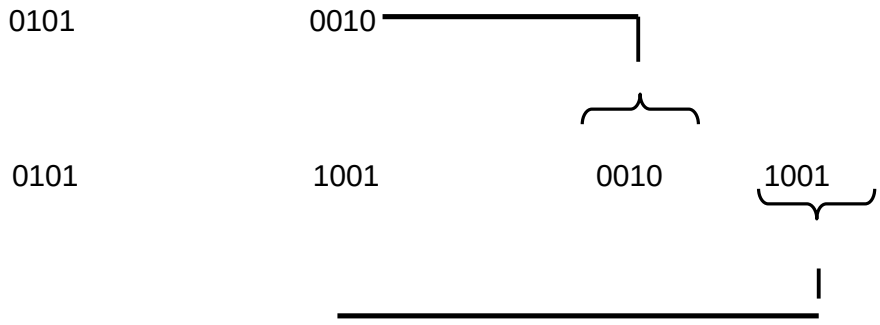
P es un bit de paridad.

Nota

Cuando se transmiten o almacenan datos binarios, frecuentemente se añade un bit adicional (denominado bit de paridad) el cual se utiliza en la detección de errores, ver sección (3.3).

En los equipos de entrada/salida se utiliza el código de ocho niveles para representar los caracteres. Una vez introducidos en la computadora, los caracteres pueden ser tratados del modo más conveniente para las distintas operaciones. Por ejemplo, las cifras decimales no necesitan mantenerse en la computadora como palabras de ocho bits. Para las cifras 0-9 se han elegido, intencionadamente, los 4321 en coincidencia con la forma codificada **BCD**. Al eliminar los bits 765X, la computadora solo necesita conservar los cuatro bits de la forma **BCD** para representar las cifras decimales. Si la longitud de la palabra normaliza es de ocho bits, como hemos indicado, la computadora puede procesar internamente las cifras decimales agrupando en una palabra los dos dígitos **BCD** de cuatro bits.

Por ejemplo, si deseamos agrupar en una palabra de ocho bits dos dígitos codificados en **BCD.** El número 29, que en las unidades de entrada/salida se representa como 01010010 01011001, puede reagruparse en la computadora en la forma



que forma una palabra de ocho bits con dos cifras decimales

Los diseñadores de computadoras han utilizado el término descriptivo "**byte**" a un grupo de 8 bits que pueden representar una palabra o algunos caracteres. La información se procesa por bytes en lugar de por bits, en el interior de una computadora. El byte de ocho bits (octeto) podría utilizarse para representar un carácter ASCII, al introducirlo en el computador o para representar dos cifras decimales en las operaciones aritméticas. De este modo las palabras pueden representarse mediante un número prefijado de bytes.

Nota

Un ***bit***, por definición, es un digito binario y el cual puede tomar el valor de “0” o “1”.

Por estar íntimamente ligado al byte u octeto (y por consiguiente a los enteros que van del 0 al 127), el problema que presenta es que no puede codificar más que 128 símbolos diferentes (128 es el número total de diferentes configuraciones que se pueden conseguir con 7 dígitos binarios (0000000, 0000001,..., 1111111), usando el octavo dígito de cada octeto (como bit de paridad) para detectar algún error de transmisión). Un cupo de 128 es suficiente para incluir mayúsculas y minúsculas del abecedario inglés, además de cifras, puntuación, y algunos "caracteres de control" (por ejemplo, uno que permite a una impresora que pase a la hoja siguiente), pero el ASCII no incluye ni los caracteres acentuados ni el comienzo de interrogación que se usa en castellano, ni tantos otros símbolos (matemáticos, letras griegas,...) que son necesarios en muchos contextos y por esto que se propuso el Código **ASCII Extendido**. Debido a las limitaciones del ASCII se definieron varios códigos de caracteres de 6 u 8 bits entre ellos el código **BCDIC** (6 Bits) y el código **ASCII Extendido**. El código ASCII Extendido es un código de 8 bits que complementa la representación de caracteres faltantes del código **ASCII** estándar.

Sin embargo, el problema de estos códigos de 8 bits es que cada uno de ellos se define para un conjunto de lenguas con escrituras semejantes y por tanto no dan una solución unificada a la codificación de todas las lenguas del mundo. Es decir, no son suficientes 8 bits para codificar todos los alfabetos y escrituras del

mundo, por lo tanto hay que buscar otros códigos de codificación más eficientes. Una posible solución al problema de la codificación de todas las lenguas del mundo es utilizar el código Unicode, como lo vamos describir en la siguiente sección

Código Unicode

Como solución a estos problemas, desde 1991 se ha acordado internacionalmente utilizar la norma **Unicode**, que es una gran tabla, que en la actualidad asigna un código a cada uno de los más de cincuenta mil símbolos, los cuales abarcan todos los alfabetos europeos, ideogramas chinos, japoneses, coreanos, muchas otras formas de escritura, y más de un millar de símbolos especiales.[1]

3.2.2 BCDIC

El código **BCDIC (del idioma inglés, Standard Binary Coded Decimal Interchange Code)** usualmente utiliza 6 bits de codificación (2^6 = hasta 64 caracteres) y en ocasiones se adhiere un bit adicional para verificar posibles errores de transmisión o grabación.

Un inconveniente de este código es que con 6 bits son insuficientes para representar (codificar) los caracteres alfabéticos, numéricos, símbolos especiales, etc. Debido a este inconveniente se propuso el **BCDIC extendido**, el cual explicaremos a continuación.

3.2.3 EBCDIC

El código de **Intercambio BCD Extendido** -**EBCDIC-** (del idioma inglés, **Extended BCD Interchange Code)** es un código de 8 bits y se utiliza para representar hasta 256 caracteres (símbolos) distintos. En este código, el bit menos significativo es el **b7** y el más significativo es el **b0**. Por consiguiente, en

[1] Wikipedia: “Codificación de caracteres”, actualizado el 28/01/09, disponible en: http://es.wikipedia.org/wiki/Codificaci%C3%B3n_de_caracteres, recuperado el 23/02/09.

el código **EBCDIC** se transmite primero el bit de mayor orden, **b7**, y al último se transmite el bit de menor orden, **b0**. Este código no facilita el uso de un bit de paridad.

La tabla 3.5 muestra la representación (codificación) de los caracteres con el código **EBCDIC** y su comparación con el código ASCII. Con respecto a la columna del código EBCDIC, las letras mayúsculas de la **A** a la **Z**, se dividen en tres grupos (A-I), (J-R), (S-Z) y en las primeras cuatro posiciones se identifica el grupo al cual pertenece la letra y en las restantes cuatro posiciones el dígito correspondiente a la posición de la letra en el grupo. Los dígitos del 0 al 9 se identifican con un uno en las primeras cuatro posiciones y en las restantes cuatro posiciones el dígito en binario.

3.3 Códigos por detección de error

La detección y/o corrección de errores es un campo de estudio de mucho interés y aplicación creciente en la transmisión, codificación, compresión y almacenamiento de datos digitales debido a las limitaciones del canal de transmisión.

Si en el envío de una información, por ejemplo 1000, por efecto del canal de transmisión, se recibe como 1001, ambos números pertenecen al mismo código y no será posible saber si ha habido algún error en la información que se recibe. Una medida que se toma para detectar si hubo algún error, es la de agregar a cada símbolo o carácter alfanumérico un bit a la izquierda del mismo, dicho bit recibe el nombre de "Bit de paridad". Estos bits pueden ser de paridad, par o impar según el método de verificación de paridad que se tenga y los explicaremos a continuación.

3.3.1 Paridad par

La paridad par consiste en verificar que la suma de todos los 1´s existentes en un carácter alfanumérico o un símbolo es par, incluyendo en dicha suma el bit de paridad. De lo anterior se desprende, que un bit de paridad par será aquel

que asumirá el estado de 1 si la suma de los 1´s restantes es impar y será 0 si la suma de los 1´s en el carácter es par.

La verificación de paridad par deberá cumplir con la ecuación siguiente:

$$X^n + X^{n-1} + X^{n-2} + \ldots + X^1 + X^0 + P = 0$$

donde

p es el bit de paridad y las X^n son los bits que forman el carácter.

3.3.2 Paridad impar

La paridad impar se obtiene verificando que la suma de todos los 1´s existentes en un carácter alfanumérico o un símbolo, incluyendo al bit de paridad, sea un número impar, y por tanto un bit de paridad impar será aquel que sumando a los 1´s restantes deberá producir un número impar por tanto este bit será 1 si la suma de los 1´s restantes es impar.

La verificación de paridad impar deberá cumplir con la ecuación siguiente:

$$X^n + X^{n-1} + X^{n-2} + \ldots + X^1 + X^0 + P = 1$$

donde

P es el bit de paridad y las X^n son los bits que forman el carácter.
Puesto que **P** puede tomar los valores de 0 ó 1, se puede demostrar que:

$$P(PAR) = X^n + X^{n-1} + X^{n-2} + \ldots + X^1 + X^0 + 1$$

$$P(IMPAR) = X^n + X^{n-1} + X^{n-2} + \ldots + X^1 + X^0$$

Por otro lado, el bit de paridad también se puede aplicar a otros códigos como se muestra a continuación.

Por ejemplo, si modificamos el código BCD mediante un bit de paridad. Este bit se añade a la derecha de la posición 2^0. En un código con **paridad par**, el bit de paridad añadido hace **par** el número total de **1´s** y en un código de **paridad impar**, se selecciona el bit de paridad de modo que haga **impar** el número total de **1´s**. Cuando se recibe una palabra codificada, se compara su paridad (par o impar, según haya sido elegida previamente) y se acepta como correcta si pasa la prueba. La tabla 3.6, muestra los códigos **BCD**, **BCD con paridad impar** y **BCD con paridad par**.

Decimal	BCD	BCD con paridad impar	BCD con paridad par
0	0000	0000 1 o sea 00001	0000 0 o sea 00000
1	0001	0001 0 00010	0001 1 00011
2	0010	0010 0 00100	0010 1 00101
3	0011	0011 1 00111	0011 0 00110
4	0100	0100 0 01000	0100 1 01001
5	0101	0101 1 01011	0101 0 01010
6	0110	0110 1 01101	0110 0 01100
7	0111	0111 0 01110	0111 1 01111
8	1000	1000 0 10000	1000 1 10001
9	1001	1001 1 10011	1001 0 10010

Tabla 3.6 Paridad en el código BCD

Ejemplo Verifique la existencia de errores en las siguientes palabras, codificadas en **BCD** con paridad par.

Solución

	Palabra	Bit de paridad	Tipo de paridad
a)	1001	0	paridad impar
b)	1000	0	paridad impar
c)	0001	0	paridad par
d)	0110	1	paridad impar

Los ejemplos (a) y (c) son incorrectos y los (b) y (d), correctos.

La paridad, también se puede utilizar en otros códigos distintos del código BCD. Cuando se envía un conjunto de palabras con paridad añadida, el bit de paridad se elige de manera similar.

Ejemplo Verifique la existencia de errores en las siguientes palabras.
Solución

	Palabra	Bit de paridad	Tipo de paridad
a)	0110111101	1	paridad par
b)	1101110100	0	paridad impar
c)	1110111011	0	paridad impar
d)	1011011100	0	paridad par
e)	1010111010	1	paridad impar

Los ejemplos (b) y (c) son incorrectos: (a), (d) y (e) son correctos,

Los códigos **BCD**, **BCD con paridad impar y BCD con paridad par** no son los únicos códigos para la detección de errores. Existen otros códigos para la detección de errores de un solo bit. Entre los cuales se encuentra el código Biquinario. El código biquinario es un código ponderado de 7 bits cuya distancia mínima es dos y permite la detección de errores de un bit, como se explicará a continuación.

Código Biquinario

Para facilitar la comprobación de posibles errores cuando se transmiten datos binarios, se puede utilizar el código biquinario o la adición de un bit de paridad a cada carácter codificado. Hasta ahora se han empleado otros códigos, dependiendo de la elección del grado de fidelidad requerido, de la cantidad de información que puede enviarse y de la cuantía del equipo transmisor y receptor necesario para realizar las operaciones de comprobación, pero no un

código para la detección de errores, el cual explicaremos a continuación.

El código biquinario es un código ponderado y consta de 7 bits, de los cuales, los 2 de la izquierda y los 5 de la derecha se consideran partes separadas del conjunto. La tabla 3.7 muestra las formas codificadas del 0 a 9, así como la ponderación de cada una de las posiciones de sus bits.

Decimal	Biquinario						
	5	0	4	3	2	1	0
0	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0
2	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	1	0	0
8	1	0	0	1	0	0	0
9	1	0	1	0	0	0	0

Tabla 3.7 Código Biquinario

En esta tabla se puede observar que se necesitan siete bits para especificar una cifra decimal (mientras que en **BCD** o **Exceso en tres** se requieren cuatro bits). El código biquinario presenta, como ventaja importante, la propiedad intrínseca de indicar cuándo existe error en la palabra codificada. En general, cuando se transmite información de un lugar a otro, como sucede en el computador, resulta muy conveniente el empleo de un código que permita determinar si se ha producido un error en la transmisión.

Analizando el código biquinario de la tabla 3.7 observamos lo siguiente: cada palabra solamente tiene dos 1´s. Por consiguiente, si apareciera cualquier otro 1 extra en la respuesta significaría que se había producido un error y la palabra no debería ser aceptada. Si solamente se hubiese recibido un 1, de nuevo

sería evidente la existencia de error. Además, el reconocimiento y aceptación de una palabra, como correcta, exige que haya un solo bit entre los dos primeros de la izquierda y que haya también un solo bit entre los cinco restantes de la derecha. La comprobación se establece fácilmente, debido a que es fácil realizar un circuito que compruebe la existencia de un 1 entre dos bits y de otro circuito que detecte la presencia de un 1 entre cinco bits.

Por ejemplo: determine la existencia de errores en el siguiente grupo de palabras codificadas en biquinario.

		Biquinario	Decimal
a.)		01 10001	4
b.)		01 10010	5
c.)		10 10101	6
d.)		11 00010	6
e.)	01 01000	01 00010	31
f.)	10 10000	10 00010	31

Los ejemplos (a) y (d) son incorrectos, mientras que los (e) y (f) son correctos.

Bibliografía del tema 3

Ayala Pérez, Jaime, *Aritmética para Computadoras*, México ENEP. Aragón, UNAM, (colección de Cuadernos de la ENEP ARAGÓN No. 33), 1989.

Ayala Pérez, Jaime, *Arquitectura de Computadoras*, México, Comunicación interna, 2005.

Charles H Roth Jr., *Fundamentals of Logic Design*, 4ª ed., USA, West Publishing Company, 1992.

Hayes P. John, *Diseño de Sistemas Digitales y Microprocesadores*, MEX.,McGraw Hill, 1986.

Lee, Samuel C, *Digital circuits and logic design,* USA, PRENTINCE HALL, Inc., Englewood Cliffs, N.J., 1976.

Mandado, E., *Sistemas Electrónicos Digitales*, Madrid, Marcombo Boixareu, 1980.

Mano, Morris, M., *Arquitectura de Computadoras*, México, Prentice Hall Hispanoamericana, 1988.

Mano, Morris, M., *Diseño Digital*, México, Prentice Hall Hispanoamericana, 1987.

Nashelky, Louis, *Teoría de las calculadoras numéricas automáticas, Madrid,* Alhambra, 1979.

Tocci, Ronald, *Digital Design, Principles and Applications*, EE.UU., Prentince Hall, 1977.

Actividades de aprendizaje

A.3.1 Escribe en código BCD el número $(7351)_{10}$.

A.3.2 Escribe en código BCD los números decimales desde 10 a 31.

A.3.3 Escribe en código BCD el número $(28897)_{10}$.

A.3.4 Escribe en código exceso tres el número decimal 7514.

A.3.5 Escribe en código biquinario el número $(2159)_{10}$.

A.3.6 Escribe en código biquinario el número BCD 1001 1000 0101. $(2573)_{10}$.

A.3.7 Convierte el número biquinario 1101101011 a código de Gray.

A.3.8 Convierte el número 110101101, representado en código de Gray, en su equivalente binario.

A.3.9 Escribe $(5712)_{10}$ en BCD con paridad PAR.

A.3.10 Escribe $(5191)_{10}$ en código de exceso en tres con paridad IMPAR

A.3.11 Escribir 37 empleando el código ASCII.

Ejercicios

1. Escribe en código BCD los siguientes números decimales:

a) 1773.

b) 19221

c) 1001

d) 5.1279

2. Escribe los números siguientes en el código de Exceso en tres.

a) 279

b) 301

c). 2176

d) 1568

e) 769

3. Convierte las siguientes palabras codificadas en código Gray a su forma binaria.

a) 1 0 0 1 1 1 0 1 0 1 1 0 1

b) 1 0 1 0 1 0 1 0 1 0 1 0

c) 0 1 1 1 0 1 1 1 0 1 1 1 1 0

d) 1 0 0 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 1

e) 1 1 1 1 1 1 1 1 1 1 1 1 1

4. Conviértanse los números binarios dados a código Gray.

a) 0 1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 1

b) 1 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 0

c) 1 0 1 0 0 1 0 1 1 0 1 0 0 1

d) 1 1 1 1 1 1 1 1 1 1

e) 0 0 1 0 1 0 1 1 1 0 1 1 1

5. Escribir 137 empleando el código EBCDIC

6. Escribir en código biquinario el número BCD 1101 1010 0101.

7. Convierte el número biquinario 10 01110 10 11101 a Código de Gray.

8. Convierte el número 1010101101, representado en código de Gray, en su equivalente binario.
9. Escribe $(3719)_{10}$ en BCD con paridad PAR.
10. Escribe $(7195)_{10}$ en código de exceso en tres con paridad IMPAR

Cuestionario de autoevaluación

1. ¿Qué es un código?
2. ¿Cuáles son los códigos para la codificación de números?
3. ¿Cuáles son los códigos para la representación de números letras, caracteres especiales y caracteres de control?
4. ¿Qué es un código BCD?
5. ¿Qué es un código Gray?
6. ¿Qué es el bit de paridad?
7. ¿Qué es paridad par?
8. ¿Qué es paridad impar?
9. ¿Cuáles son los códigos para la detección de errores?
10. ¿En qué consiste un código biquinario?

Examen de Autoevaluación

Relaciona las columnas escribiendo dentro del paréntesis el número que una los conceptos con sus respectivas definiciones.

1) EBCDIC		( ) Código ponderado de 7 bits cuya distancia mínima es dos y permite la detección de errores de un bit.
2) Código ASCII		( ) Código usualmente de 6 bits de codificación (2^6 = hasta 64 caracteres) al cual se le adhiere un bit adicional para verificar posibles errores de transmisión o grabación.
3) Bit de paridad		( ) Código derivado del código **BCD**,
4) BCDIC		( ) Código más utilizados en una computadora, debido a la facilidad de construir cualquier “sistema” en base a solamente 2 dígitos: 0 y 1.
5) Código Gray		( ) Código normalizado que es aceptado por los fabricantes de computadoras y el cual ocho bits con los cuales se permite la representación de los números decimales (0-9), caracteres alfabéticos (letras minúsculas y mayúsculas), signos especiales (por ejemplo, *,+,=, etc.) y más de treinta órdenes o instrucciones de control.
6) Código biquinario		( ) Código utilizado en las computadoras para representar los números decimales 0 a 9 empleando el sistema de numeración binario.
7) Código Unicode		( ) Es el código cíclico más común, debido a que cambia solamente uno de sus bits al pasar a la siguiente posición, es decir, el cambio entre dos números progresivos es de un bit y además facilita la conversión a la forma binaria.
8) Código Exceso-3		( ). Bit adicional el cual se utiliza en la detección de errores.
9) BCD		( ) Código utilizado para la codificación de todos los alfabetos del mundo.
10) Código Binario		( ) Código de 8 bits y se utiliza para representar hasta 256 caracteres (símbolos) distintos, donde el bit menos significativo es **b7** y el más significativo es **b0**.

TEMA 4. ÁLGEBRA DE BOOLE

Objetivo particular

Al finalizar el tema, el alumno reconocerá los principios básicos de la electrónica digital, los diferentes tipos de compuertas lógicas y las técnicas de minimización (como son el álgebra de Boole y los mapas de Karnaugh) para funciones booleanas utilizadas para el diseño de circuitos combinatorios y circuitos secuenciales.

Temario detallado

4 .1 Principios de electrónica básica

- 4.1.1 Lógica binaria

4. 2 Propiedades fundamentales del álgebra de Boole

- 4.2.1 Leyes de Morgan
- 4.2.2 Compuertas Lógicas
- 4.2.3 Función booleana

4.3 Técnicas de minimización de funciones

- 4.3.1 Proceso algebraico
- 4.3.2 Mapas de Karnaugh

Introducción

El Álgebra de Boole representa las matemáticas necesarias para el análisis, construcción y síntesis de una función booleana. Dichas matemáticas únicamente trabajan con dos valores binarios (cero “0” lógico y uno “1” lógico) y son presentadas en forma de leyes y teoremas. Estas leyes y teoremas comprenden las leyes de de Morgan, las leyes asociativas, conmutativas y distributivas (para los operadores OR, AND y NOT), etc., las cuales logran su mayor potencial cuando se combinan con la electrónica (digital) básica.

4.1 Principios de electrónica básica

La electrónica se dedica al análisis y síntesis de circuitos electrónicos. La electrónica se puede dividir en tres áreas: Analógica, Digital e Industrial. La Electrónica Digital es aquella que trabaja con señales eléctricas discretas, esta señal únicamente tiene dos valores: cero (“0”) lógico y uno (“1”) lógico. La electrónica digital es la herramienta principal para el diseño y construcción de algunas unidades que constituyen una computadora digital, por ejemplo, el decodificador, el multiplexor, la unidad aritmética-lógica, etc., (ver tema 5) y en el diseño de circuitos secuenciales basados en flip-flops, (ver tema 6).

4.1.1 Lógica binaria

La electrónica digital utiliza dos estados: cero “0” lógico o uno “1” lógico. A la vez que dicha electrónica trabaja de dos formas: Lógica Positiva o Lógica Negativa.

La Lógica positiva define al “0” lógico como falso y al “1” como verdadero.
La Lógica negativa define al “0” lógico como verdadero y al”1” como falso.

4. 2 Propiedades fundamentales del álgebra de Boole

El álgebra de Boole es la técnica matemática empleada en el estudio de problemas de naturaleza lógica. Con el desarrollo de las computadoras, el empleo del álgebra de Boole se ha incrementado en el campo de la electrónica digital hasta alcanzar la posición que actualmente ocupa, siendo utilizada por los ingenieros como ayuda para el diseño y construcción de circuitos lógicos combinacionales y/o secuenciales. En el campo de las computadoras, el álgebra de Boole se emplea para describir circuitos cuyo estado puede caracterizarse por 0 ó 1. Los signos lógicos 1 ó 0 pueden ser los números base del sistema de numeración binario. También pueden identificarse con las condiciones de “abierto” o ”cerrado” o con las condiciones de ”verdadero” o ”falso”, que son de naturaleza binaria.

Puesto que las variables booleanas pueden adoptar dos valores y, por tanto cualquier incógnita puede ser especificada con 0 ó 1, el álgebra de Boole resultará sencilla en comparación en donde las variables son continuas.

4.2.1 Leyes de de Morgan

El álgebra de Boole se apoya en un conjunto de teoremas y leyes que permiten diseñar y construir circuitos combinacionales y secuenciales más sencillos. Dicho conjunto de teoremas y leyes se resumen en la tabla 4.1

Relación	Dual	Propiedad
$AB = BA$	$A + B = B + A$	Conmutativa
$A(B+C) = AB + AC$	$A + BC = (A+B)(A+C)$	Distributiva
$1A = A$	$0 + A = A$	Identidad
$A\,\overline{A} = 0$	$A + \overline{A} = 1$	Complemento
$0A = 0$	$1 + A = 1$	Teoremas del cero y el uno
$AA = A$	$A + A = A$	Idempotencia
$A(BC) = (AB)C$	$A+(B+C) = (A+B) + C$	Asociativa
$\overline{\overline{A}} = A$		Involución
$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$	Teorema de DeMorgan
$AB + \overline{A}C + BC = AB + \overline{A}C$	$(A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C)$	Teorema del concenso
$A(A+B) = A$	$A + AB = A$	Teorema de absorción

Tabla. 4.1 Teoremas de Algebra de Boole

En el álgebra de Boole, una variable binaria puede adoptar el valor de cero (“0”) lógico o uno (“1”) lógico. Estos valores se relacionan con los valores de 0 y 5 Volts (lógica positiva). La asignación puede invertirse en términos de las tensiones asignadas al 0 y al 1, es decir, asigna al cero (“0”) lógico el valor de 5 Volts y al uno “1” lógico el valor de 0 Volts (lógica negativa). A fin de comprender el correcto funcionamiento de los circuitos digitales, únicamente

utilizaremos los valores lógicos (“0” lógico y “1” lógico) en lugar de los valores físicos (0 Volts y 5 Volts).

Las leyes de de Morgan y los teoremas del álgebra de Boole se utilizan para reducir una función booleana, como se explicará en la sección 4.3. (Técnicas de minimización de funciones), a continuación daremos una breve explicación del uso de las leyes de Morgan.

Las leyes de Morgan son:

1.) $\overline{A + B + C + \ldots} = \bar{A}\,\bar{B}\,\bar{C}\ldots$

2.) $\overline{A\ B\ C\ldots} = \bar{A} + \bar{B} + \bar{C} + \ldots$

Para poder utilizar de manera correcta las leyes de Morgan se debe aplicar los siguientes pasos:

Paso 1

Se intercambia el operador OR (+) por el operador AND () o si es el caso intercambiar el operador AND () por el operador OR (+).

Paso 2

Se niegan cada una de las variables.

Paso 3

Se niega todo el término.

Para comprender la aplicación de los pasos mencionados anteriormente, demostraremos la segundo ley de de Morgan, únicamente para el caso de 2 variables.

$$\overline{A\ B} = \overline{A + B}$$

Paso 1. Intercambio del operador AND () por el operador (+).

$$\overline{A\ B} \quad => \quad \overline{A + B}$$

La negación del término en este paso se sigue conservando debido a que únicamente se intercambio el operador.

Paso 2. Se niega cada una de las variables

$$\overline{A\ B} \quad => \quad \overline{\bar{A} + \bar{B}}$$

Paso 3. Se niega todo el término

$$\overline{A\ B} \quad => \quad \overline{\overline{\bar{A} + \bar{B}}}$$

Aplicando la propiedad de involución (ver tabla 4.1) al resultado anterior

$$\overline{A\ B} \quad => \quad \overline{\overline{\bar{A} + \bar{B}}}$$

con lo cual se obtiene el resultado

$$\overline{A\ B} \quad = \quad \bar{A} + \bar{B}$$

y de esta manera queda demostrado la segunda ley de de Morgan.

Finalmente, las leyes de de Morgan se pueden utilizar para cualquier número de variables, siempre y cuando se tomen dos variables a la vez.

4.2.2 Compuertas Lógicas

Una compuerta lógica es un dispositivo físico que implementa una función básica del álgebra de Boole. La electrónica digital utiliza tres **compuertas básicas** como son: la compuerta **OR**, **AND** y **NOT** (ver figura 4.1) y a partir de estas compuertas se crean **compuertas complementarias** como son: **NAND**, **NOR**, **OR-exclusiva** y **NOR-exclusiva**, las cuales explicaremos a continuación:

- **Compuerta OR**

La figura 4.1a muestra el símbolo lógico, la tabla de verdad y la ecuación característica de la compuerta OR. La compuerta OR presenta en su salida un nivel alto, si cualquiera de sus entradas A o B están en nivel alto. La salida tiene un nivel bajo si todas las entradas tienen un nivel bajo o “0”.

En la tabla de verdad de la figura 4.1, el dígito binario 1 representa un nivel alto de voltaje, y el dígito binario 0 un nivel bajo de voltaje.

- **Compuerta AND**

Podemos decir que la compuerta AND es un circuito en el cual la salida será un nivel alto solamente cuando todas las entradas se encuentren en el nivel alto. La salida es un nivel bajo si cualquiera de las entradas (A o B) está en nivel bajo. La figura 4.1b muestra el símbolo lógico, su tabla de verdad y su ecuación característica de dicha compuerta.

- **Compuerta NOT**

La compuerta más sencilla es la compuerta inversora o NOT. La compuerta inversora es aquella en la cual su salida tiene un nivel bajo (“0”) cuando en su entrada presenta un nivel alto (“1”) y viceversa, es decir, su salida tiene un nivel alto (“1”) cuando en su entrada tiene un nivel bajo (“0”). La figura 4.1c presenta el símbolo lógico, la tabla de verdad y la ecuación característica de la compuerta NOT.

La correcta combinación de la compuerta NOT con las compuertas AND y OR produce una serie de compuertas complementarias como lo son: las

compuertas NAND, NOR, OR-exclusiva y NOR-exclusiva, ver figura 4.2 Las razones de la popularidad de las puertas inversoras (NOR y NAND) son:

a) Baratas

b) Rápidas, y

c) Disipan menos potencia

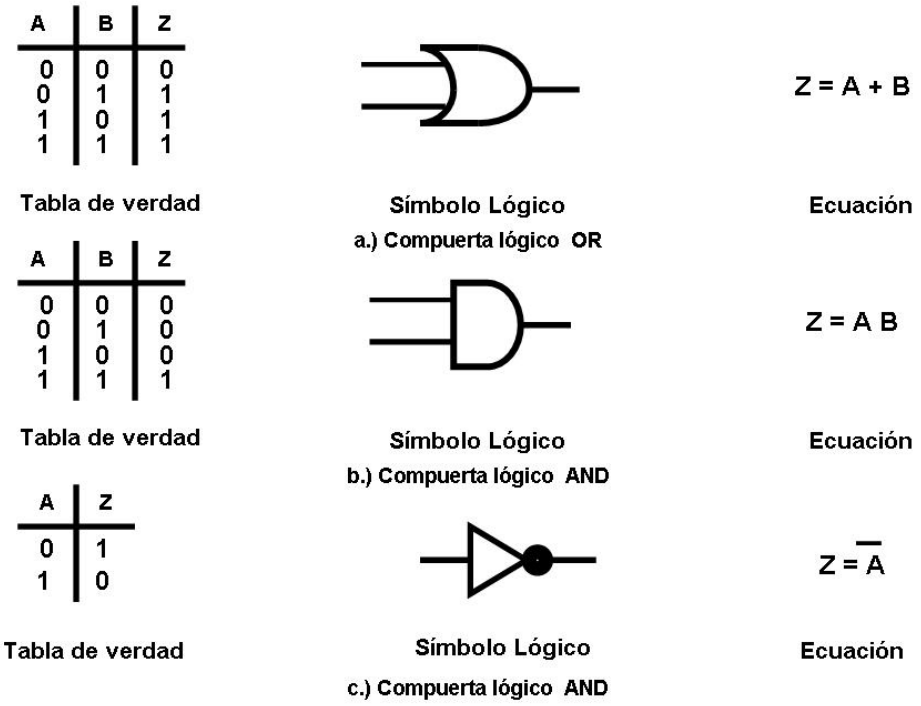


Figura 4.1 Compuertas básicas

- **Compuerta NAND**

La compuerta NAND es equivalente a una compuerta AND seguida de una compuerta NOT, tal como se muestra en la figura 4.2a. El funcionamiento de esta compuerta es el siguiente: La salida presenta un nivel bajo solamente si todas las entradas están en nivel alto (“1”). La salida tiene un nivel alto si cualquiera de las entradas está en nivel bajo “0”).

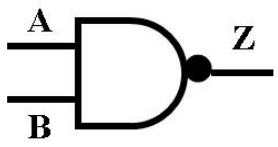
- **Compuerta NOR**

La compuerta NOR es equivalente a una compuerta OR seguida de una compuerta NOT, tal como se muestra en la Figura 4.2b.

La compuerta NOR es aquella en la cual la salida presenta un nivel bajo o “0” si sus dos entradas está en un nivel alto o “1”y su salida presente un nivel alto ó “1” cuando al menos una de sus entradas tiene un nivel bajo o “0”. La Figura 4.2b se presenta el símbolo lógico, tabla de verdad y la ecuación característica de la compuerta NOR.

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Tabla de Verdad



Símbolo Lógico

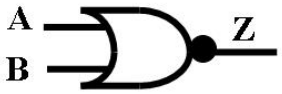
$$Z = \overline{AB}$$

Ecuación

4.2a Compuerta Lógica NAND

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Tabla de Verdad



Símbolo Lógico

$$Z = \overline{A + B}$$

Ecuación

4.2b. Compuerta Lógica NOR

Figura 4.2 Compuertas Complementarias

- **Compuerta OR-Exclusiva**

La compuerta OR-exclusiva es aquella en la cual la salida es un nivel bajo si sus entradas son iguales (son 0 ó 1) y presentan un nivel alto cuando sus entradas son diferentes. La Figura 4.2c muestra el símbolo lógico, tabla de verdad y la ecuación característica.

- **Compuerta NOR-Exclusiva**

La compuerta NOR-exclusiva es aquella en la cual la salida es un nivel bajo (“0”) si las entradas son diferentes (son “0” ó “1”) y presentan un nivel alto (“1”) cuando sus entradas son iguales. La Figura 4.2d. muestra el símbolo lógico, tabla de verdad y ecuación característica.

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

Tabla de Verdad

Símbolo Lógico

$$Z = A \oplus B$$

$$Z = \bar{A}B + A\bar{B}$$

Ecuación

4.2.c Compuerta Lógica OR-EXCLUSIVA

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Tabla de Verdad

Símbolo Lógico

$$Z = A \overline{\oplus} B$$

$$Z = \bar{A}\bar{B} + AB$$

Ecuación

4.2.d Compuerta Lógica NOR-EXCLUSIVA

Figura 4.2 Compuertas Complementarias (Continuación)

4.2.3 Función booleana

Una función booleana representa el análisis y síntesis de un problema determinado. Una función booleana depende de **n**-variables de entrada y representa a una sola salida.

Definición

Una función booleana es la combinación de variables (de entrada) y operadores lógicos que representan el análisis y/o síntesis de un problema determinado. Una función booleana en algunos casos se puede obtener a partir de una tabla de verdad.

Tabla de verdad

Una contribución fundamental del álgebra de Boole es el desarrollo del concepto de tabla de verdad. Una tabla de verdad captura e identifica las relaciones lógicas entre las n-variables de entrada y las m-funciones lógicas de salida en forma tabular.

4.3. Técnicas de minimización de funciones

La expresión algebraica de una función booleana no siempre es fácil de reducir y generalmente exige cierta intuición e ingenio. Se han desarrollado muchas técnicas para ayudar a la reducción de una función booleana entre las cuales se encuentran el proceso algebraico y los mapas de Karnaugh las cuales se explicarán a continuación.

4.3.1 Proceso algebraico

El proceso algebraico es una técnica para reducir de manera sistemática una función lógica utilizando las propiedades (teoremas y leyes) fundamentales del álgebra de Boole. Para entender en qué consiste la técnica mostramos una serie de ejemplos a continuación.

Ejemplo Reduzca la siguiente función utilizando el Algebra de Boole

Solución

$$f(A,B,C) = \overline{AB + C} + \overline{\bar{A}C\bar{B}} + AC(B + BA)$$

Primero marcamos cada uno de los minitérminos

$$f(A,B,C) = \underbrace{\overline{AB + C}}_{I} + \underbrace{\overline{\bar{A}C\bar{B}}}_{II} + \underbrace{AC(B + BA)}_{III}$$

Factorizando el término III y utilizando el teorema de complemento

$$f(A,B,C) = \underbrace{\overline{AB + C}}_{I} + \underbrace{\overline{\bar{A}C\bar{B}}}_{II} + \underbrace{AC(\,B(1 + A))}_{III} \quad (1 + A \rightarrow 1)$$

y ordenando(prop. conmutativa) la ecuación

$$f(A,B,C) = \overline{AB + C} + \overline{\bar{A}\bar{B}C} + ABC$$

Aplicando la ley de Morga a los terminos I y II de la expresión anterior

$$f(A,B,C) = \overline{\overline{\overline{AB} + \overline{C}}} + \overline{\overline{\overline{\overline{\bar{A}\bar{B}} + \overline{C}}}} + ABC$$

Con lo cual obtenemos la expresión

$f(A,B,C) = (\overline{AB} + \overline{C}) + (\overline{\overline{A}\,\overline{B}} + \overline{C}) + ABC$

A la expresión anterior aplicamos Morgan a los términos I y II

$f(A,B,C) = \overline{(\overline{A} + \overline{B})}\,\overline{C} + \overline{\overline{(A + B} + \overline{C})} + ABC$

Se obtiene la siguiente expresión

$f(A,B,C) = (\overline{A} + \overline{B})\,\overline{C} + (A + B + \overline{C}) + ABC$

Utilizando la prop. distributiva

$f(A,B,C) = (\overline{A}\,\overline{C} + \overline{B}\,\overline{C}) + (A + B + \overline{C}) + ABC$

factorizando y utilizando la prop. de complemento, se obtiene

$f(A,B,C) = \overline{A}\,\overline{C} + C(\cancel{1}^{1} + B) + A + B + ABC$

$f(A,B,C) = \overline{C}(\cancel{A}^{1} + 1) + A(\cancel{BC}^{1} + 1) + B$

$f(A,B,C) = A + B + \overline{C}$

4.3.2 Mapas de Karnaugh

El método de los mapas de Karnaugh es un técnica gráfica que puede utilizarse para obtener los términos mínimos de una función lógica utilizando las variables que les son comunes. Las variables comunes a más de un término mínimo son candidatas a su eliminación. Aunque la técnica puede emplearse para cualquier número de variables, raramente se utiliza para más de seis. El mapa está formado por cajas (o celdas), cada una de las cuales representa una combinación única de las variables. Para una variable, solamente se necesitan dos cajas. Dos variables requieren cuatro combinaciones, ver figura 4.3. Para tres variables se requieren $2^3 = 8$ cajas, ver figura 4.4 y para cuatro variables $2^4 = 16$ cajas, ver figura 4.5, etc.

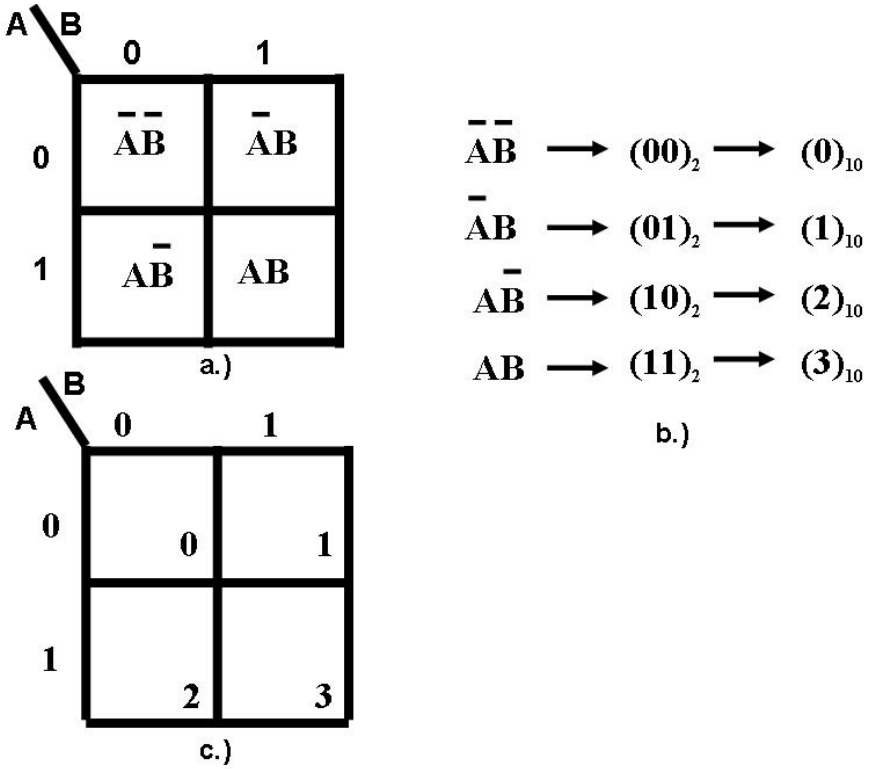


Figura 4.3. Mapa de Karnaugh para 2 variables

La figura 4.3 muestra las cajas o celdas adyacentes del mapa de Karnaugh para dos variables. En dicha figura se muestra las cuatro únicas combinaciones del mapa, figura 4.3a. La figura 4.3b muestra en forma binaria el valor lógico de cada una de las combinaciones en función de las dos variables, las cuales se pueden pasar al sistema decimal, con lo cual cada una de las cuatro combinaciones anteriores tiene una posición (en el sistema decimal) en cada una de las cajas o celdas en el mapa.

Para tres (ver figura 4.4), cuatro (ver figura 4.5) o más variables los mapas se construyen de forma que se solapen cada una de las variables a fin de producir todas las combinaciones requeridas.

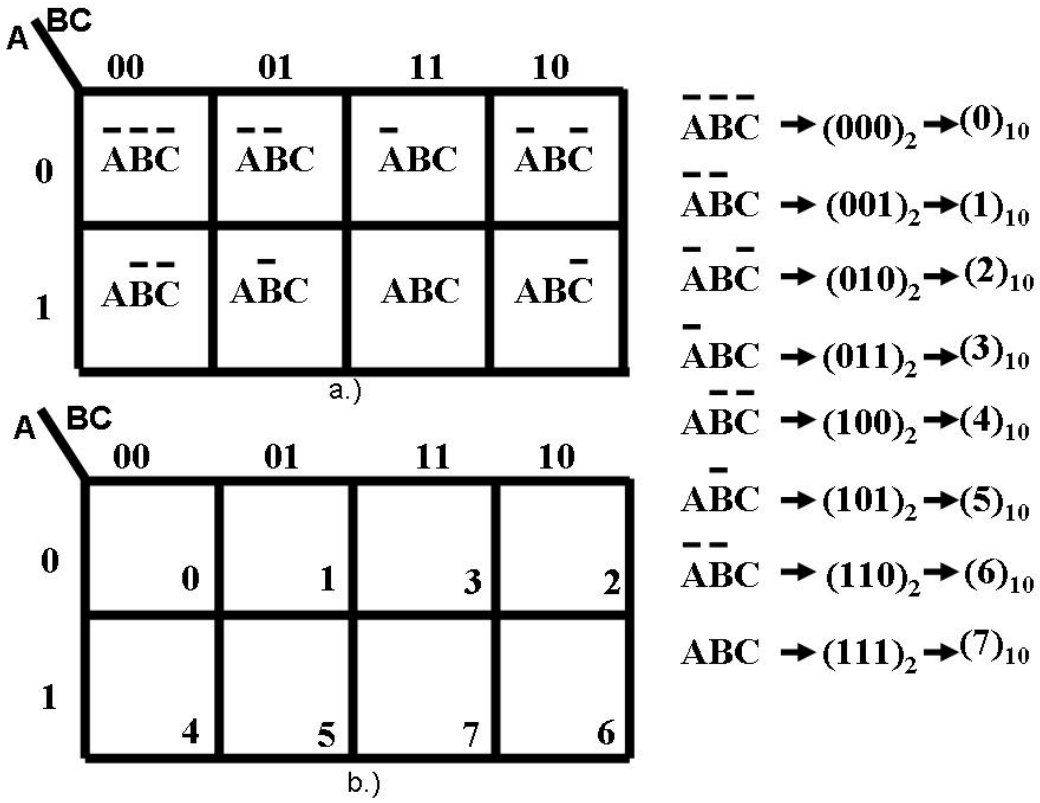


Figura 4.4. Mapas de Karnaugh para 3 variables

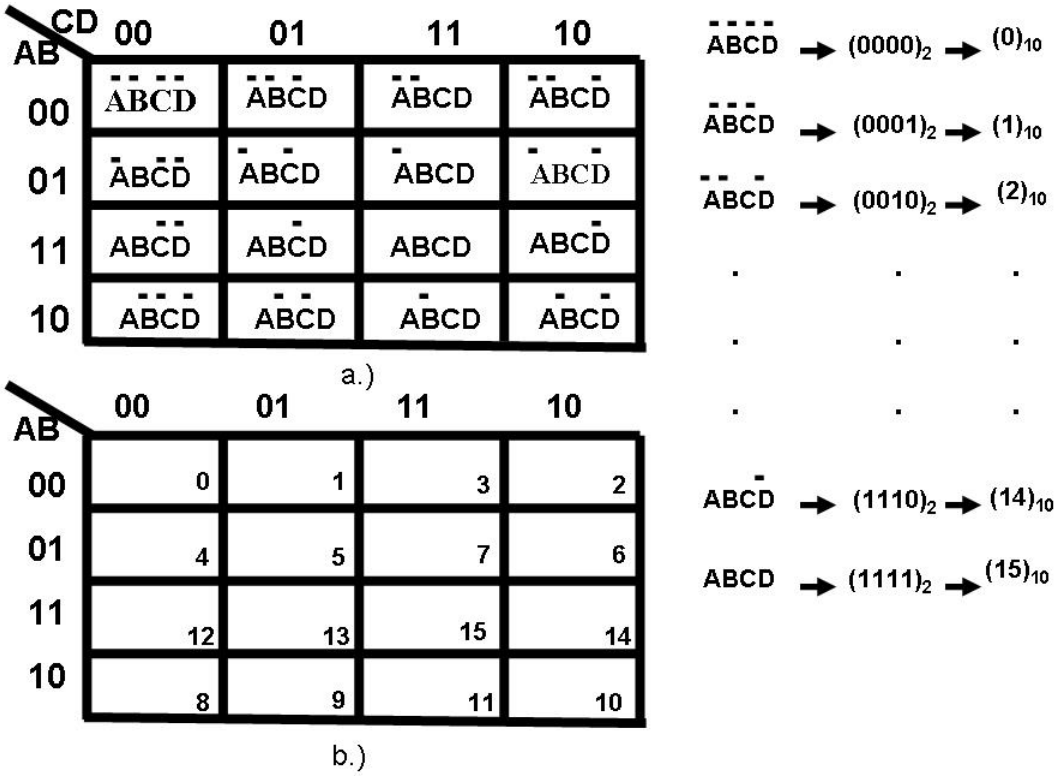


Figura 4.5. Mapas de Karnaugh para 4 variables

Procedimiento de reducción utilizando mapas de Karnaugh

El proceso de reducción de una expresión booleana utilizando mapas de Karnaugh consiste de la aplicación de los pasos siguientes:

Paso 1

Definir el tamaño del Mapa de Karnaugh

El tamaño del mapa de Karnaugh se define en función del número de las variables de entrada (*n*) que forman la expresión booleana, por ejemplo si se tienen 3 (*n*=*3*) variables, el tamaño del mapa de Karnaugh es de 8 (2^n) celdas contiguas, si tuviera cuatro variables de entrada (*n* = 4) se forma o construye un Mapa de Karnaugh de 16 celdas ($2^4 = 16$), etc.

Paso 2

Depositar en cada una de las celdas el valor de “1” donde la función es verdadera y el valor de 0 en las celdas donde la función es falsa. Por claridad únicamente se depositan los “1”s.

Paso 3

Realizar encierros de cajas o celdas (cuyo contenido sea “1”) adyacentes y contiguos de tamaño 2^n, 2^{n-1},2^{n-2},…,2^0., cuyos contenidos tengan el valor de uno. Los encierros de celdas se deben realizar a partir de la potencia de 2 más alta y posteriormente se realizan encierros de una potencia de 2 menor que la anterior y así sucesivamente hasta 2^0.

Los encierros o agrupaciones de cajas o celdas adyacentes se realizan en cantidades de términos mínimos que deben ser potencias de dos, tales como 1, 2, 4 y 8. Estos grupos se conocen con el nombre de implicantes primos[1].

Las variables booleanas se van eliminando a medida que se logra el aumento de tamaño de estos grupos. Con el objeto de mantener la propiedad de adyacencia, la forma del grupo debe ser siempre rectangular, y cada grupo debe contener un número de celdas que corresponda a una potencia entera de dos.

Los unos adyacentes de un mapa Karnaugh de la figura 4.4 satisfacen las condiciones requeridas para aplicar la propiedad de complemento del álgebra de Boole. Dado que en el mapa Karnaugh de la figura 4.4 existen unos adyacentes, puede obtenerse una simplificación sencilla.

Paso 4

Se obtiene la función booleana reducida a partir de cada uno de los grupos (encierros) formados en el punto 3.

Paso 5

Realizar el diagrama lógico de la función reducida.

Para mostrar el procedimiento exponemos una serie de ejemplos a continuación.

Ejemplo Utilizando los mapas de Karnaugh reduce la siguiente función booleana

Solución

$$f(A,B,C) = \sum(3,5,6,7)$$

Paso 1

Esta función booleana depende de 3 variables (A, B y C) por lo tanto tenemos un mapa de Karnaugh de 8 celdas como se muestra en la figura 4.6.

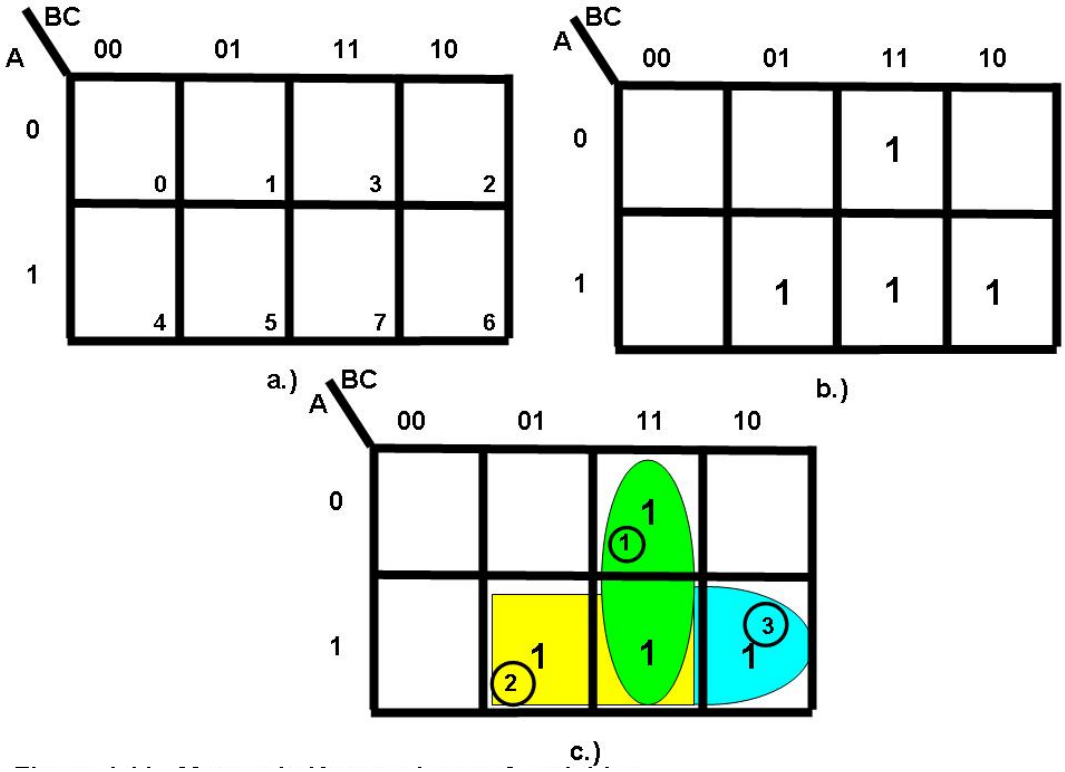


Figura 4.6 a-c Mapas de Karnaugh para 3 variables

Nota

Cada una de las celdas que forman el mapa de Karnaugh se puede enumerar con la facilidad de vaciar el valor de “1“en cada una de las celdas, ver figura 4.6 a.

Paso 2

En cada una de las celdas que forman el mapa de Karnaugh se coloca el valor de 1 cuyos términos en la función sean verdaderos. A partir de la función observamos los términos que son verdaderos (3, 5, 6 y 7) y los términos que no son verdaderos (0, 1, 2 y 4), por claridad no se colocan los ceros, ver figura 4.6. b.

Paso 3

Agrupar las celdas en grupos de tamaño 2^n

Para agrupar (o realizar los encierros) las celdas cuyo contenido es uno, se agrupan las mismas en potencia de 2, a partir de la potencia mayor hacia una potencia menor o viceversa. En nuestro ejemplo utilizamos la primera forma, es decir, de mayor a menor. Empezamos preguntándonos si se pueden formar grupos de 8 celdas cuyo contenido es uno. No. Si la respuesta es No, preguntamos nuevamente. ¿Se pueden formar grupos de 4 celdas cuyo contenido es uno? No. Si la respuesta es No, preguntamos nuevamente. ¿Se pueden formar grupos de 2 celdas cuyo contenido es uno? Sí. Si la respuesta es Sí. Enumeramos todos los encierros de dos celdas formados en nuestro caso tenemos tres encierros de 2 celdas cada uno, ver figura 4.6.c. Y preguntamos nuevamente. ¿Se pueden formar grupos de una 1 celda cuyo contenido es uno? No. Si la respuesta es No, empezamos a obtener cada término de la función reducida a partir de todos los encierros encontrados.

Paso 4

Se obtiene la función booleana reducida a partir de cada uno de los grupos (encierros) formados en el punto 3. En este ejemplo, se formaron tres grupos de dos celdas cada uno, como se muestra en la figura 4.6 c. Cada celda con un uno tiene al menos una celda vecina con un 1, por lo que no quedaron grupos de una celda. Al analizar los grupos formados por dos celdas, se observa que todos los elementos unitarios se encuentran cubiertos por grupos de dos elementos. Una de las celdas se

incluye en los tres “encierros”, lo que es permitido, en el proceso de reducción.

Para obtener la función lógica reducida procedemos de la manera siguiente. El primer grupo (encierro 1) nos proporciona el término: AC, el segundo grupo (encierro 2) nos proporciona el término: BC y finalmente el tercer grupo (encierro 3): AB, que finalmente agrupando los tres términos tenemos la función booleana reducida siguiente:

$$f(A,B,C) = AC + BC + AB$$

Paso 5

Finalmente a partir de la ecuación reducida construimos el circuito lógico correspondiente, el cual se muestra en la figura 4.7.

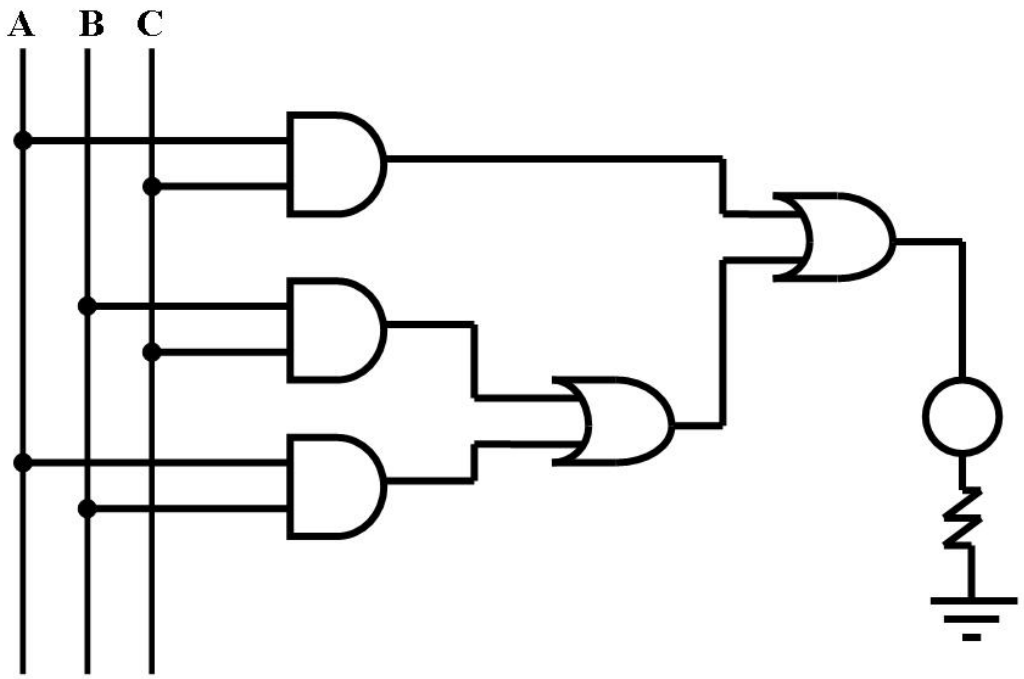


Figura 4.7. Circulo lógico

Ejemplo Utilizando los mapas de Karnaugh reduce la siguiente función

Solución

$$f(A,B,C) = A\,\bar{B}\,\bar{C} + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C + \bar{A}\,B\,C + A\,\bar{B}\,\bar{C}$$

a partir de la función tenemos los términos y su equivalencia en binario y decimal.

$\bar{A}\,\bar{B}\,\bar{C} = (000)_2 = (0)_{10}$ $\quad A\,\bar{B}\,\bar{C} = (001)_2 = (1)_{10}$ $\quad \bar{A}\,B\,\bar{C} = (010)_2 = (2)_{10}$

$\bar{A}\,\bar{B}\,C = (100)_2 = (4)_{10}$ $\quad A\,\bar{B}\,C = (101)_2 = (5)_{10}$ $\quad \bar{A}\,B\,C = (110)_2 = (6)_{10}$

con lo cual la función se puede escribir de la manera siguiente:

$$f(A,B,C) = \sum(0,1,2,4,5,6)$$

En conclusión tenemos dos formas de colocar los 1 en cada una de las celdas del mapa de Karnaugh y son utilizando los términos de la expresión o utilizando la forma canónica de la función a reducir.

Nota

La representación de una función lógica a base de “1” se llama ***forma canoníca*** (lógica positiva).

Paso 1 Definir el tamaño del Mapa de Karnaugh

El tamaño del mapa de Karnaugh se define en función del número de las variables de entrada. Para este ejemplo se tienen 3 variables, el tamaño del mapa de Karnaugh es de 8 celdas, ver figura 4.8.

Paso 2 Vaciar los términos verdaderos en el mapa.

Depositar en cada una de las celdas el valor de 1 donde la función es verdadera y el valor de 0 en las celdas donde la función es falsa. Por comodidad únicamente se depositan los 1s. El mapa de Karnaugh con los “1”s en sus celdas es el siguiente

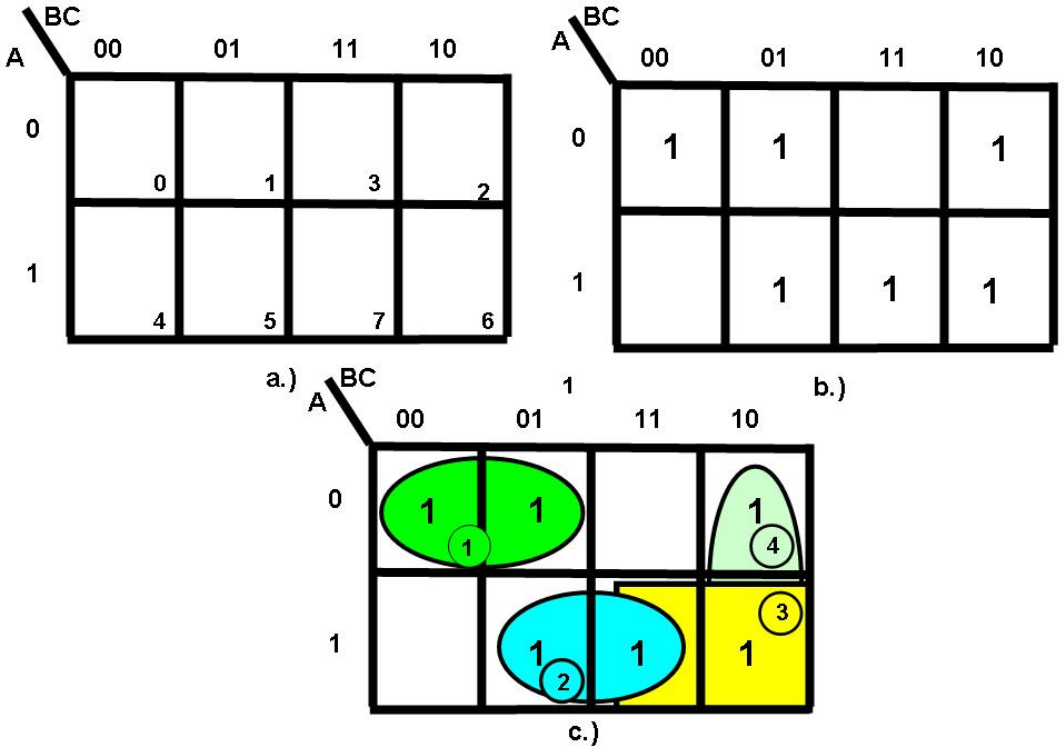


Figura 4.8 Mapas de Karnaugh para 3 variables

Paso 3

Agrupar las celdas en grupos de tamaño 2^n

Para agrupar las celdas cuyo contenido es uno, se agrupan a partir de la potencia mayor hacia una potencia menor. Empezamos preguntándonos ¿se pueden formar grupos de 8 celdas cuyo contenido es uno? -No. Si la respuesta es No, preguntamos nuevamente. ¿Se pueden formar grupos de 4 celdas cuyo contenido es uno? -No. Si la respuesta es No, preguntamos nuevamente. ¿Se pueden formar grupos de 2 celdas cuyo contenido es uno? -Sí. Si la respuesta es Sí, numeramos todos los encierros de dos celdas formados; en nuestro caso tenemos cuatro encierros de 2 celdas cada uno, ver figura 4.8.c. Y preguntamos nuevamente. ¿Se pueden formar grupos de una 1 celda cuyo contenido es uno? No. Si la respuesta es No, empezamos a obtener cada término de la función reducida a partir de todos los encierros encontrados.

Paso 4

Se obtiene la función booleana reducida a partir de cada uno de los encierros formados en el punto 3. En este ejemplo, se formaron cuatro encierros de dos celdas cada uno, como se muestra en la figura 4.8. c. Cada celda con un uno tiene al menos una celda vecina con un 1, por lo que no quedaron grupos de una celda. Al analizar los grupos formados por dos celdas, se observa que todos los elementos unitarios se encuentran cubiertos por grupos de dos elementos. Dos celdas (celda 6 y 7) se incluyen en dos “encierros”, lo que es permitido, en el proceso de reducción.

Para obtener la función lógica reducida procedemos de la manera siguiente:

Encierro 1 nos proporciona el término: $\overline{A}\,\overline{B}$

Encierro 2 nos proporciona el término: AC

Encierro 3 nos proporciona el término: AB

Encierro 4 nos proporciona el término: $B\overline{C}$

$$f(A,B,C) = \overline{A}\,\overline{B} + A\,C + A\,B + B\overline{C}$$

Paso 5

Realizar el diagrama lógico de la función reducida.

Finalmente a partir de la ecuación reducida construimos el circuito lógico correspondiente, el cual se muestra en la figura 4.9.

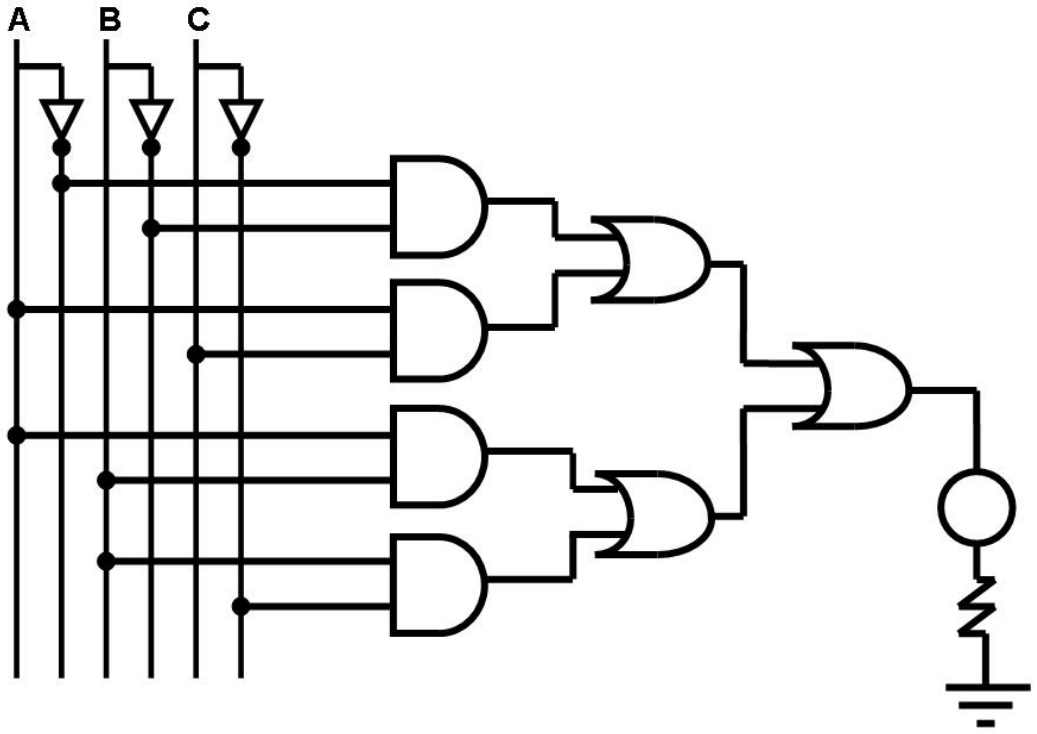
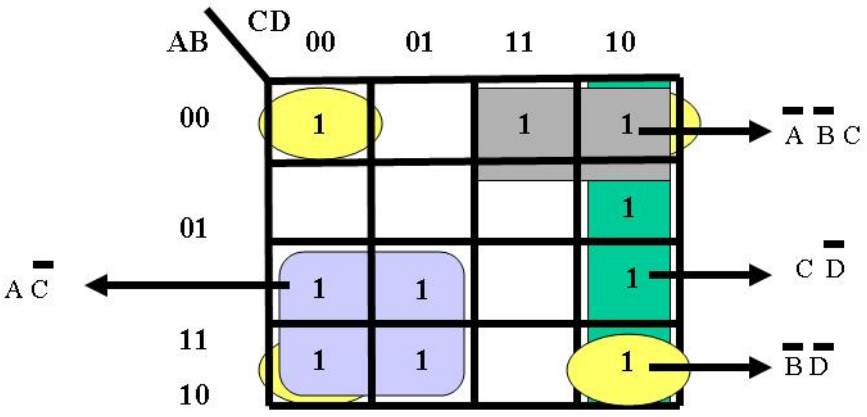


Figura 4.9.Cicuito lógico

Ejemplo Utilizando el mapa de Karnaugh reduzca la siguiente función booleana

f(A,B,C,D)= (0,2,3,6,8,9,10,12,13,14)

Solución



$$f(A,B,C,D) = A\overline{C} + \overline{B}\,\overline{D} + C\overline{D} + \overline{A}\overline{B}C$$

Figura 4.10. Mapa de Karnaugh para 4 variables f(A, B, C, D)

Ejemplo Utilizando mapas de Karnaugh reduzca la siguiente función booleana

f(A,B,C,D,E)= (2,5,6,7,8,9,10,12,13,14,18,21,22,23,24,25,26,28,29,30)

Solución

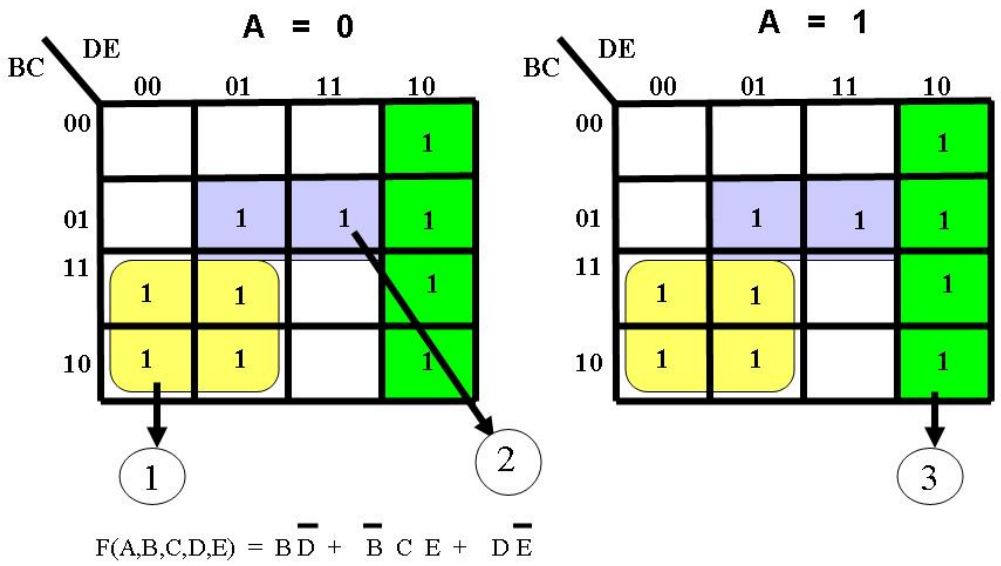


Figura 4.11. Mapa de Karnaugh para 5 variables

Bibliografía del tema 4

Ayala Pérez, Jaime, *Arquitectura de Computadoras*, México, Comunicación interna, 2005.

Charles H Roth Jr., *Fundamentals of Logic Design*, 4ª ed., USA, West Publishing Company, 1992.

Hayes P. John, *Diseño de Sistemas Digitales y Microprocesadores*, México, McGraw Hill, 1986.

Lee, Samuel C., *Digital circuits and logic design,* USA, Prentince Hall, Inc., Englewood Cliffs, N. J., 1976.

Mandado, E., *Sistemas Electrónicos Digitales*, Madrid, Marcombo Boixareu 1980.

Mano, Morris, M., *Arquitectura de Computadoras*, México, Prentice Hall Hispanoamericana, 1988.

Mano, Morris, M., *Diseño Digital*, México, Prentice Hall Hispanoamericana, 1987.

Peatman, B. John, *Microcomputer-Based Design*, Tokio, McGraw-Hill / KOGAKUSHA, LTD., 1982.

Tocci, Ronald, *Digital Design, Principles and Applications*, USA, Prentice Hall, 1977.

Actividades de aprendizaje

A.4.1. Utilizando Mapas de Karnaugh reduce las siguientes funciones

a) f(A,B,C) = Σ(0,1,3,5,6,7)
b) f(A,B,C) = Σ(0,2,4,5,7)
c) f(A,B,C) = Σ(0,3,4,5,7)
d) f(A,B,C) = Σ(0,2,4,5,7)
e) f(A,B,C) = Σ(1,2,3,4,6)

A.4.2. Utilizando Mapas de Karnaugh reduce las siguientes funciones

a) f(A,B,C,D)= Σ(0,1,3,5,6,7,10,11,13,14)
b) f(A,B,C,D)= Σ(0,2,4,5,7,9,10,12,14)
c) f(A,B,C,D)= Σ(0,3,4,5,7,8,10,12,14,15)
d) f(A,B,C,D)= Σ(0,2,4,5,7,9,11,12,14)
e) f(A,B,C,D)= Σ(1,2,4,6,8,9,10,12,14,15)

A.4.3. Utilizando Mapas de Karnaugh reduce las siguientes funciones

a) f(A,B,C,D,E)= Σ(0,1,3,4,5,8,9,10,11,14,15,17,19,20,21,22,25,24,27,29,31)
b) f(A,B,C,D,E)= Σ(0,2,4,5,7,8,10,12,14,17,18,19,20,21,22,24,27,28,29,31).
c) f(A,B,C,D,E)= Σ(0,1,5,6,7,8,10,12,14,16,18,19,20,21,23,25,27,30,31).
d) f(A,B,C,D,E)= Σ(0,2,4,5,7,8,10,12,14,17,18,19,20,21,22,24,27,28,29,31).
e) f(A,B,C,D,E)= Σ(0,2,4,5,7,8,10,12,14,17,18,19,20,21,22,24,27,28,29,31).

A.4.4. Utilizando Mapas de Karnaugh reduce las siguientes funciones

a) f(A,B,C,D,E,F)=Σ(0,2,4,7,9,10,11,13,15,17,18,19,21,24,25,28,29,30,31, 34,36,37,39,41,42,47,49,51,53,57,59,61,63).

b) f(A,B,C,D,E,F)=Σ(0,2,3,5,7,9,10,11,12,14,16,18,19,20,22,24,26,28,29,31,34, 36,39,41,44,48,50,51,53,55,59,61,63).

c) f(A,B,C,D,E,F)=Σ(0,1,3,4,5,7,9,10,11,12,15,16,18,19,20,22,23,24,26,28,29, 30,31,37,38,40,42,44,46,49,54,56,58,60,61,63).

d) f(A,B,C,D,E,F)=Σ(0,2,3,4,6,7,9,11,14,15,17,19,21,23,24,27,31,35,37,39,44, 47,48,51,55,57,59,61,62,63).

e) f(A,B,C,D,E,F)=Σ(1,3,5,6,10,12,14,15,17,19,22,23,24,27,29,31,35,37,39,41, 44,47,48,51,55,57,59,61,62,63).

A.4.5. Utilizando el Álgebra de Boole, reduce las siguientes funciones boolenas

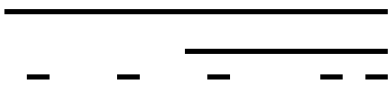
a) $f(A,B,C) = \overline{AB + (\overline{CBA + A\overline{C}})}$

b) $f(A,B,C) = \overline{AB + \overline{\overline{A}C}} + \overline{A\overline{C}\,\overline{B}}\,A$

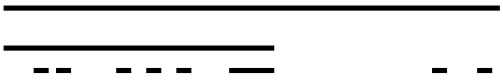
c) $f(A,B,C) = \overline{AB\,(\overline{\overline{A}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C}})}$

d) $f(A,B,C) = \overline{AB + C} + \overline{\overline{A}C\overline{B}} + AC(B + BA)$

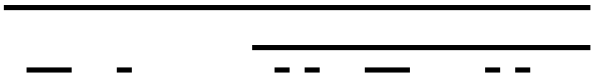
A.4.6. Utilizando el Álgebra de Boole, reduce las siguientes funciones boolenas



a.) f(A,B,C,D) = AB + AC (CDB + ADCA)

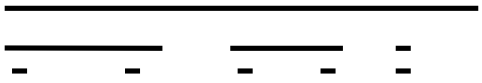


b.) f(A,B,C,D) = (ABC + A B C + AD) + (AD + AB D)



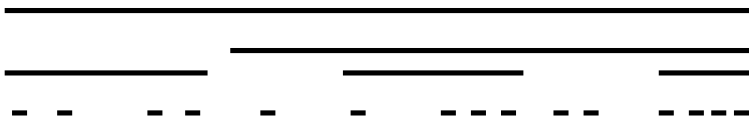
c.) f(A,B,C,D) = (A B + D +C) + (A C + D B + A B C D)

A.4.7. Utilizando el Álgebra de Boole, reduce las siguientes funciones booleanas



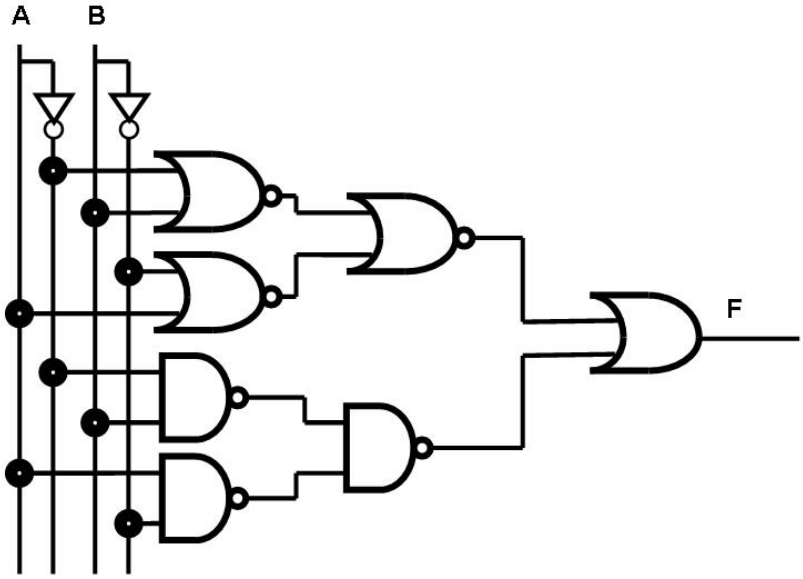
a.) f(A,B,C,D,E) = (A B C + AD) + A B D E + AC + DB

b.) f(A,B,C,D,E)=(AB CE) + AC + ACDB + ACD +(ACDA +ACE) +(ACE)(B)

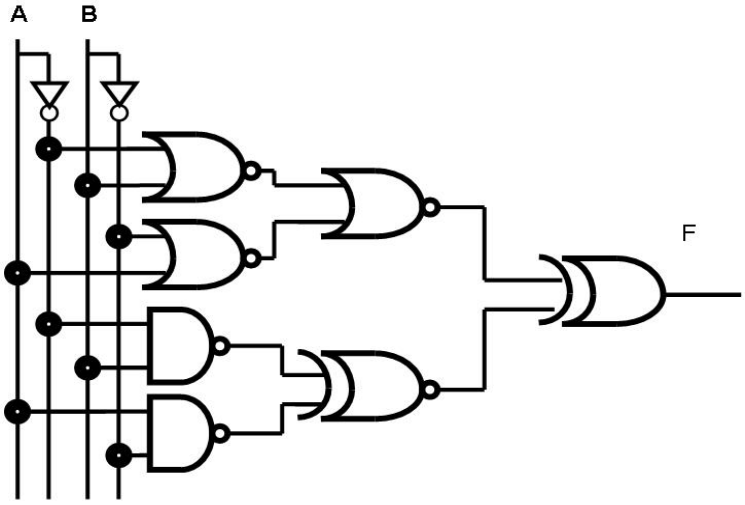


c.)f(A,B,C,B,D,E)= (ABCD) +(ACDE)+ACDA+ACD+ACDEB+ACEDA+ ABCD

A.4.8. Obtén la ecuación y la tabla de verdad del siguiente circuito



A.4.9 Obtén la ecuación y la tabla de verdad del siguiente circuito



A.4.10. Diseña e implementa un programa en cualquier lenguaje de programación para reducir una función booleana utiliza mapas de Karnaugh.

Cuestionario de autoevaluación

1. ¿Qué es el Algebra de Boole?
2. ¿Cuáles son las áreas en que se divide la electrónica?
3. ¿Qué es la lógica binaria?
4. ¿Qué es la lógica negativa?
5. ¿Qué es una función booleana (lógica)?
6. ¿Qué es una función canónica?
7. ¿Qué es una compuerta lógica?
8. ¿Cuáles son las compuertas lógicas complementarias?
9. ¿Cuáles son las técnicas para reducción de funciones booleanas?
10. ¿En qué consiste el método de reducción por mapas de Karnaugh?

Examen de autoevaluación

Elige la opción que conteste correctamente cada una de las siguientes oraciones.

1. ¿Cuáles son los valores analógicos utilizados en la lógica negativa?
    a) 0 Volts y +12 Volts
    b) -5 Volts y +5 Volts
    c) 5 Volts y 0 Volts
    d) +12 Volts y 0 Volts

2. ¿Qué es una función booleana?
    a) Es una combinación de variables continuas y operadores lógicos
    b) Es una combinación de variables discretas y operadores lógicos
    c) Es una combinación de variables discretas y operadores aritméticos
    d) Es una combinación de variables continuas y operadores aritméticos

3. ¿Cuál es el número de variables permisible para utilizar los mapas de Karnaugh?
    a) Hasta 6
    b) Mayor a 5
    c) Entre 2 y 5
    d) Menor a 8

4. ¿Cuáles son las compuertas lógicas complementarias?
    a) Las compuertas OR, AND y NOR-exclusiva
    b) Las compuertas NOR, NAND, OR-exclusiva y NOR-exclusiva
    c) Las compuertas OR, AND y NOR-exclusiva
    d) Las compuertas NOR, NAND y NOR-exclusiva

5. ¿Cuáles son los pasos para aplicar las leyes de Morgan?

a) Negar cada una de las variables, invertir el operador y negar todo el término

b) Negar todo el término, negar cada una de las variables e invertir el operador

c) Invertir el operador, negar todo el término y negar cada una de las variables,

d) Invertir el operador, negar cada una de las variables, negar todo el término

6. ¿Con qué tipo de señales trabaja la electrónica digital?

a) Señales continuas

b) Señales aleatorias

c) Señales discretas

d) Señales de potencia

7. ¿Cuál es el uso del algebra de Boole?

a) Sintetizar una función booleana

b) Construir una función discreta

c) Reducir una función booleana

d) Analizar una función discreta

8. ¿Qué es una Tabla de Verdad?

a) Establece la relación lógica entre unas variables de entrada y una función lógica de salida en forma tabular.

b) Identifica las relaciones lógicas entre **n**-variables de entrada y **m**-funciones lógicas de salida en forma tabular

c) Establece la relación lógica entre unas variables de entrada y **m** funciones lógicas de salida en forma tabular.

d) Establece la relación lógica entre **n** variables de entrada y **una** función lógica de salida en forma tabular.

9. ¿Cuántas variables se necesitan para tener un mapa de Karnaugh de 32 celdas?

a) 2

b) 5

c) 7

d) 4

10. ¿Qué es una compuerta lógica?

a) Es un dispositivo digital que implementa una función básica del álgebra de Boole.

b) Es un dispositivo físico que implementa una función básica del álgebra de Boole.

c) Es un dispositivo electrónico que implementa has 5 funciones básicas del álgebra de Boole.

d) Es un dispositivo industrial que implementa una función básica del álgebra de Boole.

TEMA 5. CIRCUITOS COMBINATORIOS

Objetivo particular

En este tema presentamos el diseño y construcción de algunos circuitos digitales basados en la lógica combinacional y que forman parte en la construcción de una computadora digital, como son: los multiplexores y demultiplexores, los codificadores y decodificadores, sumadores, etc.

Temario detallado

5.1 Multiplexores

5.2 Demultiplexores

5.3 Codificadores

5.4 Decodificadores

5.5 Sumador medio (Medio sumador)

5.6 Sumador completo

Introducción

Los circuitos combinatorios o circuitos combinacionales transforman un conjunto de entradas en un conjunto de salidas de acuerdo con una o más funciones lógicas. Las salidas de un circuito combinacional son rigurosamente función de las entradas, y se actualizan inmediatamente luego de cualquier cambio en las entradas. La figura 5.1 ilustra un modelo de unidad lógica combinacional. Esta unidad combinacional recibe un conjunto de entradas $i_0,\ldots,i_n$ y produce un conjunto de salidas $f_0,\ldots,f_m$, las que dependerán de las funciones lógicas correspondientes. En este tipo de circuito combinacional no existe retroalimentación de las salidas sobre las entradas como en el caso de los circuitos secuenciales (ver tema 6).

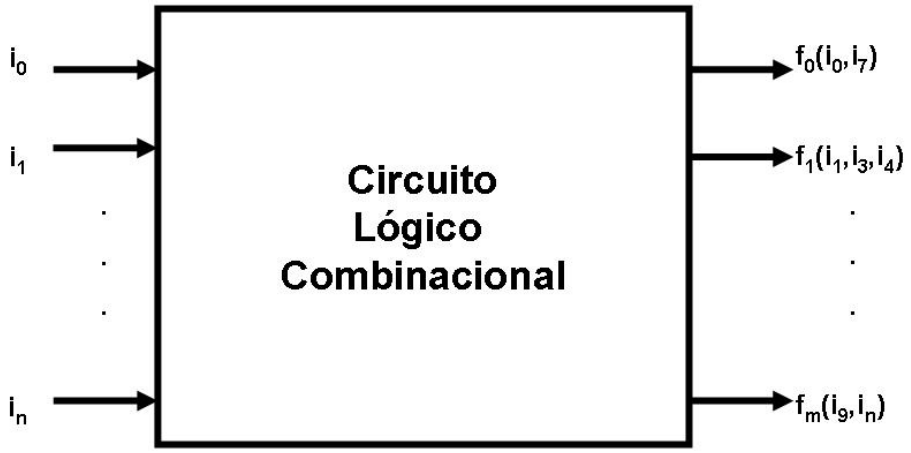


Figura 5.1 Diagrama en bloques de una unidad lógica combinacional

Un circuito combinacional recibe entradas y genera salidas en las cuales es habitual considerar como valor bajo el "**0**" lógico ó 0 Volts, en tanto que se adopta como valor alto el "**1**" lógico ó 5 Volts. Esta convención no es de uso universal. En los circuitos de alta velocidad se tiende a usar menores valores de tensión. Algunos circuitos de computadora funcionan en el dominio analógico, en el que se admite una variación continua de las señales, y en el caso de los circuitos digitales ópticos se puede utilizar variaciones de fase o polarizaciones, por lo que no es necesario plantear los conceptos de alto y bajo en este momento.

5.1 Multiplexores

Un circuito **multiplexor** (MUX) es un elemento que conecta una cantidad dada de entradas a una única salida. La figura 5.2 muestra el diagrama en bloques y la tabla de verdad de un multiplexor de 4 entradas y 1 salida. La salida ***F*** adopta el valor correspondiente a la entrada de datos seleccionada por las líneas de control ***A*** y ***B***. Por ejemplo, si A = 0 y B = 1, el valor que aparece en la

salida es el que corresponde a la entrada D_1, ver figura 5.2b. En la figura 5.2c se muestra la obtención de la función lógica del multiplexor a partir de su tabla de verdad y en la figura 5.d se presenta el diagrama lógico del multiplexor.

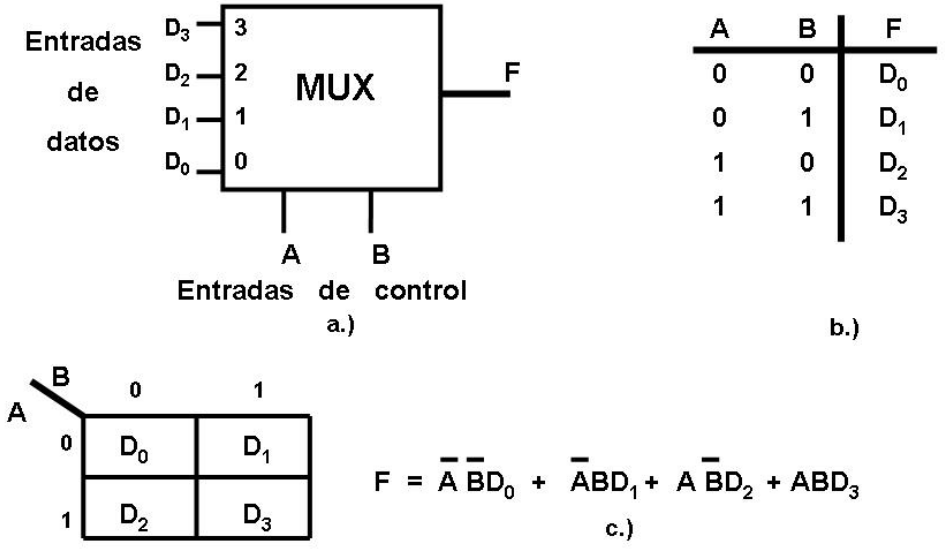


Figura 5.2 Multiplexor de 4 entradas y 1 salida.

a.) Diagrama a bloques, b.) Tabla de Verdad y c.) Función lógica

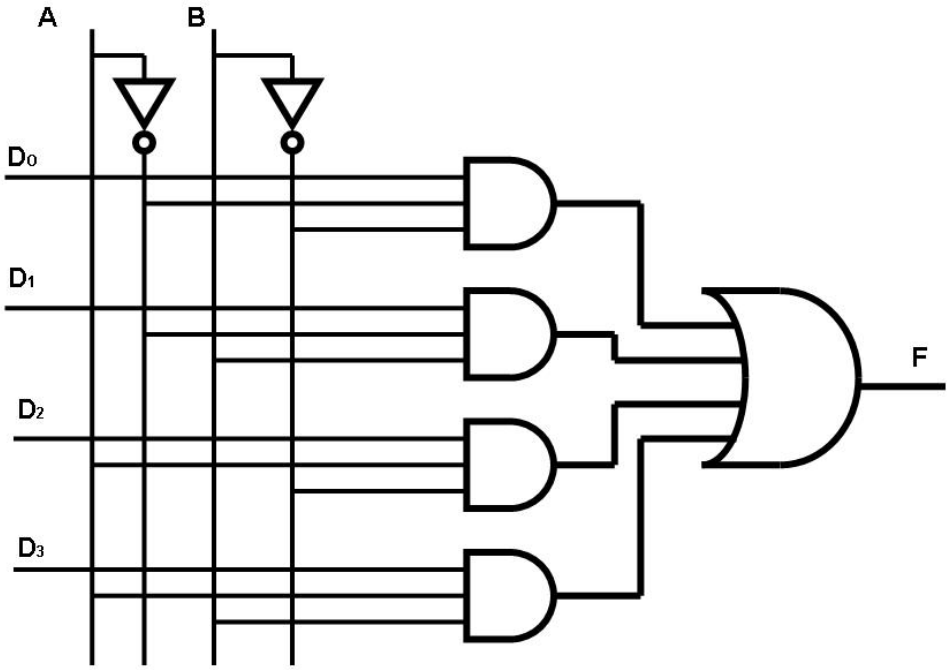


Figura 5.2 Multiplexor de 4 entradas y 1 salida (*continuación*).

d.) Diagrama lógico

Una aplicación de los multiplexores es la implementación de funciones lógicas como se muestra en la figura 5.3. En dicha figura se desea implementar una función lógica usando un multiplexor de 8 entradas y 1 salida. Las entradas de datos se toman directamente de la tabla de verdad de la función a implementar y se asignan las variables **A**, **B** y **C** como entradas de control. El multiplexor transfiere a la salida los unos correspondientes a cada término mínimo de la función. Las entradas cuyos valores son 0 corresponden a los elementos del multiplexor que no se requieren para la implementación de la función, y como resultado hay compuertas lógicas que no se utilizan. Si bien en la implementación de funciones booleanas siempre hay porciones del multiplexor que no se utilizan, el uso de multiplexores es amplio debido a que su generalidad simplifica el proceso de diseño y su modularidad simplifica la implementación.

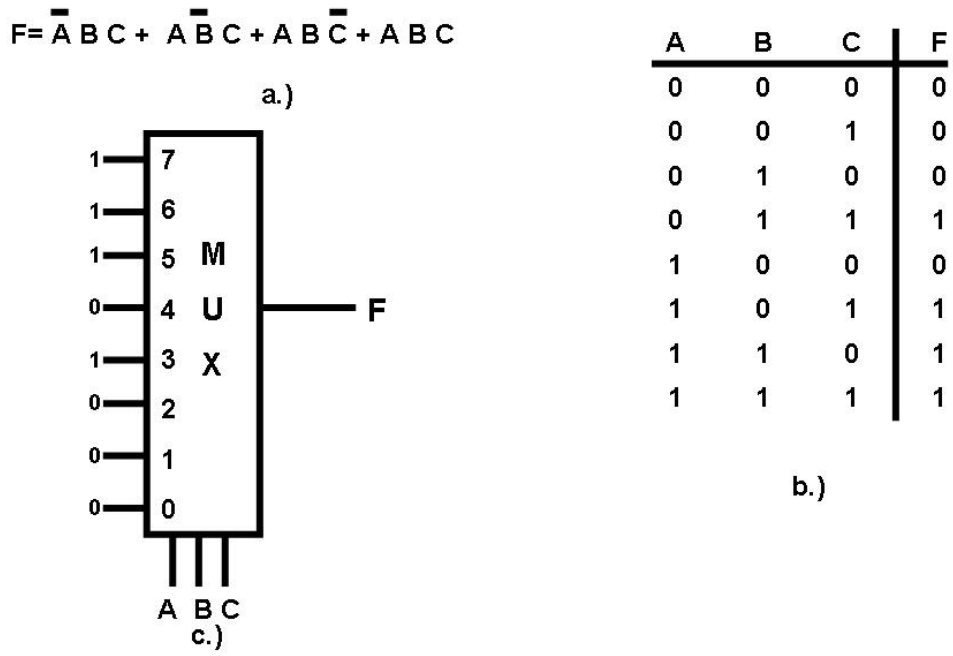


Figura 5. 3 Implementación de una función utilizando un multiplexor de 8 entradas. a.) Función a implementar, b.) Tabla de verdad y c.) Diagrama Lógico.

Otro ejemplo del uso de los multiplexores en la implementación de funciones lógicas es similar al que se muestra en la figura 5.4. La figura 5.4b ilustra la tabla de verdad de tres variables de la función lógica a implementar (ver, figura 5.4a) y el multiplexor de 4 entradas utilizado en la implementación de la función lógica. Las entradas de datos se toman del conjunto {**0, 1, C, C̄**} y la agrupación se obtiene de acuerdo con lo que se muestra en la tabla de verdad. Cuando **A =** 0**, B** = 0, la función **F** = 0 independientemente del valor de **C**, y por lo tanto, la entrada de datos **00** del multiplexor tendrá un valor fijo de **0**, Cuando **A** = 0, **B** = 1, **F** = 1, independientemente del valor de la variable **C**, por lo que la entrada de datos **01** adopta un valor de **1**. Cuando A = 1, B = 0, la función **F** = **C** dado que su valor es **0** cuando **C** es **0** y es **1** cuando **C** es 1. Finalmente, cuando **A** = **1**, **B̄** = **1**, la función **F** = **C**, por lo tanto, la entrada de datos **11** adopta el valor de **C**. De esta manera, se puede implementar una función de tres variables usando un multiplexor con cuatro entradas de datos y dos entradas de control.

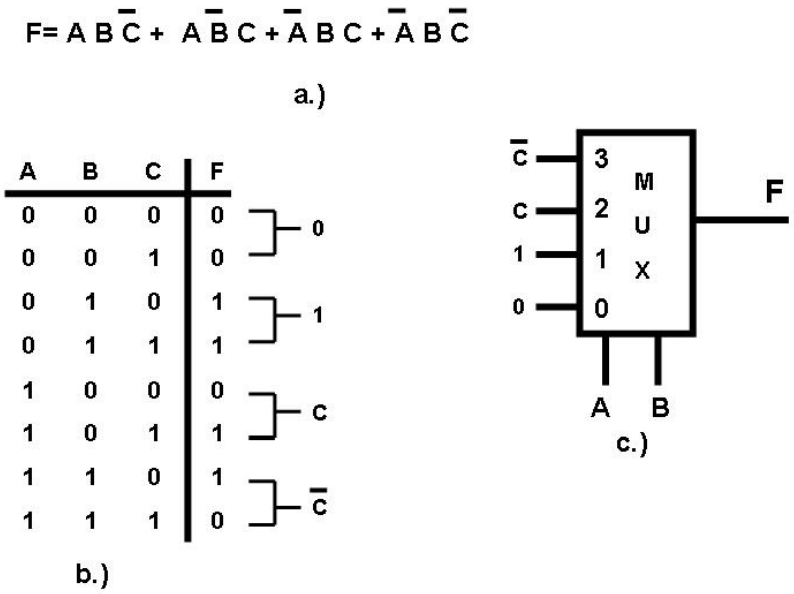
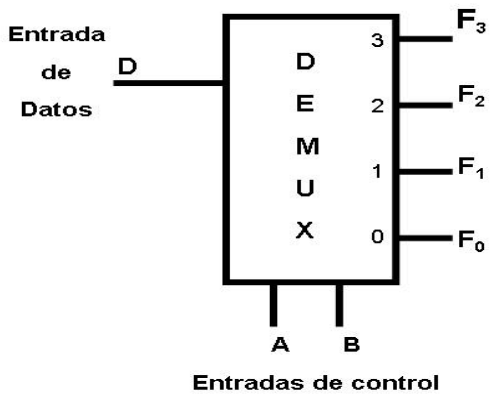


Figura 5.4 Implementación de una función utilizando un multiplexor de 4 entradas de datos. a.) Función a implementar, b.) Tabla de verdad y c.) Diagrama Lógico.

5.2 Demultiplexores

Un demultiplexor (**DEMUX**) es un circuito que cumple la función inversa a la de un multiplexor. La figura 5.5 ilustra el diagrama en bloques correspondientes a un demultiplexor de cuatro salidas, cuyas entradas de control son ***A*** y ***B***, su correspondiente tabla de verdad, su función lógica y su diagrama lógico. Un demultiplexor envía su única entrada de datos **D** a una de sus $\mathbf{F_i}$ salidas de acuerdo con los valores que adopten sus entradas de control. La figura 5.5 muestra el circuito de un demultiplexor de cuatro salidas.



a.)

D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

b.)

$F_0 = D \overline{A}\,\overline{B}$ $F_1 = D \overline{A} B$ $F_2 = D A \overline{B}$ $F_3 = D A B$

c.)

Figura 5.5 Demultiplexor de 2x4.

a.) Diagrama en bloques, b.) Tabla de verdad y c.) Las funciones de salida.

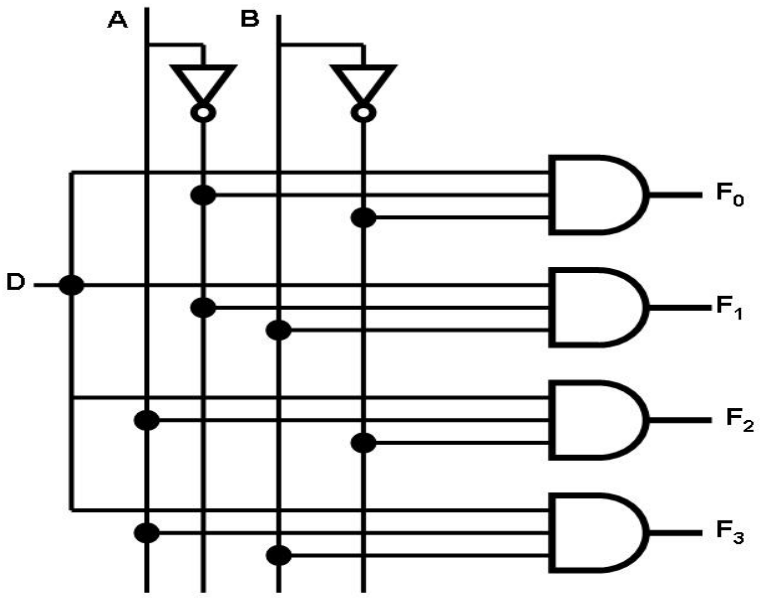


Figura 5.5 Demultiplexor de 2x4. (*continuación*)

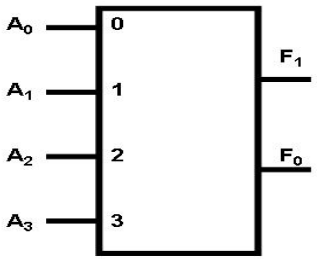
d.)Diagrama lógico.

5.3 Codificadores

Un **codificador** tiene 2^n (o menos) líneas de entrada y **n** líneas de salida. Las líneas de salidas generan el código binario para las 2^n variables de entrada. Un ejemplo de un circuito codificador es el codificador de prioridad.

Un codificador de prioridad es un codificador en el que se establece un ordenamiento de las entradas. El diagrama en bloques y la tabla de verdad de un codificador de prioridad de 4 entradas a 2 salidas se muestra en la figura 5.6. El esquema de prioridades impuesto sobre las entradas hace que $\mathbf{A_i}$ tenga una prioridad mayor que $\mathbf{A_{i+1}}$. La salida de dos bits adopta los valores $\mathbf{0_{10}}$, $\mathbf{1_{10}}$,$\mathbf{2_{10}}$ **u** $\mathbf{3_{10}}$, dependiendo de las entradas activas y de sus prioridades relativas. Cuando no hay entradas activas, las salidas llevan, por defecto, a asignarle prioridad a la entrada $\mathbf{A_0}$ ($\mathbf{F_0}$ = 0 y $\mathbf{F_1}$ = 0).

Los codificadores de prioridad se utilizan para arbitrar entre una cantidad de dispositivos que compiten por un mismo recurso, como cuando se produce el intento de acceso simultáneo de una cantidad de usuarios a un sistema de computación. La figura 5.6c ilustra el diagrama lógico para un codificador de prioridad de 4 entradas y 2 salidas.



a.)

$$F_0 = \overline{A}_0\,\overline{A}_1\,A_3 + \overline{A}_0\,\overline{A}_1\,A_2$$

$$F_1 = \overline{A}_0\,\overline{A}_2\,A_3 + \overline{A}0\,A_1$$

c.)

A_0	A_1	A_2	A_3	F_0	F_1
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

b.)

Figura 5.6 Codificador de prioridad de 4 a 2. a.) Diagrama en bloques, b.) Tabla de verdad, c.) Funciones de salida.

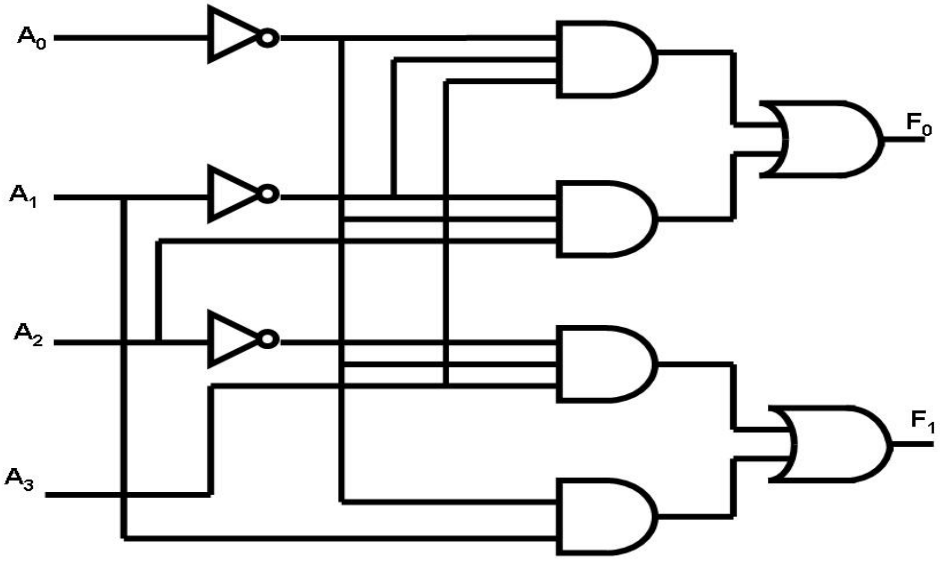
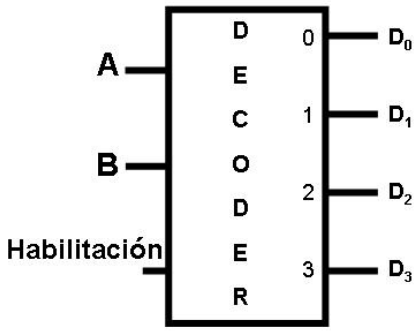


Figura 5.6 Codificador de prioridad de 4 a 2. (continuación) d.) Diagrama lógico.

5.4 Decodificadores

Un decodificador traduce una codificación lógica binaria hacia una ubicación espacial. En cada momento, solo una de las salidas del decodificador está en el estado activo (**"1"** lógico), según lo que determinen las entradas de control. La figura 5.7 muestra el diagrama en bloques, la tabla de verdad de un decodificador de 2 entradas a 4 salidas, cuyas entradas de control son ***A*** y ***B***. El diagrama lógico correspondiente a la implementación del decodificador se muestra en la figura 5.7c. Un circuito decodificador puede usarse para controlar otros circuitos, aunque a veces resulta inadecuado habilitar cualquiera de esos otros circuitos. Por esta razón, se incorpora en el circuito decodificador una línea de habilitación, la que fuerza todas las salidas a nivel "**0**" (inactivo) cuando se le aplica un "**0**" en la entrada.



a.)

$D_0 = \overline{A}\,\overline{B}$ $D_1 = \overline{A}B$

$D_2 = A\overline{B}$ $D_3 = AB$

c.)

Habilitación = 1					
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Habilitación = 0					
A	B	D_0	D_1	D_2	D_3
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

b.)

Figura 5.7 Decodificador 2 a 4. a.)Diagrama a bloques, b) Tabla de verdad y c.) funciones de salida.

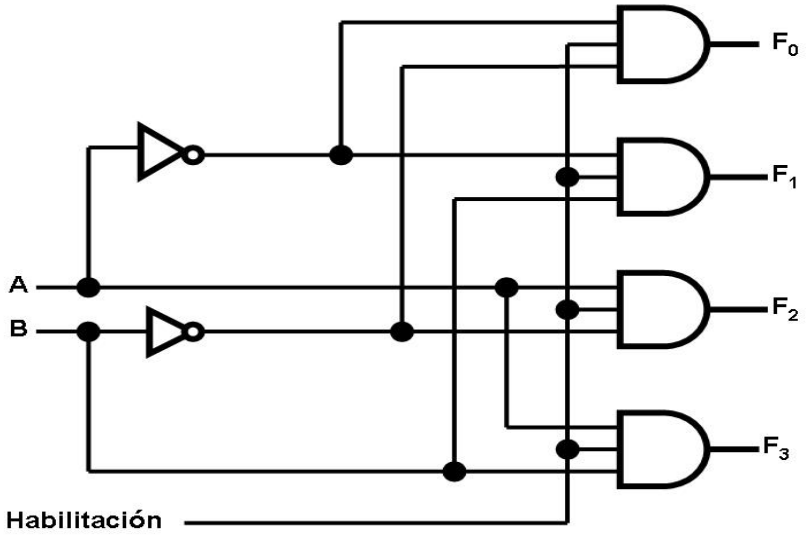
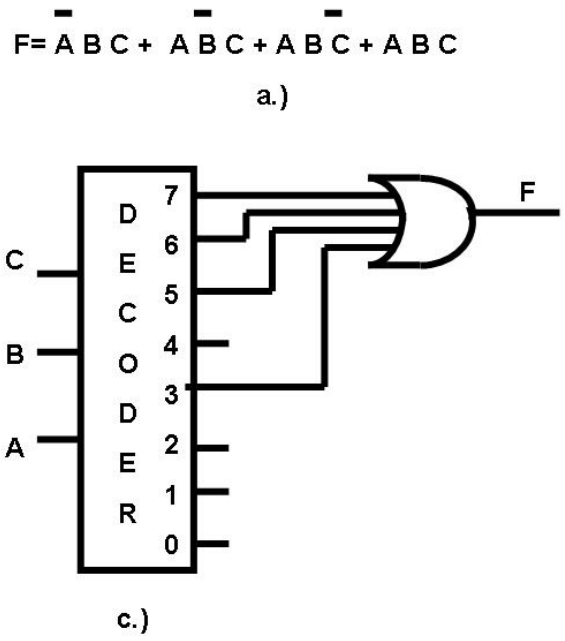


Figura 5.7 decodificador 2 a 4 (continuación) d.) Implementación de un decodificador 2 a 4.

Una aplicación para un circuito decodificador puede ser la traducción de direcciones de memoria a sus correspondientes ubicaciones físicas o para la implementación de funciones lógicas. Para el caso de implementación de funciones, dado que cada línea de salida corresponde a un término mínimo distinto, puede implementarse una función por medio de la suma lógica de las salidas correspondientes a los términos que son ciertos en la función. Por ejemplo en la figura 5.8 se puede ver la implementación de la función con un decodificador de 3 a 8. Las salidas no utilizadas se dejan desconectadas.



A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b.)

Figura 5.8 Implementación de una función utilizando un decodificador 3 a 8. a.) Función a implementar, b.) Tabla de verdad y c.) Diagrama Lógico.

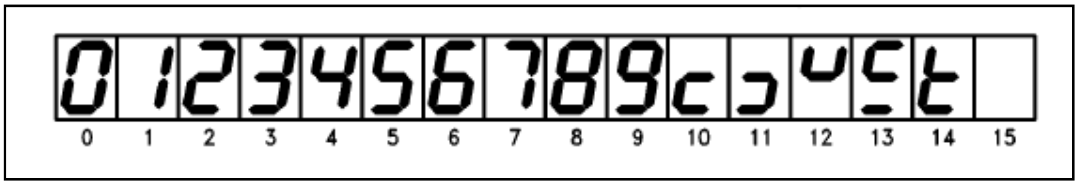
Diseño de un Decodificador BCD

Un decodificador, también puede utilizarse en la visualización de información de un "formato" a otro "formato" como lo es desplegar información en un "Display" de 7 Segmentos. Este circuito decodifica la información cuya entrada está en BCD a un código de siete segmentos adecuado para que se muestre en un visualizador de siete segmentos. El diseño de dicho decodificador se presenta a continuación:

Paso 1: Se enuncia el problema

Diseñe un decodificador BDC a siete segmentos utilizando compuertas

básicas.



Paso 2: Se determina el número requerido de variables de entrada (*n*) y el número

de funciones de salida (*N*).

$$n = 4, N = 2^n = 2^4 = 16$$

Para representar 16 combinaciones (una por cada símbolo) necesitamos cuatro entradas y siete salidas.

Paso 3: Se le asigna letras a las variables de entrada y a las funciones de salida.

entradas => A, B, C, y D

salidas => f_a, f_b, f_c, f_d, f_e, f_f, y f_g.

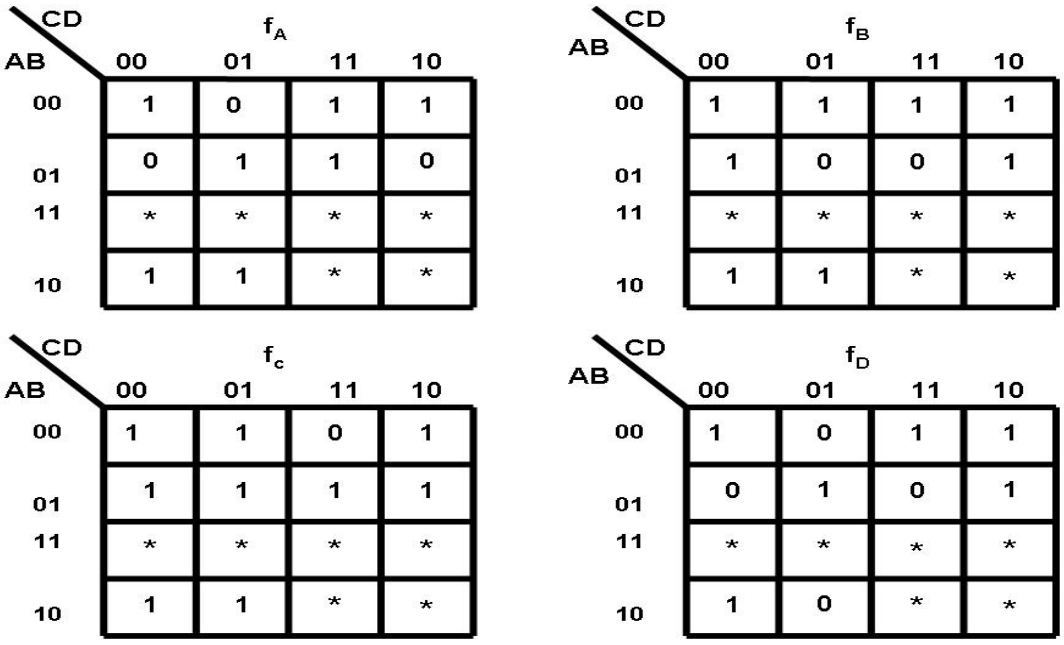
Paso 4: Se deduce la tabla de verdad que define las relaciones entre las entradas

y las salidas.

Entradas					Salidas						
A	B	C	D		a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	0	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	0	0	1	1
1	0	1	0	10	0	0	0	1	1	0	1
1	0	1	1	11	0	0	1	1	0	0	1
1	1	0	0	12	0	1	0	0	0	1	1
1	1	0	1	13	1	0	0	1	0	1	1
1	1	1	0	14	0	0	0	1	1	1	1
1	1	1	1	15	0	0	0	0	0	0	0

TABLA DE VERDAD BCD 7 DE SEGMENTOS

Paso 5: Se obtiene la función de Boole simplificada, en este caso utilizamos el método de Karnaugh a cada una de las salidas.



f_E

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	*	*	*	*
10	1	0	*	*

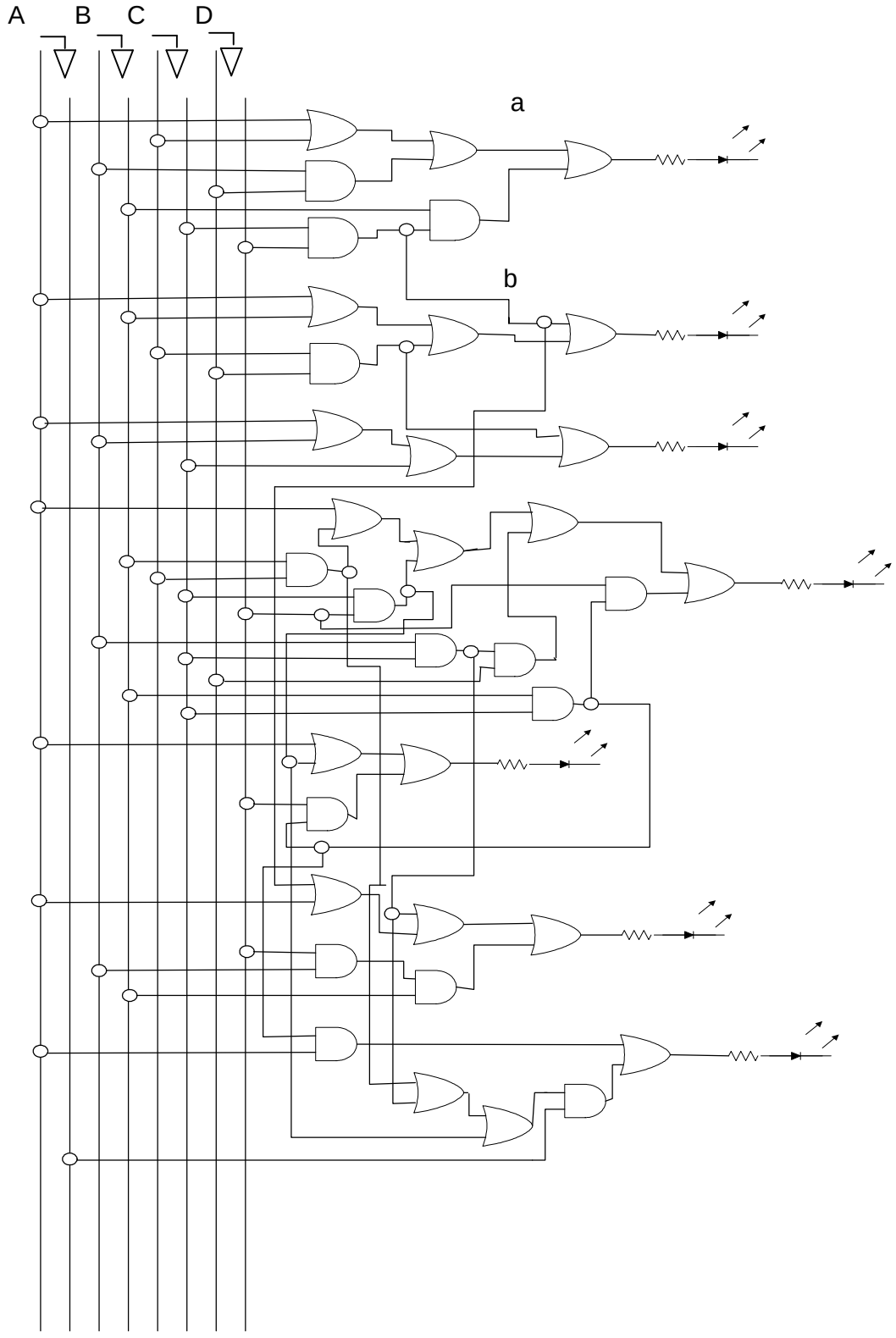
f_F

AB \ CD	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	*	*	*	*
10	1	1	*	*

f_G

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	*	*	*	*
10	1	1	*	*

Paso 6: Se dibuja el diagrama lógico del decodificador 7 segmentos



5. 5 Sumador medio (Medio sumador)

El sumador binario es un circuito combinacional básico en una computadora digital. Este circuito combinacional tiene una característica importante, y es que trabaja en "cascada", es decir, puede realizar la suma de **n-bits** a la vez. Este sumador inicia con un circuito combinacional llamado ***medio sumador*** y le siguen **n-1** ***sumadores completos***. Para diseñar un sumador binario de **n–bits**, empezamos por definir qué es un medio sumador y un sumador completo para posteriormente diseñar un medio sumador y un sumador completo.

Definiciones:

Un **medio sumador** es un circuito combinacional que suma dos bits.

Un **sumador completo** es un circuito combinacional que suma tres bits.

Diseño de un medio sumador

Para diseñar el circuito combinacional denominado *medio sumador* partimos de que deseamos un sumador de dos números de 1 bit cada uno de ellos, y de esta manera tenemos las siguientes combinaciones:

Con 2 variables, se tienen $2^2 = 4$ combinaciones

A_0	0	0	1	1
+	+	+	+	+
B_0	0	1	0	1
C_0 S_0	0 0	0 1	0 1	1 0
	C S	C S	C S	C S

donde

S es el bit del resultado de sumar dos bits, y

C es el bit de acarreo al momento de sumar dos bits

A partir de estos resultados obtenemos la tabla de verdad del medio sumador, la cual presentamos a continuación

Tabla de verdad: Medio Sumador

A_0	B_0	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

A partir de la tabla de verdad, podemos encontrar la ecuación de salida para el resultado **S_0** de la suma de dos bits, así como la ecuación de salida del bit de acarreo **C_0** utilizando Mapas de Karnough, como se muestra en la figura 5.9.

$$S_0 = A_0\bar{B}_0 + \bar{A}_0B_0$$

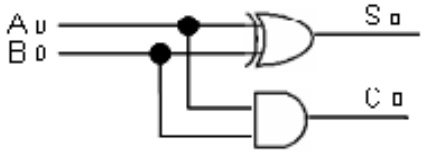
$$S_0 = A_0 \oplus B_0$$

$$S_0 = A_0B_0$$

Figura 5.9. Obtención de la ecuación de S_0 y C_0 utilizando mapas de Karnaugh

La implementación (diagrama lógico) de la ecuación del medio sumador para S_0 y C_0 nos quedaría de la siguiente forma:

Diagrama lógico: Medio sumador



5.6 Sumador completo

Diseño de un sumador completo

Para diseñar el circuito combinacional llamado *sumador completo* partimos de que deseamos un sumador de tres números de 1 bit cada uno de ellos, y de esta manera tenemos las siguientes combinaciones:

3 variables – (2^3) = 8 Combinaciones

C_i	0	0	0	0
+ A_{i+1}	+ 0	+ 0	+ 1	+ 1
B_{i+1}	0	1	0	1
C_{i+1} S_i	0 0	0 1	0 1	1 0
	C S	**C S**	**C S**	**C S**

C_i	1	1	1	1
+ A_{i+1}	+ 0	+ 0	+ 1	+ 1
B_{i+1}	0	1	0	1
C_{i+1} S_{i+1}	0 1	1 0	1 0	1 1
	C S	**C S**	**C S**	**C S**

donde

S_{i+1} es el bit del resultado de sumar tres bits, y

C_{i+1} es el bit de acarreo al momento de sumar tres bits.

A partir de estos resultados obtenemos la tabla de verdad del sumador completo, la cual presentamos a continuación

Tabla de verdad **Sumador completo**

C_i	A_{i+1}	B_{i+1}	C_{i+1}	S_{i+1}
0	0	0	0	0
0	0	1	0	1

0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

A partir de la tabla de verdad, podemos encontrar la ecuación de salida para el resultado $\mathbf{S_{i+1}}$ de la suma de tres bits, así como la ecuación de salida del bit de acarreo $\mathbf{C_{i+1}}$ utilizando Mapas de Karnough, como se muestra en la figura 5.10.

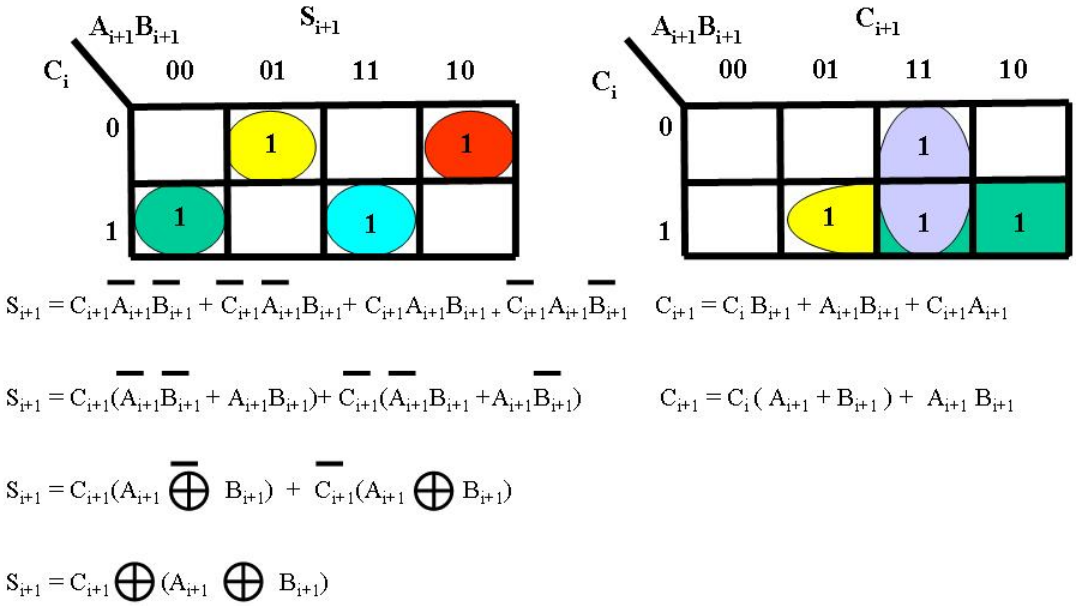
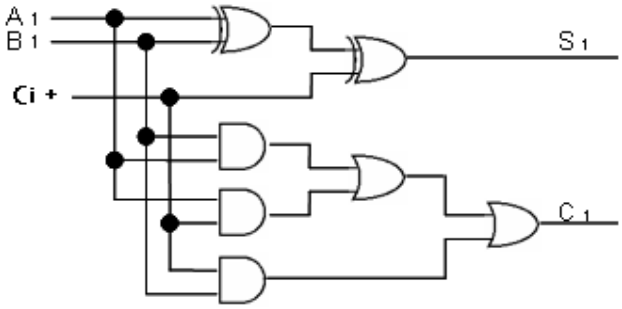


Figura 10. Obtención de las ecuaciones de S_{i+1} y C_{i+1} empleando mapas de Karnaugh

La implementación (diagrama lógico) de la ecuación del sumador completo para $\mathbf{S_{i+1}}$ y $\mathbf{C_{i+1}}$ nos quedaría de la siguiente forma:

Diagrama lógico: Sumador completo



Sumador completo de n-bits

En algunos casos se desea sumar dos números de **n-**bits, lo que se hace es poner un medio sumador y **n-1** sumadores completo en cascada y de esta manera tenemos un sumador de **n** bits, como se muestra en la figura 5.11.

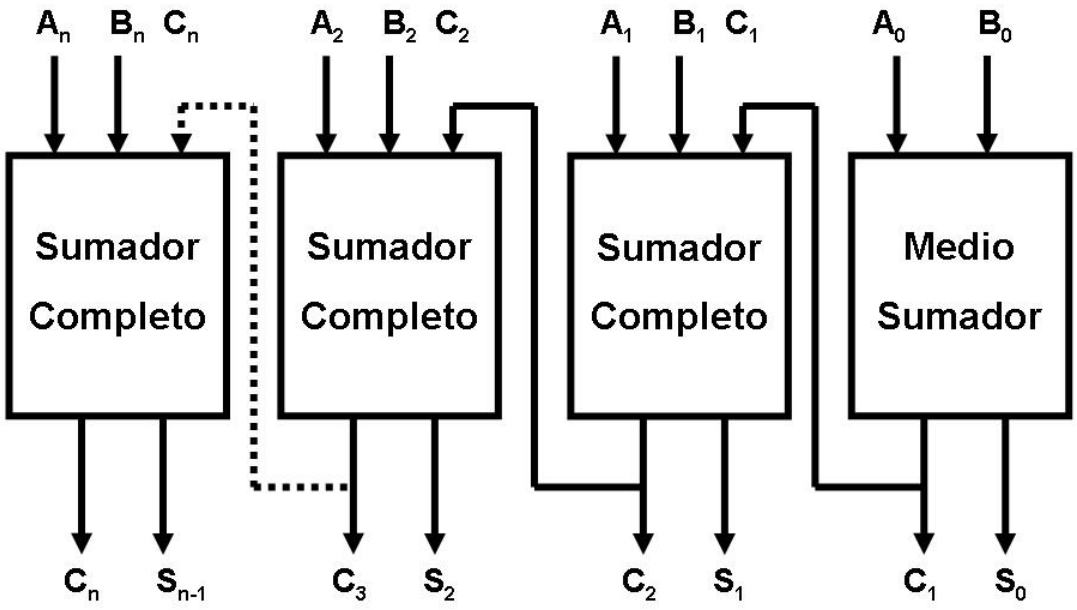


Figura 11. Sumador de n-bits implementados con n-1 sumadores completos

A partir del diagrama a bloques del sumador de 4 bits (ver figura 5.11), se construye el diagrama lógico el cual se presenta en la figura 5.12 y su respectivo diagrama eléctrico en la figura 5.13.

Diagrama lógico: Sumador de 4 bits

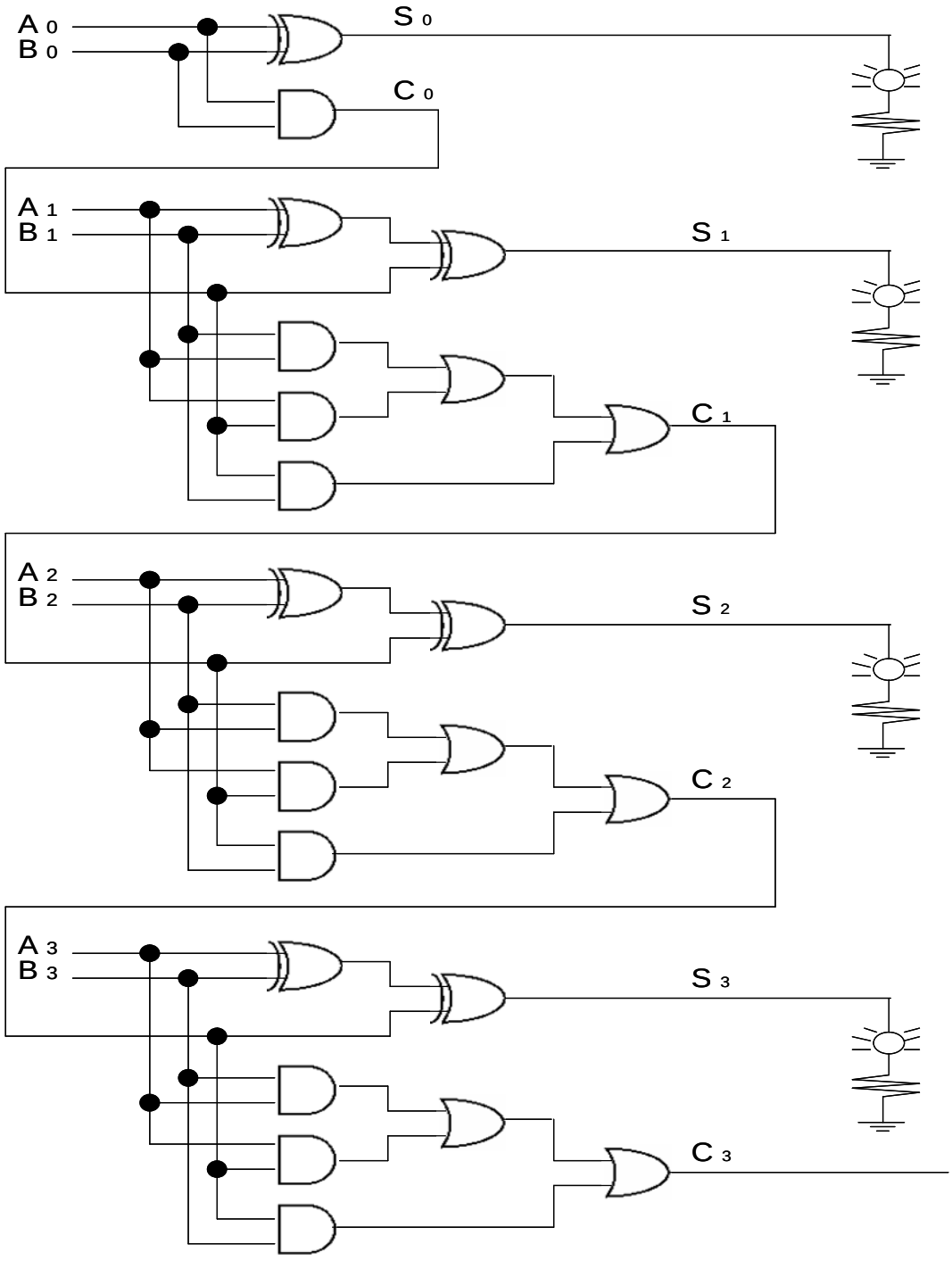


Figura 5. 12 Diagrama lógico de un Sumador de 4 bits en cascada

Diagrama eléctrico: Sumador de 4 bits

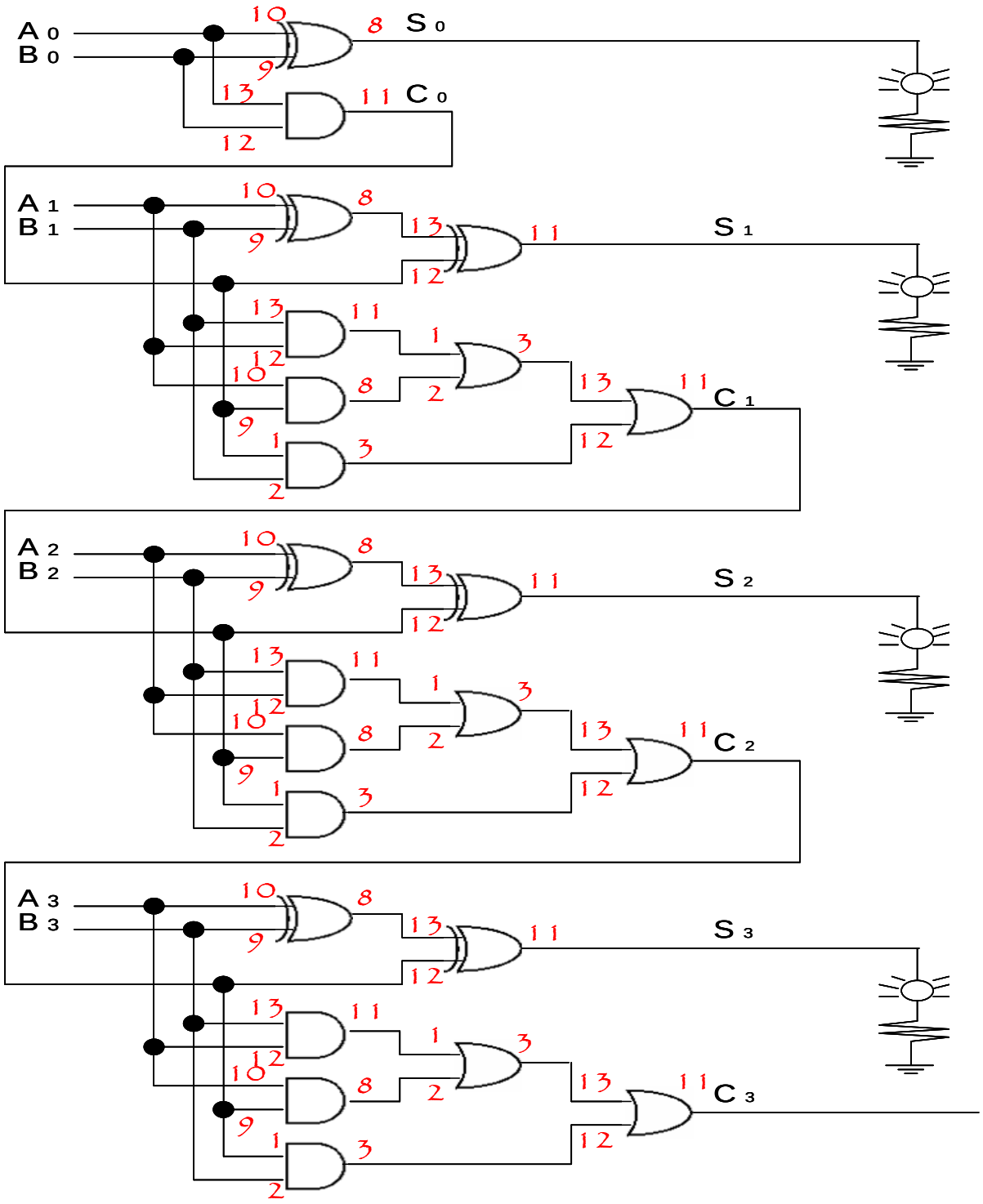


Figura 5.13 Diagrama eléctrico de un sumador de 4 bits en cascada

Para ampliar más este tema visita el Apéndice A, Diseño de circuitos combinacionales.

Bibliografía del tema 5

Mano, Morris, M., *Arquitectura de Computadoras*, México, Prentice Hall Hispanoamericana, 1988.

Mano, Morris, M., *Diseño Digital*, México, Prentice Hall Hispanoamericana, 1987.

Hamacher, V. C.,*Computer Organization*", EE.UU., McGraw Hill International Book Company,1982.

Peatman, B. John, *Microcomputer-Based Design*, Tokio, McGraw-Hill / Kogakusha, LTD, 1982.

Mandado, E., *Sistemas Electrónicos Digitales*, Madrid, Marcombo Boixareu, 1980.

Tocci, R., *Digital Design, Principles and Applications*, EE.UU., Prentice Hall, 1977.

Charles H Roth Jr.,*Fundamentals of Logic Design*, 4ª ed., USA, West Publishing Company, 1992.

Lee, Samuel C., *Digital circuits and logic design,* EE.UU., Prentice Hall, Inc., Englewood Cliffs, N. J., 1976.

Hayes P. John, *Diseño de Sistemas Digitales y Microprocesadores*, México, McGraw Hill, 1986.

Ayala Pérez, J., *Arquitectura de Computadoras*, México, Comunicación interna, 2005.

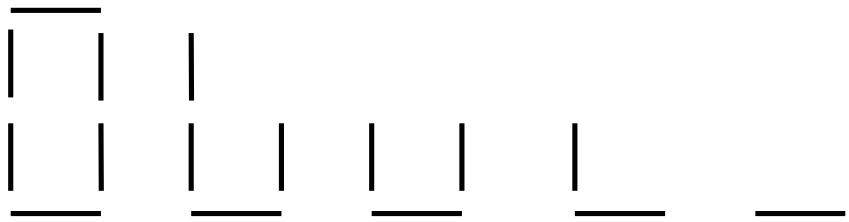
Actividades de aprendizaje

A.5.1. Diseña un comparador de 2 números de 3 bits cada uno.

A.5.2. Diseña un decodificador de binario a decimal.

A.5.3. Diseña un decodificador de binario a hexadecimal.

A.5.4. Diseñe un decodificador (utilizando un display de 7 segmentos) para indicar el espacio de un disco duro.



Disco lleno | Disco a 3/4 | Disco a 1/2 | Disco a 1/4 | Disco Vacío

A.5.5. Diseña un multiplexor de 8x3 utilizando dos multiplexores de 4x2.

A.5.6. Diseña un medio restador.

A.5.7. Diseña un restador completo.

A.5.8. Diseña un restador de 4 bits.

A5.9. Diseñe un sumador completo de 4-bits utilizando únicamente sumadores completos.

A.5.10. Diseñe un circuito combinacional que realice la conversión de código binario a código Grey.

Cuestionario de autoevaluación

1. ¿Qué es un circuito combinacional?.
2. ¿En qué consiste la lógica combinacional?.
3. ¿Qué es un multiplexor?
4. ¿Cuál es el uso de un multiplexor?
5. ¿Qué es un demultiplexor?
6. ¿Qué es un codificador?

7. ¿Cuál es el uso del decodificador?
8. ¿Qué es un decodificador de prioridad?
9. ¿Que es un decodificador?
10. ¿Que es un medio sumador?
11. ¿Qué es un sumador completo?

Examen de autoevaluación

Relaciona las columnas escribiendo dentro del paréntesis el número que una los conceptos con sus respectivas definiciones.

1) Medio Sumador		( ) Circuito Combinacional que despliega los números **0-9** en un "**Display**" de 7 **sgmentos**
2) Circuito Combinacional		( ) Circuito Combinacional que realiza la suma de tres números de **1 bit** cada uno.
3) Multiplexor		( ) Circuito combinacional que está formado a base de un medio sumador y **n**-1 sumadores completos.
4) Codificador de Prioridad		( ) Circuito combinacional que realiza la comparación "***magnitud***" de 2 números de ***n***-bits.
5) Sumador Completo		( ) Circuito combinacional que tiene una sola entrada de datos **D**, ***n*** líneas de control y ***m*** ($m = 2^n$) salidas.
6) Decodificador		( ) Circuito Combinacional de 2^n entradas y ***n*** salidas de tal forma que cuando una de las entradas adopta un estado lógico, a la salida aparece la combinación binaria correspondiente al número decimal asignado a dicha entrada.
7) Comparador de magnitud		( ) Circuito combinacional utilizado para arbitrar entre una cantidad de dispositivos que compiten por un mismo recurso.
8) Codificador		( ) Circuito Combinacional que tiene ***m*** entrada de datos, ***n*** líneas de control y una sola salida.
9) Sumador completo de n-bits		( ) Circuitos que transforman un conjunto de entradas en un conjunto de salidas de acuerdo con una o más funciones lógicas.
10) Demultiplexor		( ) Circuito Combinacional que realiza la suma de dos números de **1 bit** cada uno.

TEMA 6. CIRCUITOS SECUENCIALES

Objetivo particular

Al finalizar el tema, el alumno aprenderá qué es un circuito secuencial, un circuito síncrono y asíncrono, los diferentes tipos de flip-flops (**JK, RS T** y **D**), así como sus usos tanto en registros de corrimiento como en contadores.

Temario detallado

6.1 Circuitos síncronos
6.2 Circuitos asíncronos
6.3 Flip-Flops (JK, RS, T, D)
6.4 Registradores de corrimiento
6.5 Temporizadores
6.6 Contadores

Introducción

Un circuito lógico secuencial, también conocido como máquina de estados finitos, toma una entrada y un estado actual para generar una salida y un nuevo estado. Un circuito lógico secuencial se distingue de un circuito combinacional en que la historia (memoria) de las entradas del circuito secuencial tiene influencia sobre sus estados y sobre su salida. Este concepto es importante en el diseño e implementación de Unidades de Control en una computadora digital.

El modelo clásico de un circuito secuencial se ilustra en la figura 6.1. Este modelo consta de dos partes: Un Bloque de lógica combinacional y un Bloque de Memoria. El bloque de lógica combinacional toma entradas desde las líneas $\mathbf{x_0}$-$\mathbf{x_n}$, externas al circuito secuencial, además de las entradas desde las líneas de estado $\mathbf{Q_0}$-$\mathbf{Q_k}$ (internas al circuito secuencial). El circuito combinacional genera los bits de salida $\mathbf{f_0}$-$\mathbf{f_m}$ formando un nuevo conjunto de bits de estado. Los elementos de memoria (flip-flops) mantienen el estado actual del circuito

secuencial hasta que una señal de sincronía (señal de reloj) provoque la carga de los valores $\mathbf{D_k}$ en los elementos $\mathbf{S_k}$, apareciendo después en $\mathbf{Q_k}$ como los nuevos bits de estado.

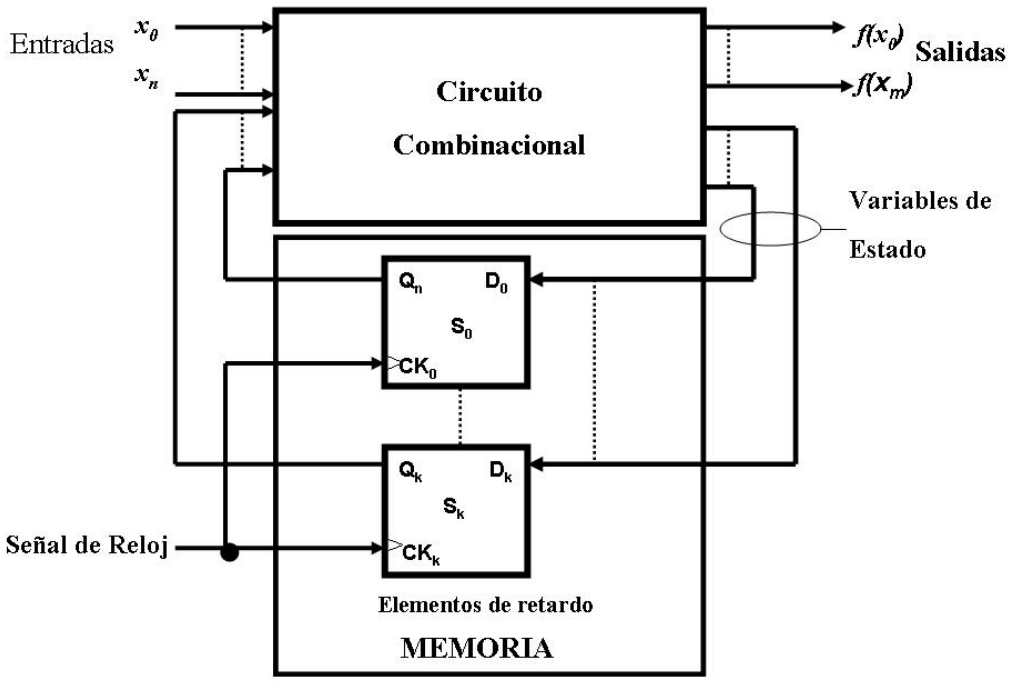


Figura 6.1 Modelo clásico de un Circuito Secuencial

Existen dos tipos de circuitos secuenciales: los síncronos y los asíncronos. Los circuitos secuenciales síncronos los explicaremos en la siguiente sección y los circuitos secuenciales asíncronos se explicarán en la sección 6.2.

6.1 Circuitos síncronos

Además, el cambio de las variables internas se puede producir de dos maneras en un sistema (circuito) secuencial síncrono.

Por nivel
Este sistema permite que las variables de entrada actúen sobre el sistema en el instante en el que la señal de reloj toma un determinado nivel lógico (“0” ó “1”).

Por flanco o cambios de nivel

Cuando la acción de las variables de entrada sobre el sistema se produce cuando ocurre un flanco activo del reloj. Este flanco activo puede ser de subida (cambio de 0 a 1) o de bajada (cambio de 1 a 0).[2]

6.2 Circuitos asíncronos

Los circuitos secuenciales asíncronos no requieren de una señal de reloj y los cambios en las variables de estado interno y en los valores de salida se producen, sencillamente, al variar los valores de las entradas del circuito. Son sistemas más difíciles de diseñar.

En este tema nos vamos a centrar especialmente en el estudio de los circuitos secuenciales asíncronos, los cuales son capaces de almacenar, (si no existe orden exterior de cambio), la información en ellos. Los contadores y registros de desplazamiento, que como observaremos, son también circuitos secuenciales formados por una cadena de flip-flops (biestables). Todos estos dispositivos son de aplicación general, y de gran importancia en el diseño y construcción de una computadora.

6.3 Flip-flops

Los elementos de memoria utilizados en los circuitos secuenciales síncronos se llaman **flip-flops**. Estos circuitos son celdas binarias capaces de almacenar un bit de información. Un flip-flop o circuito biestable mantiene estable el estado de la salida aún después de que las entradas pasen a un estado inactivo. La salida de un flip-flop queda determinado tanto por las entradas actuales como por la retroalimentación (historia) de las mismas. Un flip-flop está construido por un conjunto de compuertas lógicas, normalmente compuertas NAND y NOR.

Los flip-flops se pueden utilizar para:

[2] Rafael López Ahumada, “Sistemas secuenciales”, material electrónico, p.2, disponible en *Cursos anteriores* de la Universidad de Huelva, http://www.uhu.es/rafael.lopezahumada/Cursos_anteriores/fund01_02/tema7.pdf, (fecha de recuperación 06/10/08)

a) Diseñar y construir un circuito secuencial de una unidad de control de una computadora.

b) Construir bloques de memoria RAM (estática y/o dinámica) de una computadora.

Existen diferentes tipos de flip-flops los cuales explicaremos a continuación.

Tipos de Flip Flop (JK, RS, T, D)

Los *flip-flops* o *circuitos biestables* son la forma más sencilla de un circuito secuencial. Existen diferentes tipos de flip-flop entre los cuales se pueden mencionar los siguientes:

- Flip-flop JK
- Flip-flop SR
- Flip-flop T, y
- Flip-flop D

y todos ellos tienen las siguientes propiedades:

- El flip-flop es un dispositivo electrónico con dos estados. El flip-flop siempre se encuentra en uno de los dos estados, en ausencia de una señal de entrada, por lo cual se dice que siempre está recordando el último estado. De esta manera, el flip-flop funciona como una memoria de un bit en el diseño de un circuito secuencial.

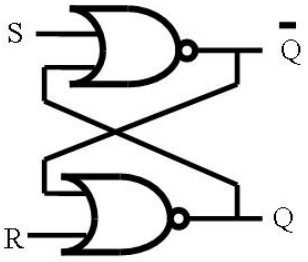
- Para que un flip-flop cambie de estado, es necesario introducir una señal de entrada.

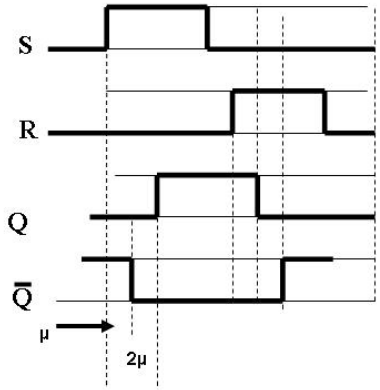
- El flip-flop tiene dos salidas, $\overline{\mathbf{Q}}$ y $\mathbf{Q}$, las cuales son siempre complementarias.

A continuación explicaremos cada uno de los diferentes tipos de flip-flop utilizados en el diseño de circuitos secuenciales en una computadora.

- **Flip-Flop SR**

El flip-flop SR es un circuito biestable que retiene o almacena un único bit de información. El flip-flop SR tiene dos entradas, S (Set) y R (Reset), y dos salidas, $\overline{\mathbf{Q}}$ y $\mathbf{Q}$, y puede estar construido a partir de dos puertas NOR unidas por una retroalimentación, (ver figura 6.2), o por dos compuertas NAND también unidas por una retroalimentación, (ver figura 6.3.)





Q_t	S_t	R_t	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	(Prohibido)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	(Prohibido)

Figura 6.2 Un circuito Flip-Flop S-R a base de compuertas NOR

El funcionamiento de este flip-flop SR es el siguiente: primero supongamos que **S** y **R** valen 0 y que **Q** es 0. Las entradas a la compuerta NOR superior son Q=0 y S=0. Entonces, la salida $\overline{\mathbf{Q}}$=1 alimenta a la entrada de la compuerta NOR (inferior) y con R = 0, produce salida **Q** = 0. Por tanto, el estado del circuito permanece estable mientras **S** = **R** = 0.

Como se había mencionado al inicio de esta sección, este tipo de flip-flop puede funcionar como una memoria de 1 bit. A partir de la figura 6.2, podemos ver la salida **Q** como el "valor" del bit. Las entradas **S** y **R** sirven para escribir los valores 1 y 0, respectivamente, en la memoria. Para ver esto, consideramos el estado **Q** = 0, $\overline{\mathbf{Q}}$=1, **S** =0, **R** = 0. Supongamos que **S** cambia al valor 1. Ahora las entradas a la compuerta NOR inferior son **S**=1, $\overline{\mathbf{Q}}$=0.

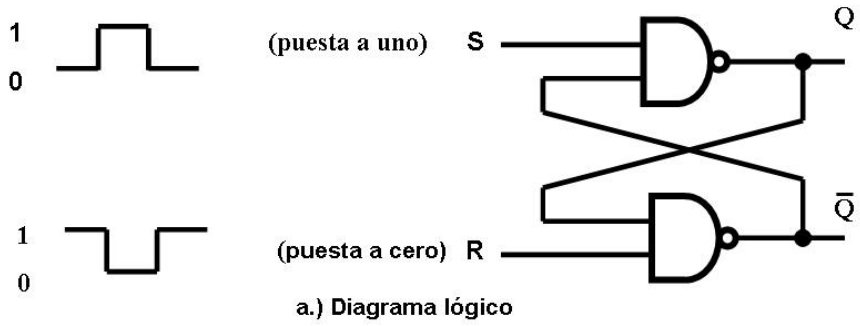
Después de cierto tiempo de retardo (μ, la salida de la puerta NOR inferior será $\overline{\mathbf{Q}}$=0).

Así que, en este momento, las entradas a la compuerta NOR superior pasan a ser **R**=0, $\overline{\mathbf{Q}}$ =0. Después de otro retardo de puerta de (μ), la salida **Q** pasa a 1. Este de nuevo es un estado estable. Las entradas de la parte inferior son ahora **S**=1, **Q**=1, que mantienen la salida **Q**=0. Mientras **S**=1 y **R**=0, las salidas seguirán siendo **Q**=1, $\overline{\mathbf{Q}}$=0. Además, si **S** vuelve a 0, las salidas permanecerán sin cambiar. Resumiendo, cuando la entrada **S** toma el valor de 1 a dicha acción se le conoce como "***PRESET***" y por lo tanto coloca la salida **Q** en 1.

La entrada **R** realiza la función contraria a la entrada **S**, es decir, cuando **R** tiene el valor de 1, coloca las salidas con los valores de **Q**=0, $\overline{\mathbf{Q}}$=1, sin importar el estado previo de **Q** y $\overline{\mathbf{Q}}$. A esta operación se le conoce como "***RESET o CLEAR***", debido a que coloca la salida **Q** en 0. De nuevo, hay un tiempo de retardo de (2μ) antes de que se restablezca la estabilidad.

El flip-flop **SR** se puede definir a partir de una tabla parecida a una tabla de verdad llamada *tabla característica*, que muestra el siguiente estado o estados de un circuito secuencial en función de los estados y entradas actuales. En el caso del flip-flop **SR** el estado se puede definir por el valor de **Q**. La figura 6.2 muestra la tabla característica resultante. A partir de dicha tabla, se observa que las entradas **S**=1, **R**=1 no están permitidas, ya que producirán una salida inconsistente ($\overline{\mathbf{Q}}$ y **Q** iguales a 0).

Como se mencionó al inicio de esta sección, existen diferentes formas de construir un flip-flop RS, utilizando compuertas básicas interconectadas, entre las cuales se encuentra el flip-flop RS construido a partir de dos compuertas NAND interconectadas como se muestra en la figura 6.3.



a.) Diagrama lógico

S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	(Despúes de S = 1, R = 0)
0	1	1	0	
1	1	1	0	(Después de S = 0, R = 1)
0	0	1	1	

b.) Tabla de Verdad

Figura 6.3 Flip-flop SR a base de compuertas NAND

En la figura 6.3, se presenta el flip-flop **SR** a base de compuertas NAND y el cual tiene dos entradas **S** (**Set**, puesto a uno) y **R** (**Reset**, puesta a cero), dos salidas **Q** y $\overline{\mathbf{Q}}$ una tabla de verdad.

Como se mencionó anteriormente, el flip-flop **SR** puede implementarse utilizando dos compuertas NAND interconectadas, caso en el que el estado de reposo es el que corresponde a **S**=**R**=1. Utilizando el teorema de de Morgan, se puede convertir las compuertas NOR de un flip- flop **SR** en compuertas AND, según se ve en la figura 6.4. Operando con inversores, se reemplazan las compuertas AND por compuertas NAND, luego se invierten los sentidos activos de **S** y **R** para eliminar los inversores de entrada restantes.

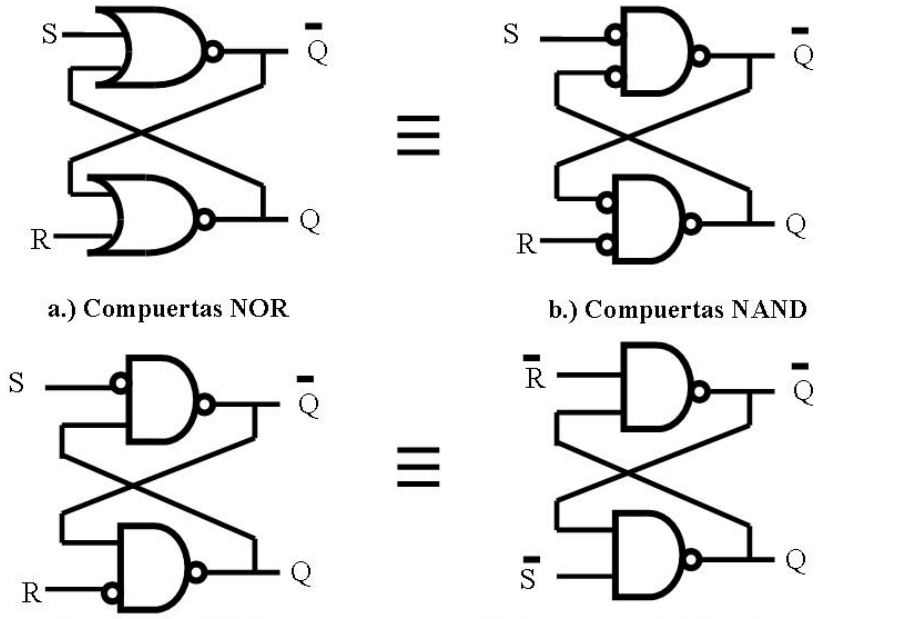
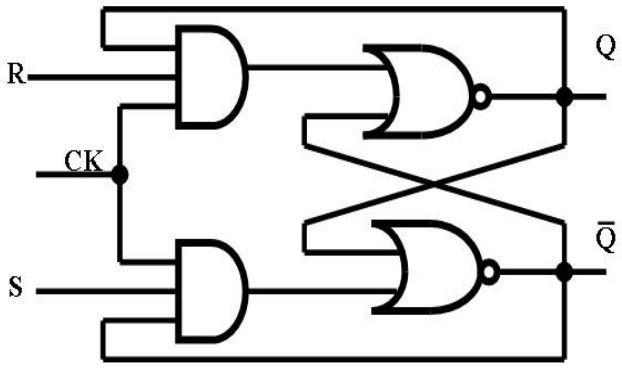


Figura 6.4 Implementación del flip - flop SR a partir de diversas compuertas básicas

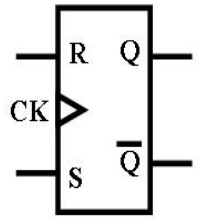
Existen otros tipos de flip-flops (**RS, JK, T** y **D**) a los cuales se les conoce como *flip-flop temporizados* y los cuales son muy utilizados en el diseño e implementación de circuitos secuenciales, veámoslos:

Flip-flop RS síncrono

Este flip-flop funciona mediante la sincronización con un pulso de reloj, y de esta manera los cambios ocurren sólo con el pulso de reloj. La figura 6.5 muestra la configuración de este flip-flop, al cual se denomina flip-flop *RS síncrono*. Nótese que las entradas **R** y **S** se aplican a las entradas de las puertas **AND** sólo durante el pulso de reloj. En dicha figura se muestra su símbolo lógico, tabla característica, tabla de excitación y ecuación característica, las cuales son muy empleadas en el diseño e implementación de circuitos secuenciales, como lo mostraremos más adelante.



a.) Diagrama Lógico



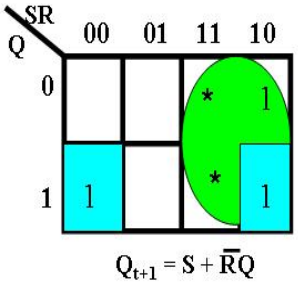
b.) Símbolo Lógico

Q	S	R	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indefinido
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indefinido

c.) Tabla Característica

Q_t	Q_{t+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

d.) Tabla de excitación



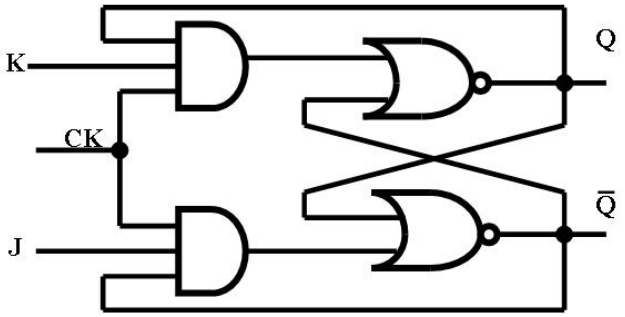
$Q_{t+1} = S + \overline{R}Q$

e.) Ecuación Característica

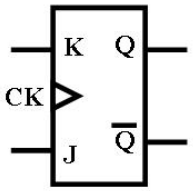
Figura 6.5 Flip Fliop RS Temporizado

- **Flip-flop JK Temporizado**

El flip-flop *JK temporizado* es otro de los flip-flops más utilizados en el diseño de circuitos digitales. El flip-flop JK temporizado se propone como una mejora al flip-flop **RS** temporizado ya que este flip-flop presenta dos estados indefinidos. El flip-flop **JK** se comporta en forma similar al flip-flop **RS**, excepto porque cuando las dos entradas valen simultáneamente 1, el circuito conmuta el estado anterior de su salida. La figura 6.6 muestra una implementación a base de compuertas del flip flop **JK**, además de mostrar su símbolo lógico, tabla característica, tabla de excitación y ecuación característica, las cuales son muy empleadas en el diseño e implementación de circuitos secuenciales.



a.) Diagrama Lógico



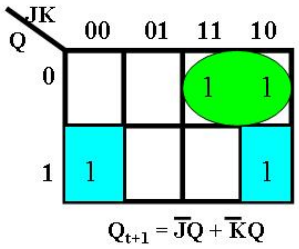
b.) Símbolo Lógico

Q	J	K	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

c.) Tabla Característica

Q_t	Q_{t+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

d.) Tabla de excitación



e.) Ecuación Característica

Figura 6.6 Flip-flop JK Temporizado

Las entradas **JK** solo realizan la función de puesta a 1, causando que la salida sea 1; la entrada **K** solo realiza la función de puesta a cero, provocando que la salida sea 0. Cuando **J** y **K** son 1, la función realizada se denomina función de conmutación: la salida se invierte.

Otra vez, puede surgir algún inconveniente cuando en un flip-flop **JK** se tienen las dos entradas **J** y **K** en 1 y se lleva la señal de reloj a su estado activo. En esta situación el estado puede cambiar de estado más de una vez mientras el reloj está en su estado alto. Esta es otra situación en que se hace apropiado el uso de un flip-flop **JK** de estructura maestro-esclavo.

El esquema de un flip-flop **JK** maestro-esclavo se ilustra en la figura 6.6f. El problema de la “oscilación infinita” se resuelve con esta configuración, aun cuando la misma crea otro inconveniente. Si se mantiene una entrada en nivel alto, el flip-flop puede llegar a ver el 1 como si fuera una entrada válida, durante un tiempo dado mientras la señal de reloj se encuentra activa, aunque fuese

porque se encuentre en una transición previa a establecerse. La situación se resuelve si se eliminan los riesgos en los circuitos que controlan las entradas.

Se puede resolver el problema de la “captura de unos” por medio de la construcción de flip-flops activados por flanco, en los que el estado de la entrada se analiza solo en las transiciones del reloj (de alto a bajo) si el circuito se activa por flanco negativo o de bajo a alto, se trata de un flip-flop activado por flanco positivo, instantes en los cuales las entradas deberían estar estables.

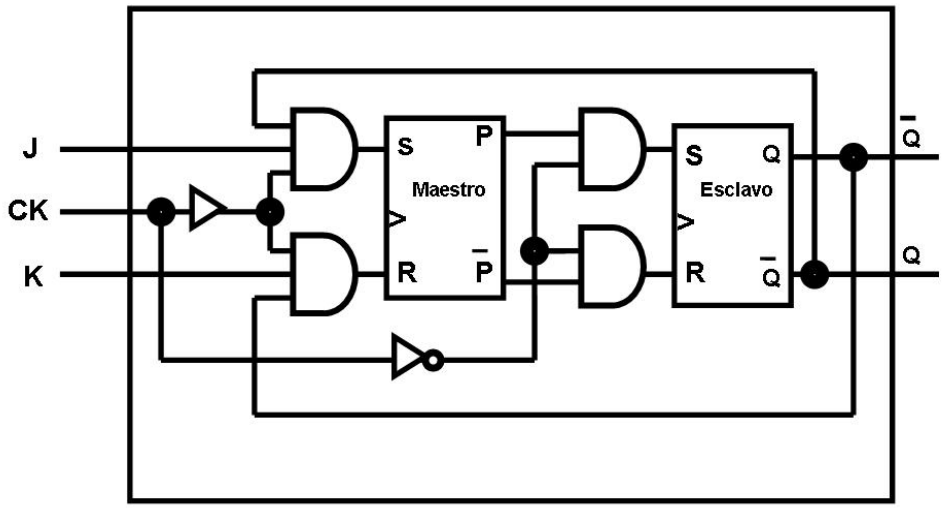
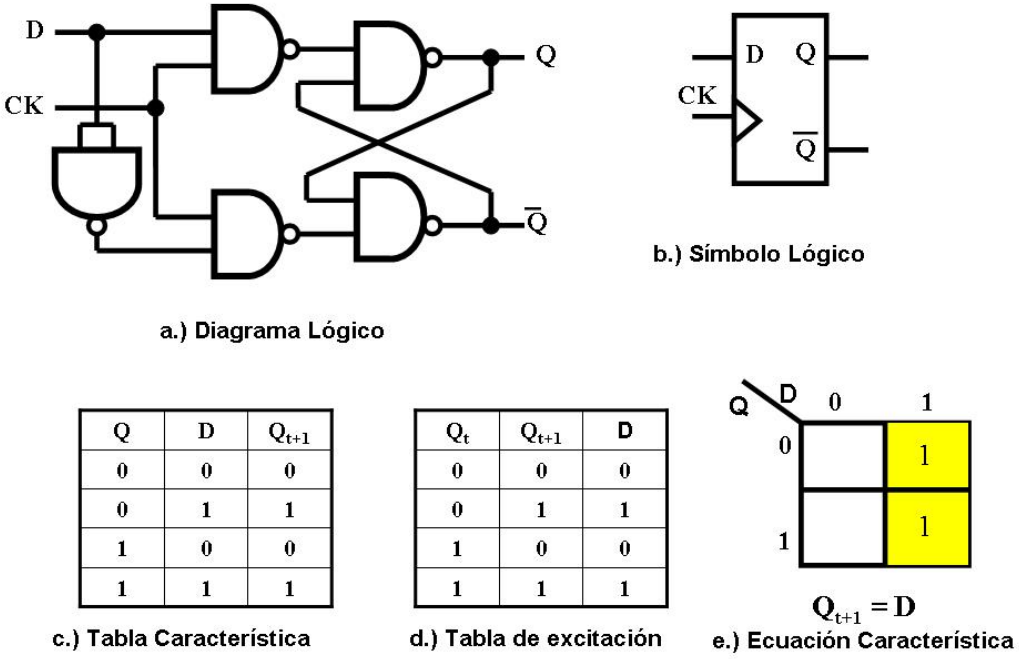


Figura 6.6f Flip-flop Maestro-Esclavo JK (Continuación)

- **Flip-flop tipo D**

El problema con los flip-flop **RS** es que la condición **R**=1, **S**=1 debe ser evitada. Una manera de hacerlo es permitir solo una única entrada. El flip-flop tipo **D** lo cumple. La figura 6.7 muestra una implementación con compuertas **NAND**, la tabla característica, tabla de excitación y ecuación característica del flip-flop tipo **D**.



Q	D	Q_{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Figura 6.7 Flip Flop D temporizado

El flip-flop tipo **D** a veces se denomina “flip-flop de datos”, porque en efecto, almacena un bit de datos. La salida del flip-flop tipo **D** es siempre igual al valor más reciente aplicado a la entrada, por tanto, recuerda y produce la última entrada. También se le llama biestable de retardo, porque retrasa un cero o un uno aplicado a la entrada durante un pulso de reloj.

Un flip-flop tipo **D** se usa en situaciones en las que exista realimentación desde la salida hacia la entrada a través de otros circuitos, esta realimentación puede provocar que el flip-flop cambie una sola vez por ciclo de reloj, se suele cortar el lazo de realimentación a través de la estructura conocida como maestro-esclavo que se muestra en la figura 6.7f. El flip-flop **maestro-esclavo** consiste en dos flip flops encadenados, donde el segundo utiliza una señal de sincronismo que está negada con respecto a la que se utiliza en el primero de ellos. El flip-flop maestro cambia cuando la entrada principal de reloj está en su estado alto, pero el esclavo no puede cambiar hasta que su entrada no vuelva a bajar. Esto significa que la entrada **D** se transfiere a la salida $\mathbf{Q_s}$ del flip-flop esclavo recién cuando la señal de reloj sube y vuelve a bajar. El triángulo utilizado en el símbolo del flip-flop maestro-esclavo indica que las transiciones

de la salida ocurren solo en un flanco creciente (transición 0-1) o decreciente (transición 1-0) de la señal de reloj. No se producen transiciones continuas en la salida cuando la señal de reloj se encuentra en su nivel alto, como ocurre con el circuito sincrónico simple. Para la configuración de la figura 6.f., la transición de la salida se produce en el flanco negativo de la señal de sincronismo.

Un flip-flop activado por nivel puede cambiar sus estados en forma continua cuando la señal de reloj está en su estado activo (alto o bajo, según como se haya diseñado el flip-flop). Un flip-flop activado por flanco solo cambia en una transición creciente o decreciente de la señal de reloj.

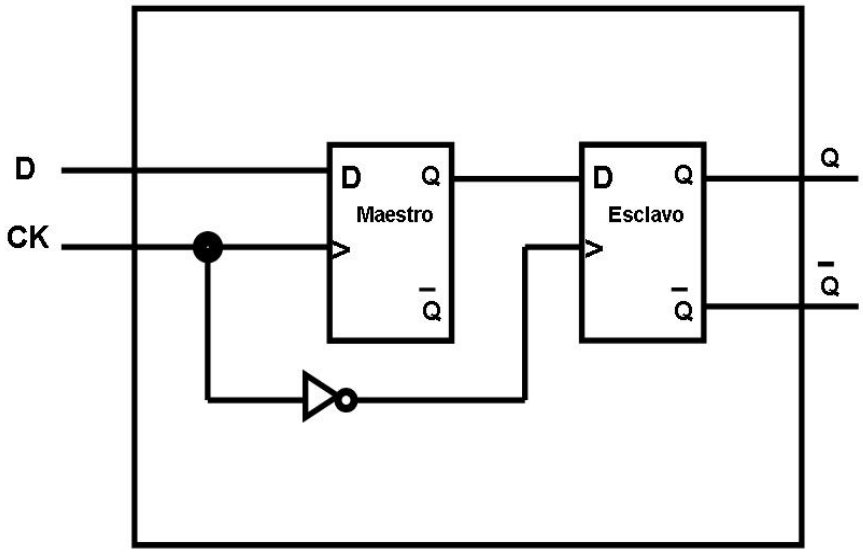
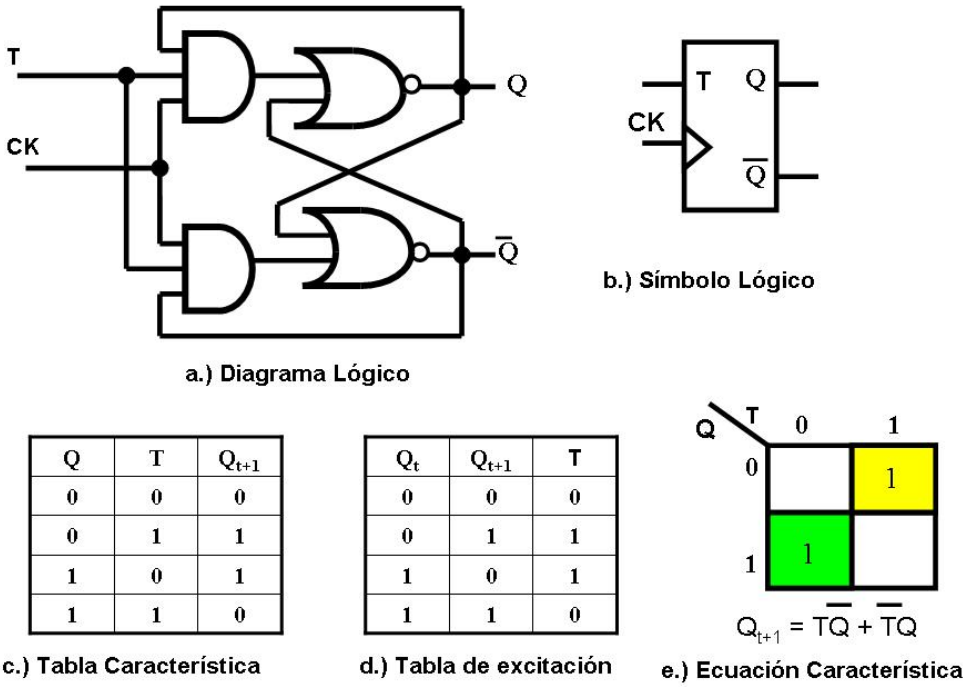


Figura 6.7f Flip-flop Maestro-Exclavo D (Continuación)

- **Flip-flop T**

El flip-flop T (por “toggle”) alterna sus estados, como ocurre en el flip-flop **JK**, cuando sus entradas están ambas en 1. Este flip-flop se comporta en forma similar al flip-flop **SR**, excepto porque cuando las dos entradas valen

simultáneamente 1, el circuito conmuta el estado anterior de su salida, (ver figura 6.8). En dicha figura se muestra su símbolo lógico, tabla característica, tabla de excitación y ecuación característica.



Q	T	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

c.) Tabla Característica

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

d.) Tabla de excitación

Figura 6.8 Flip Flop T temporizado

Diseño de un Circuito Secuencial

Haciendo nuevamente referencia al circuito secuencial de la figura 6.1 y utilizando el modelo Mealy, se puede diseñar e implementar los elementos que constituyen el bloque de memoria utilizando flip-flops del tipo **RS**, **JK**, **T** o **D**, en tanto que la señal de sincronización puede generarse a través de una señal de reloj del sistema (Temporizador), ver sección 6.5.

El procedimiento para diseñar un circuito secuencial síncrono es el siguiente:

1) **Enunciado del problema**

   Se establece la descripción en palabras del comportamiento del circuito, esto puede acompañarse por:

   El diagrama de estado

Un diagrama de tiempos, u

Otra información pertinente (diagrama de flujo, carta asm, etc.)

2) Obtención tabla de estado

De la información recabada del punto anterior, se obtiene la tabla de estado.

3) Reducción del número de estados en el circuito secuencial

El número de estados puede reducirse por algún método de reducción de estados, siempre y cuando el circuito secuencial pueda caracterizarse por las relaciones de entrada-salida independientemente del número de estados.

4) Asignación de valores binarios a cada estado

Se asigna valores binarios a cada uno de los estados. Esto se realiza si en la tabla de estado obtenida en el paso 2 o en la tabla de estado reducida (obtenida en el punto 3) contienen símbolos de letras o números.

5) Se obtiene el número de Flip-flops a utilizar

Se determina el número de flip-flops necesarios para cubrir el número total de estados. Esto se logra despejando el valor de **n** en la siguiente ecuación:

$$N = 2^n$$

es decir,

$$n = \frac{\lg(N)}{\lg(2)}$$

donde

n Es el número de flip-flops necesarios

N Número total de estados

6) Elección del flip-flop por utilizar

Se selecciona el tipo de flip-flops que se va a utilizar en el circuito secuencial.

7) Obtención de la ecuación de excitación

A partir de las tablas de estado se deduce la excitación (ecuación) del circuito y la tabla de salida (si fuera el caso).

8) Obtención de las funciones de salida

Usando cualquier método de simplificación (por ejemplo, mapas de Karnauhg o álgebra de Boole) se deducen las funciones de salida del circuito **n** flip-flops.

9) Dibujar el diagrama lógico

Se dibuja el diagrama lógico (y se comprueba el circuito secuencial).

10) Dibujar el diagrama eléctrico (opcional)

Se dibuja y se alambra el diagrama eléctrico.

A continuación realizamos algunos ejemplos para mostrar el diseño de un circuito secuencial síncrono.

Ejemplo 1

Solución

1. Se desea diseñar un circuito secuencial síncrono utilizando flip-flops del tipo **JK** a partir del siguiente diagrama de estados, (ver figura 6.9)

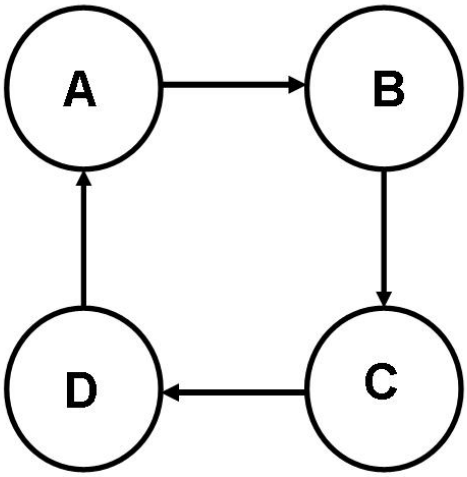


Figura 6.9. Ejemplo 1. Diagrama de Estados

2. Obtención tabla de estado

Estado Presente	Estado Futuro
A	B
B	C

Tabla de estados

3. Reducción de estados

No se aplica la reducción de estados.

4. Asignación de estados

Estado	Valor
A	00
B	01
C	10
D	11

Utilizando esta asignación de estados, la tabla de estado queda de la siguiente manera:

Estado Presente	Estado Futuro
00	**01**
01	**10**
10	**11**
11	**00**

Tabla de estados

5. Se determina el número de Flip-flops por utilizar

A partir de la ecuación:

$$N = 2^n$$

donde

N Número de estado

n Número de flip-flop a utilizar

en nuestro caso **N** = 4 estados, hay que determinar el valor de **n**.

Despejando **n** obtenemos

$$n = \frac{\lg(N)}{\lg(2)} = \frac{\lg(4)}{\lg(2)} = \frac{0.602059991}{0.30102999566} = 2$$

Por lo tanto se requieren de 2 flip-flops para representar los cuatro estados, los **A, B, C** y **D**.

6. Elección del flip-flop por utilizar

En este ejemplo se seleccionó (a partir del enunciado del problema) el flip-flop JK.

7. Obtención de la ecuación de excitación

A partir de las tablas de estado se deduce la excitación del circuito y la tabla de salida.

Estado Presente		Estado Futuro	
Q_0	Q_1	Q_0	Q_1
0	**0**	**0**	**1**
0	**1**	**1**	**0**
1	**0**	**1**	**1**

Tabla de estados

8. Obtención de las funciones de salida

Usando cualquier método de simplificación (por ejemplo, mapas de Karnaugh o álgebra de Boole) se deducen las funciones de salida del circuito de los **n** flip-flops.

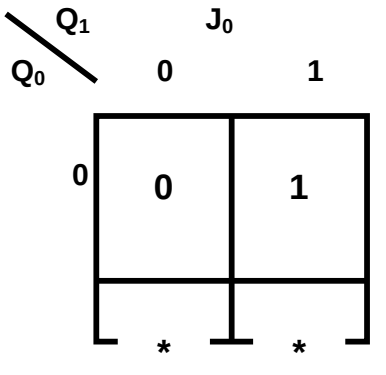
Utilizando la tabla característica del flip-flop **JK** se obtiene las funciones de salida del circuito y las funciones de entrada de los 2 flip-flops de la siguiente manera:

Q_t	Q_{t+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

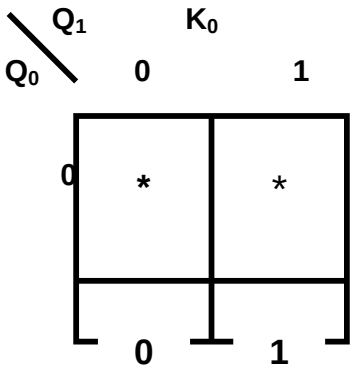
Tabla de Excitación

Estado Presente		Estado Futuro	
Q_0	Q_1	Q_0	Q_1
1	**0**	**0**	**1**
0	**1**	**1**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**

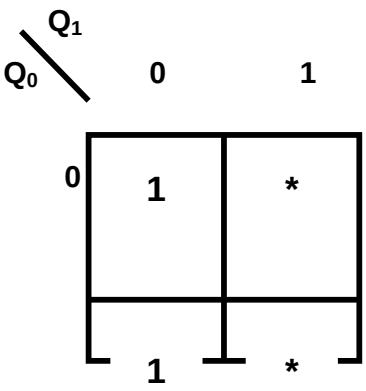
Tabla de estados

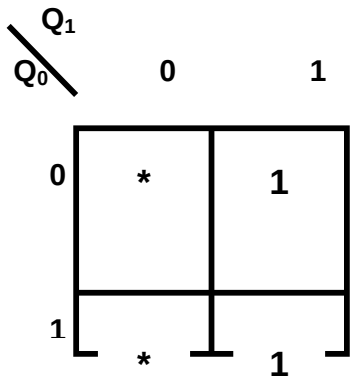


$J_0 = Q_1$



$K_0 = Q_1$



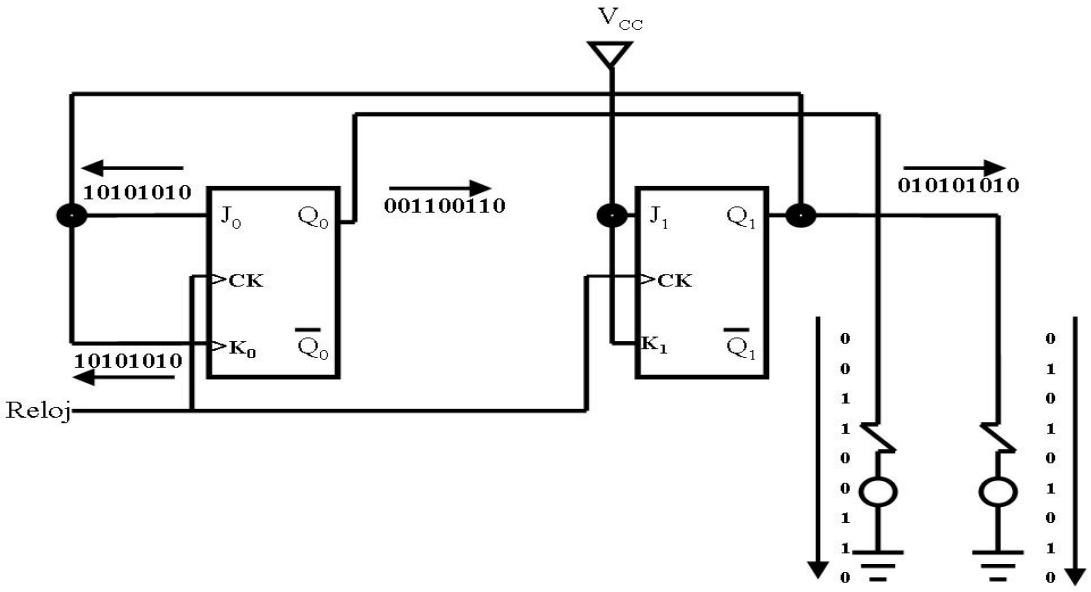


$J_1 = 1$

$K_1 = 1$

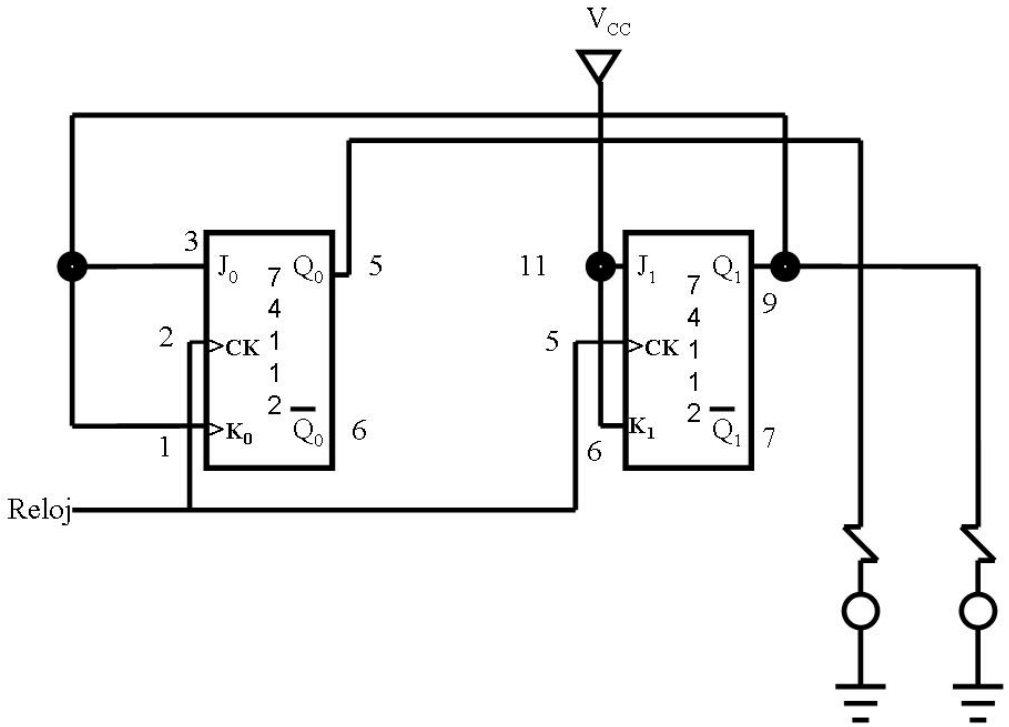
9) Dibujar el diagrama lógico

Comprobar el circuito secuencial



10) Dibujar el diagrama eléctrico

Se alambra el diagrama eléctrico



Ejemplo 2 Circuito Secuencial

Solución

1. Enunciado del problema

Se desea diseñar un circuito secuencial temporizado cuyo diagrama de estados se muestra en la figura 6.10 y utilizando flip-flop's **JK**.

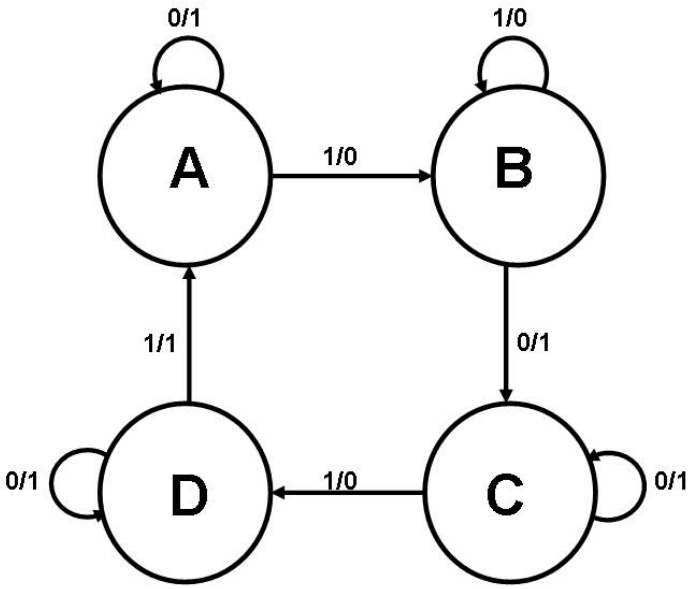


Figura 6.10. Ejemplo 2. Diagrama de Estados

Nota

La notación **x/y** significa que **x** es la variable de entrada y **z** es la salida.

2. Obtención de la tabla de estado

Estado Presente	Estado Futuro		Salida	
	X = 0	X = 1	X = 0	X=1
A	A	B	1	0
B	C	B	1	0

3. Reducción de estados

No se aplica la reducción de estados.

4. Asignación de estados

A	0	0
B	0	1
C	1	0
D	1	1

5. Número de Flip-flops

A partir de la ecuación:

$$N = 2^n$$

Donde

N Número de estado

n Número de flip-flop a utilizar

en nuestro caso **N** = 4 estados, hay que determinar el valor de **n**.

Despejando **n** obtenemos:

$$n = \frac{\lg(N)}{\lg(2)} = \frac{\lg(4)}{\lg(2)} = \frac{0.602059991}{0.30102999566} = 2$$

Por lo tanto se requieren de 2 flip-flops para representar los cuatro estados (**A, B, C** y **D)**.

6. Elección del flip-flop a utilizar

En este caso se seleccionó el flip-flop JK

7. Obtención de la ecuación de excitación

A partir de las tablas de estado se deduce la excitación del circuito y la tabla de salida.

Estado Presente	Estado Futuro		Salida	
	X = 0	X = 1	X = 0	X = 1
Q_0 Q_1	Q_0 Q_1	Q_0 Q_1		
0 0	**0 0**	**0 1**	**1**	**0**
0 1	**1 0**	**0 1**	**1**	**0**

Tabla de estados

8. Obtención de las funciones de salida

Usando cualquier método de simplificación (por ejemplo, mapas de Karnaugh o álgebra de Boole) se deducen las funciones de salida del circuito y las funciones de entrada de los **n** flip-flops.

Utilizando la tabla característica del flip-flop JK se obtiene las funciones de salida del circuito y las funciones de entrada de los 2 flip-flops de la siguiente manera:

Q_t	Q_{t+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

Tabla de Excitación

Estado Presente	Estado Futuro		Salida	
	X = 0	X = 1	X = 0	X = 1
Q_0 Q_1	Q_0 Q_1	Q_0 Q_1		
1 0	0 0	0 1	1	0
0 1	1 0	0 1	1	0

Tabla de estados

Q_1Q_0 / X	00	01	11	10
0	*	1	*	*
1	*	0	*	*

$J_0 = \overline{X}$

Q_0Q_1 / X	00	01	11	01
0	1	*	0	0
1	1	*	1	0

$K_0 = \overline{Q_0} + XQ_1$

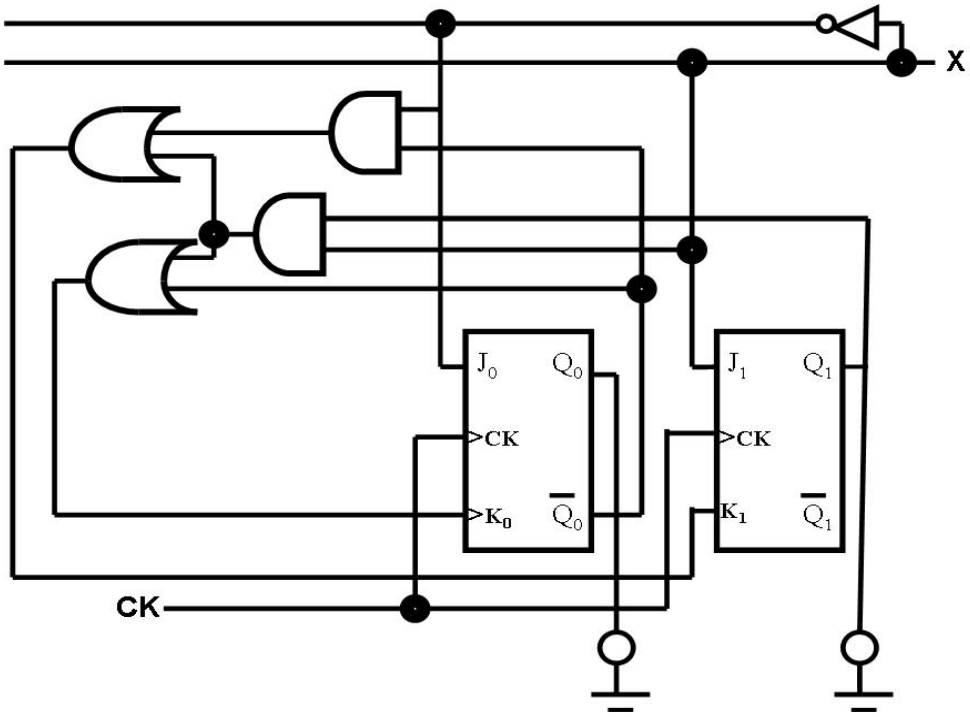
Q_1Q_0 / X	00	01	11	10
0	0	*	*	0
1	1	*	*	1

$J_1 = X$

Q_0Q_1 / X	00	01	11	10
0	*	1	0	*
1	*	0	1	*

$K_1 = \overline{X}\,\overline{Q_0} + X\,Q_1$

9. Dibujar el diagrama lógico



6.4 Registradores de corrimiento

Registro

Un registro es un grupo de celdas donde se almacena información binaria. Un registro está compuesto por un grupo de flip-flops, debido a que cada flip-flop es una celda binaria que almacena un bit de información. Un registro de **n**-bits tiene un grupo de **n** flip-flops y tiene la capacidad de acumular cualquier información binaria que contengan **n**-bits. Un registro, además de contar con **n**-flip-flops, emplea compuertas lógicas que controlan (el) cuándo y (el) cómo se transfiere la nueva información al registro.

Un registro puede ser

- Registro de corrimiento
- Registro en paralelo
- Registro universal

Registro de corrimiento

Un registro de corrimiento acepta y/o transfiere información vía serie. Un registro se puede construir utilizando alguno de los diferentes tipos de flip-flops, por ejemplo **RS**, **JK**, **T** y el **D**. En esta sección mostramos un registro de corrimiento (entrada serie-salida serie) de 4 bits utilizando flip-flop tipo **D**, (ver 6.11a). En la 6.11 b se muestra un diagrama de tiempos del mismo registro de corrimiento pero ahora introduciendo los datos: **0 1 0 1 0 0 0 0**.

El funcionamiento de este registro es el siguiente: Primero ponemos a todos los flip-flops en condiciones iniciales, es decir, “0”, esto se realiza con la operación de limpiar (del inglés **Reset**), es decir, colocar todos los flip-flops en “0”. A continuación, colocamos el dato “0” en la entrada del primer flip-flop y durante el primer pulso de reloj y esperamos el ***flanco de subida*** (es decir, el instante de tiempo que pasa de un nivel bajo a un nivel alto) en ese momento reconoce el dato “0” y lo muestra a la salida del primer flip-fliop ($\mathbf{Q_0}$) y los demás “0”s se recorren una posición hacia la derecha. Enseguida, introducimos el dato “1” en

la entrada del primer flip-flop y durante el segundo pulso de reloj esperamos el siguiente *flanco de subida*, el flip-flop 0 muestra el dato “1” en su salida, y los demás datos (“0”s) se recorren a la derecha una posición. En el tercer pulso de reloj, se introduce el dato “0” en la entrada del flip-flop 0, se espera el *flanco de subida* y este dato se presenta a la salida del flip-flop 0, el dato “1” que se tenía anteriormente, se recorre una posición a la derecha y se presenta en la salida del flip-flop 1 (**Q_1**) y los demás datos se recorren una posición hacia la derecha. En el cuarto pulso de reloj, se introduce el dato “0” en la entrada del flip-flop 0, durante el *flanco de subida*, este dato se presenta a la salida del flip-flop 0, el “1” que se tenía a la salida de este flip-flop se recorre a la derecha y se presenta en la salida del flip-flop 1 y el “0” que se tenía en esta salida se recorre una posición hacia la derecha y los demás datos se recorren una posición a la derecha, y así sucesivamente hasta introducir todos los datos en el registro de corrimiento.

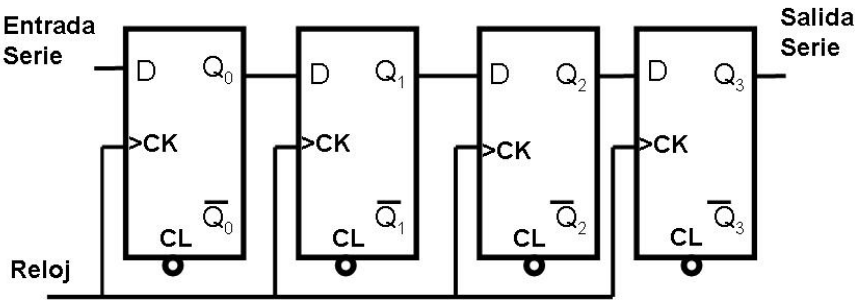


Figura 6.11a. Registro de desplazamiento de 4 bits.

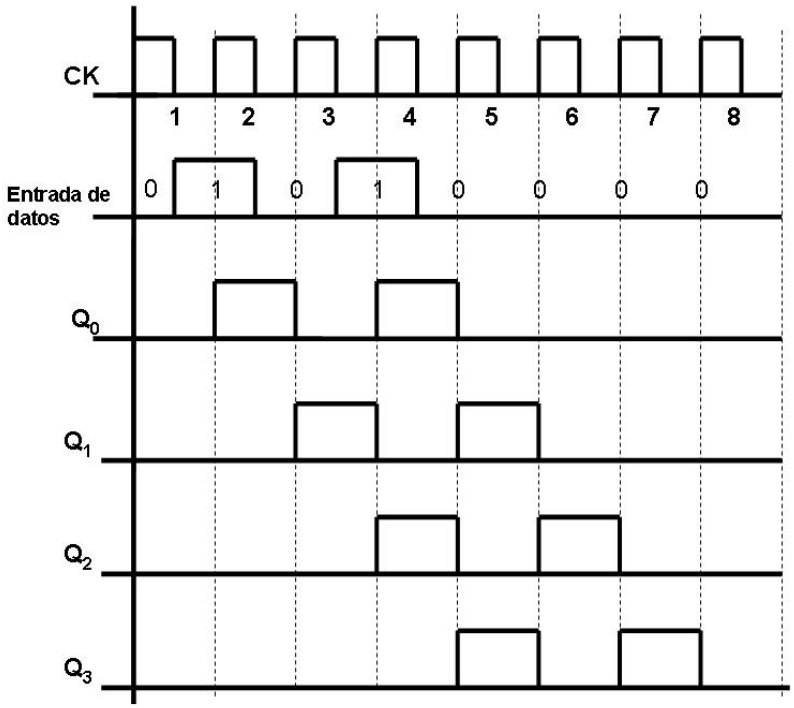


Figura 6.11b. Registro de corrimiento (entrada serie-salida serie)

Registro en paralelo

Un registro paralelo consiste en un conjunto de flip-flops en los cuales se puede leer o escribir simultáneamente. Un registro paralelo de 8 bits se muestra en la figura 6.12. El funcionamiento de este registro consiste en que una señal de control, llamada *validación de dato de entrada*, controla la escritura en los registros de los valores provenientes de las líneas de señales, de la $\mathbf{D_{10}}$ a la $\mathbf{D_{17}}$.

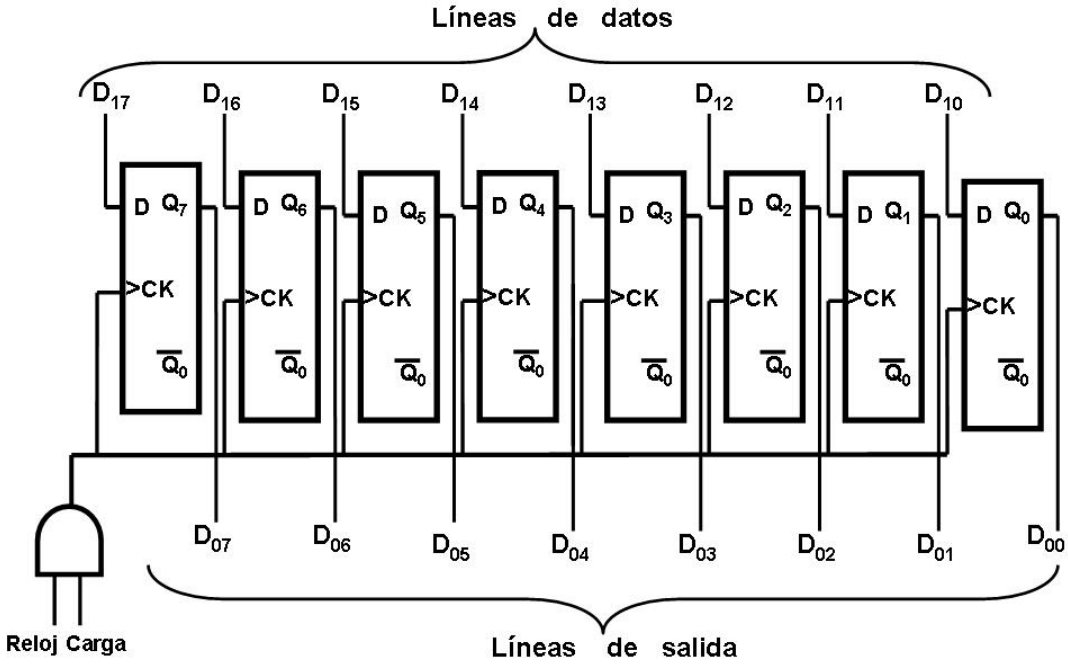


Figura 6.12. Registro en paralelo de 8 bits.

Registro universal

Un registro universal es una combinación del registro de corrimiento y el registro en paralelo para leer o escribir simultáneamente, introducir datos en serie por la derecha, sacar los datos en serie por la izquierda, cargar los datos en paralelo, sacar los datos en paralelo, cargar los datos en paralelos, sacar los los datos en serie por la derecha o por la izquierda. Un registro universal está formado por un conjunto de flip-flops y contiene una serie de señales de control que permiten realizar todas las operaciones mencionadas anteriormente.

6.5 Temporizadores

Un temporizador es un circuito generador de onda de una frecuencia específica. Un temporizador trabaja en los modos:

- Monoestable
- Biestable, y
- Astable

Los circuitos multivibradores **monoestables** encuentran amplia aplicación en las computadoras. Los multivibradores **biestables** se emplean en los contadores binarios para generar señales de tiempo para las distintas operaciones de la computadora y en los registros de desplazamiento para recorrer los datos binarios a todas las unidades de la computadora.

El multivibrador **astable** se utiliza para modificar la forma de onda de las diversas señales, prolongando su duración si son demasiado breves o acortándola si son demasiado largas; también se emplea para modificar la forma de onda de una señal que se origine con un retardo prefijado, una vez disparado el circuito **monoestable**.

Una forma de realizar un generador de onda cuadra, el cual va a funcionar como reloj para los diferentes circuitos que componen una computadora, es utilizando transistores o circuitos integrados.

Un circuito temporizador se puede implementar con el C.I. LM 555 (en modo **astable**), para generar una onda cuadrada, ver figura 6.13.

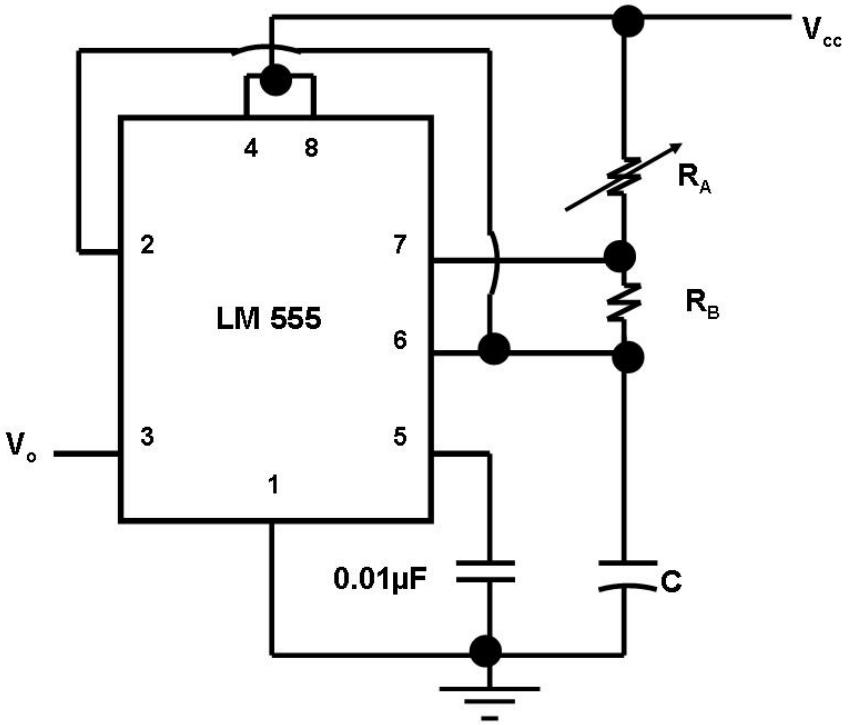


Figura 6.13 C.I. LM555 (Configuración Astable)

Para generar un oscilador de onda cuadrada que tenga un funcionamiento de 1 [Hz] a 20 [Hz], utilizando el C.I. LM 555 (en configuración astable) se realiza de la manera siguiente:

Procedimiento

Se tiene que f1 = 1Hz = t1 = 1 / f1 = 1seg

f2 = 20 Hz = t2 = 1 / f2 = 0.05 seg.

$$T = t_1 + t_2 \quad (1)$$

$$t_1 = (0.693)\, C\, (R_A + R_B) \quad (2)$$

$$t_2 = (0.693)\, C\, (R_B) \quad (3)$$

$$T = (0.693)\, C\, (R_A + 2R_B) \quad (4)$$

Se propone que C=200[µF] (Capacitor electrolítico de valor comercial)

Despejando R_B de la ec. (3)

$$R_B = \frac{0.05}{(0.693)(220 \times 10^{-6})} = 327.90 \text{ [ohm]}$$

de la ec. (2) y (4)

6.6 Contadores

Un circuito secuencial que pasa por una secuencia preestablecida de estados después de cada pulso de reloj se llama un contador. En un contador la secuencia de estados puede seguir una cuenta binaria o cualquier otra secuencia de estados.

Se tienen varios tipos de contadores entre los cuales destacan el contador binario y el contador binario en décadas (contador decimal) los cuales explicaremos a continuación.

Contador binario

Un contador de **n**-bits que sigue la secuencia binaria se llama contador binario. Un contador binario de **n**-bits consiste de **n** flip-flops y puede contar en binario de **0** hasta $\mathbf{2^n-1.}$ En la figura 6.14 se muestra un contador binario de cuatro etapas en el que la señal de entrada (señal de reloj) se aplica a la etapa 2^0. La salida de cada etapa es designada por el número de orden de la etapa (2^0, 2^1, 2^2, etc), el cual se toma de la salida $\mathbf{Q^n}$ del flip-flop. Obsérvese que, en este caso, el disparo para cada etapa sucesiva procede también de un valor positivo. Cada vez que la señal de entrada de reloj cambia en sentido negativo, se completa la etapa 2^0.

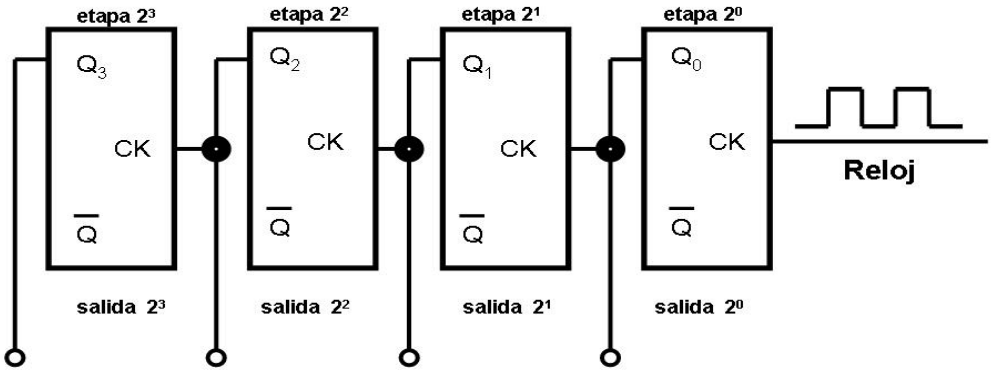


Figura 6.14 Diagrama a bloques de un Contador Binario

Utilizando lógica positiva, resulta que el flip-flop terminará cuando la entrada cambie de 1 a 0. Puesto que el primer impulso de reloj, aplicado a la entrada, cambia la salida de la etapa 2^0 de 0 a 1, la etapa 2^1 no se terminará. Solamente cambia de estado la etapa 2^0. La entrada del segundo impulso hará que se complemente de nuevo la etapa 2^0, pero pasando ahora de 1 a 0. Este cambio hace complementar a la etapa 1, con lo que su salida pasará de 0 a 1. Ninguna de las restantes etapas queda afectada por estos cambios. Mostrando estos

pasos en forma de tabla se observará fácilmente el mecanismo de funcionamiento (ver tabla 6.1).

Reloj	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1 ←
2	0	0	1	0
3	0	0	1 ←	1 ←
4	0	1	0	0
5	0	1	0	1 ←
6	0	1	1	0
7	0	1 ←	1 ←	1 ←
8	1	0	0	0
9	1	0	0	1 ←
10	1	0	1	0
11	1	0	1 ←	1 ←
12	1	1	0	0
13	1	1	0	1 ←
14	1	1	1	0
15	1 ←	1 ←	1 ←	1 ←
16 ó 0	0	0	0	0

Tabla 6.1 Contador binario de cuatro etapas

A partir de la tabla 6.1 se puede observar las flechas que indican cuando el cambio de 1 a 0 produce el disparo de una etapa sucesiva. Obsérvese que la etapa 2^0 cambia en cada uno de los ciclos, la 2^1 solamente en cuatro, la 2^2 solamente en dos, y la 2^3 en uno. Este hecho puede interpretarse como una disminución de la velocidad del ciclo para las etapas de orden superior. Con 16 impulsos, la primera etapa describe el ciclo ocho veces ($16/2^1$), la siguiente cuatro veces ($16/2^2$); la tercera etapa, dos veces ($16/2^3$), y la cuarta, una vez

($16/2^4$). Esta disminución del ciclo puede representarse, también, mediante un diagrama de tiempos como lo indica la figura 6.15. Esta figura muestra, la señal de entrada (reloj) con las señales de salida de los diferentes flip-flops (**Q_3, Q_2, Q_1**, y **Q_0**) de cada una de las etapas indicadas.

Puede verse que la frecuencia del ciclo de cada etapa se reduce en un factor de 2. Por consiguiente, al circuito lógico de la figura 6.14 también se le conoce como un divisor de frecuencia. Si la frecuencia de la señal de entrada, por ejemplo hubiese sido de 256 000 [Hz] la señal de salida es de 256000/16.

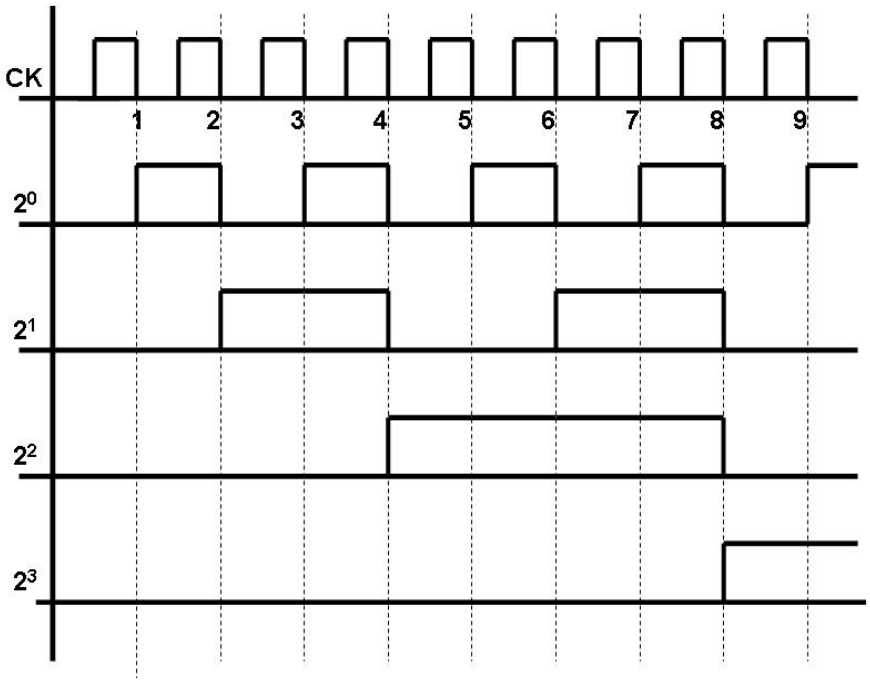


Figura 6.15 Formas de onda de un Divisor de frecuencias

Algunas veces se necesita otro factor de recuento. Normalmente, el factor debe ser diez, de modo que el recuento sea algún múltiplo de diez para su empleo en operaciones decimales. Existen diversas técnicas para modificar un contador binario. El método más difundido consiste en utilizar realimentación con objeto de adelantar el conteo. Cuando se desea un cierto valor de conteo se elige el número de etapas de modo que sea proporcional al número binario

inmediato más alto, y se emplea el conteo en un número igual al número de pasos excedentes.

Por ejemplo, para contar 6 unidades en un contador de tres etapas (conteo hasta 8) ha de utilizarse realimentación para adelantar el conteo en dos pasos. Ocho menos dos proporciona el conteo deseado, es decir, seis, (ver figura 6.16).

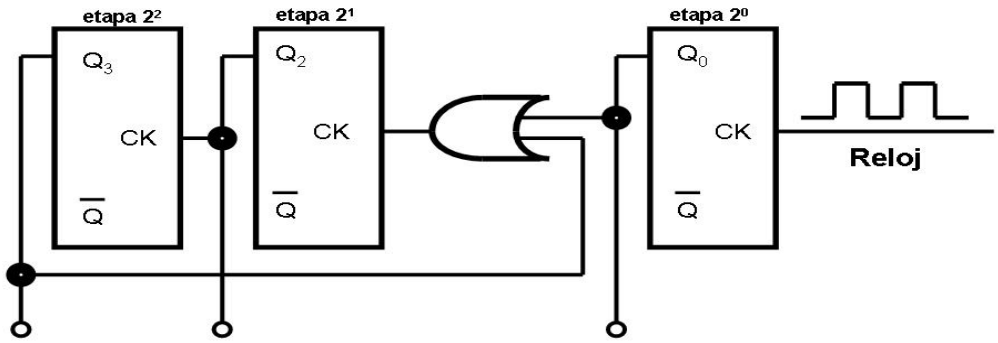


Figura 6.16 Diagrama a bloques de un Contador Binario mod 6

En la figura 6.16 se muestra un circuito para el conteo. Un contador para el conteo de 6 se suele llamar “módulo 6” (generalmente se abrevia mód 6), indicando el módulo del contador, es decir, el valor del impulso particular para el cual vuelve de nuevo a cero. Puesto que el impulso de disparo se produce mediante un cambio predeterminado de tensión (positivo o negativo), deberá tenerse presente su sentido al diseñar el circuito. Cuando se emplea lógica positiva, la tensión más positiva es el 1 y la menos positiva es el 0. Por consiguiente, al pasar de 1 a 0 se produce un cambio negativo. Cuando se utiliza lógica negativa, al ser menos positivo el 1 que el 0, se genera un escalón de tensión de sentido positivo.

Algunas veces el contador de décadas tiene que proporcionar el conteo binario equivalente para cada impulso decimal. Suele utilizarse una compuerta decodificadora para detectar el conteo final (10 en este caso) y para poner a cero el controlador. Un circuito de este tipo es el representado en la figura 6.17 y su tabla de recuento es la tabla 6.2

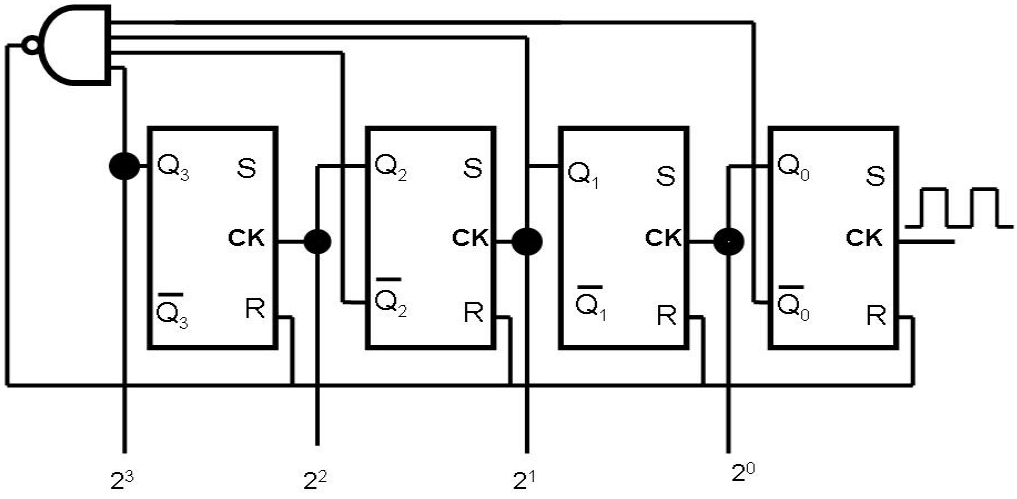


Figura 6.17 Contador binario de décadas

La compuerta produce un nivel bajo “0” cuando se alcanza el conteo 1010 (decimal 10) y repone (puesto a “0”) las cuatro etapas. La tabla 6.2 muestra que solamente es necesario poner los flip-flops 2^1 y 2^3 para volver a contar desde cero. Se reponen todas las etapas para asegurar que el contador empieza en cero en todo momento. La tabla también indica que el contador binario de cada paso decimal es exactamente el contador binario equivalente a dicho conteo decimal.

Reloj	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
↓	↓	↓	↓	↓
0	0	0	0	0

Tabla 6.2 Contador binario decimal

Para conseguir un contador en escala de 10 pueden asociarse varios contadores de décadas (ver figura 6.18).

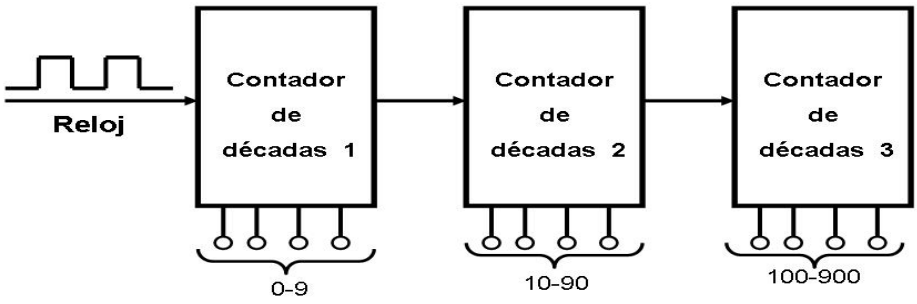


Figura 6.18 Contador de décadas para conteo en la escala de 10

Bibliografía del tema 6

Mano, Morris M., *Arquitectura de Computadoras,* México, Prentice Hall, 1988.

_____*: Diseño Digital,* México, Prentice Hall, 1987.

Ayala Pérez, Jaime, *Apuntes de Arquitectura de Computadoras, México,* Comunicación interna. 2007.

Mandado, E., *Sistemas, Electrónicos Digitales*, Madrid, Marcombo Boixareu, 1980.

Tocci, Ronald, *Digital Design, Principles and Applications*, USA, Prentice Hall, 1977.

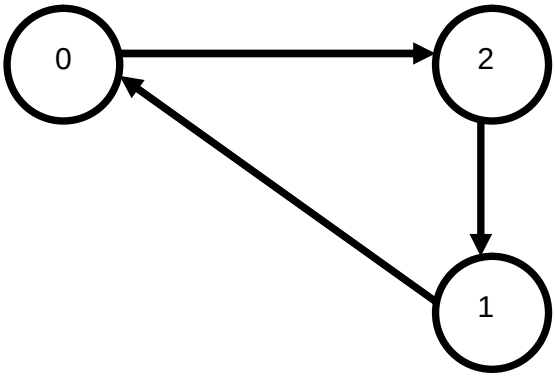
Charles H Roth Jr., *Fundamentals of Logic Design*, 4ª ed., USA, West Publishing Company, 1992.

Hayes P. John, *Diseño de Sistemas Digitales y Microprocesadores*, México, McGraw-Hill, 1986.

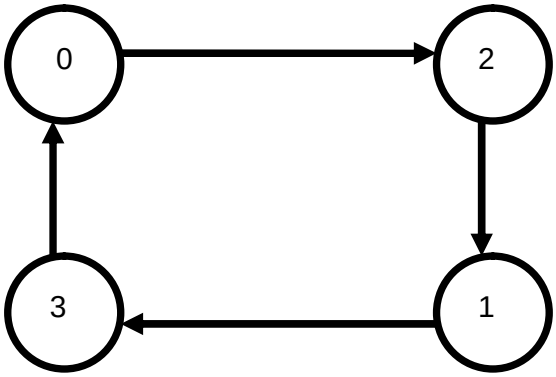
Lee, Samuel C., *Digital circuits and logic design,* Englewood Cliffs, Nueva Jersey*,* Prentice Hall, 1976.

Actividades de aprendizaje

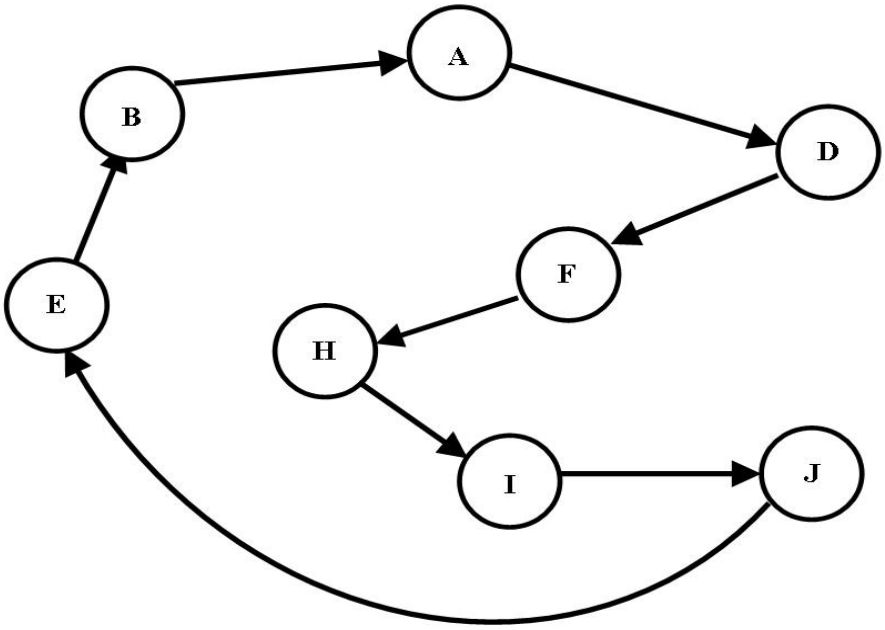
A.6.1. Diseña e implementa un circuito secuencial síncrono utilizando flip-flop **JK** a partir del siguiente diagrama de estados.



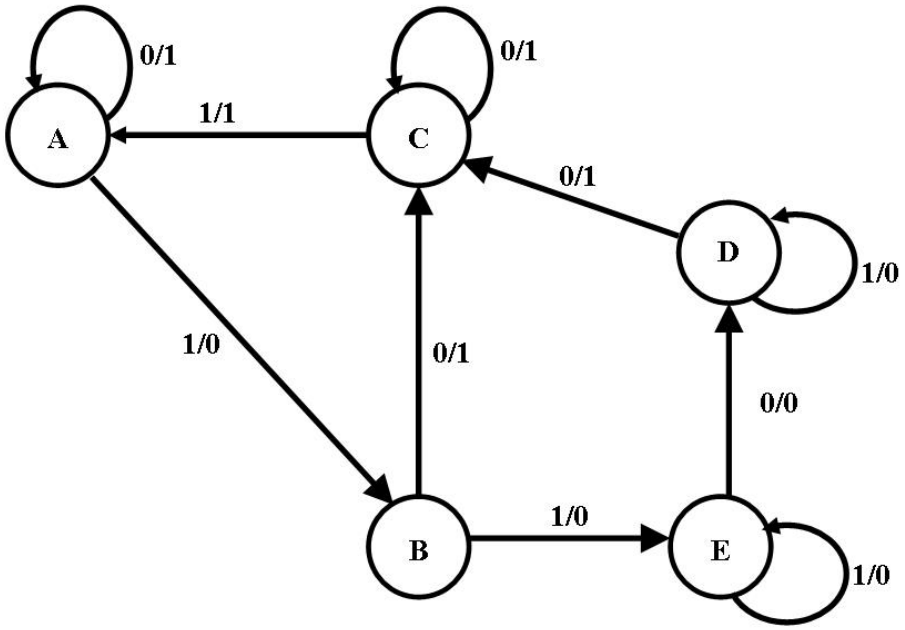
A.6.2. Diseña e implementa un circuito secuencial síncrono utilizando flip-flop **D** a partir del siguiente diagrama de estados.



A.6.3. Diseña e implementa un circuito secuencial síncrono utilizando flip-flop **D** a partir del siguiente diagrama de estados.

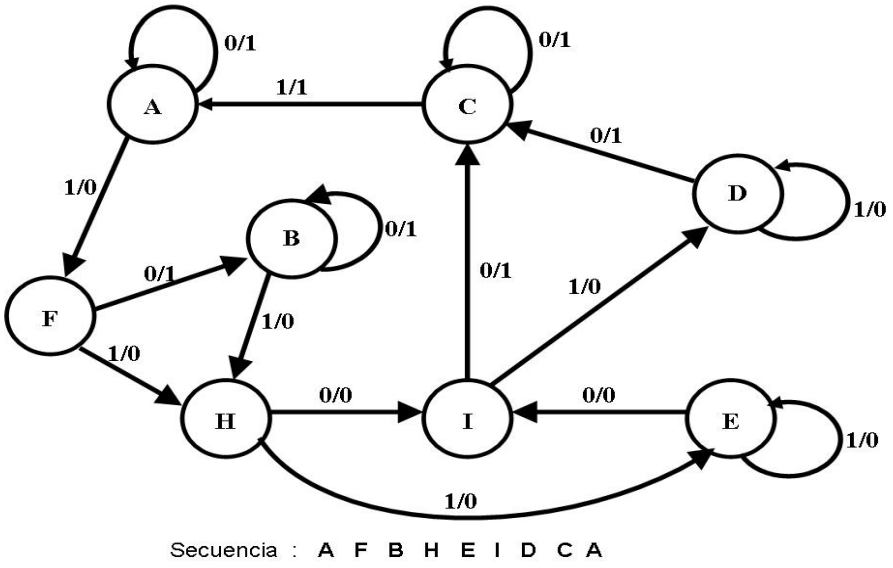


A.6.4. Diseña e implementa un circuito secuencial síncrono utilizando flip-flop **T** a partir del siguiente diagrama de estados.



Secuencia : A B E D C A

A.6.5. Diseña e implementa un circuito secuencial síncrono utilizando flip-flop **D** a partir del siguiente diagrama de estados.

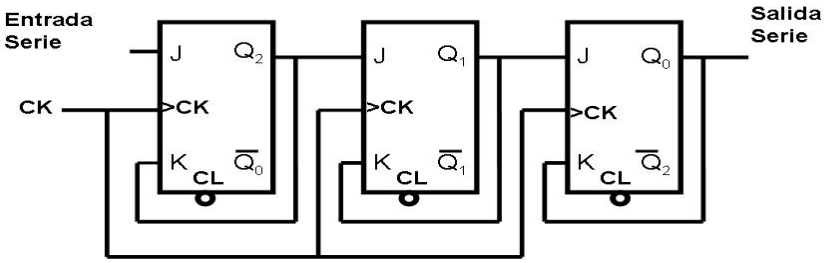


A.6.6. Diseña e implementa un oscilador de onda cuadrada que tenga un funcionamiento de 1 [Hz] a 50 [Hz], utilizando el C.I. LM 555 (configuración astable) y un capacitor electrolítico de 1000 [μF] a 25 Volts.

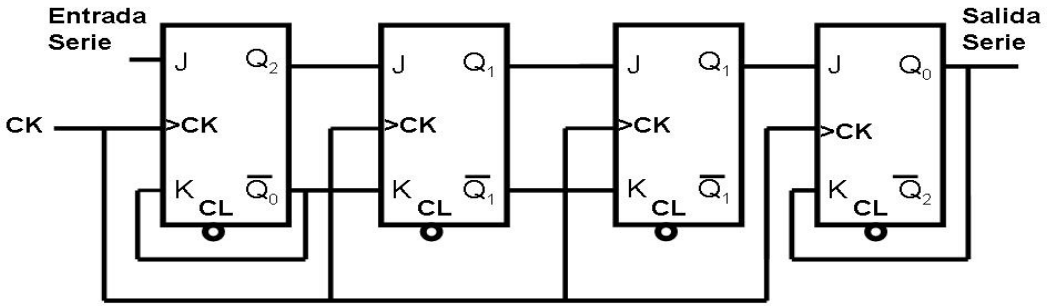
A.6.7. Dibuja el diagrama de tiempos de un registro de corrimiento de 6 bits (construido con flip-flops **D**) para una secuencia 1011010110101.

A.6.8. Dibuja el diagrama de tiempos de un registro de corrimiento de 6 bits (construido con flip-flops **T**) para una secuencia 10010110101.

A.6.9. Dibuja el diagrama de tiempos del siguiente registro de corrimiento de 3 bits para la secuencia 1001011. Implementa el circuito para comprobar sus resultados.



A.6.10. Dibuja el diagrama de tiempos del siguiente registro de corrimiento de 4 bits para la secuencia 10010110. Implemente el circuito para comprobar sus resultados.



Cuestionario de autoevaluación

1. ¿Qué es un circuito secuencial?
2. ¿Cuáles son los tipos de circuitos secuenciales?
3. ¿Qué es un flip-flop?
4. ¿Cuáles son los diferentes tipos de un flip-flop?
5. Menciona los pasos de diseño de un circuito secuencial síncrono.

6. ¿Qué es un temporizador?
7. ¿Cuáles son los modos de funcionamiento de un temporizador?
8. ¿Qué es un registro de corrimiento?
9. ¿Cuáles son los diferentes tipos de registro de corrimiento?
10. ¿Cuál es la diferencia entre flanco positivo y flanco negativo?
11. ¿Qué es un contador binario?
12. ¿Qué es un contador de décadas?

Examen de autoevaluación

Relaciona las columnas escribiendo dentro del paréntesis el número que una los conceptos con sus respectivas definiciones.

1) Contador		( ) Grupo de celdas donde se almacena información binaria
2) Flip flop T		( ) Circuitos que requieren una señal de reloj para funcionar
3) Astable		( ) Circuito secuencial que pasa por una secuencia preestablecida de estados después de cada pulso de reloj.
4) Flip-flop		( ) Bloque principal en un circuito secuencial
5) Reset		( ) Multivibrador utilizado para modificar la forma de onda de las señales en breves o largos tiempos
6) Registro		( ) Circuito generador de señales (pulsos)
7) Flip-flop D		( ) acción de colocar las salida del flip flop a cero
8) Temporizador		( ) flip-flop denomina “flip-flop de datos”
9) Circuito síncrono		( ) Elemento para almacenar un bit de información
10) Memoria		( ) flip-flop que alterna sus estados cuando sus entradas están ambas en 1.

TEMA 7. MEMORIAS

Objetivo particular

Al finalizar el tema, el alumno identificará los tipos básicos de memoria que componen una computadora digital, los procesos de lectura, escritura y actualización así como el funcionamiento de las memorias cache y memoria virtual.

Temario detallado

7.1 Tipos de memoria
- 7.1.1 RAM
- 7.1.2 ROM

7.2 Ciclos de memoria
- 7.2.1. Lectura
- 7.2.2. Escritura
- 7.2.3. Actualización

7.3 Mapa de memoria
- 7.3.1 Memoria expandida y extendida
- 7.3.2 Organización de memoria

7.4 Memoria cache

7.5. Memoria virtual

Introducción

La memoria es una de las unidades más importantes (después de la Unidad Central de Control -CPU-) en una computadora digital, debido a que en ella se almacenan las **instrucciones** (codificadas en binario) y los **datos** de un programa. La memoria es una de las unidades que más se ha desarrollado en la computadora digital, una de las razones radica en el hecho de que la CPU debe acceder y modificar los datos y programas almacenados en ella lo más rápido posible y por lo tanto la memoria no debe tener tiempos muertos.

Actualmente, debido a las nuevas tecnologías de fabricación de semiconductores, se está diseñando y construyendo nuevos dispositivos de memoria con mayor capacidad de almacenamiento, con una longitud de palabra mayor, tiempo de acceso para las operaciones de lectura y escritura más rápido, etc. tanto para las memorias RAM y memorias ROM.

7.1 Tipos de memoria

Los tipos de memoria que tiene una computadora digital básicamente son: memoria **RAM** y memoria **ROM**. La memoria **RAM** construida a base de arreglos de flip-flops (elementos biestables) y de una memoria **ROM** construida con circuitos electrónicos digitales, las cuales explicaremos a continuación.

7.1.1 Memorias RAM

Las memorias a las que se les puede cambiar el contenido de sus localidades con la función de "**Escritura**", lo mismo que obtener los contenidos de sus localidades con la función de "**Lectura**", se llaman **memorias de acceso aleatorio** o **RAM** (del inglés **Random Access Memory**).

Las memorias **RAM** se utilizan para almacenar datos y programas. Los datos pueden ser resultados parciales o finales. Los programas almacenados en discos, cintas u otros dispositivos, que en un momento dado se quiera ejecutar o procesar, tienen que pasar a la memoria **RAM** antes de su ejecución. Las memorias **RAM** se usan también durante la edición, ensamble y depuración de los programas.

Con el avance de la tecnología de los semiconductores se comenzó la fabricación de memorias **RAM** en circuitos integrados (C.I.) o "chips" de 256 bits con la técnica bipolar. Con el advenimiento de la tecnología NMOS se comenzó la fabricación de circuitos **RAM** de mayor capacidad, como lo son las memorias **RAM** de 1024 bits (1024 x 1). Actualmente las memorias **RAM** se fabrican con diversas tecnologías con lo cual se fabrican tarjetas de memoria

con capacidades de hasta un máximo de 4 Gbytes con un tiempo de acceso de "pocos nanosegundos". Las memorias fabricadas a base de semiconductores desplazaron a las memorias fabricadas a base de núcleos magnéticos con relación al costo y funcionamiento.

Características de las memorias RAM

En el diseño con memorias RAM se debe considerar algunas características, entre las más importantes se puede mencionar: **Tecnología**, **tipos, organización**, **velocidad** y **configuración**.

Tecnología en las memorias RAM

Las primeras memorias RAM fueron construidas con tecnologías **TTL** y **NNOS** y a la vez están siendo reemplazadas por memorias **RAM** con tecnología **I2L**, **VMOS** y **MNOS** que ofrecen mejor funcionamiento, mayor densidad de circuitos y menor costo. Algunas tecnologías tienen características más especiales que son deseables en ciertas aplicaciones. Por ejemplo, las memorias **RAM CMOS** requieren de poca energía y por lo tanto utilizan baterías o las memorias **RAM MNOS** que son "no volátiles", es decir, conservan los datos aún cuando se apague la fuente de alimentación. La mayoría de las memorias **RAM** son "volátiles".

Tipos de memorias RAM

Las memorias RAM se dividen por su diseño en tres tipos: **Estáticas, dinámicas** y **pseudo-estáticas**.

*** Memorias estáticas**

Las memorias estáticas están formadas por un flip-flop de dos transistores o multivibrador biestable. Al activar (direccionar) el flip-flop, éste se carga con el dato de entrada “0” o “1”. El dato cargado se conserva hasta que se activa de nuevo el flip-flop o se quita la alimentación, de ahí su nombre de memoria "estático".

* **Memorias dinámicas**

Las memorias dinámicas utilizan una estructura co-activa. El término "dinámicas" se refiere a que cambian de estado, la memoria dinámica puede conservar por muy pocos milisegundos (2 milisegundos la mayoría) la carga depositada en ellas. Para no perder la información, las memorias dinámicas deben recargar (refrescar o actualizar) el contenido de cada localidad.

Para no recargar una por una cada localidad de memoria **RAM**, se emplea el método de actualizar simultáneamente todas las localidades de memoria de una fila en el arreglo en un integrado **RAM** "siempre" que se efectúe la lectura de cualquiera de las localidades de la fila. Es decir, se aprovecha el hecho de leer una localidad de memoria para actualizar a todas las localidades de la fila a la que corresponde la localidad que se está leyendo.

La lógica más utilizada para refrescar memorias es un circuito externo que genera números de filas de manera secuencial por el Bus de Dirección y activa la señal **MEMR** cuando la CPU no está accediendo a las memorias. Esto para no interrumpir el funcionamiento normal de la CPU.

*** Memorias pseudo-estáticas**

Las memorias "pseudo-estáticas" o "cuasi-estáticas" combinan las ventajas de las memorias dinámicas y estáticas. Las memorias "pseudo-estáticas" son básicamente memorias dinámicas con circuitos adicionales para colocar periódicamente carga adicional en las localidades con nivel lógico 1.

Debido a que las localidades de memoria estáticas requieren de más componentes que las localidades de memoria dinámicas, las memorias dinámicas siempre están un paso adelante de las estáticas en cuanto a la densidad de las memorias. Actualmente en el mercado se encuentran memorias dinámicas de 4 Gbytes y estáticas de 512 Kbytes. Las memorias dinámicas son más baratas que las estáticas.

Organización de las memorias RAM

Las memorias **RAM** tienen líneas de dirección, líneas de datos, líneas para la alimentación de la energía, una o más líneas para habilitar el integrado ($\overline{CS}$ o $\overline{CE}$) y líneas de control para indicar la dirección del flujo de datos (Lectura o Escritura).

Una memoria **RAM** consta de dos bloques funcionales: **Arreglo de localidades de memoria** y los **circuitos internos de interfaz**, ver figuras 7.1a y 7.2b. El arreglo de localidades de la memoria es generalmente una matriz cuadrada de localidades de uno o más bits arregladas en filas y columnas, tal es la razón de que la capacidad de las **RAM** generalmente son potencias de 2 (1K, 2k, 4K, etc.). Los circuitos de la memoria toman una dirección y la dividen para seleccionar la fila y columna correspondiente a la localidad de la dirección.

Los circuitos internos, una vez seleccionado el integrado y la localidad de la memoria, determinan si se va a recibir un dato (escribir) o si se va a enviar (leer).

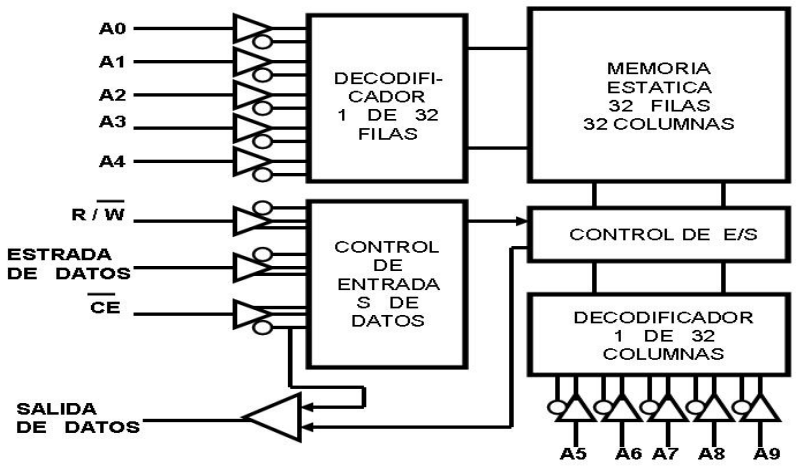


Figura 7.1a) Diagrama interno de la Memoria RAM 2102

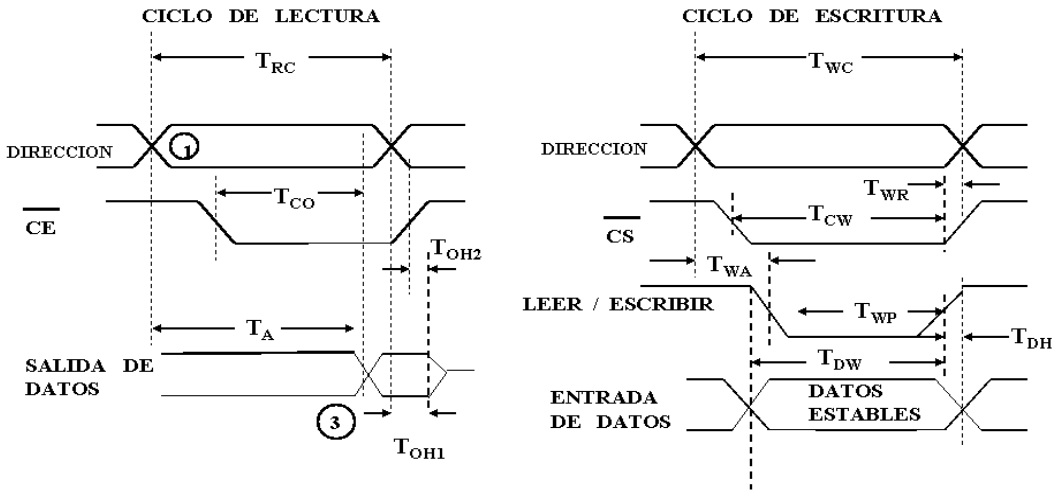


Figura 7.1b Diagramas de tiempo de la memoria RAM 2102

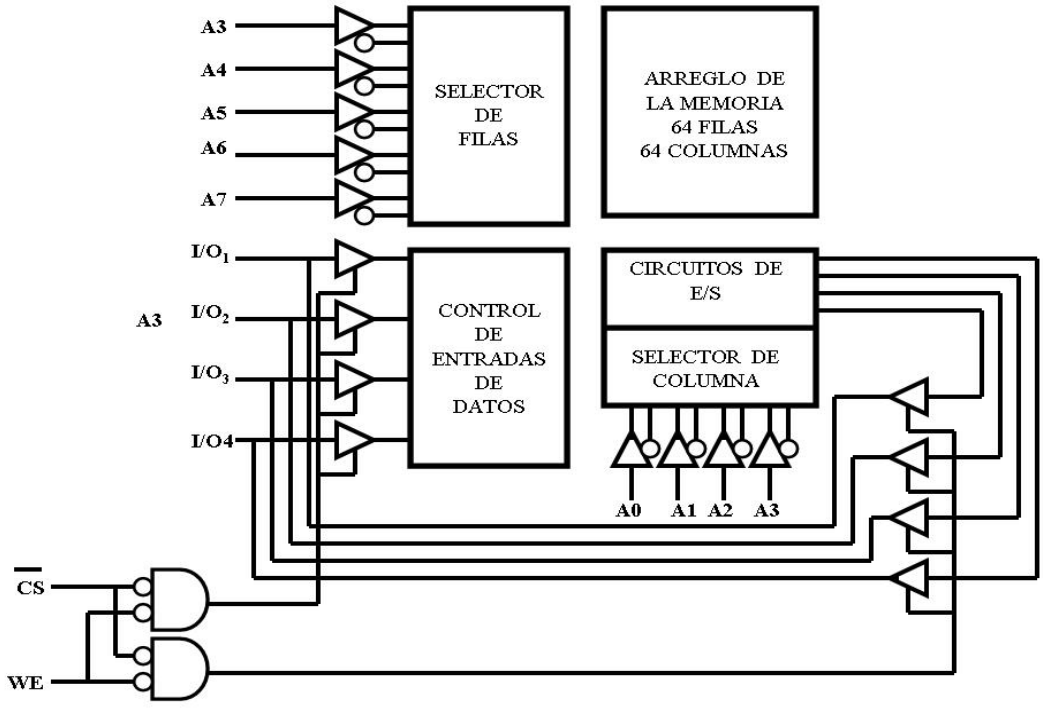


Figura 7.2a) Memoria RAM 2114 - Diagrama a Bloques -

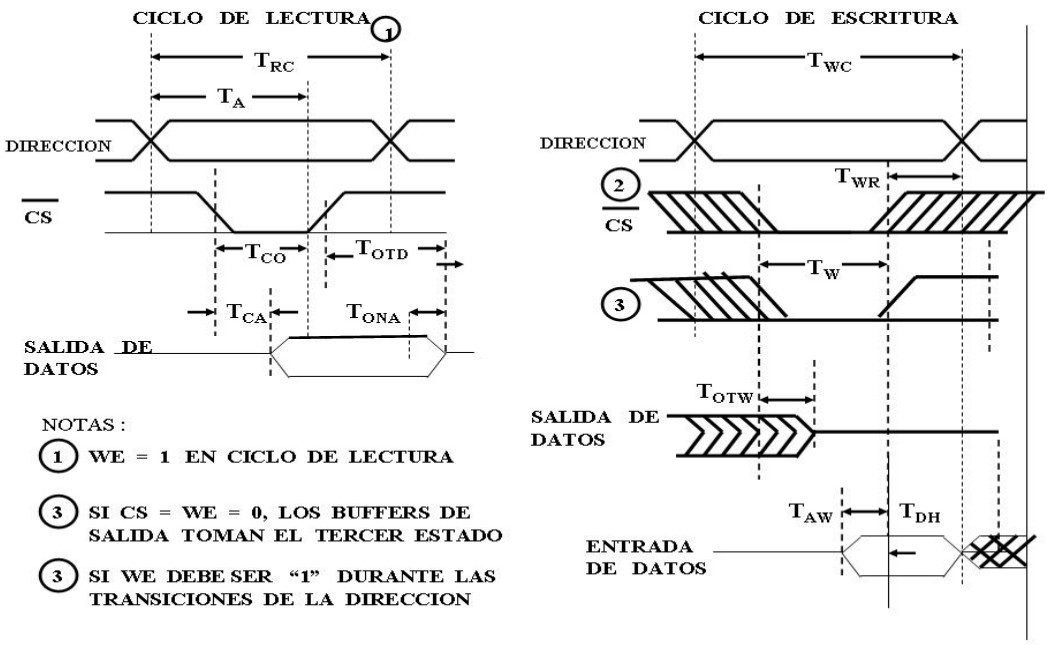


Figura 7.2b.) Diagramas de tiempo de la memoria RAM 2114

Velocidad de las memorias RAM

El **tiempo de acceso de una memoria** y el **tiempo de acceso del sistema** son dos conceptos que se deben considerar en el diseño de los módulos de memoria.

El tiempo de acceso de una memoria, **Ta**, es el tiempo que toma a la lógica de la memoria decodificar la dirección y estar lista para presentar en las salidas de datos el contenido de la localidad direccionada después de que recibe una dirección válida. El dato se presenta en las salidas de dato únicamente cuando la línea $\overline{CS}$ o $\overline{CE}$ se activa. El tiempo de acceso del sistema, **Tco,** es el tiempo que toma a la lógica de la memoria en presentar el dato de la localidad direccionada en las salidas del dato después de que se activa la entrada $\overline{CE}$ (en el C.I. 2102) o $\overline{CS}$ (en el C.I. 2114).

Toh2 (en el C.I. 2102) y **Toha** (en el C.I. 2114) es el tiempo durante el cual el dato de salida continúa válido después de que se desactiva la línea $\overline{CS}$ o $\overline{CE}$.

Cada microprocesador tiene un tiempo de acceso del sistema, el cual consiste

en el tiempo que sucede desde que el microprocesador envía una dirección válida y hasta que lee el Bus de datos. La 8085A envía la dirección válida al inicio del estado **T1** y lee el Bus de datos al inicio del estado **T3**.

Si el **tiempo de acceso de la memoria** usada con el microprocesador es menor que el **tiempo de acceso del sistema**, el microprocesador puede funcionar a su máxima velocidad. Pero si es mayor, se requiere de circuitos externos para sincronizar las memorias lentas con el microprocesador.

Configuración de las memorias RAM

Existen cuatro configuraciones de Entrada/Salida de datos en las memorias **RAM** estáticas:

1. E/S separadas sin líneas OD.
2. E/S comunes sin línea OD.
3. E/S separadas con línea para deshabilitar salidas (OD, Output Disable).
4. E/S comunes con la línea para deshabilitar salidas (OD).

1. Memorias con E/S separadas

Un ejemplo de memoria con entrada y salida separadas es el C.I. RAM 2102 de Intel, ver figura 7.1a. La figura 7.1a muestra la conexión de las líneas de entrada y salida de una memoria **RAM** 2102 con el Bus de Datos Bidireccional. Las memorias **RAM** deben operar como sigue: durante un ciclo de **Lectura** de memorias a las salidas de datos se deben conectar eléctricamente al Bus de Datos y durante el ciclo de **Escritura** en memoria las entradas de Datos se deben conectar eléctricamente al Bus de Datos. Las memorias con líneas de entrada y salida separadas tienen el problema de que las operaciones de escribir las salidas de Datos se activan, lo que puede alterar el dato de entrada. Por ejemplo, si una localidad tiene nivel 0 y se está escribiendo un nivel 1, el nivel 0 de salida puede alterar el nivel de entrada.

Para lograr desconectar eléctricamente las salidas durante operaciones de **Escritura** se usan buffers de tres estados en las salidas de las **RAM**, ver figura

7.3b. La figura 7.4 muestra la conexión de dos circuitos buffer 74LS367 a las salidas de 8 memorias **RAM** 2102. La corriente de entrada requerida por el 74LS367 en el nivel 0 es de 0.3 mA y 20 microA en el nivel 1. Las salidas del 74LS367 puede proporcionar 2 mA en el nivel 1 y 12 mA en el nivel 0. Esto es más que suficiente para las necesidades del **Bus de Datos**.

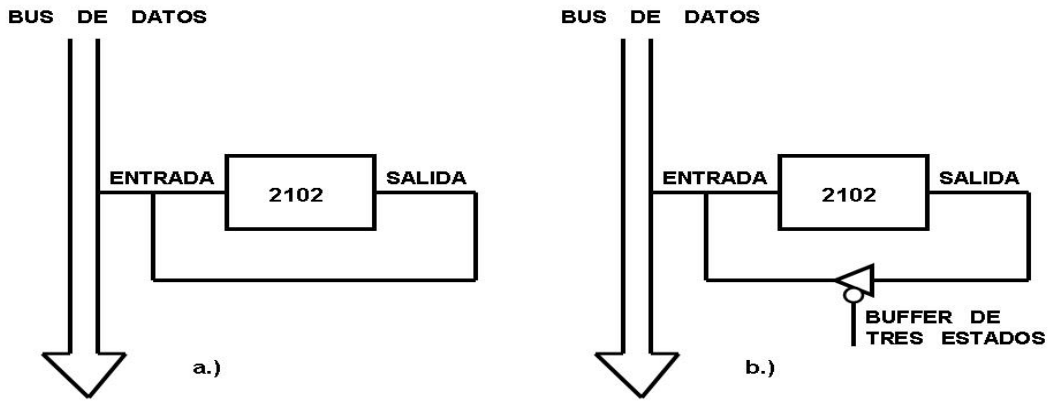


Figura 7.3 Conexión de un 2102 al Bus de Datos

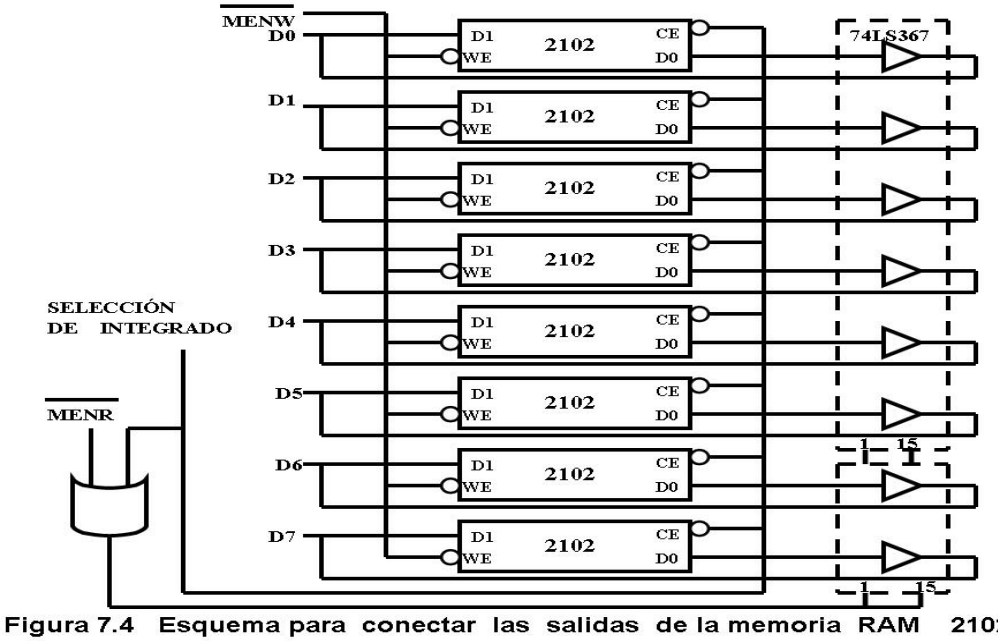


Figura 7.4 Esquema para conectar las salidas de la memoria RAM 2102

El C.I. 74LS367 se deshabilita (pasa al tercer estado) cuando las terminales 21 y 15 están en nivel 1. Cuando la computadora digital va a escribir en la memoria **RAM** 2102, las terminales 1 y 15 del 74LS367 deben tener un nivel 1 para deshabilitar las salidas. Es tarea de los circuitos externos asegurar que estas terminales tengan nivel 1 cuando el microprocesador va a escribir en la RAM 2102.

Las líneas de selección de módulo y la línea de control $\overline{\boldsymbol{MEMR}}$ se combinan para activar al buffer 74LS367. Durante una operación de Lectura (Leer de Memoria) la entrada 3 ($\boldsymbol{R/\overline{W}}$) debe tener nivel 1, al conectar esta entrada a la señal $\overline{\boldsymbol{MEMW}}$ se cumple este requisito. Cuando la 2102 ha sido seleccionada y se activa la línea $\overline{\boldsymbol{MEMR}}$, el 74LS367 se activa y permite que las salidas del C.I. 2102 se presentan en el Bus de Datos.

Durante una operación de **Escritura**, el buffer 74LS367 no se activa dejando desconectadas las salidas de los C.I. 2102.

2. Memorias con E/S comunes

Un ejemplo de memoria de entrada y salida comunes es la RAM 2114, ver figura 7.2a. La figura muestra la conexión de las líneas entrada E/S del C.I. 2114 con el **Bus de Datos Bidireccional**. Las memorias **RAM** con E/S comunes utilizan las mismas terminales para recibir y enviar datos. No existen problemas en el integrado ya que cuenta con la entrada $\overline{\boldsymbol{WE}}$ para indicar una operación de leer ($\overline{\boldsymbol{WE}}$ = 1) o una de escribir ($\overline{\boldsymbol{WE}}$ = 0) y no se puede ordenar las dos funciones al mismo tiempo. Durante una operación de escribir las líneas E/S representan las entradas y los buffers de las líneas de salida se deshabilitan.

3. Memorias con E/S separadas y con línea OD

Este tipo de memorias son semejantes a las memorias con E/S separadas, pero con una línea adicional que permite el control de las líneas de Salida de Datos. Esta línea se conoce como deshabilitar salidas (**OD, Disable Output**). Mientras la línea **OD** no esté activa, las líneas de Salida de Datos se

encuentran en el tercer estado.

La memoria **RAM** 2101 de Intel es de este tipo de memorias (figura 7.1a). Cuando la memoria está habilitada (**CE = 0** y **CE = 1**) y la línea OD tengan nivel 0, se activan las salidas de datos. Durante las operaciones escritura (escribir) la línea OD debe tomar nivel 1 para poner las líneas de Salida de Datos en el tercer estado.

4. Memorias con E/S comunes y con línea OD

Este tipo de memorias es semejante a las memorias con E/S comunes, pero con la línea adicional OD para el control de las líneas de salida de datos interna. La memoria **RAM** 2114 de Intel es de este tipo de memorias, (figura 7.2b).

7.1.2 ROM

Las memorias a las que se les puede realizar la función de leer los contenidos pero no la función de escribir se conocen como "memorias sólo para leer" o **ROM** (del inglés **Read Only Memory**). Los datos se almacenan durante la fabricación de la memoria y estas memorias son no **volátiles**.

Las memorias **ROM** se utilizan para almacenar en forma "permanente" datos y programas. Los programas muy importantes tales como el programa monitor y/o los programas de control se almacenan en memoria **ROM**. Al encender la microcomputadora generalmente se genera una señal de "**RESET**", la cual causa que el **Contador del Programa** (**PC**) tome la dirección cero, y a partir de esta dirección se inicie el proceso de la CPU. Tal es la razón de que algunas localidades de memoria comienzan con la dirección cero sean del tipo **ROM**, las cuales están cargadas con programas que le permiten al usuario tomar el control de la CPU.

Existen algunas variaciones de las memorias **ROM** que permiten más versatilidad a los microcomputadores, tales como el **PROM** (**PROM Programable**), **EPROM** (**ROM Programable y Borrable**) y **EEROM** (**ROM**

Electrical Erase). Las memorias **PROM** son semiconductores que contienen pequeños fusibles que controlan el nivel lógico de los bits de las localidades, un fusible por bit. Un fusible completo genera un nivel 1 y un fusible quemado genera un nivel 0. La memoria **PROM** se fabrica con todos los fusibles completos, toda la memoria tiene nivel 1. Para programar las memorias **PROM**, el usuario debe quemar (generalmente con pulsos de alto voltaje y alta corriente, 20 a 30 volts y 20 a 50 mA) uno por uno los fusibles de los bits que deben tener nivel 0. Desafortunadamente, los bits de los **PROM** se pueden programar sólo una vez. Los circuitos internos adicionales en las memorias **PROM** causa que estas memorias ocupen más espacio que las **ROM**, son menos densos, es decir, menos bits por cm^3. La capacidad de almacenamiento de los **PROM** está cerca del 50% de los **ROM**. Las memorias **PROM** tienen la ventaja de que son programables y no se requiere la ayuda del fabricante.

Las memorias **EPROM** son más populares que las **ROM** y **PROM**. Las **EPROM** en lugar de fusibles almacenan cargas en las celdas cuando se les aplica pulsos de alto voltaje. Estas cargas permanecen atrapadas, representando un nivel 0, hasta que no se le aplica una energía externa, tal como las luz ultravioleta. Al eliminarse la carga, la celda representa un nivel 1. Las **EPROM** tienen una pequeña ventana transparente por lo que se puede iluminar directamente el circuito integrado interno con la luz ultravioleta. Los diferentes **EPROM** requieren de diferentes intensidades de luz ultravioleta.

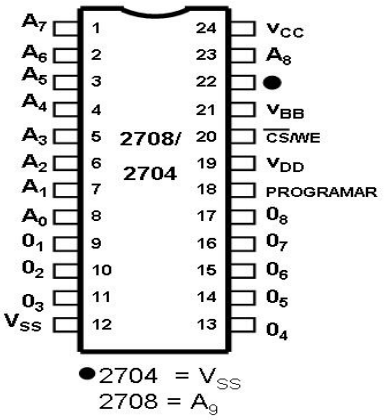
En las especificaciones del fabricante se indican las características para programar y borrar las memorias **EPROM**. Una buena costumbre es la de tapar las ventanas para prevenir las exposiciones accidentales de luz ultravioleta que pudieran alterar el contenido de las celdas. Las memorias **EPROM** más populares se muestran en la Tabla 7.1

Nombre	Configuración	Alimentación	Tiempo de Acceso
1720A	256 x 8	+5, -9	650 -1700 nseg.
2708	1K x 8	+5, +12, -5	450 nseg.
2716	2K x 8	+5	450 nseg.
2732	4K x 8	+5	450 nseg.

Tabla 7.1 Características de memorias EPROM

Los **EEPROM** son semejantes a los **EPROM**, pero en lugar de requerir luz ultravioleta para borrar el contenido de las localidades de memoria requieren de un voltaje aplicado en una de sus “pins”.

Los C.I. 2708 y 2704 son dos **EPROM** de 8192 bits (1024 x 8) y de 4096 bits (512 x 8) respectivamente. Los dos están fabricados con técnica **MOS** con canal **N** de silicio en integrados de 24 “pins”, ver 7.5a. Estos integrados tienen una ventana transparente por los que se accede directamente el circuito interno.



NOMBRE DE LOS TERMINALES

A_0 - A_9	ENTRADAS DE DIRECCION
O_1 - O_8	SALIDAS DE DATOS
$\overline{CS}$ - WE	SELEC. INTEGRADO/HABILITAR ESCRIBIR

Figura 7.5a) Descripción de terminales y tiempos del EPROM 2708/2704

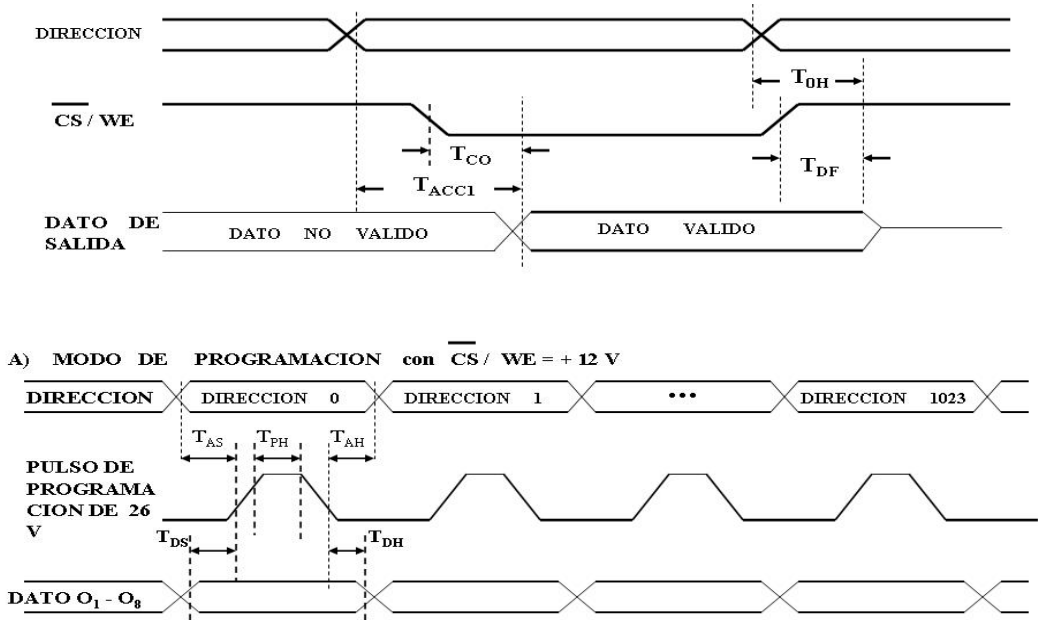


Figura 7.5b Configuración y diagramas de tiempo de la memoria EPROM 2708 / 2704

Proceso de BORRAR

De fábrica y/o después de borrar todos los bits del C.I. 2708/2704, su contenido es “1” lógico. La programación consiste en cargar ceros en los bits necesarios. Los pasos requeridos son:

1. La entrada **CS/WE = +12 V**
2. Presentar la dirección en las entradas **A_0 - A_9**
3. Presentar el dato a cargar en las entradas **01-08**
4. Presentar un pulso de **+26V** en la entrada **PROGRAMAR**.

Estos pasos se deben repetir para cada dirección. El hecho de realizar de manera secuencial estos pasos desde la dirección 0 a la 1023 se conoce como "**Lazo de programación**" (**Program Loop**). El número de Lazos de programación que se requiere está en función del ancho del pulso. La fórmula para determinar el número de lazos es:

N x Tpw x 100 milisegundos

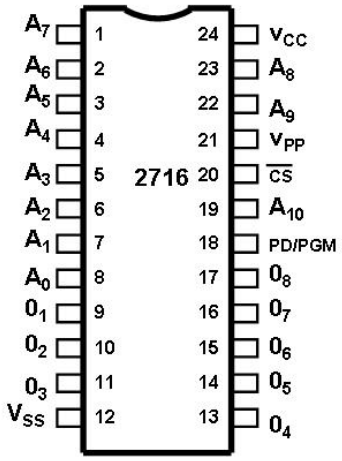
donde

N es el número de lazo de programación requeridos, y
Tpw es el ancho del pulso en milisegundos.

El ancho del pulso puede variar de 0.1 a 1 milisegundos. Por tanto, **N** puede variar de 100 a 1000, dependiendo del **Tpw** usado. Independientemente del **Tpw** usado, siempre se debe ejecutar un lazo aplicando sólo un pulso a cada localidad. No se puede aplicar **N** pulsos a una sola dirección y después a la siguiente dirección. Se deben programar todas las direcciones en cada sesión de programación y en total cada dirección debe recibir **N** pulsos y la suma de los pulsos debe ser igual o mayor de 100 milisegundos.

EPROM 2716

La 2716 es una memoria **EPROM** de 16384 bits (2048x8). La 2716 es una extensión del 2704 y 2708. El “pin” 19 se utiliza para la línea **A10** en lugar de **Vdd** de +12V en el 2704/2708. La 2716 requiere únicamente de una fuente de alimentación de +5V, (ver figura 7.6). El “pin” 18 tiene el nombre de **PD** de **PGM**. Este “pin” tiene la doble función del control de "**baja alimentación**" (Power Down) y la entrada del pulso de programación. Para el proceso de borrar tiene las mismas características que el 2708/2704.



MODOS DE SELECCION

MODO \ PIN	PD/PGM	CS	V_{PP}	V_{CC}	SALIDAS
LEER	V_{IL}	V_{IL}	+5	+5	SALIDAS
DESHABILITAR	NADA	V_{IH}	+5	+5	3er. Estado
SIN ALIMEN-TACION	V_{IH}	NADA	+5	+5	3er. Estado
PROGRAMAR	Pulso de V_{IL} o V_{IH}	V_{IH}	+25	+5	ENTRADAS
VERIFICAR PROGRAMA	V_{IL}	V_{IL}	+25	+5	SALIDAS
INHIBIR PROGRAMA	V_{IL}	V_{IH}	+25	+5	3er. Estado

NOMBRE DE LOS TERMINALES

A_0 - A_{10}	ENTRADAS DE DIRECCION
O_1 - O_8	SALIDAS DE DATOS
$\overline{CS}$	SELEC. INTEGRADO/HABILITAR ESCRIBIR
PD / PGM	SIN ALIMENTACIÓN /PROGRAMAR

Figura 7.6 Descripción de terminales del EPROM 2716

La 2716 tiene seis modos de operación. Todas las entradas son nivel TTL. En la entrada **Vpp** se debe proporcionar una alimentación de +25V durante los tres modos de programación y de +5V durante los otros tres modos, los cuales explicaremos a continuación.

MODO DE LECTURA

El dato de la localidad direccionada se presenta en las salidas **O1-O8** en el modo de lectura. El tiempo de acceso es de 450 nseg. Después que la dirección se hace estable cuando **CS**=0, o de 120 nseg. (Tco) después que **CS** = 0 cuando la dirección está estable.

MODO DESHABILITADO

Cuando la entrada **CS**=1, las salidas **O1-O8** están deshabilitadas y cuando **CS**=0 las salidas toman los valores del contenido de la localidad direccionada.

MODO DE BAJA ALIMENTACIÓN

En este modo el 2716 reduce la disipación de energía en 75%, de 525 mW a 132 mW. Este modo se logra aplicando nivel 1 en la entrada **PD/PGM**. En este modo las salidas pasan al tercer estado.

PROGRAMACIÓN

De fábrica y después de borrar todos los bits de la 2716 contienen nivel “1”. La programación consiste en cargar ceros en los bits necesarios. Los pasos requeridos para la programación son (ver figura 7.6):

1. Alimentar +25V a la entrada **Vpp**.
2. **CS = 1**
3. Proporcionar la dirección en las líneas **A0-A10**
4. Proporcionar el dato a cargar en las líneas **O0-O7**
5. Cuando la dirección y los datos están estables, aplicar un pulso con nivel 1 (TTL) con duración de 50 milisegundos en la entrada **PD/PGM**.

Se debe aplicar un pulso a cada localidad por programar. Se puede programar cualquier localidad en cualquier momento, sea individualmente, secuencial o aleatorio. Debido a su facilidad de programación, se pueden programar varias 2716 en paralelo con los mismos datos.

INHIBICIÓN DE PROGRAMACIÓN

Durante la programación múltiple de circuitos 2716 también se puede programar con diferentes datos. Con excepción de las entradas **PD/PGM**, todas las entradas iguales (**O0-O7**) incluyendo **CS** se amarran en paralelo. Aplicar nivel 1 a las entradas **PD/PGM** de los 2716 que desean estar "**Inhibidos de Programación**". Cuando **PD/PGM=0** las líneas **O0-O7** toman el tercer estado.

VERIFICACIÓN DE PROGRAMACIÓN

Se recomienda efectuar una verificación de los bits programados para

determinar que fueron programados correctamente. En el proceso de verificación **Vpp = 25V** y **CS=0**. Al presentar una dirección, el contenido de la localidad se presenta en las líneas **O0-O7**.

Arquitectura de memorias ROM

La arquitectura con memorias **ROM** es semejante a la arquitectura con memorias

RAM. Las memorias **ROM** se pueden seleccionar con direccionamiento "**absoluto**" y "**no-absoluto**". Lo que se mencionará para las memorias **ROM** se aplica a las **PROM**, **EPROM** y **EEROM**.

DIRECCIONAMIENTO NO-ABSOLUTO EN MEMORIAS ROM

Las figuras 7.7a y 7.7b ilustran el uso de decodificadores para seleccionar las memorias ROM de 1K conectando las salidas de los decodificadores con las entradas **CS** de las **ROM**. Las salidas del 8205 permiten seleccionar hasta ocho memorias de 1K (2708) y las salidas del 74LS42 permiten seleccionar hasta 16 memorias de 1K. El rango de direcciones de la figura 7.7a es de 0 a 1FFFH y el de la figura 7.7b es de 0 a 3FFFH. Los dos decodificadores se habilitan únicamente en ciclos de Lectura de Memoria, **MEMR=0**

Los decodificadores de la figura 7.8 permiten seleccionar 8 (el 8205) y 16 (el 74LS42) memorias **ROM** de 2K (2716). El decodificador de la figura 7.9 permite seleccionar 8 memorias **ROM** de 4K (2732). Todos estos decodificadores proporcionan direccionamiento "**no-absoluto**".

Figura 7.7 Direccionamiento no absoluto con decodificador para a.) 8K y b.) 16K con EPROM's 2708

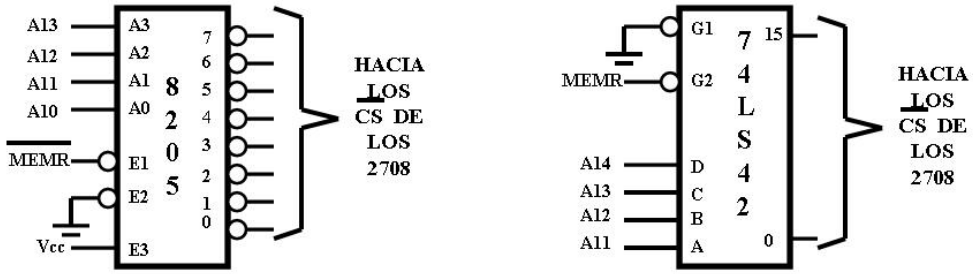


Figura 7.8 Direccionamiento no absoluto con decodificador para a.) 16 K y b.) 32 K con EPROM's 2716

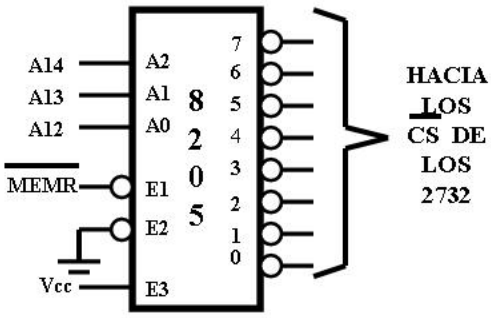


Figura 7.9 Direccionamiento no absoluto con decodificador para 32K con EPROM's 2732.

DIRECCIONAMIENTO ABSOLUTO EN MEMORIAS ROM

En el direccionamiento "absoluto" se debe usar las 16 líneas de dirección. Generalmente las memorias de las microcomputadoras utilizan memorias **RAM** y **ROM** (**PROM o EPROM**), y las **ROM** tienen las direcciones más bajas, comenzando con 0. Esto debido a que al encender el sistema de energía de la

microcomputadora o activar la señal **RESET** el contador del programa se carga con 0, iniciando, a partir de esta dirección, el procesamiento. Generalmente a partir de esta dirección se encuentra el programa Monitor o un programa que tome el control del CPU.

El circuito de la figura 7.10 muestra un arreglo para seleccionar 64 módulos de memoria de 1K bytes. El integrado 74LS154 #0 tiene la característica de que la entrada **G2** se puede conectar directamente por medio de un puente a la línea $\overline{MEMR}$; para que las memorias conectadas a las salidas del 74LS154 se debe conectarlas a las entradas $\overline{CS}$ de las memorias **ROM**. La memoria **ROM** conectada a la salida 0 del 74LS154 #0 tendrá un rango de direcciones de 0000H a 03FFH, y la conectada a la salida 15 tiene el rango de direcciones de 3C00H a 3FFFH.

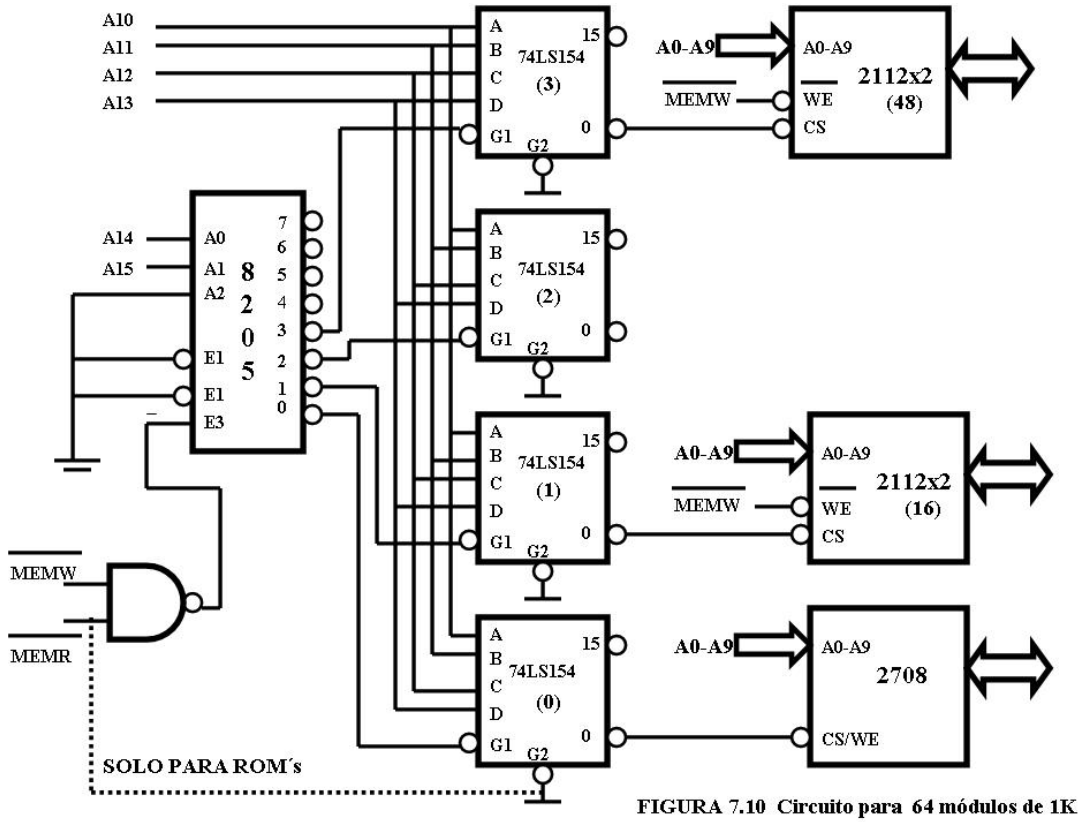


FIGURA 7.10 Circuito para 64 módulos de 1K

Los circuitos de las figuras 7.11a y b también se pueden utilizar para seleccionar memorias **ROM**, notando que se debe conectar una de las entradas de habilitar (**G1** o **G2**) de cada 74LS154 a una de las salidas de los decodificadores y la otra a la línea $\overline{MEMR}$, para que los 74LS154 se habiliten en ciclos de leer memoria.

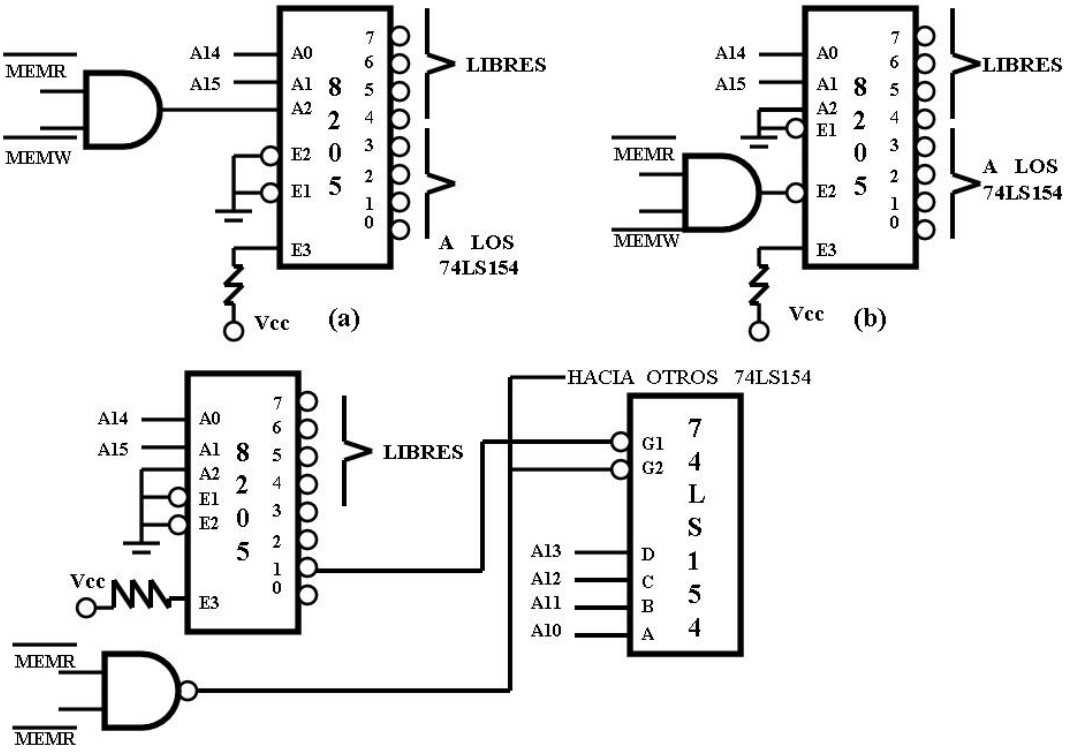


Figura 7.11 Variantes de los decodificadores de la figura 7.10

Con las memorias **ROM** también se puede usar las técnicas de Selector de Bloque y Banco de Memoria.

7.2 Ciclos de memoria

Para asegurar la operación correcta de una memoria existen restricciones de tiempo en la secuencia que deben seguir las direcciones, datos y señales de control. Estas restricciones están marcadas en las hojas de especificaciones del fabricante, como parte de las características de funcionamiento y en los diagramas de tiempos.

El tipo de memoria más simple, en términos de su operación, es la memoria **ROM**. Los pasos de operación básica de una memoria **ROM** son:

1 Se aplica una dirección a las entradas de dirección de la memoria **ROM**.

2. Se selecciona el circuito de la **ROM** activando sus entradas de selección de circuito (**chip select CS** o **chip enable CE**).

3. El contenido de la localidad de memoria seleccionada aparece en las salidas de datos de la **ROM**, después de un periodo igual a su tiempo de acceso.

7.2.1 Lectura

Supón que un circuito de memoria es seleccionado con el nivel lógico apropiado en su entrada de selección (**CS**). El tiempo transcurrido desde la subsecuente aplicación de una dirección en sus entradas de dirección hasta la aparición, a la salida de la memoria, de una copia estable del dato seleccionado, es el tiempo de acceso, $\mathbf{t_A}$.

Si existe un valor estable en las entradas de dirección de una memoria **ROM** y la entrada de selección (**Chip Select**) se activa para seleccionar la memoria ROM, el retraso entre la activación de las señales de selección (**Chip Select**) apropiada y la estabilización del dato a la salida es $\mathbf{t_{CO}}$ o tiempo de acceso de **CS** (Chip Select). Los parámetros, $\mathbf{t_A}$ y $\mathbf{t_{CO}}$, se muestran en el diagrama de tiempos de la figura 7.12. En dicho diagrama se muestra la aplicación de las señales de dirección y de selección (**Chip Select**) al tiempo correcto para que

sus efectos a la salida ocurran simultáneamente. El punto de referencia es la aparición de datos válidos a la salida. Como se ve en el diagrama, $\mathbf{t_{CO}}$ es generalmente menor que $\mathbf{t_A}$. Esto se debe a que la lógica de selección (**Chip Select**) está conectada directamente a los buffers de salida.

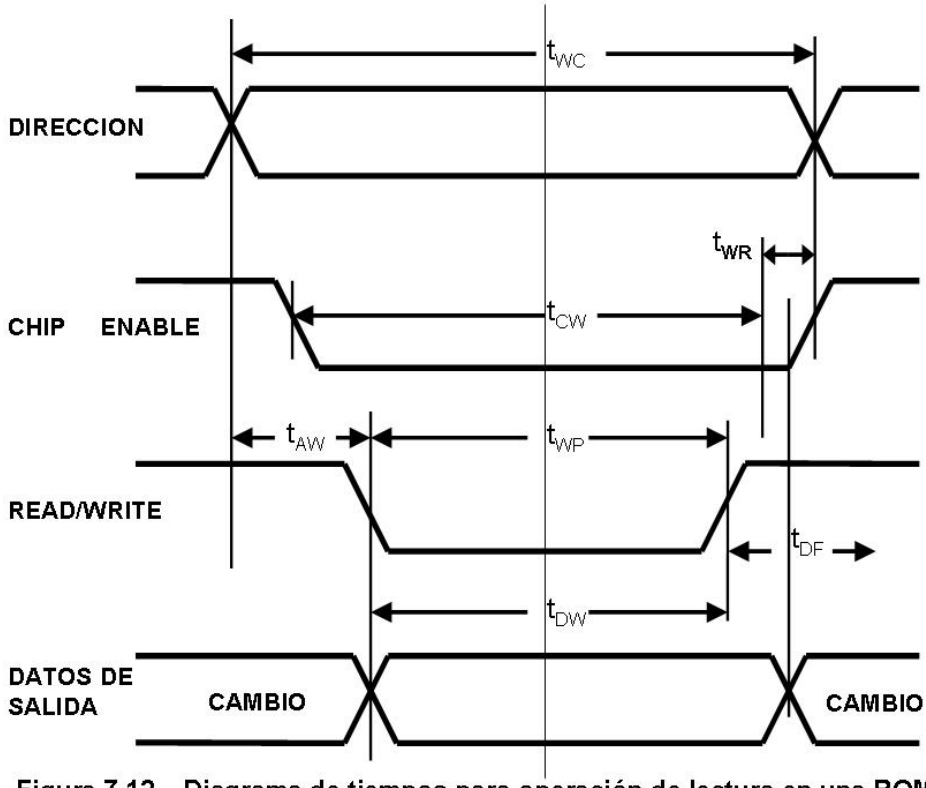


Figura 7.12 Diagrama de tiempos para operación de lectura en una ROM

Hay otros tres parámetros en el diagrama de tiempos, dos de ellos son el tiempo de sostenimiento de la salida (**output hold time**), $\mathbf{t_{OH}}$, y el tiempo que transcurre desde la desactivación de la línea de selección (**chip select**) hasta que las salidas flotan - tercer estado - (**chip deselect to output float time**) $\mathbf{T_{DF}}$. Estos parámetros están referidos al instante en que el dato de salida válido pasa a ser un dato inválido o las líneas de datos se ponen a flotar. El $\mathbf{t_{OH}}$ indica cuánto tiempo es todavía válido el dato de salida después de que la dirección ha cambiado. El $\mathbf{t_{DF}}$ indica el tiempo que permanece válido el dato de salida cuando se ha dejado de seleccionar el circuito de memoria. El tercer parámetro, tiempo del ciclo de lectura (**Read Cycle Time**), $\mathbf{t_{RC}}$, especifica la

velocidad máxima a la cual diferentes localidades de memoria pueden ser leídas sucesivamente.

7.2.2 Escritura

El ciclo de escritura se entiende mejor con una memoria **RAM** estática o dinámica. Las memorias **RAM** estáticas tienen los requisitos de operación más sencillos. Además de las líneas de dirección, habilitación de chip (**chip enable**) y datos de salida necesarios en las memorias **ROM**, las memorias **RAM** requieren líneas de datos de entrada y una línea de control que determina si la operación es de lectura o de escritura. La línea de control **R/W** se mantiene en 1 lógico para una operación de lectura y en 0 de escritura. Las entradas de dirección, habilitación de chip (**chip enable**), inutilización (o des-habilitación) de salida (output disable) y **R/W**, deben de seguir un orden apropiado para la operación correcta de la memoria. La secuencia de operaciones y los requerimientos de tiempo para leer una memoria **RAM** estática son similares a los de una **ROM**.

La secuencia básica de operaciones para escribir en una memoria **RAM** estática es la siguiente:

1. Se aplica una dirección a las entradas de dirección de la memoria **RAM**.

2. Se aplica el circuito de la **RAM** activando sus entradas de habilitación de chip (**chip enable**).

3. El dato que va a ser escrito (almacenado) en la memoria se aplica en las entradas de datos.

4. Se envía un pulso negativo, de 1 a 0 lógico, por la línea **R/W.**

5. Las señales de dirección y de selección de chip (**chip enable**) pueden cambiarse para la lectura o escritura de otra localidad de memoria.

La figura 7.13 muestra el diagrama de tiempos para la operación de escritura una memoria **RAM**. El pulso de escritura (del idioma inglés, ***write pulse***), generalmente un 0 lógico debe ocurrir en la entrada **R/W** por un periodo mínimo $\mathbf{t_{WP}}$; como se indica en la realización de la operación de escritura es una referencia conveniente para especificar otros parámetros asociados con ellas. Empieza cuando la línea **R/W** sufre una transición de 1 a 0 lógico. Sin embargo, en la mayoría de las memorias, los niveles lógicos en las líneas de datos no son importantes si no hasta que la línea **R/W** cambia de nuevo de 0 a 1 lógica, porque casi todas ellas no aceptan el dato de entrada sino hasta esta transición. Esta es una situación similar a la que existe en un flip-flop disparado en la transición positiva.

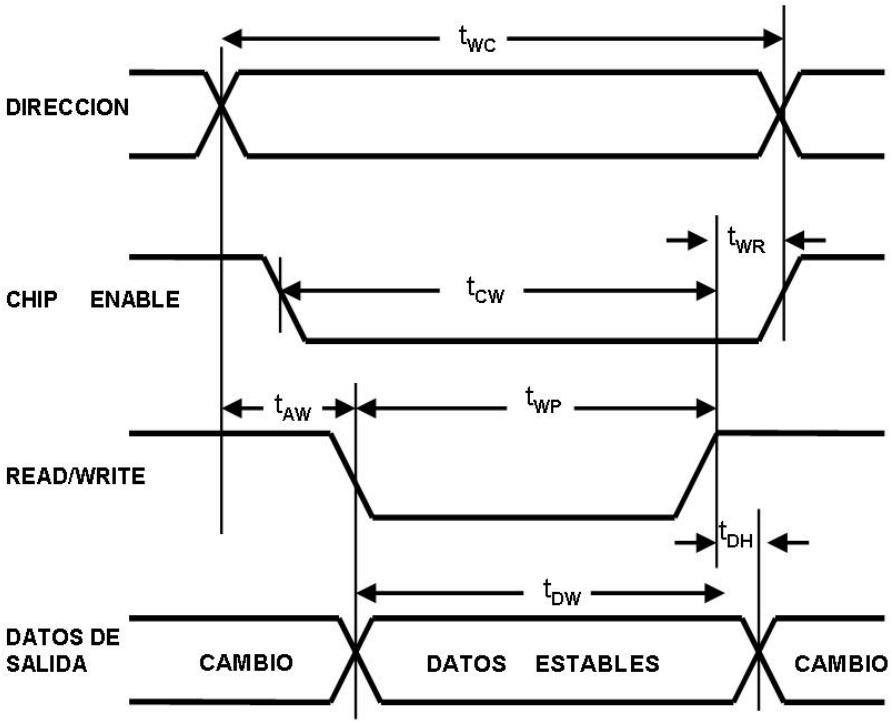


Figura 7.13 Diagrama de tiempos para operación de escritura en una RAM

Para almacenar información de datos que sea válida éstos deben permanecer estables durante un intervalo previo a la transición de 0 a 1 de la línea **R/W**; a este intervalo se le llama tiempo de preparación del dato (del idioma inglés, data set up time), $\mathbf{t_{DW}}$. También es necesario mantener el dato estable durante un periodo de tiempo después de la transición de 0 a 1 de la línea **R/W**; a éste

se le conoce como el tiempo de sostenimiento del dato (dato hold time), t_{DH}.

Siempre que las entradas de dirección cambian, transcurre cierto tiempo antes de que las salidas de los decodificadores de dirección se hayan estabilizado en su valor final. Durante estos transitorios, otras localidades de memoria son direccionadas involuntariamente, si el pulso de escritura se aplica antes de localidades de memoria, además de aquellas a la que está destinado. Para eliminar esto, las líneas de dirección deben ser estables durante un periodo de tiempo que anteceda y siga a la ocurrencia del pulso de escritura. El tiempo de retraso de escritura (**write delay**), $\mathbf{t_{AW}}$, indica un dato tiempo antes de la transición de 1 a 0 del pulso de escritura, la dirección debe ser estable; y el tiempo de recuperación de escritura (**write recovery time**), $\mathbf{t_{WR}}$, indica cuánto tiempo debe de mantenerse estable la dirección después de la transición de 0 a 1 de **R/W**.

Análogamente a las entradas de dirección, las de habilitación de chip (**chip enable**) deben ser estables por un periodo conocido como el tiempo de habilitación de chip (**chip enable**) a escritura (**chip enable to write time**), $\mathbf{t_{CW}}$, especifica el tiempo mínimo entre operaciones de escritura, en la memoria, las estáticas, los ciclos de lectura y escritura duran lo mismo.

Explicaremos con mayor profundidad los ciclos de lectura y escritura utilizando la memoria **RAM** (C.I. 2114).

Lectura de datos

Para leer datos de las memorias **RAM** se debe cumplir lo siguiente:

1. Una dirección debe estar presente en el Bus de Dirección.
2. El integrado debe estar seleccionado (**CS** =0 en el 2114 o **CE** = 0 en el 2102).
3. La señal de control de leer debe tener el nivel adecuado (**R/W**=1 en el 2102 o **WE**=1 en el 2114)

Un microprocesador, como el 8085A, envía una dirección en el estado **T1** en un ciclo de leer memoria, de las cuales, algunas se conectan al integrado (por ejemplo, el 2102 y el 2114, A0 – A9). Las líneas que sobran (**A10-A15**, para el 2102 ó 2114) se deben utilizar para habilitar los integrados, usando direccionamiento "absoluto" o "no-absoluto".

Una vez direccionada la localidad y habilitado el integrado, la memoria está lista para enviar o recibir un dato.

Durante un ciclo de Lectura de Memoria, las señales de control **MEMR=0** y **MEMW=1**. Conectando la señal **MEMW** la entrada **CE** o **CS** pueden satisfacer la tercera condición. Una vez cumplidas las tres condiciones y después que ha pasado el Tiempo de Acceso (**TA**), el contenido de la localidad direccionada se presenta en el Bus de Datos, de donde debe ser tomado por la 8085A. Esta función es similar a la lectura de una memoria **ROM**.

Escribir datos

Para escribir datos en las memorias **RAM** se debe cumplir lo siguiente:

1. Una dirección debe estar presente en el Bus de Dirección.
2. El integrado debe estar seleccionado.
3. Un dato de 8 bits se debe enviar por el Bus de Datos.
4. La señal de control de escritura/escribir debe tener el nivel adecuado **R/W=0** (en el 2102) o **WE=0** (en el 2114).
5. La transición bajo-alto de la señal de control de escritura/escribir carga en la localidad direccionada el dato presente en el Bus de Datos. Esto sucede cuando la memoria está fabricada con Flip-Flops. Si está fabricada con Latches con el nivel **0** es suficiente para cargar el dato.

Durante un ciclo de escritura en memoria, la 8085 envía una dirección en el estado **T1** (se cumplen las condiciones 1 y 2) y un dato en el estado **T2** (se cumple la condición 3). Durante este mismo ciclo; **MEMR** = 1 y **MEMW** = 0 de tal manera que conectando la señal **MEMW** a la entrada **CE** o **CS** se puede

cumplir la cuarta condición.

Conectando la señal **MEMW** con las entradas **CE** o **CS** se pueden cumplir las condiciones de estas entradas tanto en el ciclo de lectura como en el ciclo de escritura en memoria.

Considerar que se conecta la señal **MEMW** a la entrada **WE** de los dos circuitos 2114 y que las líneas **CS** se activan cuando las líneas de dirección A15-A10 tienen el valor de 01100 y **MEMR** = 0 o **MEMW** = 0.

De tal manera, que las direcciones del módulo están en el rango de 0110 0000 0000 0000 (6000H) a 0110 0011 1111 1111 (63FFH). Al efectuar una lectura a la localidad 6000H se tendría lo siguiente:

Dirección

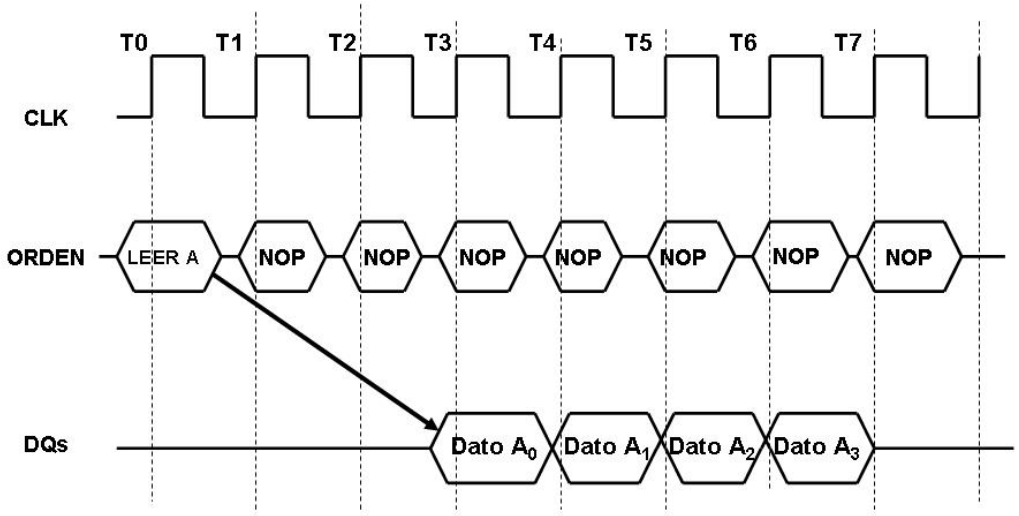
Byte Alto	Byte Bajo	MEMR	MEMW	I/OR	I/ OW
01100000	10100000	0	1	1	1

7.2.3 Actualización

Las memorias **RAM** se pueden clasificar en memorias estáticas y dinámicas. Las memorias **RAM** dinámicas tienen la propiedad de que los datos almacenados decaen o se desvanecen espontáneamente y deben ser actualizados a intervalos regulares debido a que están fabricadas con la tecnología CMOS. Además los datos se almacenan como cargas eléctricas en condensadores. Ya que los condensadores tienen una tendencia natural a descargarse. Las **RAM** dinámicas requieren actualizaciones periódicas para mantener memorizados los datos. El término dinámico hace referencia a esta tendencia a que la carga almacenada se pierda, incluso manteniéndola siempre alimentada.

Una de las memorias **RAM** dinámicas más utilizada en una Computadora Digital es la memoria **RAM** Dinámica Síncrona (SDRAM), la cual intercambia datos con el microprocesador de forma sincronizada con una señal de reloj externa, funcionando a la velocidad tope del bus del microprocesador/memoria, sin importar estados de espera.

La figura 7.14 muestra un ejemplo muestra el funcionamiento de una SDRAM. En este caso, la longitud de ráfaga vale 4 y la latencia es 2. La orden de lectura en ráfaga se inicia teniendo **CS** y **CAS** en bajo mientras se mantienen **RAS** y **WE** en alto al llegar el flanco ascendente del reloj. Las entradas de direcciones determinan la dirección de columna inicial para la ráfaga, y el registro de modo indica el tipo de ráfaga (secuencial o entrelazada) y la longitud de la ráfaga (1, 2, 4, 8, página completa). El retardo desde el inicio de la orden hasta que el dato de la primera celda que aparece en las salidas coincide con el valor de latencia de que se ha fijado en registro de modo.



7.14 Temporización de una lectura de SDRAM (longitud de ráfaga =4, latencia de $\overline{CAS}$ = 2)

7.3 Mapa de memoria

Una computadora digital, para tener un mejor control sobre la ejecución de las instrucciones (en binario) y de los datos por utilizar para el desarrollo de un programa, del funcionamiento de un sistema operativo (programa monitor), etc., organiza su memoria **RAM** y memoria **ROM** en un Mapa de Memoria.

Se le llama ***Mapa de memoria*** a la representación de los bloques en que se ha dividido el espacio de memoria direccionable por el microprocesador. Cada bloque o partición corresponde al rango de direcciones ocupado por un circuito de memoria, de acuerdo con la asignación que se haya hecho de las líneas del bus de direcciones que no van conectadas a las entradas de direcciones del circuito de memoria, ver figura 7.15.

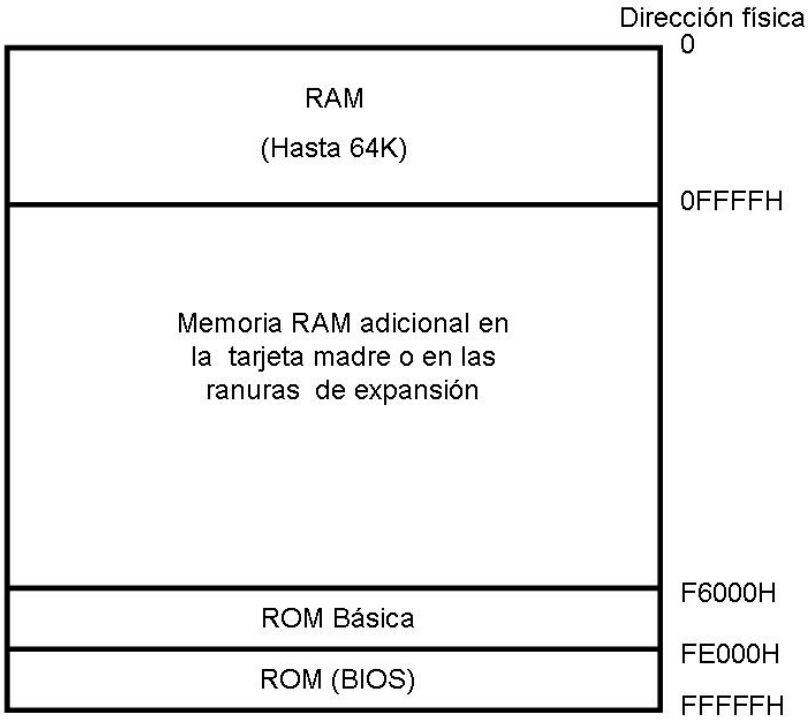


Figura 7.15 Mapa de Memoria para el Sistema

En la figura 7.15 podemos observar la ubicación de la memoria RAM y la memoria ROM en un mapa de memoria de 1 Mb basado en un microprocesador 8086. En dicha figura se observa lo siguiente:

- Hay un área de memoria **RAM** básica de 64 Kb, la cual sirve para el almacenamiento temporal de instrucciones y datos tanto de programas del usuario como programas del sistema.
- Hay un área dedicada a la memoria **ROM** en la cual se almacenan

pequeños programas del sistema.

- Hay un área de memoria, la cual tiene diferentes usos, como por ejemplo expansión de la memoria **RAM** en la tarjeta madre.

7.3.1 Memoria expandida y extendida

A medida que fue avanzando el desarrollo de programas y aplicaciones por parte de un usuario, los requerimientos de memoria aumentaron considerablemente. En las primeras computadoras basadas en microprocesadores de 8 bits (8085, Z80, 6800, etc.) y en algunos microprocesadores de 16 bits (8086, Z8000, 68000, etc.) se desarrollaron los conceptos de **memoria expandida** y **memoria extendida**, los cuales ayudaron a resolver cierta problemática con la demanda de memoria.

Memoria expandida

La memoria expandida se presentó en las primeras computadoras personales de IBM y en los compatibles, en la cual presentaban una organización lógica de memoria, de hasta 8MB que puede utilizarse en las máquinas que ejecutan MS-DOS en modo real (emulación de 8086).

La memoria expandida (ver figura 7.16) es una técnica de software utilizada para acceder a la memoria por encima de 1Mb. La memoria expandida es una memoria a la que normalmente no acceden los programas que ejecutan MS-DOS, la memoria expandida requiere una interfaz denominada **EMM** (Gestor de Memoria Expandida), que asigna páginas (bloques) de bytes de la memoria expandida según se necesite. Sólo el software compatible con **EMS** (Especificación de Memoria Expandida) puede utilizar la memoria expandida.

Como solamente se puede trabajar con 64K de información a la vez, es necesario copiar continuamente datos desde la memoria expandida (más de 1Mb) a la memoria superior y viceversa. Otra solución más rápida y eficiente es la llamada memoria extendida.

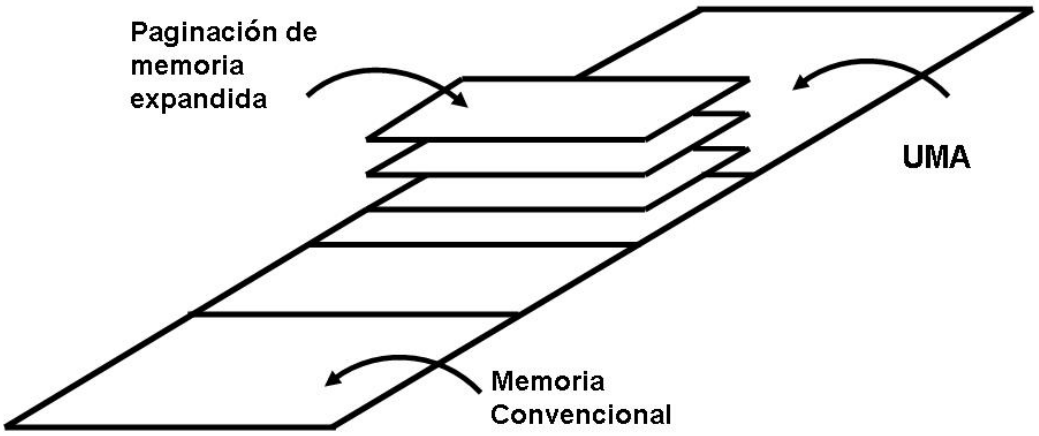


Figura 7.16 Memoria Expandida

Memoria extendida

La memoria extendida es la memoria por encima de 1Mb de la memoria convencional y el **Área de Memoria Superior UMA (**del inglés, ***Upper Memory Area***), ver figura 7.17. Para poder alcanzar esta región, el microprocesador debe trabajar en un modo llamado modo protegido. Aunque el MS-DOS no es capaz de operar en este modo, la mayoría de las aplicaciones sobre MS-DOS emplean diversas técnicas para acceder a memoria extendida.

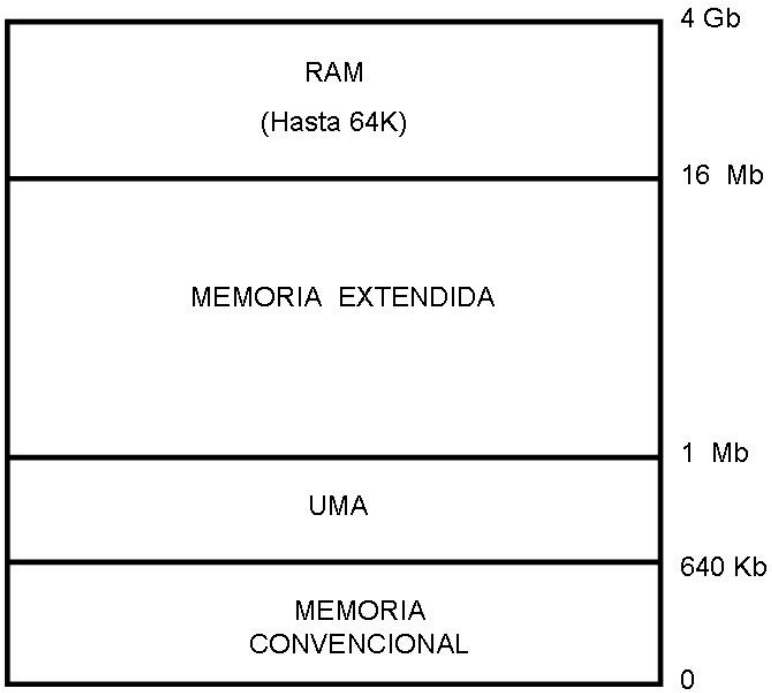


Figura 7.17 Memoria Extendida

En un IBM PC o compatible con un microprocesador 80286 o posterior, la memoria extendida se refiere a la memoria por arriba del primer MByte de espacio de dirección. La memoria extendida está solamente disponible en computadoras personales basadas en el procesador 80286 de Intel o un procesador más alto. Solamente estos chips pueden acceder a más de 1MB de **RAM**. En un microprocesador 286 o posterior, en computadoras personales equipadas con más que 640KB de **RAM**, la memoria adicional por arriba de esos 640KB es generalmente re-mapeada por arriba de 1MB, haciendo que toda ella sea disponible a programas corriendo en modo protegido. Incluso sin este re-mapeo, las máquinas con más de 1MB de **RAM** pueden tener acceso a la memoria sobre el 1MB.

Solamente las aplicaciones ejecutándose en modo protegido pueden usar directamente la memoria extendida.

Otros tipos de memoria[3]

- **Memoria convencional**

En el mapa de memoria, el área que va desde la localidad 00000H hasta la localidad A0000H se le conoce como memoria convencional. Esta memoria tiene un tamaño de 640Kb de memoria. Debido a que el sistema operativo MS-DOS administra por sí mismo la memoria convencional, no necesitará un administrador adicional para usar la memoria convencional. Todos los programas basados en MS-DOS utilizan memoria convencional.

- **Área de memoria superior (UMA, del inglés, Upper Memory Area)**

Son los 384 Kb de memoria que se encuentran a continuación de los 640 Kb de memoria convencional. El área de memoria superior es usada por el hardware del sistema, como por ejemplo, el adaptador de video. Las partes de la memoria superior que no se usan se llaman bloques de memoria superior (**UMB**); en un equipo 80386 o 80486, los bloques (**UMB**) se podrán utilizar para ejecutar controladores de dispositivos y programas residentes en memoria.

- **Área de memoria alta**

Son los primeros 640 Kb de memoria extendida. Esto es únicamente válido para Computadoras Personales que cuenten con memoria extendida.

7.3.2 Organización de memoria

Los requerimientos de memoria de una computadora personal basado en microprocesadores, frecuentemente no se pueden satisfacer con un solo circuito de memoria. En estos casos, varios de ellos deben ser interconectados para formar un sistema de memoria. En un sistema de memoria, la capacidad de almacenamiento se puede ampliar incrementando el número de palabras y/o creciendo la longitud de palabra por encima de los valores que se obtienen con un solo dispositivo. La longitud de palabra se incrementa colocando las salidas

[3] Véase, TodoBytes: "Tipos de memoria", material en línea, disponible en: http://www.todobytes.net/Articulos/Tipos_de_Memoria/tipos_de_memoria.html, recuperado el 20/02/09.

de dos o más circuitos de memoria en paralelo. Por ejemplo, **m** memorias de 1024 x 1 bit se pueden configurar en paralelo para formar un sistema de memoria de 1024 x **m** bits, ver figura 7.18.

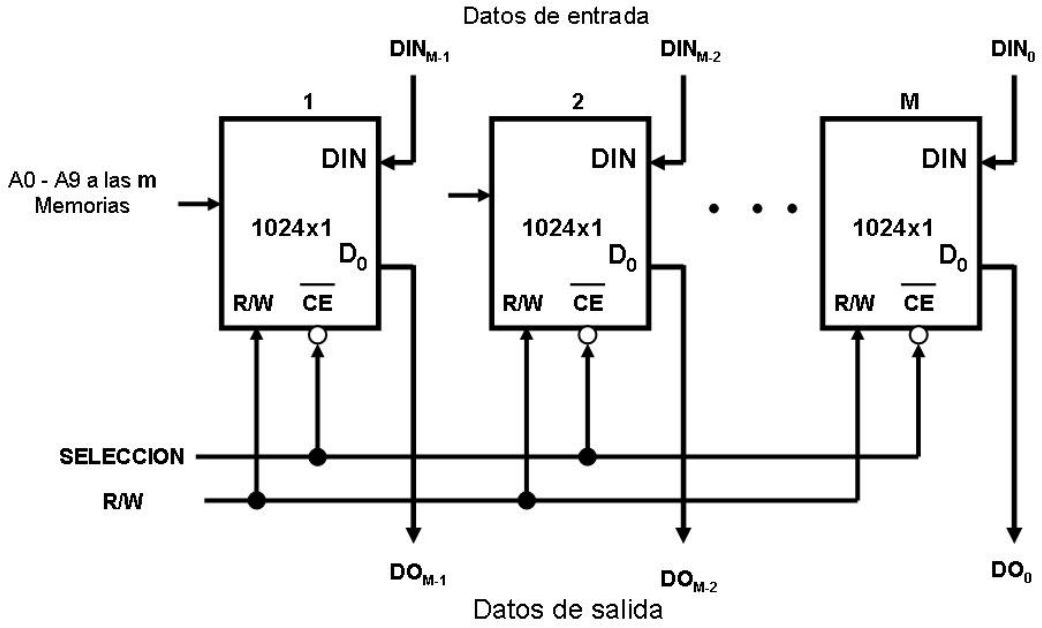


Figura 7.18 Memoria se 1024 x m bits formado a partir de circuitos de memoria de 1024 x 1.

En la figura 7.18 se observa que el número de palabras en un sistema de memoria se aumenta multiplexado las salidas de dos o más dispositivos de memoria. Para poder realizar este multiplexado se utilizan las entradas de selección (**chip select** o **chip enable**) que existen en todos los tipos de memorias y los cuales sirven para este propósito. Las entradas de selección (**chip select**) proporcionan la lógica interna que minimiza el número de componentes externos requeridos para seleccionar un circuito de memoria específico dentro del sistema, ya que con ellas se controla que las salidas de datos estén activas o en tercer estado.

7.4. Memoria cache

La memoria cache es una memoria de menor capacidad, de rápido acceso y diseñada para resolver las diferencias de velocidad entre una CPU muy rápida y una memoria principal muy lenta. Lo hace almacenando una copia de los datos de uso frecuente en una memoria de fácil acceso en vez de la memoria principal, cuyo acceso es más lento. Un tamaño de memoria cache razonablemente pequeño puede generar mejorías significativas en el rendimiento. Dado que la memoria cache es un pequeño espacio que contiene relativamente pocos datos, el procesador tiene acceso a sus datos e instrucciones con mayor rapidez que si tuviera que recuperarlos de la memoria principal. La memoria cache está situada entre el microprocesador y la memoria principal.

Funcionamiento

El objetivo de la memoria cache es lograr que la velocidad de la memoria sea lo más rápido posible, consiguiendo al mismo tiempo un tamaño grande al precio de memorias semiconductores menos costosas, ver figura 7.19. Hay una memoria principal relativamente grande y más lenta, junto con una memoria cache más pequeña y rápida. La memoria cache contiene una copia de partes de la memoria principal. Cuando el microprocesador intenta leer una palabra de memoria, se hace una comprobación para determinar si la palabra está en la cache. Si es así, se entrega dicha palabra al procesador. Si no, un bloque de memoria principal, consistente en un cierto número de palabras se transfiere a la cache y después la palabra es entregada al microprocesador. Debido al fenómeno de localidad de las referencias, cuando un bloque de datos es capturado por la cache para satisfacer una referencia a memoria simple, es probable que se hagan referencias futuras a la misma posición de memoria o a otras palabras del mismo bloque.

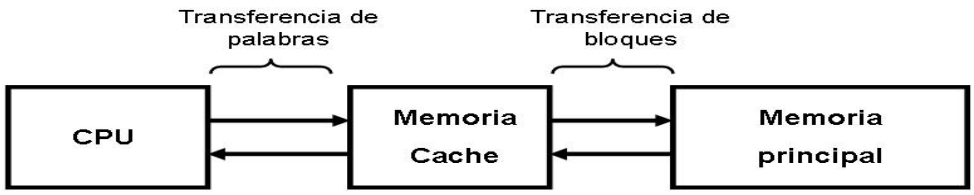


Figura 7.19 Memorias Caché y principal

Esto se logra debido a que en todos los ciclos de instrucción el procesador accede al menos una vez para leer la instrucción, y con frecuencia, una o más veces adicionales, para leer y/o almacenar los resultados. La velocidad a la que el procesador puede ejecutar instrucciones está claramente limitada por el tiempo de ciclo de memoria (el tiempo que se tarda en leer o escribir una palabra de la memoria). Esta limitación ha sido de hecho un problema significativo debido a la persistente discrepancia entre la velocidad del procesador y la de la memoria principal. La velocidad del procesador se ha incrementado constantemente de forma más rápida que la velocidad de acceso a la memoria.

Por otro lado, el diseñador se encuentra con un compromiso entre velocidad, costo y tamaño al construir la memoria principal. Idealmente, se debería construir la memoria principal con la misma tecnología que la de los registros del procesador, consiguiendo tiempos de ciclo de memoria comparables con los tiempos de ciclo del procesador. Esa estrategia siempre ha resultado demasiado costosa. La solución consiste en aprovechar el principio de la proximidad utilizando una memoria pequeña y rápida entre el procesador y la memoria principal, denominada **memoria cache**.

Finalmente, el propósito de la memoria cache es proporcionar un tiempo de acceso a memoria próxima al de las memorias más rápidas disponibles y, al mismo tiempo, ofrecer un tamaño de memoria grande que tenga el precio de los tipos de memorias de semiconductores menos costosos y con lo cual el procesador tiene acceso a sus datos e instrucciones con mucha mayor rapidez que si tuviera que recuperarlos de la memoria principal. Un controlador de cache determina la frecuencia con que se utilizan los datos, transfiere los que se usan a menudo a la memoria cache y los elimina cuando identifica datos de uso menos constante. Los datos en la memoria cache se deben considerar como temporales. En el caso de una falla de energía, se pierden y no se pueden recuperar, a diferencia de los datos escritos en el almacenamiento secundario.

Diseño de la memoria cache

A continuación se resumen brevemente los aspectos de diseño de la memoria cache y el cual se divide en las siguientes categorías:

- Tamaño de la cache
- Tamaño del bloque

Un **tamaño de la memoria cache** razonablemente pequeño puede tener un impacto significativo en el rendimiento. Otro aspecto relacionado con la capacidad de la cache es el **tamaño del bloque**: el cual es la unidad de datos que se intercambia entre la cache y la memoria principal.

Según el tamaño del bloque se incrementa desde muy pequeña a tamaños mayores, al principio la tasa de aciertos aumentará debido al principio de la proximidad: la alta probabilidad de que accedan en el futuro inmediato a los datos que están en la proximidad de una palabra a la que se ha hecho referencia.

Según se incrementa el tamaño de bloque, se llevan a la cache más datos

útiles. Sin embargo, la tasa de aciertos comenzará a decrecer cuando el tamaño del bloque siga creciendo, ya que la probabilidad de volver a usar los datos recientemente leídos se hace menor que la de utilizar nuevamente los datos que se van a expulsar de la cache para dejar sitio al nuevo bloque.

7.5 Memoria virtual

La memoria virtual es la técnica que permite correr a los programas aún cuando no están ciento por ciento en memoria. Da al usuario la ilusión de que está disponible una gran cantidad de memoria principal cuando de hecho no es así.

El tamaño del almacenamiento virtual está limitado por el esquema de direccionamiento del sistema de computación y por la cantidad de memoria disponible y no por el tamaño de memoria principal.

El espacio de almacenamiento direccionable es aquel en el cual las direcciones virtuales se traducen a direcciones reales.

La memoria virtual es una utilidad que permite a los programas direccionar la memoria desde un punto de vista lógico, sin importar la cantidad de memoria principal física disponible. La memoria virtual fue concebida como un método para tener múltiples trabajos de usuario residiendo en memoria principal de forma concurrente, de forma que no exista un intervalo de tiempo de espera entre la ejecución de procesos sucesivos, es decir, mientras un proceso se escribe en almacenamiento secundario, se lee el proceso sucesor. Debido a que los procesos varían de tamaño, si el procesador planifica un determinado número de procesos, es difícil almacenarlos compactamente en memoria principal. Se introdujeron los sistemas de paginación, que permiten que los procesos se compriman en un número determinado de bloques de tamaño fijo, denominados **páginas**. Un programa hace referencia a una palabra por medio de una **dirección virtual**, que consiste en un número de página y un desplazamiento dentro de la página. Cada página de un proceso se puede localizar en cualquier sitio de memoria principal. El sistema de paginación

proporciona una proyección dinámica entre las direcciones virtuales utilizadas en el programa y una **dirección real**, o dirección física, de memoria principal.

Administración de la memoria virtual

La administración de la memoria virtual tiene varias ventajas:

- El tamaño de una tarea ya no queda sujeta al tamaño de la memoria principal (del espacio libre dentro de la memoria principal).

- La memoria se utiliza con más eficiencia porque las únicas secciones de un área almacenadas en la memoria son las que se necesitan de inmediato, en tanto que las que no se precisan se mantienen en almacenamiento secundario.

- Permite una cantidad ilimitada de multiprogramación.

- Elimina la fragmentación externa cuando se utiliza con la paginación, suprime la fragmentación interna cuando se usa con la segmentación.

- Permite compartir códigos y datos.

- Facilita el enlace dinámico de segmentos de programa.

Desventajas

- Costos de hardware de procesador más altos

- Mayor carga general para el manejo de las interrupciones de paginación.

- Incremento de la complejidad del software para evitar la hiperpaginación.

Bibliografía del tema 7

Intel, *Microsystem Components Handbook,* EE.UU., Intel Corporation. Literature Department, 1985. Vol. 1 y 2.

García Narcia, O., *8085A e Interfaces, México,* IPN, agosto 1982.

McGlynnn, R., Daniel, *Modern Microprocessor System Design,* EE.UU., John Wiley & sons, 1980.

Uruñuela Martínez, José Ma., *Microprocesadores Programación e interconexión*, México, McGraw-Hill, 1989.

Stallings, William, *Organización y Arquitectura de Computadores*, Madrid, Pearson Educación, 2006.

Actividades de aprendizaje

A.7.1. Realiza un tabla comparativa de las memorias **RAM** (estáticas y dinámicas), considerando fabricante, longitud de palabra, tiempo de acceso.

A.7.2. Realiza una tabla comparativa de las memorias **ROM** (PROM Y EEPROM), considerando fabricante, longitud de palabra, tiempo de acceso.

A.7.3. Diseña un banco de memoria **RAM** dinámica de 1 Mb x 8 bits.

A.7.4. Diseña un banco de memoria **RAM** estática de 16 Mb x 16 bits.

A.7.5. Diseña un banco de memoria **EEPROM** de 1Mb x 8bits.

A.7.6. Diseña un banco de memoria **RAM** dinámica (SDRAM) 1GMb x 32 bits.

A.7.7. Diseña un banco de memoria **RAM** 16 GMb x 32 bits.

A.7.8. Investiga y describe el Mapa de Memoria de una tarjeta madre fabricado por Intel para un microprocesador de 32 bits.

A.7.9. Investiga y describe el Mapa de Memoria de una tarjeta madre fabricado por Intel para un microprocesador de 64 bits.

A.7.10. Investiga y describe el Mapa de Memoria de una tarjeta madre fabricado por AMD para un microprocesador de 32 bits.

A.7.11. Investiga y describe el Mapa de Memoria de una tarjeta madre fabricado por AMD para un microprocesador de 64 bits.

Cuestionario de autoevaluación

1. ¿Cuáles son los tipos básicos de memoria en la Unidad de Memoria?
2. ¿Cuántos y cuáles son los ciclos de una memoria **ROM**?
3. ¿Cuántos y cuáles son los ciclos de una memoria **RAM** dinámica?
4. ¿Cuáles son las características de las memorias **RAM**?
5. ¿Cuáles son las tecnologías utilizadas en la fabricación de las memorias **RAM**?
6. ¿Qué es la memoria cache?
7. ¿Qué es la memoria virtual?
8. ¿Qué es un Mapa de memoria?
9. ¿Qué es la memoria expandida?
10. ¿Qué es la memoria extendida?
11. ¿Cómo está organizada la memoria en un procesador?
12. ¿Cuáles son los elementos básicos para construir una memoria **RAM** ?.
13. ¿Qué es el Área de Memoria Superior?

Examen de autoevaluación

Relaciona las columnas escribiendo dentro del paréntesis el número que una los conceptos con sus respectivas definiciones.

1.) Unidad de memoria		( ) Memoria a las que se le puede realizar la función de leer los contenidos pero no la función de escribir.
2.) Memoria **RAM**		( ) Técnica de software utilizada para acceder a la memoria por encima de 1Mb.
3.) Memoria Cache		( ) Memoria ubicada encima de 1Mb de la memoria convencional y del **UMA**
4.) Memoria **ROM**		( ) Memorias a las que se les puede cambiar el contenido de sus localidades con la funciones de "**Escritura**", y "**Lectura**".
5.) Memoria Virtual		( ) Representación de los bloques en que se ha dividido el espacio de memoria direccionable por el microprocesador
6.) Ciclo de actualización.		( ) Memoria de menor capacidad, rápido acceso y diseñada para resolver las diferencias de velocidad entre una CPU y la memoria principal.
7.) Mapa de Memoria		( ) Lectura, escritura y actualización
8.) Memoria extendida		( ) Proceso periódico de actualización de datos para las memorias **RAM** dinámicas.
9.) Memoria expandida		( ) Lugar donde se almacenan las **instrucciones** (codificadas en binario) y los **datos** de un programa.
10.) Ciclos de memoria		( ) Da al usuario la ilusión de que está disponible una gran cantidad de memoria principal cuando de hecho no es así.

TEMA 8. UNIDADES FUNCIONALES

Objetivo particular

Al finalizar el tema, el alumno reconocerá la estructura del BIOS, el funcionamiento de las interrupciones, la configuración de los puertos de comunicación (i.e., el puerto paralelo y puerto serie), así como el proceso de almacenamiento de información.

Temario detallado

8.1. Arquitectura de una PC bajo el esquema de Von Newman

- 8.1.1 El BIOS
- 8.1.2 Direcciones de entrada/salida (E/S)
- 8.1.3 Niveles de interrupción (IRQ´S)
- 8.1.4 Canales DMA
- 8.1.5 Puertos de comunicación
- 8.1.6 Sistemas de almacenamiento de información

Introducción

Las unidades funcionales son unidades complementarias para el funcionamiento interno y externo de una Computadora Personal (Microcomputadora Digital). Dentro del funcionamiento interno de una computadora dichas unidades llevan a cabo ciertas tareas que originalmente las realizaba el microprocesador de la Computadora Personal. Por ejemplo una de estas tareas, consiste en establecer un camino directo entre la memoria y los dispositivos de E/S realizada por la unidad funcional de Acceso Directo a Memoria (**DMA**, del inglés **Direct Memory Access**).

Dentro de las funciones externas que realizan estas unidades funcionales está la entrada y salida de datos a partir de ciertos dispositivos especiales, por ejemplo la salida de datos en forma paralela hacia una impresora o la transmisión serial hacia un Módem.

8.1 Arquitectura de una PC bajo el esquema de von Newman

La arquitectura de una Computadora Personal -**PC**- (del inglés **Personal Computer**) presenta una serie de elementos que complementan el funcionamiento de una computadora, como lo son: el **BIOS**, los canales **DMA**, los puertos de comunicación, etc., los cuales se explicarán a continuación.

8.1.1 El BIOS

El Sistema Básico de Entrada/Salida -**BIOS**- (del inglés **Basic Input Output System**) es un conjunto de programas que están almacenados permanente en una memoria **ROM** de la Computadora Personal. Cuando se enciende una Computadora Personal, el **BIOS** carga el sistema operativo en memoria **RAM** para su ejecución. Las funciones mínimas del **BIOS** son:

1) Proporcionar una comunicación de bajo nivel.
2) Configurar el Hardware.
3) Poner en funcionamiento un pequeño sistema operativo (**Programa Monitor**) que, como mínimo, maneja el teclado y proporciona como salida básica la bocina (emitiendo pitidos por la bocina de la computadora si se producen fallos) durante el arranque.

En resumen, el **BIOS** proporciona el software mínimo necesario para controlar varios programas (sistema operativo) que ponen a funcionar una Computadora Personal. El **BIOS** se almacena en un área de memoria ROM, la cual forma parte del Mapa de Memoria, ver figura 8.1.

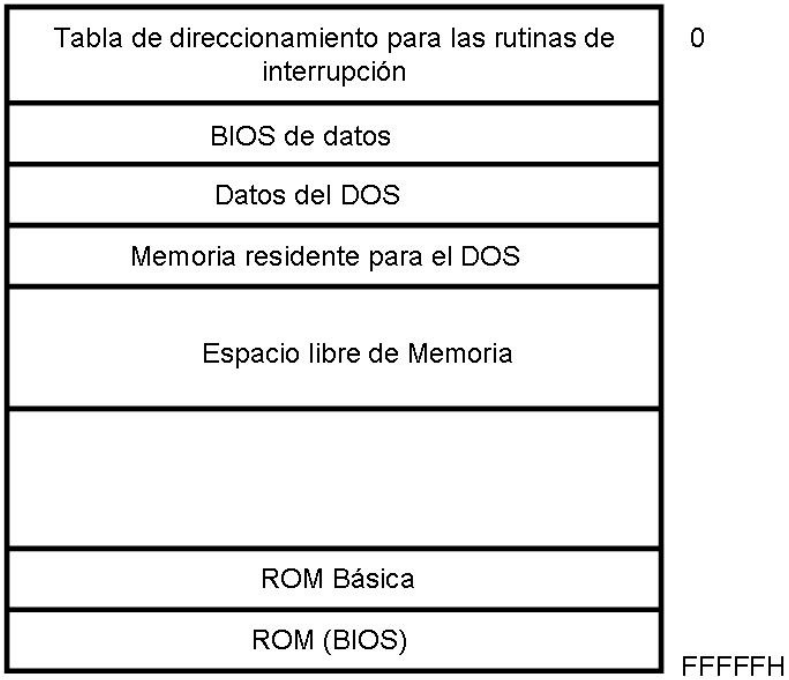


Figura 8.1 Ubicación del BIOS en el Mapa de Memoria

En las primeras versiones de los sistemas operativos para computadoras personales (por ejemplo, CP/M y DOS), el **BIOS** todavía permanecía activo tras el arranque y funcionamiento del sistema operativo. El acceso a dispositivos como el diskette, el disco duro se hacía a través del **BIOS** activo. Sin embargo, los sistemas operativos más modernos realizan estas tareas por sí mismos, sin necesidad de llamadas a las rutinas del **BIOS**.

8.1.2 Direcciones de entrada/salida (E/S)

Las direcciones de entrada/salida son direcciones (localidades) de memoria establecidas por el diseñador de computadoras las cuales permiten capturar y/o enviar datos a través de las diferentes unidades funcionales (por ejemplo, Puerto Paralelo, Puerto Serial, Controlador de interrupciones, etc.). Las direcciones de la entrada estándar (teclado) y la salida estándar (Monitor) de una Computadora Personal basada en el microprocesador 8088 se indican en

la Tabla 8.1.

VIA INVOCADA	REQUERIMIENTOS DE ENTRADA	REGISTRO	SALIDA / RESULTADOS
INT 10H	SALIDA DEL MONITOR AH = 0, AL = MODE: AL=0: 40x25 BW AL=1: 40x25 COLOR AL=2: 80x25 BW AL=3: 80x25 COLOR	AX, SI, DI	COLOCA EL MONITOR EN EL MODO ESPECIFICADO POR EL REGISTRO AL
	AH = 2, BH = 0, DH = renglón DL = columna	AX, SI, DI	COLOCA EL CURSOR EN LA POSICION ESPECIFICADA. RENGLON 0, COLUMNA 0.
	AH = 14, BX = 0, AL = carácter	AX, SI, DI	MUESTRA EL CARÁCTER ESPECIFICADO EN LA PANTALLA DEL MONITOR EN LA POSICION ACTUAL DE CURSOR.
INT 13H	ENTRADA/SALIDA FLOPPY		
INT 14H	ENTRADA/SALIDA SERIAL		

Tabla 8.1 Direcciones de entrada/salida

VIA INVOCADA	REQUERIMIENTOS DE ENTRADA	REGISTRO	SALIDA / RESULTADOS
INT 16H	TECLADO AH = 0	AX	LEE UN CARÁCTER DESDE EL TECLADO Y LO PONE EN EL REGISTRO AL
	AH = 1	AX	COLOCA ZF=0, SI UNA TECLA ES OPRIMIDA, O ZF=1 EN CUALQUIER OTRO CASO.
	AH = 2	AX	REGRESA EL STATUS DE LA TECLA OPRIMIDA EN LOS BITS DE EL REGISTRO AL: Bit 7 = INSERT KEY Bit 6 = CAPS LOCK Bit 5 = INSERT KEY Bit 4 = SCROLL LOCK Bit 3 = ALT SHIFT Bit 2 = CTL SHIFT Bit 1 = LEFT SHIFT Bit 0 = RIGHT SHIFT

Tabla 8.1 Rutinas del BIOS (Continuación)

El controlador de interrupciones es otra unidad funcional presente en una Computadora Personal, dicho controlador se encarga de atender una serie de peticiones realizadas por ejemplo un dispositivo externo. Dicho controlador tiene dos registros los cuales se encuentran ubicados en las direcciones siguientes:

20H	Registro de Comando de Interrupción
21H	Registro de Máscara de Interrupción

Direcciones del C. I. 8259 Controlador de Interrupción

El Puerto Paralelo Programable se encarga de enviar datos hacia el exterior en forma paralela. Su uso más común es enviar datos hacia una impresora. Esta unidad funcional posee 4 registros, tres de los cuales sirven para enviar y recibir datos (Puerto A, Puerto B y Puerto C). El cuarto registro (Registro de Comando) sirve para programar o establecer el modo de trabajo de los tres primeros registros. Estos cuatro registros cuentan con una dirección la cual presentamos a continuación:

60H	Puerto de entrada “PA”
61H	Puerto de entrada/salida “PB”
62H	Puerto de entrada “PC”
63H	Registro de Comando del 8255

Direcciones del C. I. 8255 Periférico Programable de E/S Paralelo

8.1.3 Niveles de interrupción (IRQ´s)

Una interrupción es una señal recibida por el microprocesador de una Computadora Personal, indicando que debe "interrumpir" el curso de ejecución actual y pasar a ejecutar una llamada a una subrutina para atender esta solicitud de interrupción. La petición de una interrupción es asíncrona, esto es, puede ocurrir en cualquier momento durante la ejecución del programa

principal. Para atender el uso de las interrupciones se propusieron dos técnicas: una utiliza al microprocesador y la otra utiliza una unidad funcional conocida como controlador de interrupciones.

1. **Atención de una interrupción utilizando un microprocesador**
   Esta primera técnica emplea al propio microprocesador, el cual se encarga de sondear (**polling**) el dispositivo cada cierto tiempo para averiguar si tenía pendiente alguna comunicación para él. El principal inconveniente es que es muy ineficiente, ya que el procesador constantemente consume tiempo en realizar todas las instrucciones de sondeo.

2. **Atención de una interrupción utilizando un controlador de interrupciones**
   Esta segunda técnica, utiliza un nuevo mecanismo de interrupciones, el cual le permite al microprocesador olvidarse de esta problemática y delegar a un dispositivo la responsabilidad de comunicarse cuando lo necesite. El microprocesador, en este caso, no sondea a ningún dispositivo, sino que queda a la espera de que estos le avisen (le "interrumpan") cuando tengan un evento, una transferencia de información, una condición de error, etc. Una computadora personal utiliza este tipo de técnica.

El mecanismo de las interrupciones consiste en cada dispositivo que desea comunicarse con el microprocesador por interrupciones; debe tener asignada una línea única capaz de avisar a éste de que le requiere para una operación. Esta línea es la llamada petición de interrupción **IRQ** (del inglés "**Interrupt ReQuest**"). Las **IRQ** son líneas que llegan al controlador de interrupciones, el cual es un componente de hardware dedicado a la gestión de las interrupciones y que puede o no estar integrado en el microprocesador. El controlador de interrupciones debe ser capaz de habilitar líneas de interrupción (operación llamada "enmascarar") y establecer prioridades entre las distintas interrupciones habilitadas. Cuando varias líneas de petición de interrupción se activan a la vez, el controlador de interrupciones utilizará estas prioridades para escoger la interrupción sobre la que informará al microprocesador.

Un microprocesador (con el controlador de interrupciones integrado) suele tener una única línea de interrupción llamada habitualmente **INT**. Esta línea es activada por el controlador de interrupciones cuando tiene una interrupción que servir. Al activarse esta línea, el microprocesador consulta los registros del controlador de interrupciones para averiguar qué **IRQ** es la que ha de atender. A partir del número de **IRQ** busca en el vector de interrupciones qué rutina debe llamar para atender una petición del dispositivo asociado a dicha **IRQ**.

Las rutinas de interrupción generalmente toman un pequeño tiempo de ejecución y la mayoría no pueden ser interrumpidas cuando se están atendiendo, porque al entrar en ellas se almacena el estado de los registros en una pila y si se interrumpen muchas veces, la pila se puede desbordar.

Pasos para el procesamiento de una **IRQ**

1. Terminar la ejecución de la instrucción máquina en curso.

2. Almacenar el valor de contador de programa **-PC-** (del inglés **Program Counter**), en la pila, de manera que en la CPU, al terminar el proceso, pueda seguir ejecutando el programa a partir de la última instrucción.

3. La CPU salta a la dirección donde está almacenada la rutina de servicio de interrupción (**ISR, Interrupt Service Routine**) y ejecuta esa rutina que tiene como objetivo atender al dispositivo que generó la interrupción.

4. Una vez que la rutina de la interrupción termina, el microprocesador restaurará él estado que había guardado en la pila en el paso 2 y retorna el programa que se estaba usando anteriormente.[4]

Una Microcomputadora basada en el microprocesador 8085A tiene 5 líneas de entrada para señales de control por medio de las cuales los dispositivos (circuitos) externos pueden solicitar la atención inmediata de la microcomputadora, estas señales se conocen como **"solicitudes de interrupción"**. Las líneas de **solicitudes de interrupción** solicitan a la microcomputadora interrumpir el trabajo que está ejecutando en el momento de la solicitud para atender las necesidades de los dispositivos (circuitos) externos solicitantes. La transferencia de datos por medio de interrupciones hacen más eficiente el uso del tiempo de procesamiento de las microcomputadoras, ya que

[4] Wikipedia: "Interrupción", actualizado el 27/02/09, disponible en: http://es.wikipedia.org/wiki/Interrupci%C3%B3n. Recuperado: 03 de marzo de 2009.

la microcomputadora no tiene que estar preguntando si un periférico tiene datos o está lista para recibir datos, es el periférico con la interrupción quién se lo indica.

Las interrupciones permiten que eventos tales como alarmas, fallas de energía y periféricos (que tengan datos o estén listos para recibir datos) obtengan una atención inmediata de la CPU. El programador no necesita tener a la CPU verificando continuamente si ocurrieron estos eventos.

Tipos de interrupciones

Existen tres modos básicos de interrupción: una **línea, multinivel** y **vector**.

Una línea

Una señal de interrupción a la entrada de la interrupción **INTR** de la microcomputadora causará que tome lugar una serie de actividades muy bien definidas. Varios dispositivos pueden utilizar esta entrada a través de una compuerta OR. La interfaz de cada dispositivo debe tener una bandera de interrupción (flip-flop) que se pone cuando el dispositivo solicita una interrupción. La microcomputadora puede investigar, por programa, qué dispositivo está solicitando una interrupción y una vez que investiga cuál es el dispositivo solicitante, la microcomputadora le dará atención por medio de la rutina de servicio particular.

Multinivel

La microcomputadora proporciona varias líneas independientes de interrupción y cada una causará una serie de actividades específicas. No se requiere un programa para investigar qué dispositivo está solicitando la interrupción a menos que cada línea se encuentre conectada a varios dispositivos a través de compuertas OR.

Vector

El dispositivo debe proporcionar un vector de interrupciones a la microcomputadora después de que realiza la solicitud de interrupción y ésta es

aceptada por la microcomputadora. Este vector de interrupción le indica a la microcomputadora en qué localidad debe continuar para darle atención al dispositivo que se encuentra solicitando la interrupción.

El microprocesador 8085A utiliza la técnica de interrupción por vectores y por multinivel y para entender cómo funcionan estas técnicas de interrupción, empezaremos estudiando la instrucción **Restart** (**RST**), la cual es muy importante en el proceso de interrupción.

Instrucción RST

Durante la ejecución de la instrucción **CALL**, el contenido del Contador del Programa **-PC-** (del inglés **Program Counter**) se guarda en el **Stack** de la memoria y se carga con el contenido de los bytes dos y tres de la instrucción **CALL**. La instrucción **RST** funciona en forma semejante a la instrucción **CALL** ya que almacena el contenido del **Contador del Programa** en el **Stack** y lo carga con una nueva dirección. La diferencia con la instrucción **RST** es en la forma de obtener la nueva dirección. Con la instrucción **CALL**, la 8085A obtiene la dirección en los bytes dos y tres de la propia instrucción mientras que con la instrucción **RST** la dirección la obtiene al multiplicar por 8 el valor de los bits 3, 4 y 5 del código de la misma instrucción **RST**. El código de la instrucción **RST** es el siguiente:

	7	6	5	4	3	2	1	0
Código de **Restart**	1	1	X	X	X	1	1	1

La instrucción realmente es **RST X**, en donde **X** es el valor en decimal de los bits 3, 4 y 5. Con un campo de tres bits se pueden formar 8 instrucciones, desde **RST 0** hasta **RST 7**.

La tabla 8.2 muestra las ocho posibilidades con la instrucción **RST** indicando el código y la dirección que se carga en la computadora. La ventaja de la instrucción **RST** es que se puede saltar a una de 8 posibles direcciones bien definidas y salvar el contenido del **PC** en el área de **Stack** ejecutando una

instrucción de un byte en vez de tres bytes como en el caso de la instrucción **CALL**. En la misma tabla se puede observar que las direcciones que se pueden alcanzar con la instrucción **RST** tienen entre sí una diferencia de 8H localidades.

Instrucción	Código	Dirección
RST 0	C7H	0000H
RST 1	CFH	0008H
RST 2	D7H	0010H
RST 3	DFH	0018H
RST 4	E7H	0020H
RST 5	EFH	0028H
RST 6	F7H	0030H
RST 7	FFH	0038H

Tabla 8.2 Instrucciones RST

Líneas de interrupción de la 80885A

La 8085A tiene 5 entradas por medio de las cuales los dispositivos externos pueden "**solicitarle interrupciones**" al procesamiento. Estas entradas son:

INTR

Ordena a la CPU a realizar un ciclo **Fetch** "especial" para obtener el código de una instrucción (**RST** o **CALL**) que envía un periférico por el **Bus de Datos**. La 8085A tiene un registro (flip-flop **INTE**) que controla las solicitudes de interrupción que recibe por esta línea, acepta o no las solicitudes. Esta línea es "**mascarable**".

TRAP

Ordena a la CPU guardar el contenido del **Contador del Programa** en el área **Stack** y a cargar el **Contador del Programa** con la dirección 0024H para

continuar a partir de esa dirección el procesamiento de la CPU. La línea **TRAP** genera solicitudes de interrupción "**no mascarables**", es decir, no hay máscara o bandera (flip-flop) que lo controle, no se puede deshabilitar.

RST 5.5

Ordena a la CPU a guardar el contenido del **Contador del Programa** en el área de **Stack** y cargar el **Contador del Programa** con la dirección 002CH para continuar a partir de esa dirección el procesamiento de la CPU. La 8085A tiene una máscara que controla las solicitudes de interrupción que se reciben por esta línea, acepta o no las solicitudes. Esta línea es "**mascarable**".

RST 6.5

Igual que la línea **RST 5.5** pero carga el **Contador del Programa** con la dirección 0034H.

RST 7.5

Igual que la línea **RST 5.5** pero carga el **Contador del Programa** con la dirección 0003CH.

La 8085A cuenta con un flip-flop interno denominado "Habilitar Interrupciones" flip-flop **INTE** (**Interrupt Enable**), que controla la aceptación de las solicitudes de interrupción de la línea **INTR** y las tres líneas **RST.** Cuando el flip-flop **INTE** tiene nivel 1 permite las solicitudes en cualquiera de las cuatro líneas mascarables y no las acepta cuando tienen nivel 0.

El estado del flip-flop **INTE** lo controla el programador por medio de las instrucciones **EI** (**Enable Interrupt**) y **DI** (**Disable Interrpt**). La instrucción **EI** "pone" el flip-flop **INTE** permitiendo que la 8085A acepte las solicitudes de interrupción por las cuatro líneas mascarables, y la instrucción **DI** "limpia" el flip-flop **INTE** permitiendo que la 8085A acepte las solicitudes de interrupción por las cuatro líneas mascarables.

Cuando la 8085A "acepta" una solicitud de interrupción externa su lógica interna "limpia" en forma automática el flip-flop **INTE**, con lo que se retira el

permiso de más solicitudes de interrupción "mascarables" hasta que se ejecute otra vez la instrucción **EI**. Al activarse la línea **RESET IN** también se limpia el flip-flop **INTE**. En otras palabras podemos decir que existe solo una forma de permitir interrupciones, ejecutando la instrucción **EI** y tres formas de suspender al permiso de interrupciones, ejecutando la instrucción **DI**, cuando la 8085A "acepta" una solicitud y al activarse la línea **RESET IN** de la 8085A.

La 8085A tiene dos registros: a) El **Estado de las Máscaras de Interrupción (EMI)**, el cual se lee con la instrucción **RIM** y b) El registro **Máscara de Interrupción (I)**, el cual se carga con la instrucción **SIM**.

Estos registros se utilizan para el control de las máscaras de las tres líneas **RST**. Las solicitudes en las tres líneas **RST** causan la ejecución de una instrucción **RST** (salvando el contenido del **Contador del Programa** y cargándolo con una dirección) siempre y cuando estén habilitadas las interrupciones (flip-flop **INTE** = 1) y no tengan puestas sus máscaras en el registro de "**Máscaras de Interrupción**".

Las prioridades de las líneas de interrupción de mayor a menor son:

1) **TRAP**, 2) **RST 7.5**, 3) **RST 6.5**, 4) **RST 5.5** y 5) **INTER**

Las interrupciones están arregladas de tal forma que la 8085A acepta la solicitud de interrupción de la línea de mayor prioridad cuando hay más de una solicitud. Este esquema no toma en cuenta la prioridad de la línea que está siendo atendida. Una solicitud en la línea **RST 5.5** puede interrumpir a una rutina de la línea **RST 7.5** si el flip flop **INTE** = 1.

8.1.4 Canales DMA

Los canales **DMA** (del inglés **Direct Memory Access**) son rutas del sistema usados por muchos dispositivos para transferir información directamente a la memoria en ambos sentidos. Hoy en día los **DMA** son utilizados comúnmente en diskettes y tarjetas de sonido.

En las primeras computadoras personales, el procesador era la parte principal de la máquina, es decir, el procesador hacía prácticamente todo. Aparte de hacer funcionar los programas también era responsable de transferir datos a los periféricos. Desafortunadamente, dejar que el procesador haga estas transferencias, es bastante negativo porque le impide hacer otras tareas.

La invención de la tecnología **DMA** permitió a los procesadores hacer otros trabajos y que los periféricos transfieran los datos ellos mismos, con la consiguiente mejora del rendimiento. Algunos canales especiales fueron creados permitiendo la transferencia de información sin que el procesador controlara cada aspecto de la transferencia.

Hay que tener en cuenta que los canales **DMA** solo se encuentran en los bus **ISA** (y en los **EISA** y **VLB**). Los dispositivos **PCI** no utilizan los canales **DMA** estándar en absoluto.[5]

Las computadoras personales modernas cuentan con el **DMA** en la tarjeta madre. Este **DMA** tiene ocho circuitos (o “canales") para la transmisión de datos. El canal 4 de **DMA** se reserva para que el sistema lo use.

Un inconveniente de los canales **DMA** es que son dispositivos únicos cuando hablamos de compartir recursos. Si dos dispositivos tratan de usar el mismo canal **DMA** al mismo tiempo, la información se mezclará entre los dos equipos que tratan de usarlo.

Arquitectura del controlador de DMA

La figura 8.2 muestra un diagrama a bloques de un C.I. 8237A. El C.I. 8237A está configurado para hacer transferencias por **DMA**, teniendo el CPU que programar sus 16 registros internos a través de un conjunto de 16 puertos de **E/S**. La programación se hace a través de las líneas de control $\overline{\mathbf{IOR}}$, $\overline{\mathbf{IOW}}$ y $\overline{\mathbf{CS}}$ (**chip select**), y las líneas de datos **DB0-DB7**. Cuando realmente se hace una transferencia por **DMA**, el C.I. 8237 toma las líneas de control y dirección de la CPU y genera estas señales él mismo. Esto es porque las líneas **CLOCK** y **READY**, las líneas de control $\overline{\mathbf{MEMR}}$, $\overline{\mathbf{MEMW}}$, $\overline{\mathbf{IOR}}$ e $\overline{\mathbf{IOW}}$ y las líneas de dirección están presentes. Aunque no está indicado en la figura, las líneas de

[5] Ordenadoresyportátiles.com: “¿Que son los canales DMA?”, material en línea, disponible en: http://www.ordenadores-y-portatiles.com/dma.html. Recuperado el 3 de marzo de 2009.

datos **DB0- DB7** se convierten en las líneas de dirección **A8-A15** durante la transferencia por **DMA.**

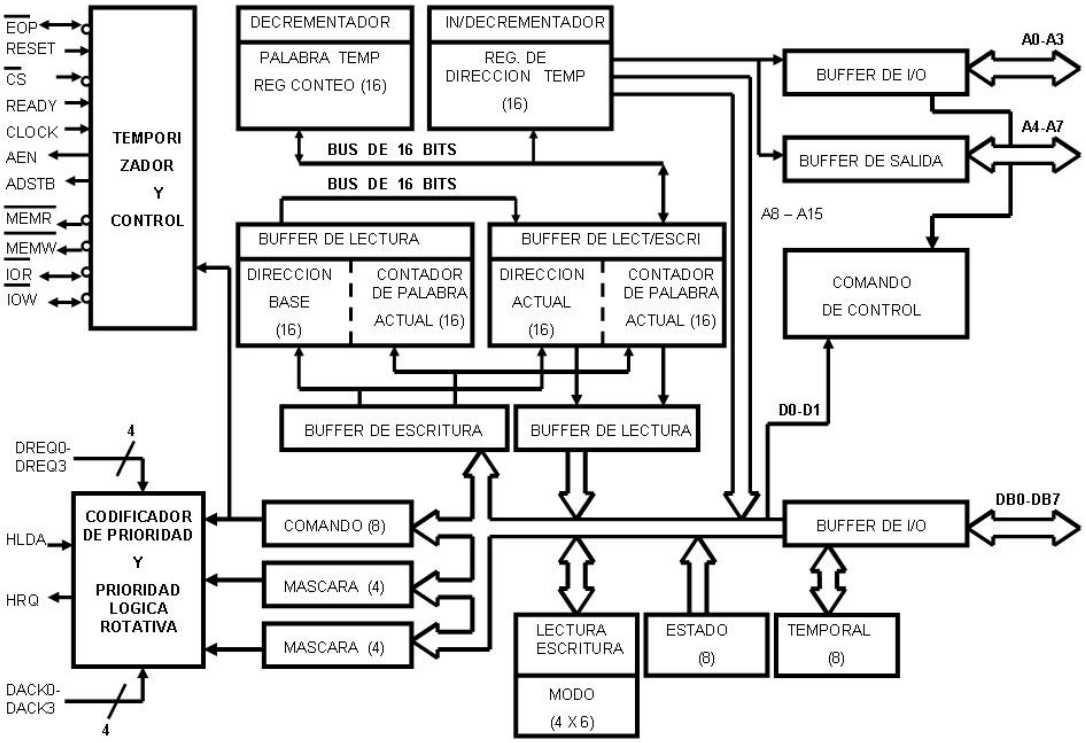


Figura 8.2 Diagrama a bloques del 8237A de Intel

Dentro de cada C.I. 8237A hay cuatro canales de **DMA**. Cada canal se puede programar independientemente para realizar su propia transferencia por **DMA**, es decir, un C.I. 8237 se comporta como cuatro controladores de **DMA** coordinados de tal manera que en un tiempo dado sólo uno de ellos toma el control del bus.

Las líneas **DREQn** (solicitud de **DMA**) son entradas que se deben conectar a los dispositivos **E/S** que necesitan servicio **DMA**. Las líneas de salida correspondientes **DACKn** (reconocimiento de canal de **DMA**) se deben conectar a aquellos dispositivos E/S que sirven como las selecciones de integrado del dispositivo **E/S**. Cuando el C.I. 8237A recibe una petición de **DMA** desde una línea **DREQn,** pone a **HRQ** (solicitud de suspensión) en alto. Esta línea se conecta al **HOLD** de la CPU, causando que la CPU ponga en tercer estado sus líneas de dirección, datos y control. La CPU le dice al 8237 que esto está hecho subiendo la línea **HLDA**. Cuando se ve que **HLDA** se va a alto, el C.I 8237A empieza su transferencia **DMA**. La línea $\overline{\mathbf{EOP}}$ (fin de proceso)

del 8237 es bidireccional. Cuando se usa como salida, $\overline{\mathbf{EOP}}$ se va abajo cuando se alcanza la cuenta final, esto es, cuando el número programado de bytes ha sido transferido. Cuando se usa como entrada, $\overline{\mathbf{EOP}}$ puede ser transferida a un nivel bajo por un dispositivo externo para terminar cualquier operación de **DMA** en curso.

Los canales de **DMA** están priorizados de tal forma que si más de una señal **DREQ** se presenta al mismo tiempo, solo un canal **DMA** a la vez se puede convertir en activo. En la computadora, los canales tienen una prioridad fija, el canal 0 tiene la prioridad más alta y el canal 7 la más baja.[6]

Registros del 8237-5

Para entender cómo se transfieren realmente los datos, necesitamos entender los registros internos del 8237A que se usan para almacenar direcciones de memoria, cuentas de bloque, modos, comandos y estados. Cada canal de **DMA** tiene una palabra de modo de 6 bits y cuatro registros de 16 bits asociados con él, a saber una cuenta y dirección actual, y una cuenta y dirección base. La dirección base (de inicio) y cuenta (número de bytes a transferirse) se envían al **DMA** antes de que tome lugar una operación y automáticamente inicialice la dirección actual y registros cuenta a los mismos valores. Después de que cada byte se transfiere, la dirección actual se incrementa o decrementa (dependiendo de si el bit 5 de la palabra de modo del canal es 0 ó 1), y la cuenta actual se decrementa. La transferencia continua hasta que la cuenta actual llega a cero (cuenta **FINAL**), o hasta que el ´"pin" $\overline{\mathbf{EOP}}$ se pone a "0" por algún dispositivo externo.

Los registros base mantienen los valores iniciales de los registros actuales correspondientes para que cuando la cuenta actual se decrementa a 0 los registros actuales se pueden recargar automáticamente si el bit 4 del registro de modo del canal esta coloca a "1". Los bits 3 y 2 del registro de modo especifican transferencia de lectura (10) o escritura (01), y los bits 7 y 6 especifican uno de cuatro posibles modos de transferencia: demanda (00), individual (01), bloque (10) y cascada (11). Los bits 1 y 0 especifican el número de canal. El registro de estado se puede leer a partir del CPU para determinar qué canales tienen peticiones **DMA** pendientes y cuales canales han alcanzado su cuenta final. Para tazas de transferencia substancialmente menores que un byte por ciclo de máquina (4 periodos de reloj), es deseable el modo de transferencia individual. Esto transfiere un solo byte por activación de la línea **DREQ** del canal. El modo de transferencia de bloque se usa cuando el dispositivo E/S

[6] Juan Ramón Jiménez Alaniz: "Arquitectura del controlador de DMA", en *Aplicación de Microprocesadores e Interfaces,* actualizado por última vez en 03/06, UAM, disponible en: http://akimpech.izt.uam.mx/Web_jr/ami61.htm. Recuperado: 03 de marzo 2009.

puede operar cercanamente tan rápido o más que el controlador **DMA**. Si el dispositivo E/S es un poco más lento que el controlador **DMA**, puede poner la línea **READY** del C. I. 8237 a "1", originando que el C. I. 8237 inserte estados de espera en el ciclo del bus.

Finalmente, hay tres **registros de control** adicionales en el C. I. 8237: un **registro de máscara de canal** que permite a los canales individualmente ser deshabilitados, un **registro de petición de canal** que permite a un programa (en lugar de la línea **DREQ**) iniciar una petición **DMA**, y un **registro de comando**.

El controlador de **DMA** 8237-5 que se usa en las computadoras actuales tiene cuatro canales **DMA**. Dos de los cuales se utilizaron en el diseño de la computadora, el canal 0 se usa en la función de refresco de memoria dinámica de la unidad del sistema, y el canal 2 se usa para transferir datos entre el controlador de disco flexible y memoria. Los canales 1, 2 y 3 están disponibles en el bus para uso de interfaces instaladas en las ranuras de tarjeta del bus. El **BIOS** de la computadora inicializa el **DMA** tal que el canal 0 tiene la más alta prioridad y el canal 3 tiene la más baja.

El 8237-5, una versión del 8237A, tiene 16 direcciones de registro de puerto **E/S** de lectura/escritura que contienen tanto los datos de inicialización como estado del dispositivo. La computadora decodifica al 8237-5 tal que las direcciones de puerto residen en el rango de 0000H a 000FH. Las direcciones de puerto están divididas en dos grupos: las direcciones 0000H a 0007H son registros de lectura-escritura que contienen las direcciones de memoria de inicio de **DMA** para cada canal, dirección de memoria actual para el siguiente ciclo **DMA** en cada canal, y la cuenta byte actual de cada canal. El segundo grupo de direcciones de puerto **E/S**, de 0008 a 000FH, contiene los registros de estado y control que define la operación de cada canal.

La tabla 8.3 muestra la función de cada una de las direcciones en el rango de 0008H a 000FH. Las funciones son diferentes para lectura y escritura, así, típicamente no es posible leer los contenidos de registros sólo de escritura.[7]

Dirección	Función de Lectura	Función de Escritura
0008	Registro De estado	Registro de Comando
0009	No usado	Registro de petición
000A	No usado	Registro bit de máscara individual
000B	No usado	Registro de modo

[7] Juan Ramón Jiménez Alaniz, "Registros del 8237-5" en *Aplicaciones en Microprocesadores e Interfaces*, actualizado por última vez en 03/06, disponible en: http://akimpech.izt.uam.mx/Web_jr/ami62.htm. Recuperación: 03 de marzo de 2009.

000C	No usado	Limpia byte apuntador de flip-flop
000D	Registro temporal	Limpiador maestro
000E	No usado	Limpia registro máscara
000F	No usado	Escribe todos los bits de registro máscara

Tabla 8.3 Direcciones del DMA

8.1.5 Puertos de comunicación

Para comunicarse con el mundo exterior, una Microcomputadora basada en el Microprocesador 8085A, utiliza un puerto paralelo (C.I. 8255) y un puerto serial (C.I. 8251) los cuales explicaremos a continuación.

Interface programable de E/S en paralelo 8255

La interface programable de **E/S** en paralelo (C.I. 8255) es una herramienta muy poderosa y flexible para conectar casi cualquier equipo periférico (impresora, tubo de rayos catódicos, etc.) a una computadora digital basada en el microprocesador 8085 sin la necesidad de lógica externa adicional.

El puerto paralelo 8255 conectado a una microcomputadora usualmente tiene una "rutina de servicio" asociado a él. La rutina maneja el programa de interface entre el dispositivo y la CPU. La definición funcional del C.I. 8255 está programada por la rutina de servicio de **E/S** y representa una extensión del programa. Examinando las características de conexión de los dispositivos de **E/S** para la transferencia de datos, los tiempos y las tablas se puede conformar una palabra de control que fácilmente cumpla las condiciones para inicializar el puerto paralelo 8255 exactamente a una aplicación determinada.

El C.I. 8255 es un dispositivo de **E/S** de propósito general programable, diseñado para usarse con los microprocesadores 8085A (aunque se puede usar casi con cualquier otro microprocesador) de 8 y 16 bits.

Las características de este puerto paralelo son:

Puerto paralelo con 24 terminales o pins de **E/S** que se pueden programar individualmente en dos grupos de 12 y utilizarse en tres modos principales de operación (**Modo 0**, **Modo 1**, y **Modo 2**).

Modo 0

En el modo 0, cada grupo de 12 pins de E/S se puede programar en grupos de 4 para entrada o salida.

Modo 1

En el modo 1, cada grupo se puede programar para tener 8 líneas de entrada o de salida. De las cuatro terminales restantes, tres se usan para señales de protocolo y una de control de interrupción.

Modo 2

En el modo 2 es un modo de Bus Bidireccional y en el cual se utilizan las 8 líneas para bus bidireccional y 5 líneas (solicitando una al otro grupo) para protocolo.

Capacidad de poner y limpiar el bit (**Set/Reset**), y

Capacidad para alimentar 1mA de corriente a 1,5 volts.

Esto permite el manejo directo de transistores Darlington en aplicaciones tales como impresoras y display de alto voltaje.

Descripción funcional del C.I. (PPI 8255)

El C.I. 8255 es un dispositivo periférico de interfaz (conexión) programable (PPI) diseñado para usarse en un sistema de computadora digital basado en el 8085A. Su función es la de un componente de E/S de propósito general para conectar equipos periféricos con el Bus de Datos. La configuración funcional de la 8255 se realiza por la programación del sistema.

Un diagrama a bloques de los componentes internos de la PPI se muestra en la figura 8.3.

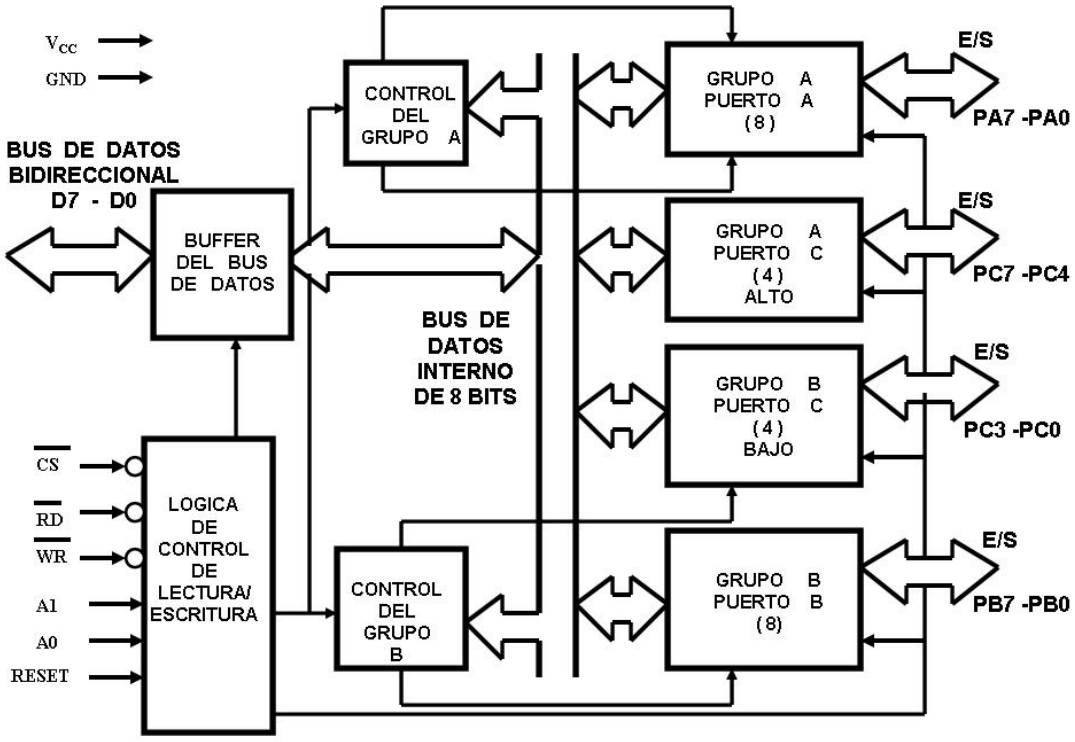


Figura 8.3) Diagrama lógico de la 8255

Líneas de Entrada y de Salida

Las líneas **D0-D7** representan el Bus de Datos Bidireccional que se comunica con el microprocesador 8085.

Las líneas **PA0-PA7**, **PB0-PB7** y **PC0-PC7** representan los buses de los puertos **A**, **B** y **C** de **E/S** que se conectan con los periféricos.

Buffer del Bus de Datos

Este buffer de 3 estados, bidireccional, de 8 bits se usa para unir el C.I. 8255 con el **Bus de Datos** del sistema 8085A. Los datos se transmiten o se reciben por el buffer durante la ejecución de las instrucciones de entrada o de salida por la 8085A. Las **palabras de control e información de estados** también se

transfieren a través del buffer del **Bus de Datos**.

Lógica de Control y de Leer/Escribir Lectura/Escritura

La función de este bloque es manejar todas las transferencias internas y externas de datos, controles y palabras de estados. Acepta entradas desde los buses de dirección y de control de la 8085A para enviar de manera inmediata comandos a los dos grupos de control (**A** y **B**).

$\overline{\text{CS}}$ Selector de Circuito Integrado

Un nivel bajo en esta entrada habilita la comunicación entre el 8255 y el microprocesador 8085A.

$\overline{\text{RD}}$ Lectura

Un nivel bajo en esta entrada habilita al CI 8255 a enviar datos o información de estados al microprocesador 8085A por el **Bus de Datos**. Básicamente, permite "leer" un dato a la 8085A desde la 8255.

$\overline{\text{WR}}$ Escribir/Escritura

Un nivel bajo en esta entrada habilita a la 8085A para escribir datos o palabras de control en la 8255.

A0 y A1 Selección de los puertos

Estas señales de entrada, en combinación con las entradas $\overline{\text{RD}}$y $\overline{\text{WR}}$, controlan la selección de uno de los tres puertos o del "**registro de palabras de control**". Ellos están normalmente conectados a los bits menos significativos del bus de dirección (**A0** y **A1**). La tabla 8.4 muestra la tabla de verdad de la transferencia de datos entre el acumulador y los cuatro registros (**A, B, C** y de **Control**).

A1	A2	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	**Operación de Entrada (Leer)**
0	0	0	1	0	Puerto A → Bus de Datos
0	1	0	1	0	Puerto B → Bus de Datos
1	0	0	1	0	Puerto C → Bus de Datos
					Operación de Salida (Escribir)
0	0	1	0	0	Bus de Datos → Puerto A
0	1	1	0	0	Bus de Datos → Puerto B
1	0	1	0	0	Bus de Datos → Puerto C
1	1	1	0	0	Bus de Datos → Registro de Control
					Función Deshabilitada
X	X	X	X	1	Bus de Datos → Tres estados
1	1	0	1	0	Función ilegal

Tabla. 8.4 Operación Básica del C. I. 8255.

RESET Limpiar

Un nivel alto en esta entrada limpia todos los registros internos incluyendo el "**registro de control**" y todos los puertos (**A**, **B**, **C**) se ponen al modo de entrada.

CONTROLES DE GRUPO A y B

La configuración funcional de cada puerto se comanda por programación. Esencialmente la 8085A envía una palabra de control a la 8255 que contiene información tal como modo de operación, qué puertos son de entrada y cuáles de salida. La figura 8.4 proporcionando la configuración funcional del C.I. 8255. Cada uno de los grupos de control (**A** y **B**) acepta comandos de la lógica de **control de Lectura/Escritura**, reciben parte de la **palabra de control** del bus de datos interno y envían comandos a sus puertos asociados.

Grupo de Control A --- Puerto A y Puerto C Superior (C7-C4)
Grupo de Control B --- Puerto B y Puerto C Inferior (C3-C0)

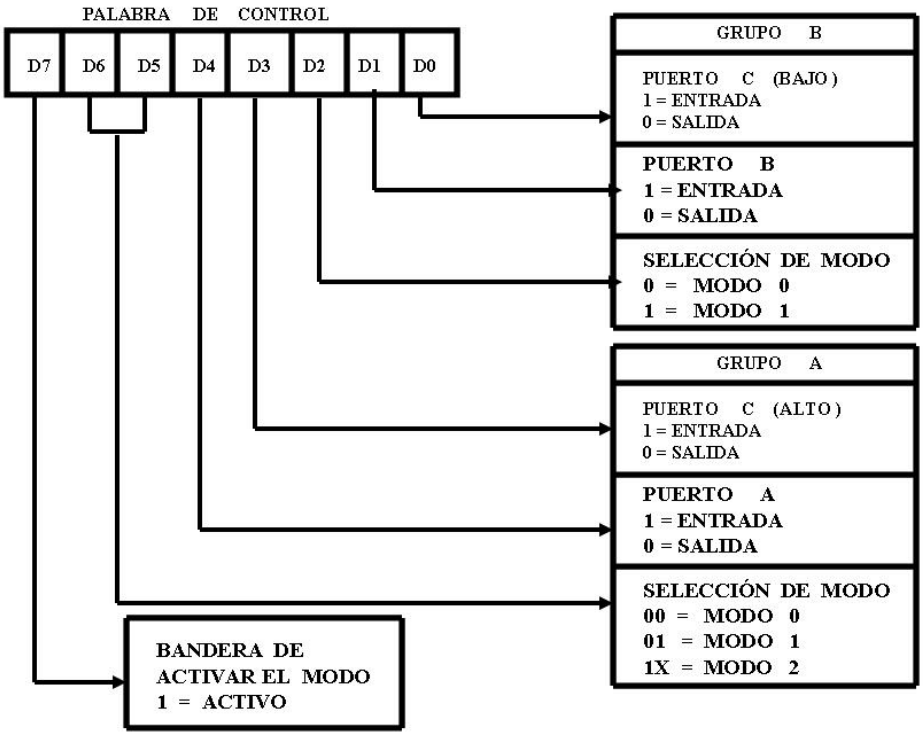


Figura 8.4 Formato de definición del modo

El **registro de control** puede solamente ser escrito, no permite la operación de leer su contenido. Los tres puertos de la 8255 tienen la siguiente configuración.

PUERTO A Consta de un latch/buffer de salida de datos de 8 bits y de un latch de entrada de datos de 8 bits.

PUERTO B Consta de un latch/buffer de entrada/salida de datos de 8 bits y un buffer de entrada de datos de 8 bits.

PUERTO C Consta de un latch/buffer de salida de datos de 8 bits y un buffer de entrada de datos de 8 bits (sin latch para entrada). Este puerto se divide en dos puertos de 4 bits cada uno bajo el control del modo de operación.

La interfaz 8255 trabaja en tres modos principales de operación.

Modo 0 Entrada/Salida básica

Modo 1 Entrada/Salida muestreada (utilizando el pulso **strobe**)

Modo 2 Bus bidireccional

Cuando la entrada **RESET** pasa a nivel alto todos los puertos de la 8255 pasan al estado de alta impedancia (tercer estado). Después de quitar el nivel alto de la entrada **RESET** el C.I. 8255 permanece en el modo de entrada. Durante la ejecución de un programa se puede seleccionar cualquiera de los tres modos de operación simplemente cargando el registro de control con la palabra de control adecuada. La figura 8.4 ilustra cómo definir la configuración de los tres puertos. Para definir la configuración, el bit 7 de la palabra de control debe tener el valor 1. La Figura 8.5 ilustra las definiciones de los puertos con los tres modos de operación.

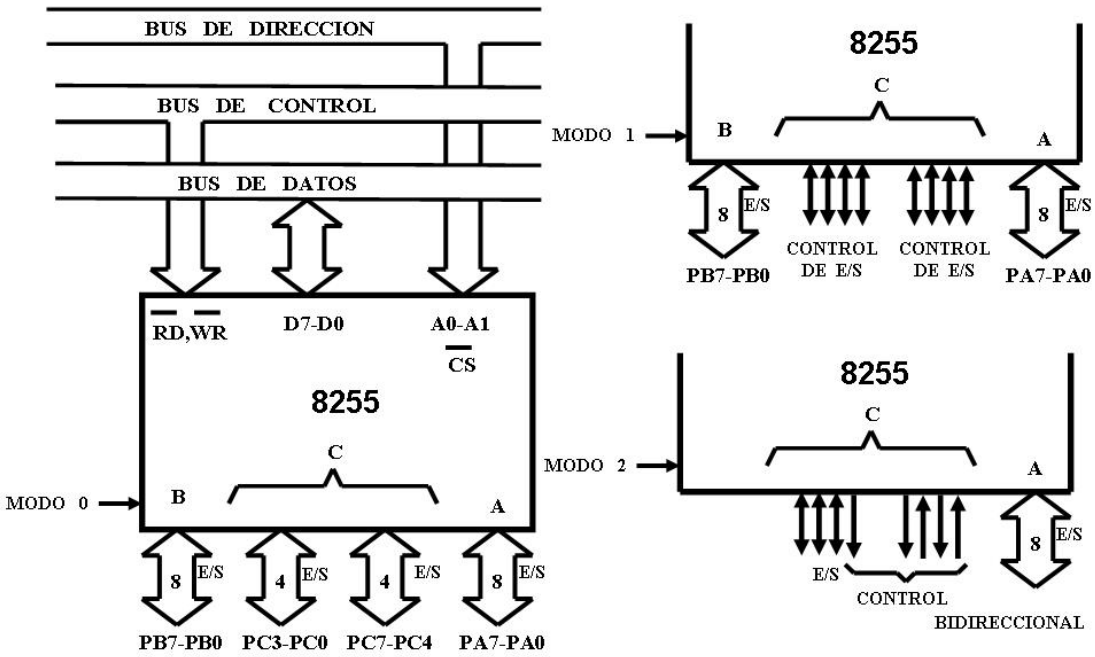


Figura 8.5 Definiciones de modo básico

MODO 0

El **modo 0** de operación proporciona operaciones simples de entrada y salida para cada uno de los tres puertos. No se requiere de un protocolo previo, los datos simplemente se leen de o se escriben en un puerto específico. El C.I. 8255 tiene las siguientes características en este modo:

- Tres puertos de 8 bits
- Cualquier puerto puede ser de entrada o de salida
- Las salidas se almacenan en latches
- Se pueden realizar 16 configuraciones diferentes de **E/S**.
- Las entradas no se almacenan en latches.

Por ejemplo

Se desea programar a los puertos de la 8255 con la siguiente configuración:

Puertos **A** y **C** como entradas y el puerto **B** como salida. Utilizando el **modo 0**.

Palabra de control = 10011001 = 99H

Por ejemplo

Se desea programar a los puertos de la 8255 con la siguiente configuración: Puertos **A** como salida y puertos **B** y **C** como entrada. Utilizando el **modo 0**.

Palabra de control = 10001011 = 8BH

MODO 1 Muestrear E/S

Esta configuración funcional proporciona una forma de transferencia de datos de E/S con un puerto utilizando señales de muestreo (**strobe)** o de protocolo. En este modo los puertos **A** y **B** se definen como puertos de entrada o de salida y el puerto **C** se utiliza para proporcionar las señales de protocolo. La 8255 programada en el modo 1 tienen las siguientes características:

- Dos grupos (**A** y **B**)
- Cada grupo contiene un puerto de datos de 8 bits y un puerto de control (Puerto **C**) de 4 bits
- Los puertos pueden ser de entrada o de salida. Tanto las entradas como las salidas se almacenan en latches
- El puerto de 4 bits de cada grupo se usa para las señales de control y de estados del puerto de 8 bits.

Es este modo de operación algunas líneas del puerto **C** se programan para utilizarse de acuerdo con la configuración de los puertos **A** y **B**, pero las líneas restantes se pueden utilizar como E/S.

MODO 2 BUS BIDIRECCIONAL DE E/S MUESTREADOS

El modo 2 permite al puerto **A** actuar como puerto bidireccional de datos. Se proporciona señales de protocolo a través del puerto **C** (cinco señales) para mantener una disciplina del flujo de datos del periférico. Estas señales de control son una combinación de las señales de control de entrada y de salida del modo 1. La generación de solicitudes de interrupción y de las funciones **Set/Reset** de los flip-flop **INTE** también se encuentra disponible. La 8255 programada en el modo 2 tienen las siguientes características:

- El puerto **A** como puerto bidireccional de datos de 8 bits y el puerto **C** como control con 5 bits

- Las salidas y entradas se almacenan en latches

La figura 8.6 ilustra la palabra de control y la configuración del puerto **A** en el **modo 2.**

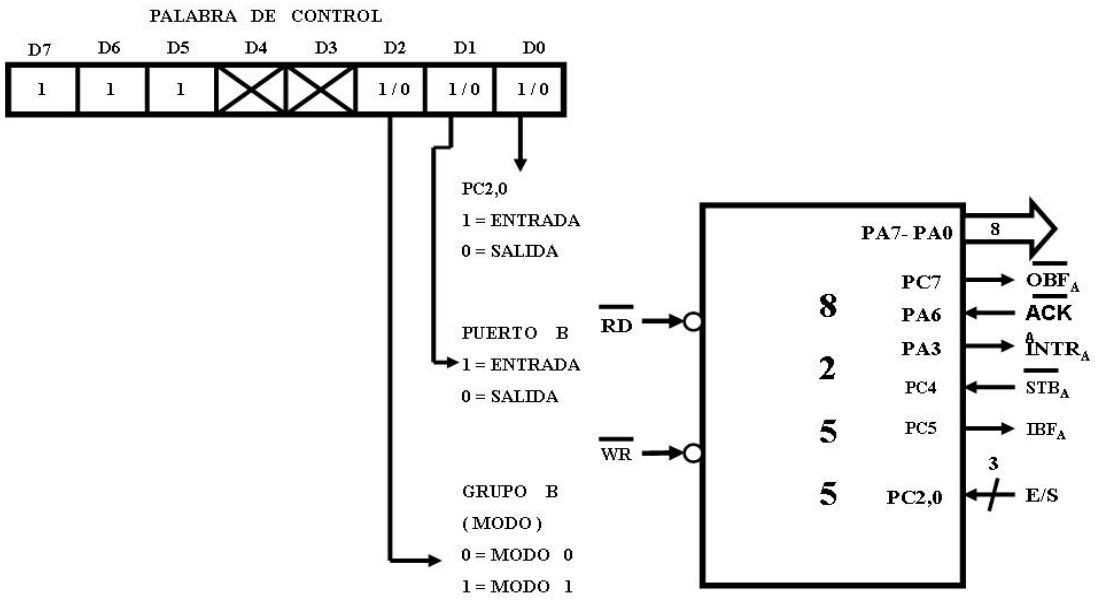


Figura 8.6 Palabra de control y configuración en el Modo 2

El **modo 2** no utiliza las terminales **PC0**, **PC1** y **PC2,** por lo que el puerto **B** se puede programar en el **modo 0** ó en el **modo 1**, la tabla 8.5 ilustra las combinaciones con el **modo 2**.

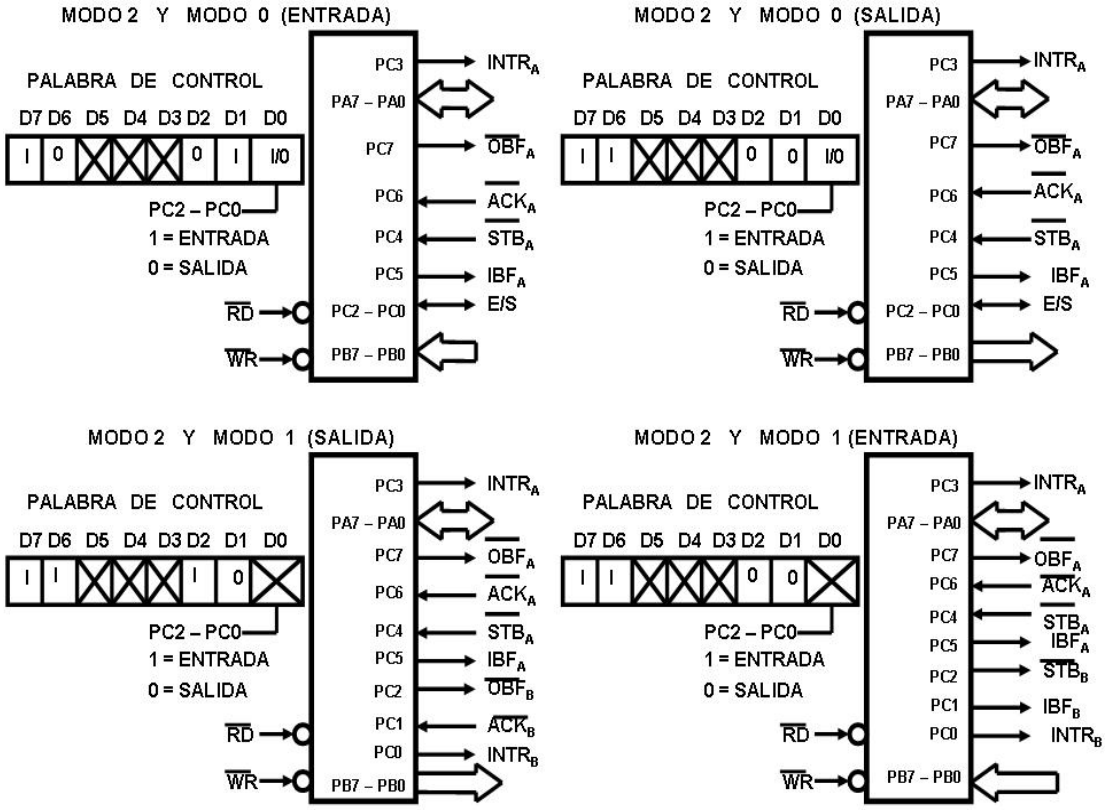


Tabla 8.5 Combinaciones con el modo 2

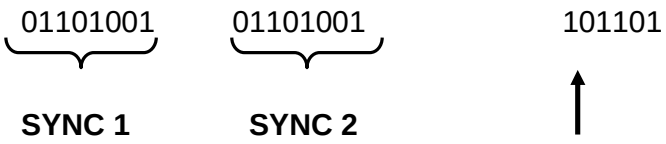
Puerto serial (Protocolo de Entrada y Salida Serie)

Una computadora también permite la comunicación en forma serial y la realiza de dos formas: **síncrona** y **asíncrona**. En esta sección explicaremos el protocolo de Entrada y Salida Serie con lo cual nos servirá para explicar la comunicación en forma serial utilizando el C.I. 8251.

Transferencia síncrona

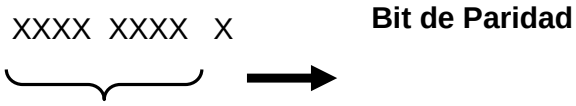
La característica principal de la transferencia de datos en serie en forma síncrona es que la corriente de datos debe ser continua. Habiendo establecido una vez la velocidad de baudios de transferencia de datos serie, el dispositivo transmisor debe transmitir un bit de datos en cada pulso de reloj, y por lo tanto, el dispositivo receptor debe tener la capacidad de interpretar las señales de los datos serie.

En el protocolo síncrono se debe definir la longitud de cada dato y se debe proporcionar al dispositivo receptor con alguna forma de sincronizar los extremos de cada dato. El carácter **SYNC** se usa para este propósito. El carácter **SYNC** es una palabra fija previamente definida. Cada flujo de datos síncronos comienza ya sea con 1 ó 2 caracteres **SYNC**.



Una vez que la transmisión serie síncrona se ha iniciado, la salida del transmisor envía continuamente bits hasta que se termina de enviar el mensaje. Se pueden transmitir datos de 5, 6, 7 y 8 bits.

Los datos en un flujo de datos serie síncrono, usualmente consta de bits de datos sin paridad; pero puede tener un bit de paridad. Enseguida presentamos un ejemplo de una **Unidad de Datos** de 9 bits, 8 bits de dato y 1 bit de paridad:



Un dispositivo receptor entra en el modo **"Hunt"** mientras espera a que comiencen a llegar los Datos Serie Síncronos. Durante la espera el receptor lee continuamente los datos serie que recibe tratando de encontrar en el flujo de datos serie el patrón estándar **SYNC**. Si su protocolo se llama por un solo carácter **SYNC**, entonces el dispositivo receptor comenzará a interpretar los datos en cuanto encuentre un carácter **SYNC**. El protocolo también puede esperar por dos caracteres **SYNC** iniciales, en cuyo caso el dispositivo receptor

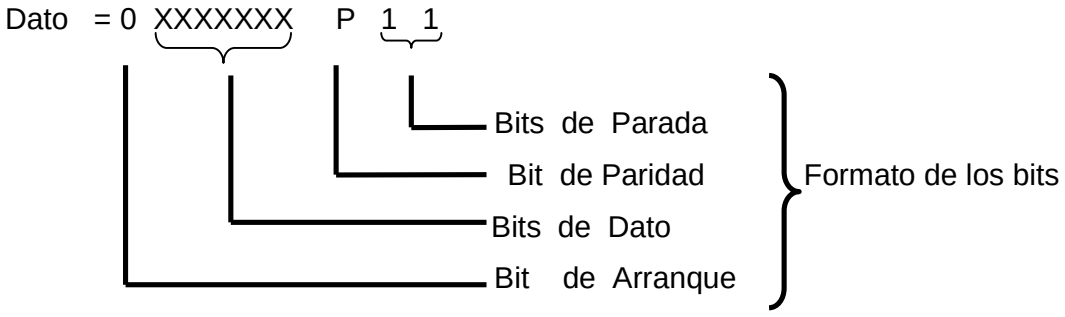
(CPU) no comenzará una interpretación de los datos hasta que haya encontrado 2 caracteres **SYNC** secuenciales.

La transmisión de datos síncronos requiere que el dispositivo transmisor mande los datos continuamente. Si el dispositivo transmisor no tiene datos listos para mandar debe meter relleno con caracteres **SYNC** hasta que el próximo carácter real está listo para transmitir.

Transferencia asíncrona

En la transferencia de datos serie en modo asíncrona, el dispositivo transmisor enviará una señal conocida como "**marca**" (usualmente de nivel alto) mientras no tenga un dato para transmitir. Para indicar que comenzará a transmitir un dato válido el transmisor envía un bit 0, el cual se conoce como señal o bit de arranque. Después del bit de arranque el transmisor envía un dato compuesto de una cantidad predefinida de bits. Para indicar que terminó la transmisión de un dato el transmisor envía una señal conocida como "**señal de parada**". Esta señal de parada puede constar de uno, uno y medio o dos bits con nivel alto.

Cada dato en la transmisión asíncrono tiene el siguiente formato: 1.- Señal de Arranque, 2.) Dato y 3.) Señal de parada. En seguida se ilustra cómo quedan arreglados los bits durante la transmisión



Usar un solo bit de arranque 0 es el más aceptado en los sistemas de microcomputadora. Existe una similitud entre los caracteres **SYNC** del flujo de datos síncrona y los bits del formato de un flujo de datos asíncronos, ambos se requieren para identificar a los datos. La transmisión de la información se

realiza con base en datos de 8 bits, pero pueden ser de 5, 6, y 7 bits.

Para los bits de parada se usan siempre bits 1. Es más frecuente encontrar dispositivos con 2 bits de parada en su formato de datos. Si se tiene 2 bits de parada, entonces cada palabra de 8 bits de datos serie contendrá 12 bits. Si se tiene un bit de parada entonces cada palabra de datos de 8 bits constará de 11 bits.

Algunos protocolos de transmisión especifican uno y medio bits de parada. El ancho de los bits de parada es una y media veces el ancho de un bit normal.

Si el dispositivo receptor no detecta los bits adecuados de Arranque y Parada para cualquier Unidad de Datos, entonces reportará un "error de formato"

Requisitos de una interfaz de comunicaciones

La interfaz de un dispositivo de entrada-salida de comunicación serie se puede visualizar como formado por 3 interfaces:

- Una para comunicarse con la CPU de la computadora,
- Una para la E/S serie externa Síncrona y
- Otra para la E/S serie externa Asíncrona.

"Cada interfaz tendrá, como es usual, línea de datos y señales de control. Para la interfaz E/S serie las señales de control se pueden agrupar en **controles generales** y **controles del módem**. Los controles generales se aplican a cualquier lógica externa, mientras que los Controles del Módem, llenan las necesidades específicas del estándar de la Industria de Módems."[8] A estas interfaces se les conoce como **USART** (**Universal Synchronous/Asynchonous Receiver/Transmitter**) cuando tienen comunicación síncrona y como **UART** cuando solo tienen comunicación asíncrona.

[8] Juan Gilberto Mateos Suárez, "Requisitos de un dispositivo interface de comunicaciones", 9 de mayo de 1998, disponible en: http://proton.ucting.udg.mx/dpto/maestros/mateos/clase/practicas/uart/req.html, Recuperado: 03 de marzo de 2009.

El dispositivo de E/S para comunicación serie debe cumplir con algunas funciones lógicas:

1. Bus de datos
2. Alimentación y reloj
3. Selección de integrado
4. Control de transferencia
5 Terminales para transmisión y recepción
6. Señales de control de E/S serie.[9]

Bus de datos, alimentación y reloj

El dispositivo de interfaz debe contar con terminales para conectarse al Bus de Datos del Sistema para la entrada y salida de datos en paralelo. La transferencia se realiza por medio de un buffer de datos, figura 8.7. El dispositivo necesita las terminales para alimentación y entrada para señales de reloj. Estas señales de reloj no son para la transmisión y recepción de datos, sino únicamente para generar los tiempos internos del dispositivo.

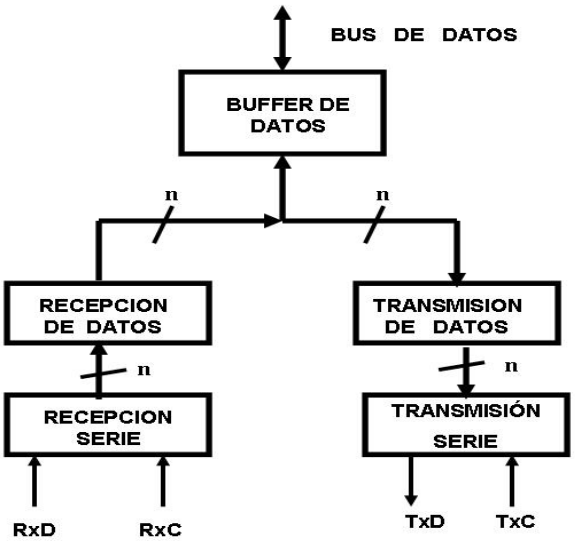


Figura 8.7 Transferencia de datos en forma Serial

[9] Juan Gilberto Mateos Suárez, "Requisitos de un dispositivo interface de comunicaciones", 9 de mayo de 1998, disponible en: http://proton.ucting.udg.mx/dpto/maestros/mateos/clase/practicas/uart/reg.html, Recuperado: 03 de marzo de 2009.

Selección de integrado y control de transferencia

Durante la transferencia de datos, el dispositivo interfaz se considera formado de dos puertos: un puerto para la transferencia de datos, y otro puerto para la transferencia de las palabras de control y de estado.

Para lograr la selección adecuada para transferencia de los datos se utilizan generalmente dos entradas: $\overline{\mathbf{CS}}$ y $\overline{\mathbf{C/D}}$ (Control/Dato). La señal $\overline{\mathbf{CS}}$ permite seleccionar al dispositivo y la señal $\overline{\mathbf{C/D}}$ permite indicar el tipo de dato durante la transferencia. La dirección de la transferencia se indica al dispositivo por medio de las entradas $\overline{\mathbf{RD}}$ y $\overline{\mathbf{WR}}$. La combinación de estas dos entradas con la entrada $\overline{\mathbf{C/D}}$ le indica al dispositivo la interpretación del dato y la dirección de la transferencia.

Terminales de transmisión y recepción

Generalmente se utilizan terminales separadas para la transmisión y recepción de los datos serie. A estas terminales se les dan los nombres de **TxD** (**Transmit Data**) y **RxD** (**Receive Data**) respectivamente. Las frecuencias de transmisión y recepción se proporcionan en dos entradas conocidas como **TxC (Transmit Clock)** y **RxC (Receive Clock)**. Estas señales de reloj se pueden alimentar de la señal de reloj del sistema de la microcomputadora pasándolas por circuitos divisores o tener su propia fuente de reloj. Normalmente se utiliza la misma velocidad para transmitir que para recibir por lo que ambas señales se alimentan de la misma fuente de reloj.

Durante la recepción, las subidas de los pulsos de reloj de recepción leerán el nivel de la línea **RxD** en el buffer de recepción serie. Cuando el buffer de recepción serie contenga la cantidad de bits especificados para formar un dato, su contenido se envía al buffer de recepción de datos. Una vez hecho esto, se comienza a cargar otra vez el buffer de recepción serie, ver figura 8.7.

Durante la transmisión, una vez que la microcomputadora ha cargado un dato en el buffer de transmisión de datos éste pasa al buffer de transmisión serie, el

cual es transmitido en forma serie. Los bits del buffer de transmisión serie se envían en orden ascendente comenzado con el bit 0. Tan pronto como el contenido del buffer de transmisión de datos se deposita en el buffer de transmisión serie, se encuentra listo para recibir otro dato de salida el cual enviará al buffer de transmisión serie en el momento en que éste termine de enviar el último dato serie, figura 8.7.

Señales de control de E/S serie

El buffer de Bus de Datos no se puede usar simultáneamente para recibir bytes de datos ensamblados del registro de recepción serie y para transmitir bytes de datos para desensamblar en el registro de transmisión. La lógica de control y las señales de control que se describen a continuación determinan qué operaciones están ocurriendo en cada momento. El dispositivo de interfaz de E/S serie ignora las señales de reloj si la lógica de control interna no se ha programado para reconocerla, también el contenido del buffer de recepción de datos se perderá si el buffer datos no está lista para recibir un byte ensamblado.

La lógica de transmisión necesita dos señales de control, para indicar que el buffer de transmisión serie está vacío y la otra para indicar que el buffer de transmisión de datos está listo para recibir otro byte de datos. Estas dos señales se llaman **TE** (**Transmit Empty**) y **TRDY** (**Transmit Ready**). Las dos señales tienen las siguientes características: cuando los datos serie se están transmitiendo en modo asíncrono, **TE** tendrá nivel bajo mientras la salida **TxD** está transmitiendo el dato del buffer de transmisión serie; sin embargo, **TRDY** será bajo para indicar que el buffer de transmisión de datos se encuentra listo para recibir otro byte de datos, aun cuando un dato se está actualmente enviando.

La lógica de recepción usa únicamente la señal **RRDY** (**Receiver Ready**). Esta señal dice a la CPU que se ha cargado un byte de datos en el buffer de datos y que puede leerse.

Control de la interface de E/S serie

Dadas las muchas opciones de la interface de un dispositivo de **E/S** serie se necesita de un **Registro de Control** para seleccionar las opciones y en algunos casos para determinar las configuraciones de las señales de control que se están enviando.

Primero se debe seleccionar el tipo de **E/S** serie: síncrono o asíncrono. En la tabla 8.6 se identifican los parámetros que se deben seleccionar bajo control del programa para cada tipo. En la tabla 8.6, se muestran los parámetros del Modo, los que usualmente no se cambian durante el transcurso de operación de E/S serie. Algunas veces se llama **E/S** síncrona a la transferencia de E/S asíncrona que usa un reloj x1.

Después de definir los parámetros del modo seleccionado, la interfaz de **E/S** serie recibirá más información de los comandos de control. Los comandos deben identificar la dirección del flujo de datos serie (transmisión o recepción) o terminar las operaciones actuales permitiendo que modifique el modo para la próxima transmisión.

Función	Asíncrona	Sincronía
Frecuencia de reloj	Razón de Bauds X1,x16 ó x64	Usualmente se usa x1
Bits de datos por byte	5, 6, 7 u 8	5, 6, 7 y 8
Paridad	Par, Impar o Ninguna	Par, Impar o Ninguna
Bits de Parada	1, 1.5 ó 2	No se Aplica
Caracteres SYNC	No se aplica	1, 2 ó SYNC Externo

Tabla 8.6 Parámetros de E/S serie

Condiciones de error de E/S serie

Las señales de entrada cuyos niveles deben ser posibles de leer son:

DSR Listo el dispositivo de datos

CTS Listo para enviar

Esta señal algunas veces no se incluye en el registro de estados; la lógica de la interfaz de **E/S** serie, entonces, debe esperar automáticamente por la señal **CTS** en alto antes de iniciar una transferencia serie de datos.

SYNC Sincronización externa

TxE El buffer transmisor vacío

TRDY El buffer transmisor listo para recibir dato de la CPU

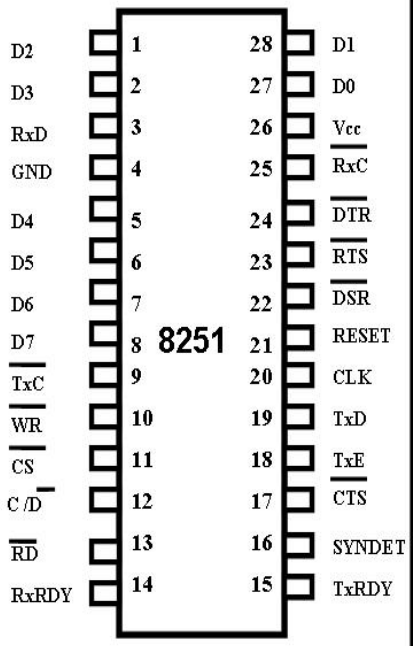
RRDY El buffer receptor listo para mandar datos a la CPU

Esta señal puede estar conectada a la lógica de interrupción y dejarla fuera del registro de estados.

Normalmente una condición de error no hace que la interface de un dispositivo de **E/S** serie aborte las operaciones. El error se reporta en el "**Registro de Estados**" y las operaciones continúan.

Interface programable de comunicación en serie 8251

El C.I. 8251 es un dispositivo **USART** de 28 terminales, el cual requiere una alimentación de +5 Volts y todas sus salidas y entradas son compatibles con TTL. El C.I. 8251 acepta caracteres de datos paralelos de la CPU y los convierte en un flujo de datos serie para transmisión. Simultáneamente puede recibir un flujo de datos serie y convertirlos en caracteres de datos paralelos para la CPU. Las terminales y señales de control del C.I. 8251 se muestran en la figura 8.8.



Pins	FUNCION
D7-DO	BUS DE DATOS (8 BITS)
C/ D	CONTROL O DATO SE VA A ESCRIBIR O LEER
RD	LEER DATO
WD	ESCRIBIR DATO O CONTROL
CS	HABILITAR INTEGRADO
CLK	PULSO DE RELOJ
RESET	LIMPIAR
TxC	RELOJ DE TRANSMISION
TxD	TRANSMISOR DEL DATO
RxC	RELOJ DE RECEPCION
RxD	RECEPTOR DEL DATO
RxRDY	RECEPTOR LISTO
TxRDY	TRANSMISOR LISTO
DSR	LISTO PARA RECIBIR
DTR	TERMINAL DE DATOS LISTA
SYNDET	DETECTOR DE SYNDET
RTS	SOLICITUD PARA ENVIAR DATOS
CTS	LIMPIAR PARA ENVIAR DATOS
TxE	TRANSMISOR VACIO
Vcc	+5 Volts
GND	TIERRA

Figura 8.8 a.) Configuración del C.I 8251

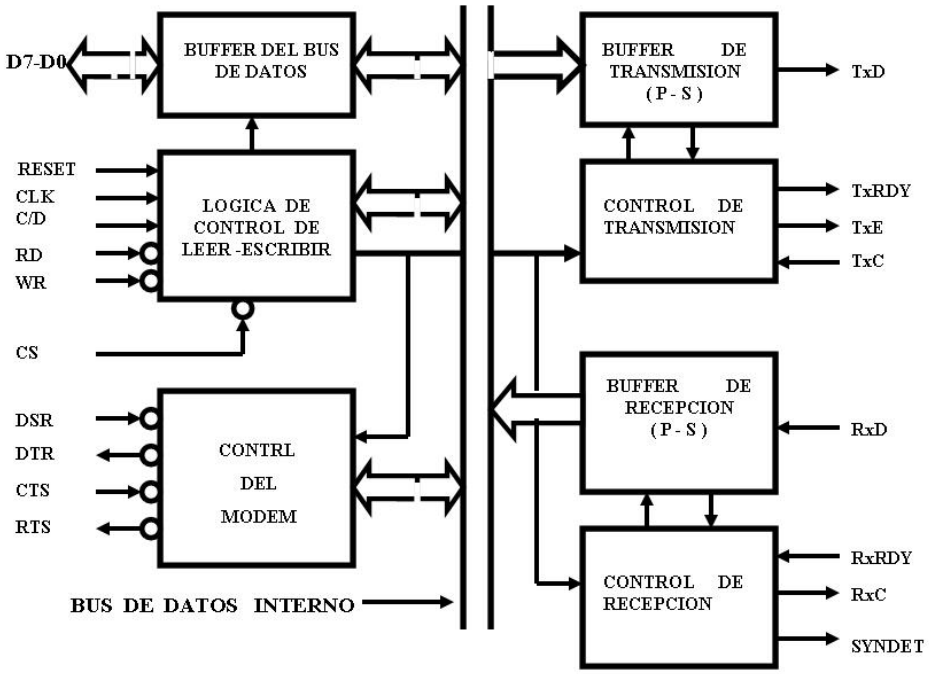


Figura 8.8 b.) Configuración del C.I. 8251 (Continuación)

Las señales se pueden dividir en cuatro categorías:

1. Control e interface con la CPU
2. Entrada serie
3. Salida serie
4. Control del módem

La figura 8.9a) muestra la comunicación entre el C.I. 8251 con los buses de datos de la 8085A. Se considerará primero las señales de control e interfaz con la CPU.

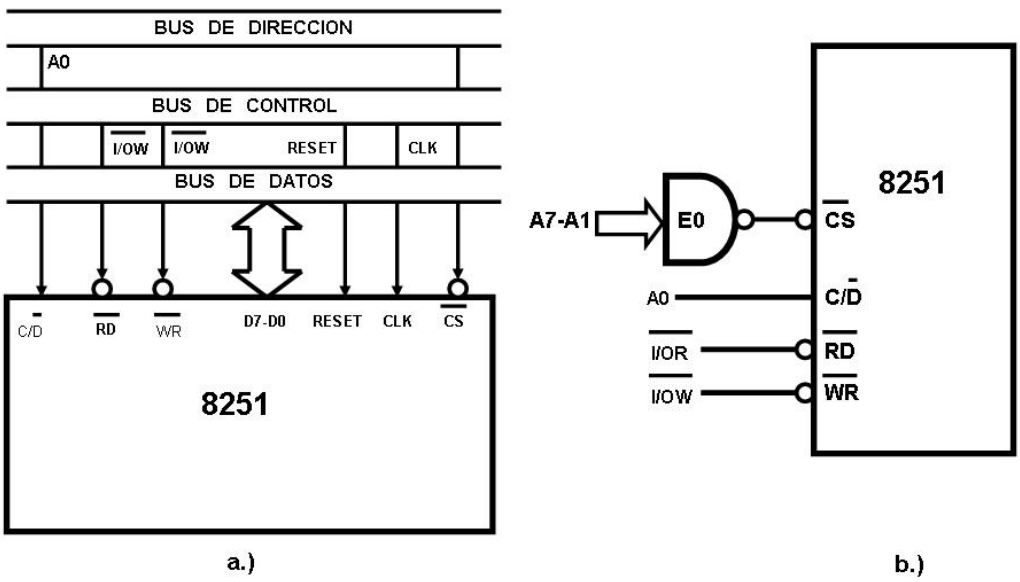


Figura 8.9 Interface del Puerto Programable Serial

a.) Conexión del 8251 al Bus del Sistema 8085A

b.) Señales de Control y Selección

Terminales D7-D0

Los terminales **D7-D0** se comunican al bus de datos de 8 bits. Cuando la CPU envía un dato paralelo de 8 bits al C.I. 8251, éste puede ser un dato al periférico o una palabra de control y/o de estado. El C.I. 8251 convierte los bytes de datos en un flujo de datos serie. Una palabra de control previamente almacenada define el protocolo que debe cumplir en la transmisión.

El C.I. 8251 se conecta con el microprocesador 8085A como dos puertos. Un puerto se utiliza para la transferencia de datos y el otro para la transferencia de las palabras de control y de estado. La lógica de selección consiste de dos entradas:

$\overline{\mathbf{CS}}$ y $\overline{\mathbf{C/D}}$

Estas señales se combinan con los valores de las entradas $\overline{\mathbf{RD}}$ y $\overline{\mathbf{WR}}$ para indicar la dirección de la transferencia del dato y la interpretación del dato. La tabla 8.7 muestra las combinaciones de estas cuatro señales y la figura 8.9

ilustra la configuración para la selección de la interface. Las líneas de dirección **A7-A1** habilitan a la entrada $\overline{\mathbf{CS}}$ cuando tienen el valor **EOH** (en este caso el dígito **0** es de 3 bits) y el valor de la línea A0 indicará el tipo de dato de la transferencia.

$\overline{CS}$	C / $\overline{D}$	$\overline{RD}$	$\overline{WR}$	TRANSFERENCIA
0	0	0	1	8251 ⟶ Bus de Datos
0	0	1	0	Bus de Datos ⟶ 8251
0	1	0	1	Estado ⟶ Bus de Datos
0	1	1	0	Bus de Datos ⟶ Control
1	X	X	X	Bus de datos ⟶ Tercer Estado

Tabla 8.7 Selección del tipo de transferencia

Cuando la señal $\overline{\mathbf{I/OR}}$ tiene nivel bajo la CPU está leyendo un dato o la palabra de estado y cuando la señal $\overline{\mathbf{I/OW}}$ tiene nivel bajo la CPU está enviando un dato o una palabra de control.

El C.I. 8251 tiene dos señales adicionales: **RESET** y **CLK**. La señal **RESET** es una señal del sistema que cuando tiene nivel alto, obliga al C.I. 8251 al estado inactivo. En este estado, se limpian todos los controles previamente definidos por lo que la CPU debe definir en las siguientes instrucciones el tipo de operación que desarrollará a continuación. Esto se logra con las palabras de control que se describirán enseguida.

La señal de **CLK** es una señal de entrada de reloj. Esta entrada de reloj no controla la velocidad de transmisión o recepción de los datos serie, se utiliza únicamente para los tiempos internos del **USART**. Pero debe ser por lo menos 30 veces la velocidad de transmisión o recepción en el modo asíncrono. Debido a especificaciones eléctricas del C.I. 8251, la señal **CLK** debe ser mayor que 0.47 MHz y menor que 2.38MHz.

Transferencia de datos y del control

El C.I. 8251 cuenta con varios buffers a través de los cuales fluyen los datos. Este flujo de datos se ilustra en la figura 8.10. Los datos serie de entrada se reciben y se ensamblan en unidades de datos de 8 bits en el buffer **RB**. Una vez que se ha ensamblado el dato en el buffer de **RB**, a continuación se transfieren al buffer **RA**. La CPU puede llevar el contenido del buffer **RA** ejecutando una instrucción **IN XXH** (donde **XXH** es el código de selección para datos, $\overline{CS}$ y **C/D(=0)**. Mientras la CPU lee el contenido del buffer **RA**, el próximo dato serie se puede estar ensamblado en el buffer **RB**. Por "ensamblar" se entiende al hecho de recibir 8 bits en serie por medio de una línea y almacenarlos en un registro para tenerlos disponibles en paralelo.

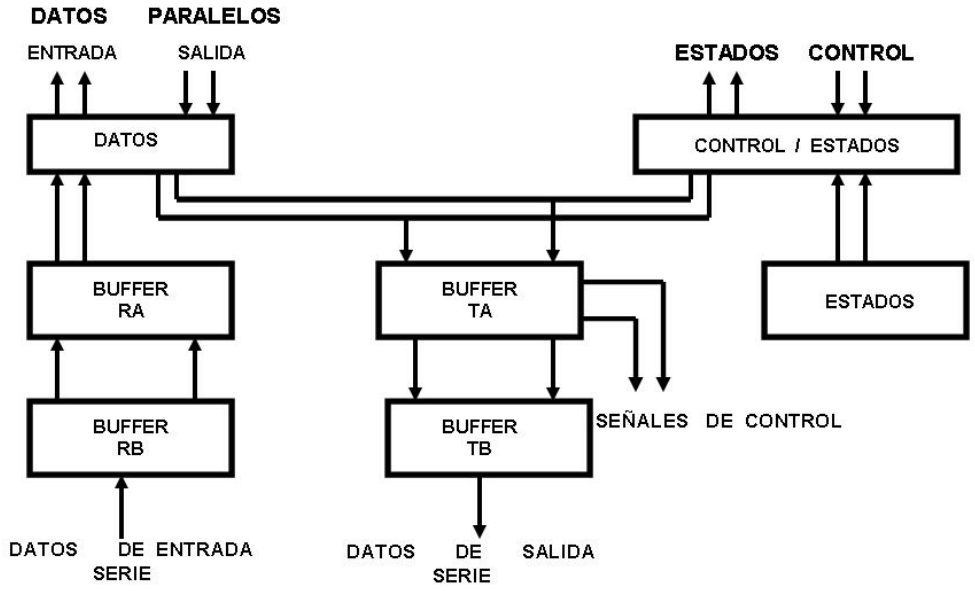


Figura 8.10 **Diagrama del flujo de datos**

Cuando el siguiente byte se ha terminado de ensamblar en el buffer **RB**, se desplazará al buffer **RA** borrando el contenido anterior de **RA**. Si el contenido anterior **RA** no se ha leído, se registra una bandera de error en el registro de estados (error de atención-**overrum)**.

Los datos de salida que envía la CPU se reciben en buffer **TA**, si el buffer **TB** está vacío, el contenido de TA se desplaza a **TB** de donde son enviados en forma serie al periférico de salida conectado a la 8251. Mientras el buffer **TB** está enviando el dato, la CPU puede cargar el siguiente dato en **TA** tan pronto el buffer **TB** se vacíe el dato nuevo se desplaza de **TA** y **TB**. Si el buffer **TA** está vació cuando el buffer **TB** termina de enviar el dato serie, el C.I. 8251 inserta caracteres **SYNC** en el buffer **TB** si se está trabajando en modo síncrono. En cambio, si se está trabajando en modo asíncrono, la línea de transmisión toma el nivel de "marca".

Las palabras de control también se reciben en el buffer **TA** pero ahora no pasan al buffer **TB** sino que su función es la de modificar la lógica de trabajo del C.I. 8251 para satisfacer el comando de control ordenado. El 8251 no tiene un buffer dedicado para la palabra de control pero cuando se reciben en el buffer **TA** alteran la lógica de funcionamiento del **USART**.

Control de transmisión asíncrona

El dato serie se envía por la terminal **TxD**. La velocidad de transmisión es controlada por la señal de reloj que alimenta la entrada **TxC.** La señal de reloj **TxC** puede o no ser derivada del reloj del sistema de Microcomputadora. El valor real de la velocidad de transmisión en el modo asíncrono puede ser 1, 1/16 o 1/64 del reloj **TxC**. Las transiciones alto-bajo del reloj **TxC** transfieren los bits del dato. Existen tres señales de control asociadas con la lógica de transmisión las cuales son: **TxRDY**, **TxE** y **RxRDY**.

Durante la transmisión se generan dos señales **TxRDY** y **TxE**. La señal **TxRDY** pasa a nivel alto tan pronto como el contenido del buffer **TA** se ha desplazado al buffer **TB** por lo que **TA** puede cargarse con el nuevo dato de salida. El estado de esta señal estará disponible en la salida **TxRDY** únicamente cuando el 8251 esté habilitado para transmitir, esto es, cuando $\overline{\mathbf{CTS}}$ esté en bajo y **TxE** en alto. Sin embargo, el bit **TxRDY**, figura 8.11, del **Registro de Estado** se pone siempre que el buffer **TA** esté vacío.

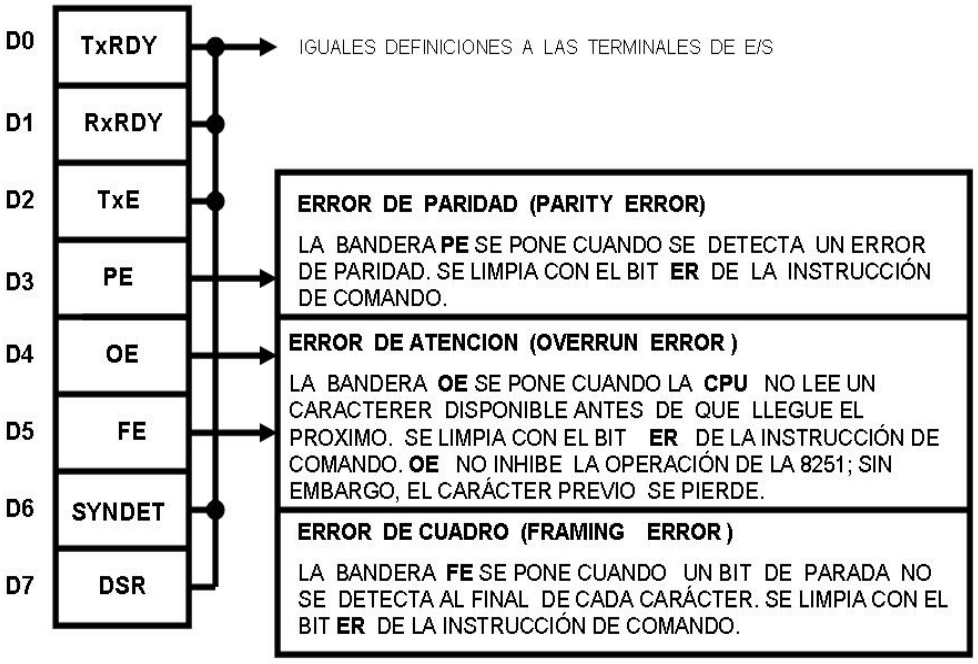


Figura 8.11 Formato para Lectura de los Estados

La salida **TxE** se pone en alto tan pronto como el dato en **TB** se ha enviado al periférico y permanece en alto mientras no se desplace un dato nuevo de **TA** o **TB**. La figura 8.12 ilustra el diagrama de tiempos durante la transmisión asíncrona de datos serie.

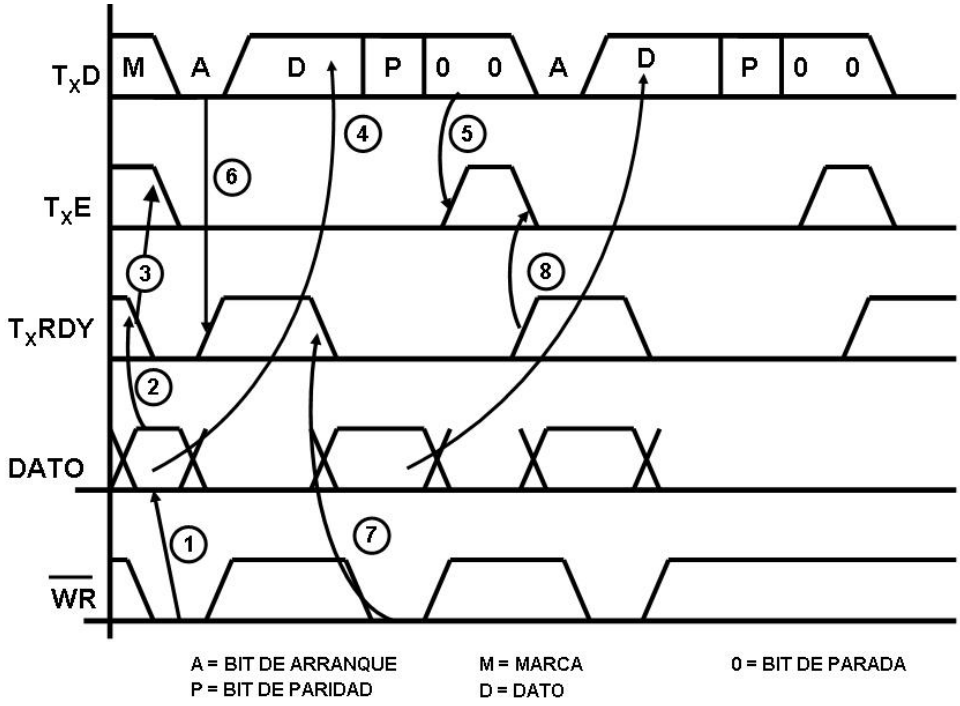


Figura 8.12 Diagrama de tiempos de transmisión asíncrona

La secuencia de los eventos se describe a continuación. Inicialmente se encuentra una señal de marca en la salida **TxD.**

1. El dato de salida se carga en **TA** con la señal $\overline{\boldsymbol{WR}}$.

2. Al cargarse **TA** con el dato, la señal **TxRDY** pasa a nivel bajo.

3. Como **TxE** está en alto, **TB** se encuentra vacío; por lo que el contenido de **TA** se transfiere a **TB** enviando a **TxE** a nivel bajo.

4. Al dato ahora se envía en forma serie al periférico desde **TB**.

5. Cuando **TB** termina de enviar el dato en forma serie al periférico, lo cual se identifica con los bits de parada, **TB** queda vacío y **TxE** pasa a nivel alto.

6. Sin embargo, tan pronto como el contenido de **TA** se transfirió a **TB** (por lo que **TB** comienza a transmitir), **TA** queda vacío. Éste ordena que **TxRDY** regrese a nivel alto indicando que puede recibir el siguiente dato. Esto se registra en la bandera de estado **TxRDY** de donde es leído por la CPU para conocer cuándo puede enviar el siguiente dato.

7. El siguiente dato se carga en **TA** con la señal $\overline{\mathbf{WR}}$.

8. Cuando **TxE** pasa a nivel alto **TA** se encuentra un dato esperando. Este dato se transfiere inmediatamente a **TB**, enviando a **TxE** a nivel bajo hay a **TxRDY** a nivel alto.

9. El dato nuevo es ahora enviando en forma serie.

El bit **RxRDY** en el registro de estado se pone cuando el buffer **TA** tiene un dato para la CPU.

La terminal de **TxRDY** se usa frecuentemente para generar solicitud de interrupción. Cuando la velocidad de salida de datos no es muy crítica se puede preguntar a través del **Registro de Estado** por el bit **TxRDY** para determinar cuando la salida **TxRDY** tiene nivel bajo y se puede enviar otro dato.

Control de recepción asíncrona

Los datos serie se reciben en la terminal $\overline{\mathbf{RxD}}$. Los bits de los datos son muestreados por las señales de reloj $\overline{\mathbf{RxC}}$, las cuales en forma semejante a **TxC** usualmente se derivan del reloj del sistema de microcomputadora. Las transiciones bajo-alto del reloj $\overline{\mathbf{RxC}}$ leen los bits de los datos en el buffer **RB**.

La lógica de recepción utiliza la señal de control **RxRDY**. Esta salida toma el nivel alto en el momento en que el buffer **RB** envía el dato recién recibido al buffer **RA**, este nivel indica a la 8085A que tiene un dato disponible para ella. Si

la 8085A no lee el contenido de **RA** antes de **RB** ensamble el siguiente dato para la **RA**, existirá un error de atención. El dato de **RB** se pierde y este hecho se reporta en el **Registro de Estado** bit **D4**.

La figura 8.13 ilustra el diagrama de tiempos en la recepción asíncrona de datos serie. La secuencia de los eventos se describe a continuación. Partimos del hecho de que inicialmente se encuentra una señal de marca en la entrada **RxD**.

1. Se ensambla un dato en **RB**.
Tan pronto como el dato se ensambla y se transfiere a **RA**.

2. Cuando **RA** recibe el dato, la señal **RxRDY** pasa a nivel alto.

3. Después de transferir el dato a **RA, RB** puede comenzar a ensamblar el siguiente dato, si lo hubiera. La CPU debe ejecutar una instrucción para leer el dato en **RA** después de censar que el valor de la bandera **RxRDY** es 1.

4. Cuando la CPU lee el dato en **RA**, la señal **RxRDY** pasa a nivel bajo.

5. Después de que **RB** ensambla el dato nuevo, lo transfiere a **RA** enviando de nuevo a nivel alto la señal **RxRDY**.

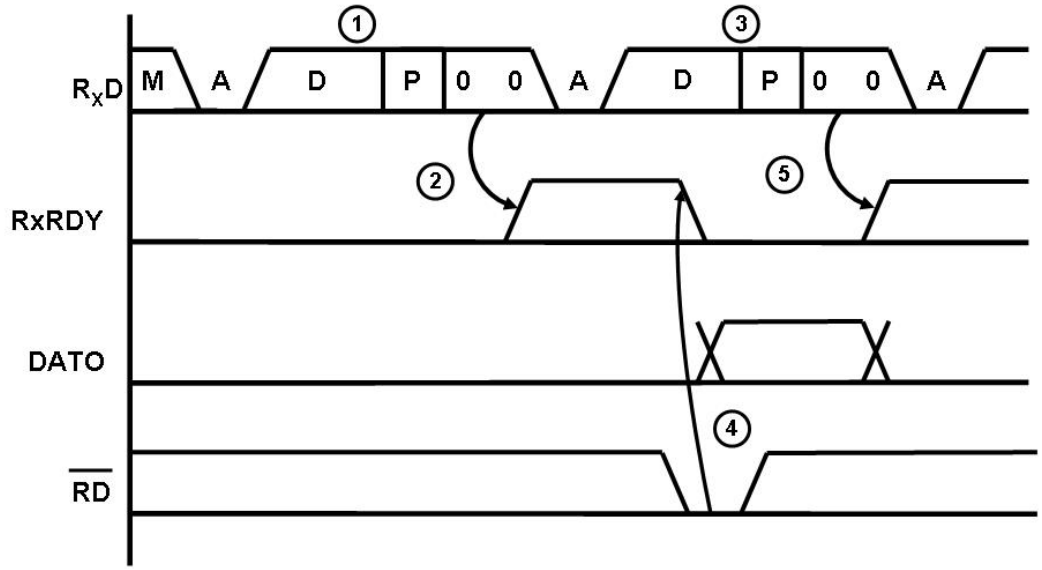


Figura 8.13 Diagrama de tiempos de recepción asíncrona

Control de transmisión y recepción síncrona

El diagrama de tiempo de la transmisión síncrona es esencialmente el mismo que el diagrama de tiempos de la transmisión asíncrona. La única diferencia se refleja en el protocolo, los datos no utilizan bits de arranque ni de parada. En lugar, un carácter **SYNC** precede a los datos. También inserta carácter **SYNC** en medio del flujo de datos cuando no hay dato listo para ser transmitido. El diagrama de tiempos de la recepción síncrono es esencialmente el mismo que el diagrama de tiempos de recepción asíncrono. Otra vez, la única diferencia es el protocolo.

Cuando el C.I. 8251 está en modo de recepción síncrona, inicialmente espera por uno o dos caracteres **SYNC** al comienzo del flujo de datos. Para detectar caracteres **SYNC** la 8251 debe estar en el modo **Hunt**. Para esto se debe enviar una palabra de control apropiada a la 8251, (ver figura 8.14, bit **D7**). En el modo **Hunt**, los datos que llegan a **RB** se comparan con el carácter **SYNC**.

Cuando un dato es igual al carácter **SYNC**, la 8251 deja el modo **Hunt** y comienza a interpretar como datos de información los siguientes bits.

La señal **SYNDET** toma el nivel alto después de que la entrada **RxD** recibe uno o dos caracteres **SYNC** al comienzo del flujo de datos, según se haya programado.

Cuando se transmiten caracteres **SYNC** en medio del flujo de datos, el C.I. 8251 los recibe y no los elimina del flujo de datos. Sin embargo, la salida **SYNDET** pasa a nivel alto identificando al carácter **SYNC** de tal manera que la CPU lo pueda descartar por programa. La señal **SYNDET** pasa a nivel bajo con la lectura del **Registro de Estado**.

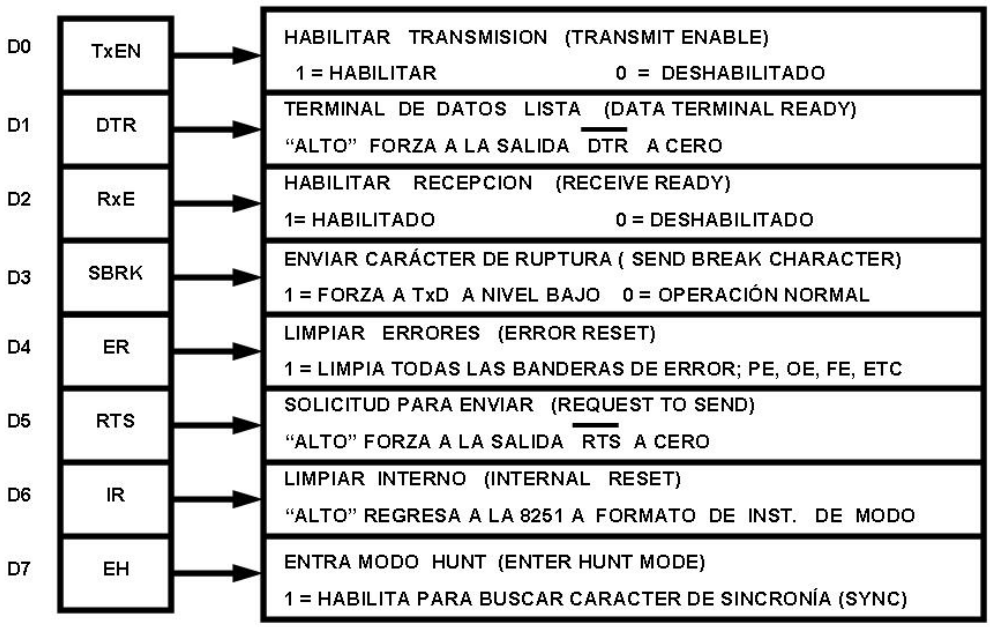


Figura 8.14 Formato de Instrucción de Comando

Control del módem

Las señales de control del módem son estándar. El C.I. 8251 utiliza la salida $\overline{\mathbf{DTR}}$ para indicar que se encuentra listo, y la entrada $\overline{\mathbf{DSR}}$ la usa para probar el estado en que se encuentra el módem. Una vez que la 8251 y el módem se encuentran listos, la transmisión se inicia por la 8251 enviando una solicitud para transmitir al módem por medio de $\overline{\mathbf{RTS}}$ y la entrada $\overline{\mathbf{CTS}}$ la utiliza el módem para indicarle al C.I. 8251 que inicie la transmisión.

Descripción de operación de la 8251

La definición funcional completa del C.I. 8251 se realiza por programación. Se debe enviar dos palabras de control por la CPU para inicializar el C.I. 8251 para soportar el formato de comunicación deseado. Estas palabras de control programarán:

- La tasa de transmisión
- La longitud del carácter
- El número de bits de parada
- La operación síncrona o asíncrona
- La paridad par o impar, etc.

En el modo síncrono también se proporciona la opción para seleccionar caracteres de sincronización interna o externa.

Una vez programada, el C.I. 8251 está listo para ejecutar las funciones de comunicación. La salida **TxRDY** pasa a nivel alto para indicar a la CPU que el C.I. 8251 está listo para recibir un carácter. Esta salida **TxRDY** se limpia automáticamente cuando la CPU escribe un carácter en el C.I. 8251. Por otro lado, el 8251 recibe datos serie desde un módem o un dispositivo de entrada/salida, al recibir un carácter completo la **RxRDY** pasa a nivel alto para indicar a la CPU que el C.I. 8251 tiene un carácter completo listo para que la CPU lo atrape. La salida **RxRDY** se limpia automáticamente al efectuarse la operación de lectura por la CPU.

El C.I. 8251 no podrá comenzar la transmisión sino hasta que el bit **TxEN** (habilitar el transmisor) no se haya programado a nivel alto con la instrucción de comando (figura 8.14) y haya recibido una entrada "listo para transmitir" (**CTS**). La salida **TxD** se mantendrá en el estado de marca después de limpiar (**Reset**) a la 8251.

Programación del C.I. 8251

Antes de comenzar la transmisión o recepción de datos, el C.I. 8251 se debe cargar con un conjunto de palabras de control generados por la CPU. Estas señales de control proporcionan la definición funcional completa del C.I. 8251 y deben seguir inmediatamente a la operación **Reset** (interna o externa).

Las palabras de control se dividen en dos formatos:

1.- Instrucción de Modo

2.- Instrucción de Comando

La figura 8.15 ilustra la secuencia de la instrucción de Modo y de Comando.

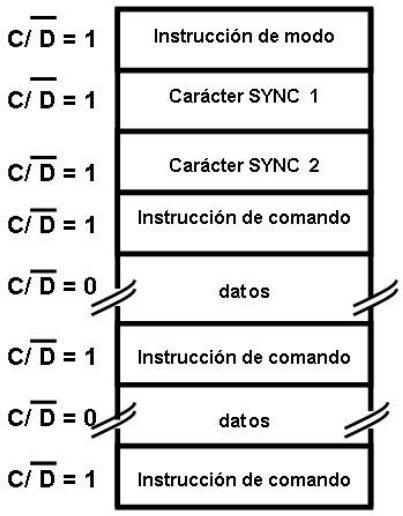


Figura 8. 15 Secuencia de la programación de las instrucciones

Instrucción de modo

Este formato define las características operacionales generales del C.I. 8251. Deberá seguir a una operación **Reset** (interna y externa). Una vez que la **instrucción modo** se ha escrito en el C.I. 8251 por la CPU se pueden insertar **instrucciones de comando** o caracteres **SYNC**.

Instrucción de comando

Este formato define una palabra de estado que se usa para controlar la operación actual del C.I. 8251. Las instrucciones de comando y de modo deben conformarse a una secuencia específica para la operación adecuada del dispositivo. La **instrucción de modo** se debe insertar inmediatamente siguiendo a una operación **Reset** antes de usar el C.I. **8251** para la comunicación de datos.

Todas las palabras de control escritas en el C.I. 8251 después de la **instrucción de Modo** se cargarán como **Instrucciones de Comando**. Las **Instrucciones de Comando** se pueden escribir en el C.I. 8251 en cualquier tiempo con el bloque de datos durante una operación del C.I. 8251 para regresar al formato de la **instrucción de Modo**, un bit (bit 6) en la palabra de **Instrucción de Comando** se puede poner a uno para iniciar una operación **reset** interna o lo cual automáticamente fija de nuevo el C.I. 8251 en el formato de la **Instrucción de Modo**. Un **reset** externo del sistema de microcomputadora se puede utilizar para realizar la misma función. La **Instrucción de Comando** debe seguir a la **Instrucción Modo** o a un carácter **SYNC**.

En el C.I. 8251 se puede utilizar para comunicación de datos síncrona o asíncrona. Para entender cómo la **Instrucción de Modo** define la operación funcional del C.I. 8251 el diseñador puede visualizar mejor al dispositivo como dos componentes separados compartiendo el mismo paquete. Uno síncrono y el otro asíncrono. La definición del formato se puede cambiar sobre la marcha, pero con el propósito de explicación los dos formatos se aislarán.

Modo asíncrono (transmisión)

Siempre que la CPU envía un carácter de dato, el C.I. 8251 le agrega automáticamente a este dato un bit de arranque (**START**, de nivel bajo) y un número programado de bits de parada (**STOP**). También, un bit de paridad par o impar se inserta antes del bit de parada, según se defina en la instrucción de modo. El carácter es entonces transmitido como un flujo de datos serie en la salida TxD. El dato serie se recorre con el borde de bajada de **TxC** a una razón igual a 1, 1/16, ó 1/64 de aquella de **TxC**, como se ha definido por la instrucción de modo.

Modo asíncrono (recepción)

La línea **RxD** está normalmente en alto. Un borde de bajada sobre esta línea, dispara el comienzo de un bit de arranque. La validez de este bit de arranque

se verifica nuevamente muestreándolo (**strobing**) en su centro nominal. Si un nivel bajo se detecta nuevamente, es un bit de arranque válido y el contador de bits empezará a contar. El contador de bits localiza el centro de los bits de datos, del bit de paridad (si existe) y de los bits de parada. Si ocurre un error en la paridad, la bandera de error de paridad se pone a uno. Los bits de datos y paridad se muestrean en el pin **RxD** con el borde de subida de **RxC**. Si se detecta un nivel bajo como el bit de parada, la bandera de error de marca se pondrá a uno.

El bit de parada señala el final de un carácter. Este carácter se carga enseguida en el buffer de E/S paralelo del C.I. 8251. La terminal **RxRDY** pasa a alto para señalar a la CPU que un carácter está listo para ser atrapado. Si un carácter previo no ha sido atrapado por la CPU, el carácter presente lo reemplaza en el buffer de E/S y la bandera de **OVERRUM** se levanta (el carácter previo se pierde). Todas las banderas de error se pueden limpiar con una **Instrucción de Comando**. La ocurrencia de cualquiera de estos errores no detiene la operación del C.I. 8251.

La figura 8.16 muestra el formato de la palabra de **Instrucción de Modo Asíncrono**. Los bits **D0** y **D1** permiten seleccionar el modo asíncrono o el modo síncrono. Si el valor de estos dos bits es diferente de 00, además de indicar que se está eligiendo **modo asíncrono** se está indicando el factor de razón de baud con el que se tiene que dividir la frecuencia de las entradas de reloj $\overline{\mathbf{TxC}}$ y $\overline{\mathbf{RxC}}$.

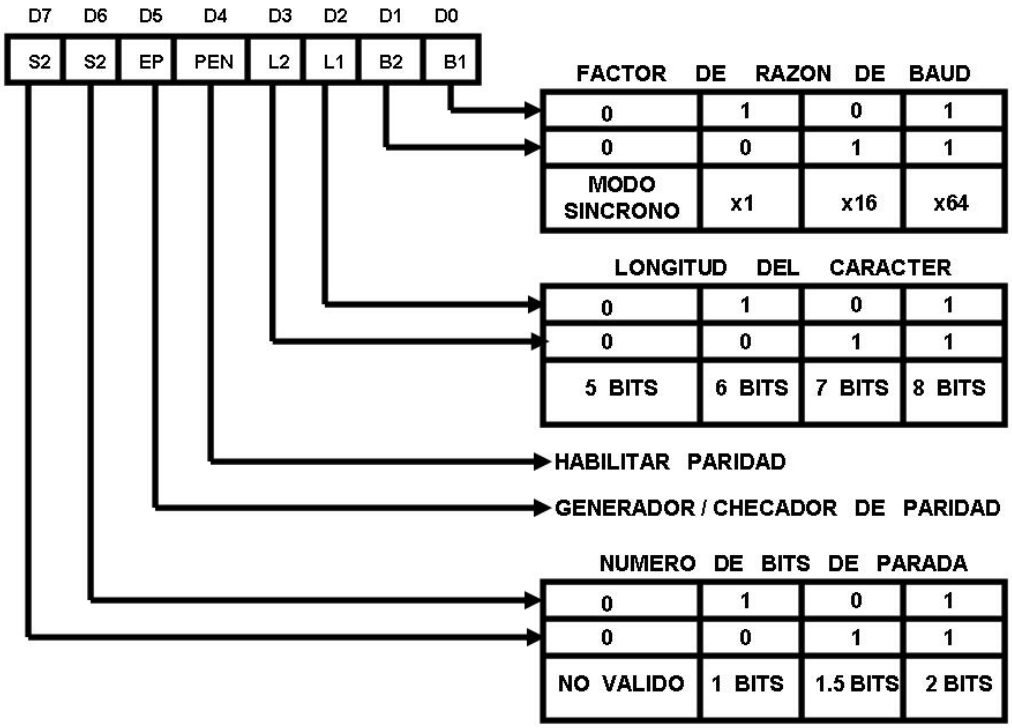


Figura 8.16 Formato de la Instrucción de Modo Asíncrono

Por ejemplo, si la frecuencia de las entradas de reloj $\overline{\mathbf{TxC}}$ y $\overline{\mathbf{RxC}}$ es 38,400 y se escoge un factor de razón de baud de **x64** (bits **D0-D1=11**), la frecuencia transmisión y recepción será de 600 bits por segundo.

Los bits **D2-D3** seleccionan la longitud del dato. El USART del C.I. **8251** puede trabajar con datos de 5, 6, y 8 bits.

El bit **D4** permite habilitar o deshabilitar el verificador de paridad. Si se ha seleccionado habilitar el verificador de paridad, con el bit **D5** se debe ordenar el tipo de paridad.

Los bits **D6-D7** seleccionan el número de bits de parada.

La figura 8.17 ilustra el formato de los bits durante la transición y recepción asíncrona de los datos. Si la longitud de los datos durante la recepción se

escoge de 5, 6 ó 7 bits, los bits que no se usan se ponen en cero. De esta forma se elimina un posible error en el dato durante la lectura en el acumulador.

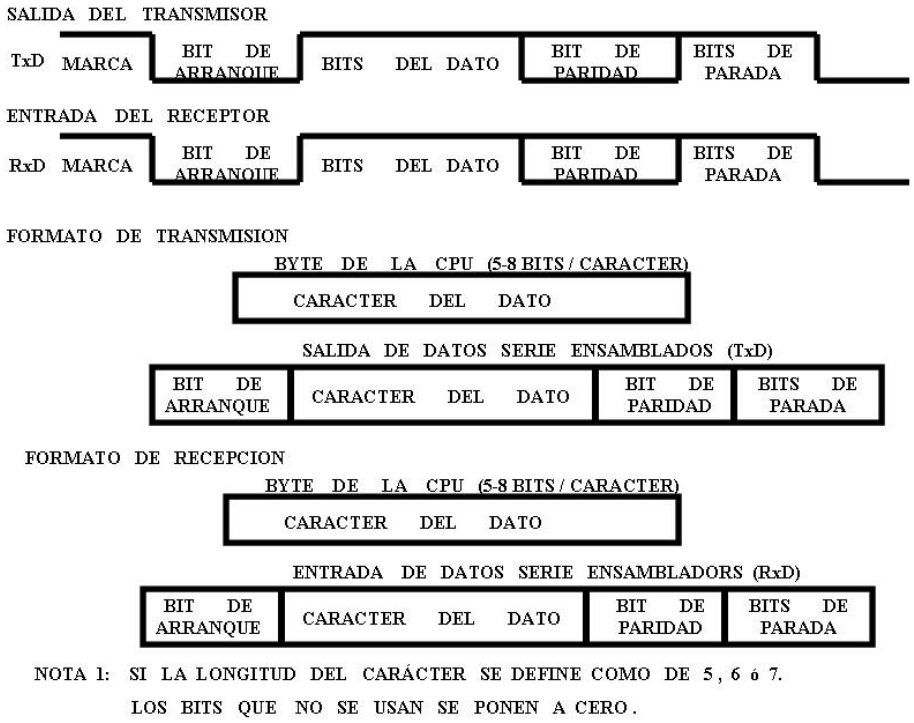


Figura 8.17 Modo Asíncrono

Ejemplo Formar la palabra de control de instrucción de modo con las siguientes características: dos bits de parada, paridad deshabilitada,
ocho bits de datos y un factor de x64.

Solución

Control = 11000 111 = CFH

Definición de la instrucción de comando

Una vez que la definición funcional del C.I. 8251 se ha programado por la **instrucción de modo**, el dispositivo se encuentra listo para usarse en la

comunicación de datos. La instrucción de comando controla la operación actual del formato seleccionado. Funciones tales como habilitar transmisión/recepción, error de limpiar (**reset**) y controles de módem son proporcionadas por la instrucción de comando.

Una vez que la instrucción de modo se ha escrito en el C.I. 8251, las posteriores escrituras de control **C/** $\overline{\mathbf{D}}$ **= 1** cargarán la **instrucción de comando**. Una operación **Reset** (interna o externa) regresará el C.I. 8251 al formato de la **instrucción de modo.** La figura 8.14 ilustra el formato de la palabra de **instrucción de comando**.

El bit 0 es el control de la señal transmisión habilitada (**TxEn**). Los datos se pueden transmitir únicamente cuando **TxEN** está en alto. Si el bit 0 tiene valor 0 la señal **TxEN** toma el valor 0 deshabilitando la transmisión de datos.

El bit 1 con valor 1 envía a la salida $\overline{\mathbf{DTR}}$ a nivel 0.

La señal $\overline{\mathbf{DTR}}$ se utiliza en sistemas de comunicación de datos que operan automáticamente o bajo control de programa. La salida $\overline{\mathbf{DTR}}$ con nivel 0 indica a la lógica del sistema de microcomputadora que el C.I. 8251 se encuentra listo para comunicarse.

El bit 2 habilita o deshabilita la señal de control **RxRDY** pero no habilita o deshabilita la lógica de recepción del C.I. 8251 recibe un dato se encuentra o no habilitada la señal **RxRDY**. Si la instrucción de comando tiene un 1 en el bit 1, entonces, la señal **RxRDy** indicará cuando **RA** haya recibido un dato.

Si el bit 3 se pone a 1, se interrumpe la salida de datos serie y forma a la salida **TxE** a nivel alto. Esto ordena enviar señales de marca.

El bit 4 permite limpiar la bandera de error del registro de estados errores durante la transmisión/recepción de datos serie, pondrá en 1 algunas banderas de error, las cuales únicamente se pueden limpiar con el bit 4 con valor 1.

El bit 5 permite enviar a nivel bajo la señal de salida RTS. Esta señal se usa en la lógica del protocolo con módems.

El bit 6, cuando está en alto, causa que la próxima palabra de control se interprete como **instrucción de modo** y no como instrucción de comando. Esta acción se conoce como limpiar interno.

El bit 7 se aplica únicamente en operación síncrona. Cuando este bit tiene el nivel alto causa que la 8251 entre en el estado **Hunt.**

Ejemplo Formar la palabra de control de instrucción de comando con las siguientes características

Solución Habilitado para transmitir, operación normal, enviar las señales $\overline{\mathbf{DTR}}$ y

$\overline{\mathbf{DTS}}$ a 0 y no entrar en estado **Hunt**.

Control = 00100111 = 27H

Banderas de estados de la 8251

En los sistemas de comunicación de datos frecuentemente es necesario examinar los estados del dispositivo activo para verificar si han ocurrido errores u otras condiciones que requieren de la atención del procesador. El C.I. 8251 tiene un registro de banderas que permite al programador leer los estados del dispositivo en cualquier tiempo. Un comando de lectura se manda por la CPU con la entrada **C / $\overline{\mathbf{D}}$** en 1 para lograr esta función. La figura 8.11 ilustra el formato de las palabras de estado de la 8251.

Los bits D0, D1, D2, D6 y D7 indican los estados de los pins, los bits D3, D4 y D5 indican cuándo se presentan estos errores. En el 8251 no intenta corregir estos errores; son responsabilidad del programador.

8.1.6 Sistemas de almacenamiento de información

En muchas ocasionas una computadora personal puede ejecutar un programa, el cual puede producir una gran cantidad de datos que deben ser almacenados en dispositivos externos de gran capacidad, entre los cuales podemos mencionar discos flexibles, discos duros, etc. Actualmente existen dispositivos (de entrada/salida) para almacenar información que van desde un 1MB hasta varios miles de TeraByte.

Una primera clasificación de los sistemas de almacenamiento se puede realizar en función de la tecnología utilizada para ello. Actualmente existen dos tipos de tecnologías: la óptica y la magnética las cuales explicaremos a continuación.

Tecnología óptica

La tecnología óptica de almacenamiento por láser es reciente. Los fundamentos técnicos en que se basa esta tecnología son sencillos de entender: Un haz láser va leyendo (o escribiendo) microscópicos agujeros en la superficie de un disco de material plástico, recubiertos a su vez por una capa transparente para su protección del polvo. Los dispositivos externos de almacenamiento fabricados con esta tecnología son: el Disco Compacto (del inglés, **Compact Disks)** y el Disco de Video Digital, (del inglés, **Digital Video Disk**), los cuales presentaremos a continuación.

Disco de Video Digital

El disco de video digital, (DVD), es un dispositivo de almacenamiento masivo de datos cuyo aspecto es idéntico al de un disco compacto (CD), aunque contiene hasta 25 veces más información y puede transmitirla a la computadora unas 20 veces más rápido que un CD-ROM. Su mayor capacidad de almacenamiento se debe, entre otras cosas, a que puede utilizar ambas caras del disco y, en algunos casos, hasta dos capas por cada cara, mientras que el CD sólo utiliza una cara y una capa. Las unidades lectoras de DVD permiten leer la mayoría de los CD, ya que ambos son discos ópticos; no obstante, los lectores de CD no permiten leer DVD.

El DVD almacenan hasta 133 minutos de información por cada cara, con una calidad de vídeo **Laser-disc** y que soportan sonido digital Dolby surround; Los DVD emplean formatos específicos para la computadora que almacenen datos y material interactivo en forma de texto, audio o vídeo, como los DVD-R, unidades en las que se puede grabar la información una vez y leerla muchas, DVD-RW, en los que la información se puede grabar y borrar muchas veces, y los DVD-RAM,

también de lectura y escritura.[10]

Los discos DVD tienen la misma forma física y el mismo tamaño, pero difieren en el formato de almacenamiento de los datos y, en consecuencia, en su capacidad. Así, los DVD-Vídeo de una cara y una capa almacenan 4,7 GB, y los DVD-ROM de dos caras y dos capas almacenan hasta 17 GB. Del mismo modo, no todos los DVD se pueden reproducir en cualquier unidad lectora.

CD

Los CD se han convertido en el medio estándar tanto para distribuir programas como para hacer copias de seguridad, grabaciones multimedia, etc., debido a su capacidad relativamente alta (hay CD de 700 Mb y de 900 Mb) y sobre todo, a un bajo costo.

Los CD es un medio idóneo para difundir programas y datos que no queramos que se alteren, ya que una vez cerrada su grabación, esta no se puede alterar, salvo en los CD del tipo regrabable, que nos permiten borrarlos para volver a utilizarlos, con una vida útil (según el fabricante) de unas 1.000 grabaciones[11].

Tecnología magnética

La tecnología magnética, "consiste en la aplicación de campos magnéticos a ciertos materiales cuyas partículas reaccionan a esa influencia, generalmente orientándose en unas determinadas posiciones que conservan tras dejar de aplicarse el campo magnético."[12] Esas posiciones representan los datos y esta operación se puede repetir un gran número de veces. Entre los elementos de almacenamiento construidos con esta tecnología se encuentra el disco duro, el cual presentaremos a continuación.

---

[10] Dispositivos de almacenamiento. http://www.monografias.com/trabajos12/dispalm/dispalm.shtml. Recuperado: 03 de marzo de 2009.

[11] Josito. Diferentes tipos de medios de almacenamiento y sus usos más frecuentes. Disponible en: http://www.configurarequipos.com/doc336.html, 11 de julio de 2008. Recuperado: 03 de marzo de 2009.

[12] Dispositivos de almacenamiento, disponible en: http://html.rincondelvago.com/dispositivos-de-almacenamiento_1.html, recuperado el 3 de marzo de 2009.

Disco duro

> Un disco duro es una unidad de almacenamiento permanente de gran capacidad. Está formado por varios discos apilados —dos o más—, normalmente de aluminio o de vidrio, recubiertos de un material ferromagnético, como en los diskettes. Un disco duro cuenta con una cabeza de lectura/escritura la cual permite grabar la información, modificando las propiedades magnéticas del material de la superficie y leerla posteriormente.[13]

Los elementos que forman un disco duro son:

- Varios discos de metal magnetizado, que es donde se guardan los datos.
- Un motor que hace girar los discos.
- Un conjunto de cabezales, que son los que leen la información guardada en los discos.
- Un electroimán que mueve los cabezales.
- Un circuito electrónico de control, que incluye el interface con la computadora y la memoria cache.
- Una caja hermética (aunque no al vacío), que protege el conjunto.

El disco duro tiene diferentes tipos entre los cuales se encuentran el disco duro **IDE, SCSI y SATA** básicamente, los cuales se explican a continuación.

Discos IDE

Los discos **IDE (ATA / PATA)** son los más utilizados. A partir del estándar ATA/133, con una velocidad de hasta 133 MBps y una velocidad de giro de 7.200 rpm, entraron en competencia directa con los HDD SCSI, con la ventaja de una mayor capacidad y un costo mucho menor. Además, un rendimiento razonablemente elevado a un precio económico y son más o menos fáciles de instalar. Sin embargo, se ven limitados a un número máximo de 4 dispositivos (y esto con las controladoras EIDE, las **IDE** originales sólo pueden manejar 2).

Su conexión se realiza mediante un cable plano con conectores con 40 pines colocados en dos hileras (aparte del cable de alimentación, que es común para todos los tipos de disco duro). Así pues, para identificar correctamente un disco **IDE** basta con observar la presencia de este conector, aunque para estar seguros al 100% deberemos buscar unos

[13] Componentes de un PC (Personal Computer) http://html.rincondelvago.com/componentes-de-un-pc_1.html ReCUPERADO: 03 de marzo de 2009.

microinterruptores ("jumpers") que, en número de 2 a 4, permiten elegir el orden de los dispositivos (es decir, si se comportan como "Maestro" o como "Esclavo").

SCSI
Esta tecnología es mucho menos utilizada, pero no por ser mala, sino por ser relativamente cara. Estos discos suelen ser más rápidos a la hora de transmitir datos, a la vez que usan menos al procesador para hacerlo, lo que se traduce en un aumento de prestaciones.[14]

Estos discos deben estar conectados a una controladora **SCSI**. Han sido más rápidos que los **IDE** y de mayor capacidad hasta la aparición del **ATA/100**, permitiendo una velocidad de transmisión de hasta 80 MBps, y discos con una velocidad de giro de unas 10.000 rpm.

El estándar **SCSI** ha evolucionado en velocidad a través del tiempo, pero también lo ha hecho la velocidad de los discos duros **SATA**, relegando a los discos **SCSI** prácticamente al sector de grandes servidores.[15]

Hoy en día, en el mundo de las computadoras personales se utilizan dos tipos de disco duro: el **IDE** y el **SCSI**. La diferencia entre estos discos duros radica en la manera de conectarlos a la Tarjeta Madre (Placa principal).

Los conectores SCSI son múltiples, como lo son las variantes de la norma: **SCSI-1, SCSI-2, Wide SCSI, Ultra SCSI**. Pueden ser planos de 50 contactos en 2 hileras, o de 68 contactos, o no planos con conector de 36 contactos, con mini-conector de 50 contactos. Una forma de identificarlos puede ser que, en una cadena de dispositivos **SCSI** (hasta 7 ó 15 dispositivos que van intercalados a lo largo de un cable o cables), cada aparato tiene un número que lo identifica, que en general se puede seleccionar. Para ello habrá una hilera de jumpers, o bien una rueda giratoria, que es lo que deberemos buscar.[16]

SATA (Serial ATA)
Es el nuevo estándar para **HDD**. Hay dos tipos. **SATA1**, con transferencia de hasta 150 Mbps y **SATA2** (o **SATA 3Gb**), con transferencia de hasta 300 Mbps.

---

[14] Información basada en Alfonso Araujo Cárdenas. Dispositivos de almacenamiento. En http://mx.geocities.com/alfonsoaraujocardenas/almacenamiento.html. 05 de junio de 2004. Recuperado: 03 de marzo de 2009.
[15] Josito: Diferentes tipos de medios de almacenamiento y sus usos más frecuentes. http://www.configurarequipos.com/doc336.html. Recuperado_ 03 de marzo de 2009.
[16] "Dispositivos de almacenamiento", disponible en: http://www.monografias.com/trabajos12/dispalm/dispalm.shtml, recuperado el 3 marzo de 2009.

La velocidad de los discos duros actuales es de 7.200 rpm, llegando a las 10.000 rpm en algunas series de discos duros de alta velocidad. En cuanto a los discos duros para portátiles, la velocidad de giro es de 5.400 rpm, si bien están saliendo al mercado algunos modelos a 7.200 rpm.[17]

Bibliografía del tema 8

Intel, *Microsystem Components Handbook*, EE.UU., Intel Corporation, Literature Department, Vol. 1 y 2,1985.

García, Narcia, O., *8085A e Interfaces*, México, IPN, Agosto 1982.

McGlynnn, R., Daniel, *Modern Microprocessor System Design,* EE.UU., John Wiley & Sons 1980.

Torres Portero, Manuel, *Microprocesadores y Microcontroladores Aplicados a la Industria*, Madrid, Paraninfo, 1988.

Willen C, David, *8088 Assembler Language Programming: The IBM PC,* EE.UU., Howard W. Sams & Company, 1987.

Leventel A., Lance, *8080A/8085 Assembly Language Programming*, Berkeley, Osborne / McGraw Hill, 1978.

Actividades de aprendizaje

A.8.1. Identifica las unidades funcionales (Puerto Programable en Paralelo, Puerto Serial, Controlador de DMA, etc.) en una tarjeta madre de 32 bits.

A.8.2. Investiga la dirección de los tres puertos (PA, PB y PC) y el registro de control en una tarjeta madre de 32 bits.

A.8.3. Investiga la ubicación del área de Memoria asignada al BIOS en una tarjeta madre de 32 bits.

[17] Josito: "Tipos de almacenamiento de datos", disponible en: http://www.configurarequipos.com/doc336.html. Recuperado el 03 de marzo de 2009.

A.8.4. Programa el puerto paralelo para que el puerto PA sea únicamente de salida.

A.8.5. Identifica las unidades funcionales (Puerto Programable en Paralelo, Puerto Serial, Controlador de DMA, etc.) en una tarjeta madre de 64 bits.

A.8.6. Investiga la dirección de los puertos (PA, PB y PC) y el registro de control en una tarjeta madre de 64 bits.

A.8.7. Investiga la ubicación del área de Memoria asignada al BIOS en una tarjeta madre de 64 bits.

A.8.8. Programa el puerto paralelo para que el puerto PB sea únicamente de salida.

Cuestionario de autoevaluación

1. ¿Qué es el BIOS?
2. ¿Qué es una interrupción?
3. ¿Cuáles son los tipos de interrupción?
4. ¿Qué es un controlador de interrupciones?
5. ¿Qué es un DMA?
6. ¿Qué es un puerto paralelo?
7. ¿Cuáles son los modos en que trabaja el puerto paralelo?
8. ¿Qué es un puerto serie?
9. ¿Qué es un UART?
10. ¿Qué es un USART?
11. ¿Cuáles son las tecnologías para el almacenamiento de datos?

Examen de Autoevaluación

Relaciona las columnas escribiendo dentro del paréntesis el número que una los conceptos con sus respectivas definiciones.

1) Interrupción multinivel	( ) Aplicación de campos magnéticos a ciertos materiales cuyas partículas reaccionan a esa influencia, generalmente orientándose en unas determinadas posiciones que conservan tras dejar de aplicarse dichos campos.
2) DMA	( ) El **BIOS**, los canales **DMA**, los puertos de comunicación.
3) Transferencia síncrona	( ) Es una unidad de almacenamiento permanente de gran capacidad, formado por varios discos apilados y fabricado con tecnología magnética.
4) Unidades funcionales	( ) Tecnología que utiliza un haz de rayo láser que va leyendo (o escribiendo) microscópicos agujeros en una superficie de un disco de material plástico y la cual se utiliza para fabricar dispositivos externos de almacenamiento.
5) Interrupción (IRQ's)	( ) Conjunto de programas que cargan el sistema operativo en memoria **RAM** para su ejecución.
6) Direcciones de entrada/salida	( ) Transferencia de datos en serie en forma síncrona y continua.
7) Tecnología Óptica	( ) Líneas independientes de interrupción y donde cada una de ellas causará una serie de actividades específicas.
8) BIOS	( ) Localidades de memoria establecidas por el diseñador de computadoras las cuales permiten capturar y/o enviar datos a través de las diferentes unidades funcionales.
9) Tecnología Magnética	( ) Señal recibida por el microprocesador de una Computadora Personal, indicando que debe "interrumpir" el curso de ejecución actual y pasar a ejecutar una llamada a una subrutina para atender esta solicitud de interrupción.
10) Disco Duro	( ) Rutas del sistema usados por muchos dispositivos para transferir información directamente a la memoria en ambos sentidos.

Bibliografía básica

Ayala Pérez, Jaime, *Aritmética para Computadoras*, México ENEP. Aragón, UNAM, (colección de Cuadernos de la ENEP ARAGÓN, No. 33,) 1989.

_____ Arquitectura de Computadoras, México, Comunicación interna, 2005.

Charles H Roth Jr., Fundamentals of Logic Design, 4ª ed., EE.UU., West Publishing Company, 1992.

García, Narcia, O., 8085A e Interfaces, México, IPN, Agosto 1982.

Hamacher, V. C., *Computer Organization*, EE.UU., McGraw Hill International Book Company, 1982.

Hayes P. John, Diseño de Sistemas Digitales y Microprocesadores, McGraw Hill, 1986.

Intel, *Microsystem Components Handbook*, EE.UU., Intel Corporation. Literature Department, Vol. 1 y 2, 1985.

Lee, Samuel C, *Digital circuits and logic design,* EE.UU., Prentice Hall, Inc., Englewood Cliffs, N. J., 1976.

Lenk, D. John, *Handbook of Microprocessors, Microcomputers, and Minicomputers*, EE.UU., Prentice-Hall, INC., Englewood Cliffs, N.J., 1979.

Leventel A., Lance, *8080A/8085 Assembly Language Programming*, Berkeley, Osborne / McGraw Hill, 1978

Mandado, E., *Sistemas Electrónicos Digitales*, Madrid, Marcombo Boixareu, 1980.

Mano, Morris, M., *Arquitectura de Computadoras*, México, Prentice Hall Hispanoamericana, 1988.

_____*Diseño Digital*, México, Prentice Hall Hispanoamericana, 1987.

McGlynnn, R., Daniel, *Modern Microprocessor System Design, EE.UU.,* John Wiley & Sons, 1980.

Motorola, *MC68340 Integrated processor User´s Manual*, EE.UU., Motorola Incorporation. Literature Department, Vol. 1, 1990.

Nashelky, Louis, *Teoría de las calculadoras numéricas automáticas, Madrid,* Alhambra, 1979.

Peatman, B. John, *Microcomputer-Based Design*, Tokio, McGraw-Hill / KOGAKUSHA, LTD., 1982.

Stallings, William, *Organización y Arquitectura de Computadores*, Madrid, Pearson Educación, 2006.

Tocci, Ronald, Digital Design, Principles and Applications, EE.UU., Prentice Hall, 1977

Torres Portero Manuel, *Micro y Microcontroladores Aplicados a la Industria*, Madrid, Paraninfo, 1989.

Uruñuela Martínez, José Ma., *Microprocesadores Programación e interconexión*, México, McGraw-Hill, 1989.

Willen C, David, *8088 Assembler Language Programming: The IBM PC,* EE.UU., Howard W. Sams & Company, 1987.

RESPUESTAS A LOS EXÁMENES DE AUTOEVALUACIÓN

ARQUITECTURA DE COMPUTADORAS

	Tema 1	Tema 2	Tema 3	Tema 4	Tema 5
1.	2	d.) 231.59375	6	c	6
2.	6	a.) 1000011111.00011	4	b	5
3.	1	b.) $(225.075)_{10}$	8	a	9
4.	10	c.) $(34001)_{8}$	10	b	7
5.	8	b.) $(5CC)_{16}$	2	d	10
6.	3	d.) Cociente: $(100101)_{2}$, Residuo: $(0)_{2}$	9	c	8
7.	5	c.) Cociente: $(C35)_{16}$, Residuo: $(0)_{16}$	5	c	4
8.	9	d.) $(74.361)_{10}$	3	b	3
9.	4	a.) $(10100)_{2}$	7	b	2
10.	7	c.)$(74.360)_{10}$	1	b	1
11.		b.) $(010011)_{2}$			
12.		a.) $(0.1001)_{2}$			

	Tema 6	Tema 7	Tema 8
1.	6	4	9
2.	9	9	4
3.	1	8	10
4.	10	2	7
5.	3	7	8
6.	8	3	3
7.	5	10	1
8.	7	6	6
9.	4	1	5
10.	2	5	2