UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
DIVISIÓN SISTEMA UNIVERSIDAD ABIERTA Y EDUCACIÓN A DISTANCIA

APUNTE ELECTRÓNICO

Arquitectura de Computadoras

Licenciatura en Informática

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

COLABORADORES

DIRECTOR DE LA FCA
Dr. Juan Alberto Adam Siade

SECRETARIO GENERAL
Mtro. Tomás Humberto Rubio Pérez

– – – –

COORDINACIÓN GENERAL
Mtra. Gabriela Montero Montiel
Jefe de la División SUAyED-FCA-UNAM

COORDINACIÓN ACADÉMICA
Mtro. Francisco Hernández Mendoza
FCA-UNAM

– – –

AUTOR
Ing. Tomás García González

REVISIÓN PEDAGÓGICA
Lic. Dayanira Granados Pérez

CORRECCIÓN DE ESTILO
Mtro. Francisco Vladimir Aceves Gaytán

DISEÑO DE PORTADAS
L.CG. Ricardo Alberto Báez Caballero
Mtra. Marlene Olga Ramírez Chavero

DISEÑO EDITORIAL
Mtra. Marlene Olga Ramírez Chavero

Arquitectura de computadoras
Apunte electrónicoes

Edición: Agosto 2017

Facultad de Contaduría y Administración
Circuito Exterior s/n, Ciudad Universitaria
Delegación Coyoacán, C.P. 04510, México, Ciudad de México.

ISBN: En trámite
Plan de estudios 2012, actualizado 2016.

Hecho en México

OBJETIVO GENERAL

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
(64 horas)

	Horas
1. Introducción	6
2. Sistemas de Numeración	8
3. Códigos	8
4. Álgebra de Boole	8
5. Circuitos combinatorios	10
6. Circuitos secuenciales	10
7. Memorias	8
8. Unidades funcionales	6

INTRODUCCIÓN

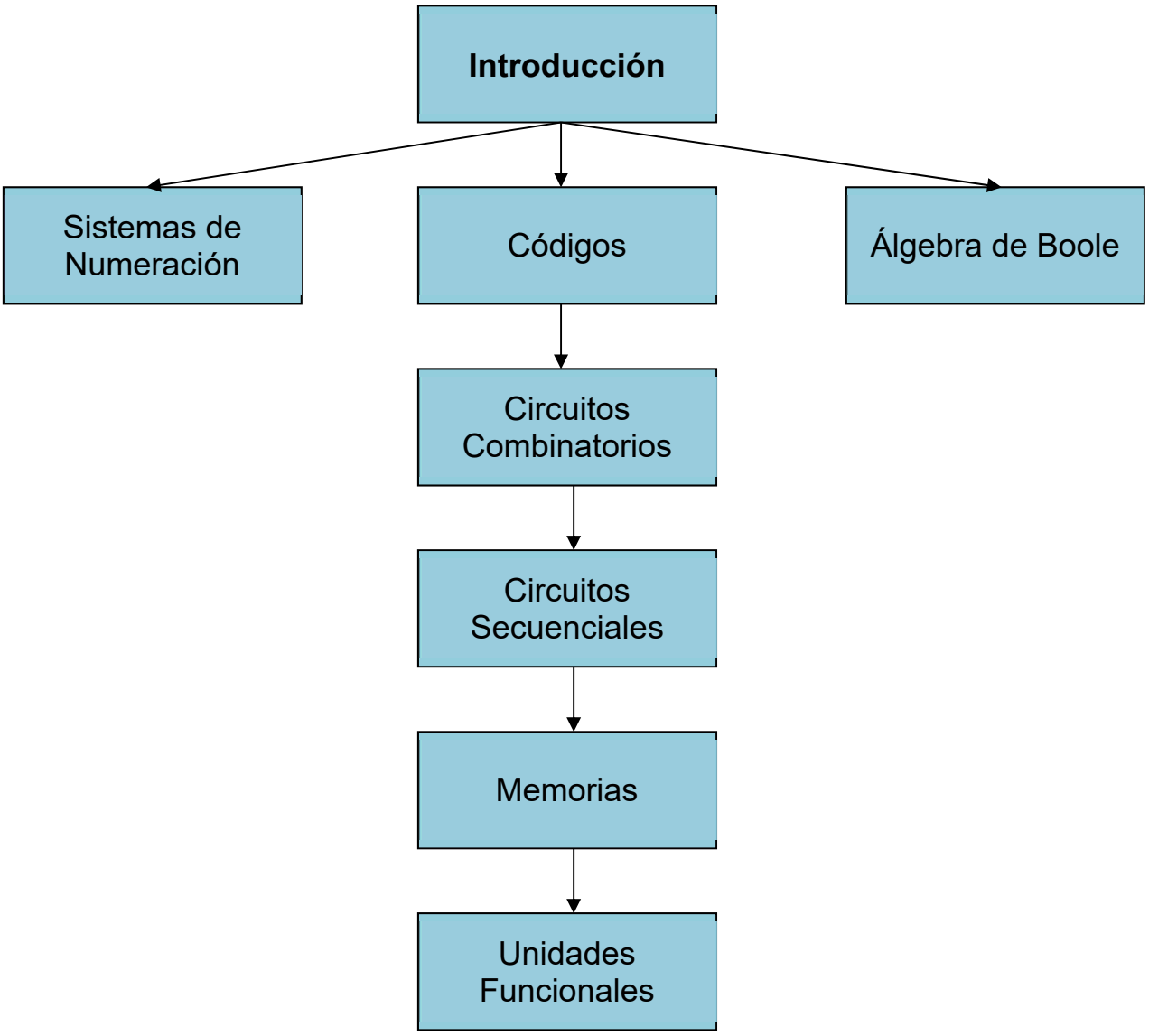
En la actualidad, la arquitectura (organización interna) de las computadoras digitales constituye un importante tema de estudio para los profesionales de la informática y sin duda alguna, su utilización e importancia aumentarán en el futuro. En esta asignatura trataremos la arquitectura básica de una computadora digital, el funcionamiento de cada uno de sus componentes y la interrelación entre sí de dichos componentes, tanto desde la parte conceptual (codificación, sistemas numéricos y álgebra de Boole) como de las partes físicas que operan mediante estos conceptos.

La unidad **uno** es introductoria y hace una presentación del modelo planteado por John von Neumann en el cual se presentan los conceptos básicos de la arquitectura de computadoras como ahora la conocemos y utilizamos, y que incluyen el uso de memoria para guardar los programas. La unidad **dos** trata de los sistemas numéricos y básicamente del uso del sistema binario y sus representaciones octal y hexadecimal. Siendo el sistema binario el utilizado por los circuitos electrónicos de una computadora, se revisan las formas de procesamiento de información numérica en este sistema. La unidad **tres** trata de la codificación o forma de representación y manejo de información, tanto numérica como alfanumérica. Se tratan asimismo los códigos de detección de error. En la unidad **cuatro** se revisan los conceptos de lógica booleana así como de la implementación de funciones binarias. Su importancia radica en que las funciones booleanas permiten el diseño y construcción de circuitos mediante elementos electrónicos. Las unidades anteriores, de la 1 a la 4, constituyen la parte conceptual de la materia. Las unidades siguientes, la parte de aplicación, desde donde se pueden crear los módulos elementales de una computadora.

En la unidad **cinco** se presentan los diversos circuitos combinatorios, los que no incluyen la variable tiempo para su funcionamiento y a partir de los cuales se pueden

crear las partes reconocibles de una computadora: memorias y unidades de control y proceso de datos. La unidad **seis**, trata de los circuitos lógicos secuenciales, en donde se involucran tiempos de proceso y señales de sincronía, así como los elementos básicos de memoria creados a partir de *flip flops*. La unidad **siete** presenta los principales tipos de memoria, así como el funcionamiento, tiempos de acceso y control de las mismas. Finalmente, en la unidad **ocho** se relacionan los elementos vistos hasta la unidad siete para integrar un modelo de computadora con sus partes básicas y subsistemas lógicos.

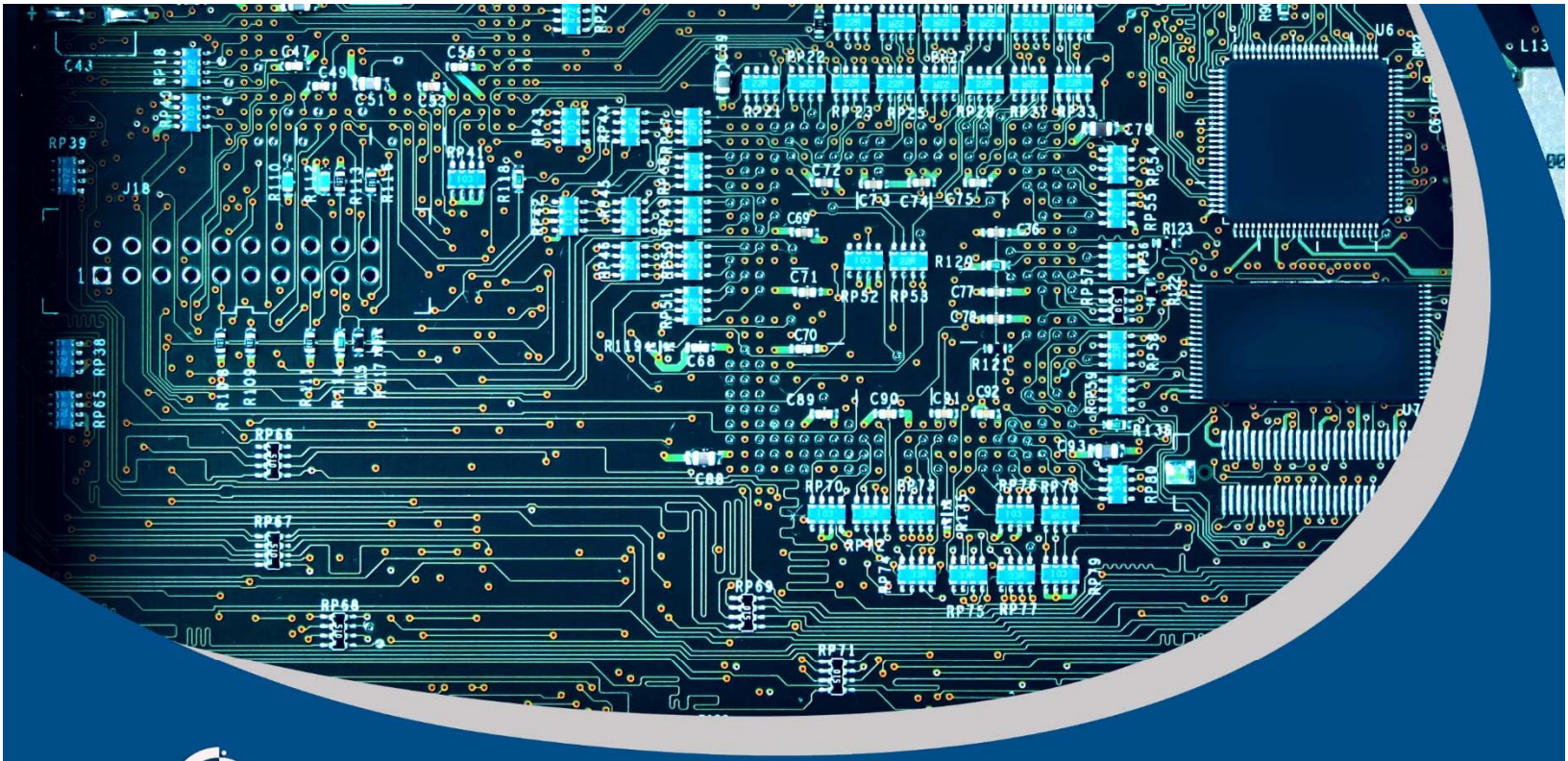
ESTRUCTURA CONCEPTUAL



Licenciatura: Informática

UNIDAD 1

Introducción



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno identificará la estructura básica de las computadoras, su organización y los elementos básicos de un microprocesador.

TEMARIO DETALLADO
(6 horas)

1. Introducción

- 1.1. Estructura básica de las computadoras
- 1.2 Organización de una computadora digital (Arquitectura de Von Neumann y de Harvard)
- 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

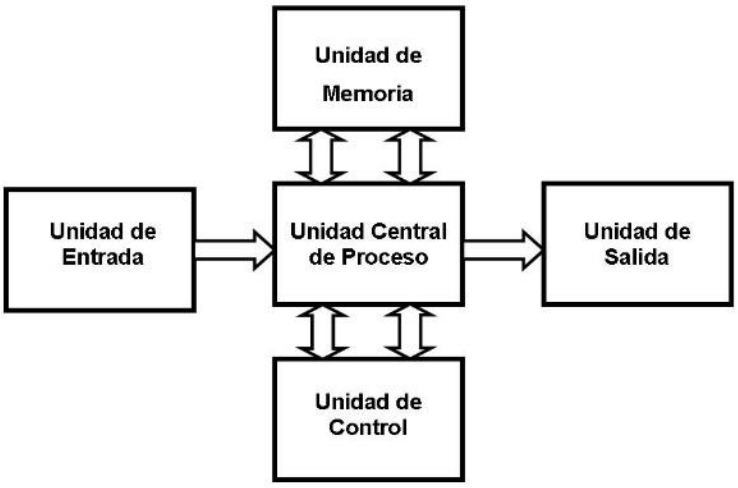
INTRODUCCIÓN

En la actualidad, la arquitectura (organización interna) de las Computadoras Digitales constituye un importante tema de estudio y una práctica cotidiana fundamental. En esta unidad 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

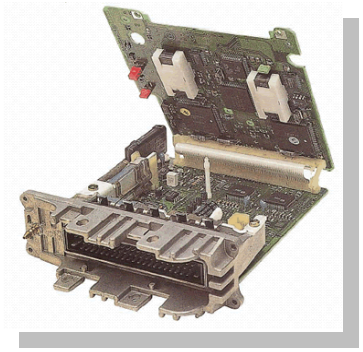
Al hablar de computadoras hoy en día se está haciendo referencia a máquinas electrónicas-mecánicas, esto es, máquinas cuyas funciones se efectúan utilizando circuitos electrónicos analógicos y/o digitales.

Definición de computadora: Una computadora es una máquina electro-mecánica que procesa información digital. Un modelo básico de una computadora consta de las siguientes partes: Unidad de Entrada, Unidad de Salida, Unidad de Central de Proceso, Unidad de Memoria y de la Unidad de Control interconectadas por buses unidireccionales y bidireccionales.



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.

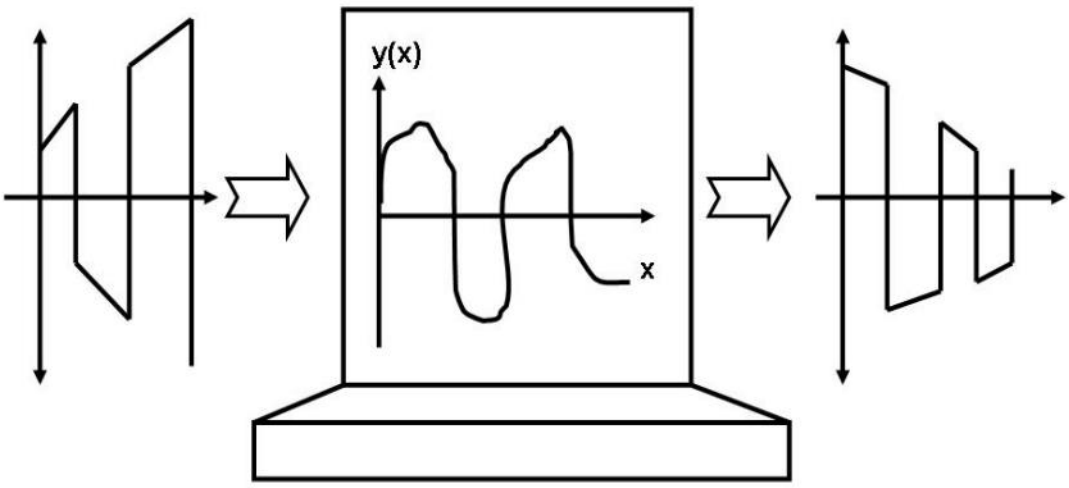
Unidad de Memoria 	Generalmente, las instrucciones (programa) y los datos se almacenan en la memoria interna de la computadora (la Unidad de Memoria), la cual se diferencia del dispositivo de entrada (que también puede ser una memoria) por su velocidad de operación. La memoria interna se diseña para 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) o pantalla de cristal liquido (LCD) como dispositivo de salida. Conforme el (programador) usuario introduce algún programa vía el teclado, cada carácter se visualiza simultáneamente en la pantalla. Además de la pantalla, la computadora cuenta con otros dispositivos de salida como son el diskette, disco duro, memoria USB, cinta magnética, disco duro, la red o impresiones en papel.

Tipos de Computadoras

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

- Computadoras analógicas
- 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 (por procesar) es una señal analógica y la señal de salida es una señal analógica representada en forma gráfica en un Tubo de Rayos Catódicos.

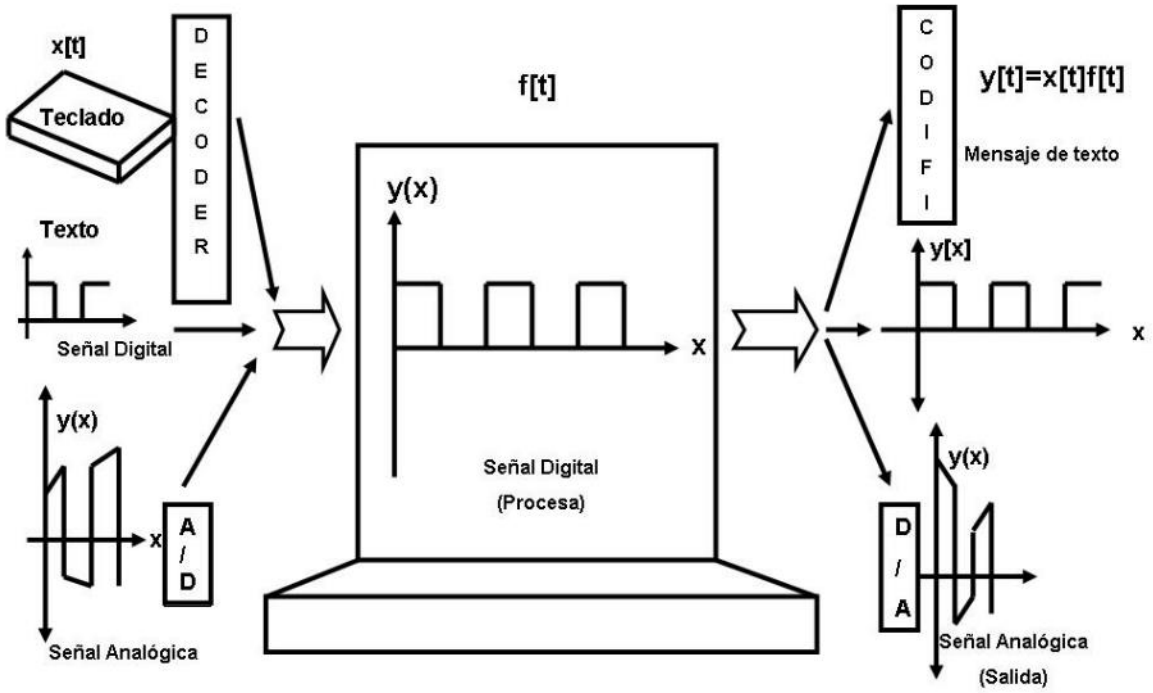


Esquema de una computadora analógica

El principal inconveniente de las computadoras analógicas es que sólo pueden alcanzar velocidades de resolución superiores a algunos centenares de ciclos por segundo o Hertz (o Hertzios). 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 (por 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.



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 Unidad 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 esta asignatura 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:

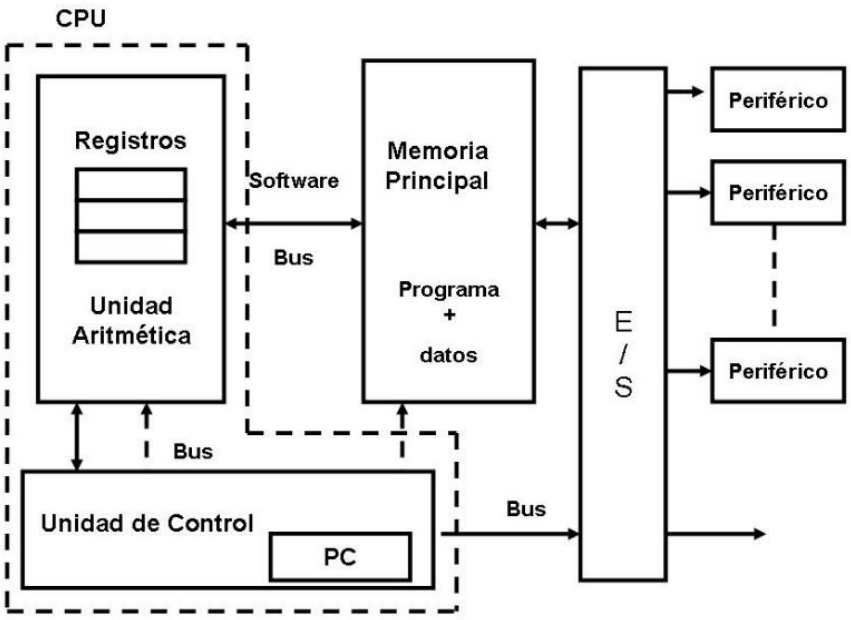
Computadoras para propósitos generales	Una *computadora para propósitos generales* se diseña de modo que pueda ser programada para resolver una amplia variedad de problemas administrativos, de medicina, de ingeniería, etc. Las computadoras de propósito general pueden estudiar, en unos cuantos minutos, 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).
Computadoras para propósitos específicos	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. Ejemplos típicos pueden ser el control de un avión, procesadores de los dispositivos de comunicación móviles o 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. A partir de este momento, haremos referencia a las computadoras digitales de propósito general.

1.2 Organización de una computadora digital (Arquitectura de Von Neumann y de Harvard)

Organización de un microcomputador (Estructura de von Neumann)

La estructura von Neumann es el modelo básico de arquitectura usado en la mayoría de las computadoras digitales actuales. Las dos principales características de esta estructura son: el uso del sistema de numeración binario y el concepto de “programa almacenado”. La estructura von Neumann está formada por:



Unidad Central de Proceso (CPU)	•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.
Unidad de Memoria	•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).
Unidad de Control	•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.
Unidad de Entrada y Salida	•Las Unidades de entrada y salida realizan la transferencia de información con los periféricos. Los periféricos permiten cargar datos y programas en la memoria principal y presentar 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 Neumann utiliza el modelo de “programa almacenado” y dicho modelo presenta las siguientes ventajas:

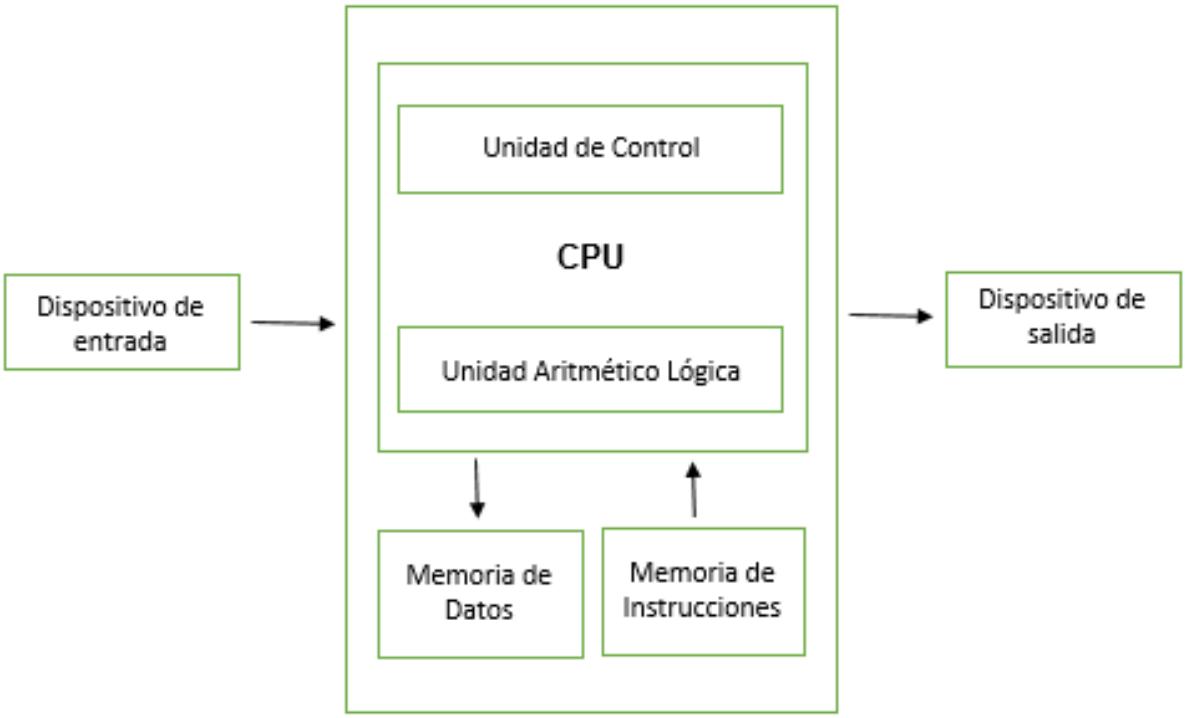
1. Se pueden ejecutar diversos programas.
2. Tiene gran velocidad de ejecución.
3. Se pueden construir programas automodificables, intérpretes, compiladores, etc.

Como se mencionó anteriormente, una de las principales aportaciones de la estructura de von Neumann 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 traduce a lenguaje de máquina y se almacena 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.

Además de la arquitectura de von Neumann, existe otra arquitectura denominada Harvard.

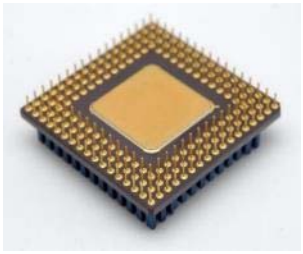


Elaboración propia.

En la arquitectura Harvard las instrucciones se encuentran en una memoria, y los datos en otra memoria. Desde el punto de vista teórico funciona. Sin embargo, desde el punto de vista físico, plantea un problema, ya que hay que duplicar la comunicación entre la memoria y el CPU, lo que hace más complejo su diseño, y además es necesario realizar una consulta a dos memorias en lugar de una, esto implica menor desempeño. En la arquitectura von Neumann también puede haber problemas de desempeño debido a que los datos y las instrucciones deben pasar por un solo bus, en esto Harvard es mejor, sin embargo, la mayoría de las computadoras están construidas bajo la arquitectura von Neumann.

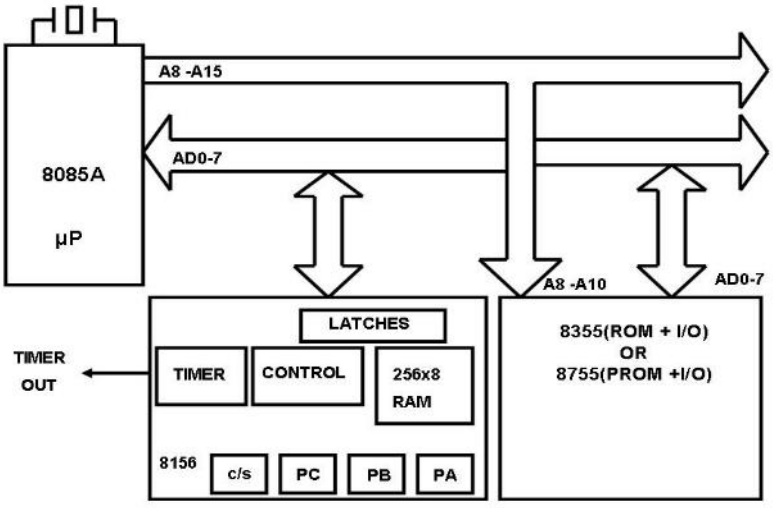
Fuente: http://latecladeescape.com/h/2015/07/arquitectura-von-neumann-y-harvard (Fecha de consulta: 19/05/2017).

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.

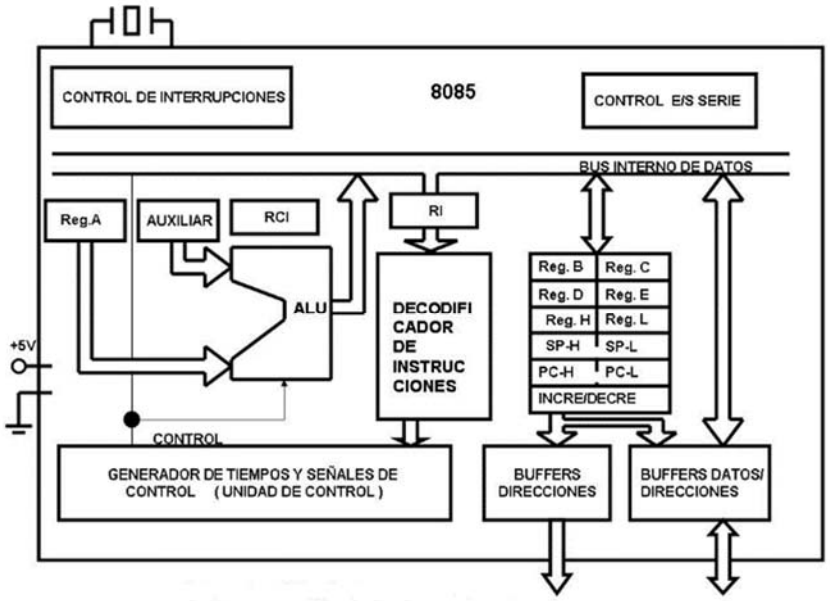
Se define como *microcomputadora* a la computadora digital en la que la CPU está formada por un microprocesador y sus componentes auxiliares por circuitos integrados que forman la familia del microprocesador (memoria, interfazs, decodificadores, *buffers*, puertos de entrada/salida, etc.). En la siguiente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



Sistema mínimo con base en 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, los cuales se explicarán a continuación.



Arquitectura de 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 por 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.

→ **Desde el punto de vista de la ingeniería del software** los requerimientos parten del mismo análisis inicial del sistema. Es una etapa temprana dentro del desarrollo del sistema que se enfoca en la obtención, análisis, especificación y validación de los requerimientos para el software.

Dentro de la ingeniería de software, entendemos como **requerimientos** a "las declaraciones que identifican atributos, capacidades, características y/o cualidades que necesita cumplir un **sistema** para que tenga valor y utilidad para el usuario". [1] Es decir, un requerimiento muestra todos los elementos que son necesarios para la construcción del sistema.

→ **Desde el punto de vista del software**, un requerimiento es una serie de elementos básicos necesarios para que las aplicaciones funcionen de manera correcta, estos

[1] Definición de requerimiento, Diccionario web de informática, disponible en línea: http://www.alegsa.com.ar/Dic/requerimientos.php, consultado el 19/03/2011.

pueden ser, cantidad de memoria de la computadora, sistema operativo, tipo de procesador, entre otros.

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

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.

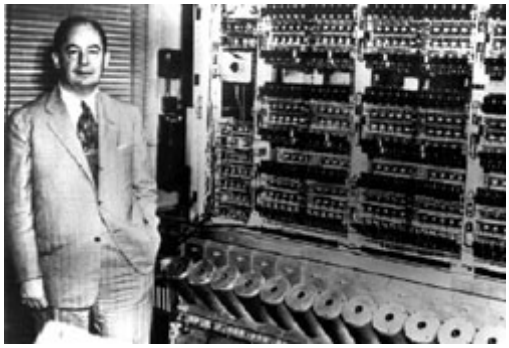
Registros de propósito especial

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.

RESUMEN

En esta unidad revisamos la estructura y organización básica de las computadoras digitales, precisando que aunque la diferencia entre ambos conceptos es relativa, se puede diferenciarlas. La organización se refiere a la distribución y conexiones entre las diferentes unidades funcionales, mientras que la arquitectura se refiere al modelo de estructura que se sigue para integrar una computadora, en función de sus requerimientos y funciones. El modelo más utilizado es el de von Neumann el cual permitió el salto para un funcionamiento eficiente de los recursos informáticos. Las aportaciones fundamentales de este modelo es el uso de la codificación binaria de datos, el uso de memorias para almacenar, compilar y ejecutar secuencias de instrucciones y el concepto de programa almacenado el cual puede ser utilizado en diversas tareas posteriores, de esta manera los programas se pueden aplicar en múltiples procesos con solo cambiar los datos de entrada.



Von Neumann y su modelo de organización de las computadoras

El modelo anterior ha permitido el rápido desarrollo de las computadoras. La computadora es un dispositivo electromecánico que recibe datos, los procesa y entrega resultados de información procesada. Las partes básicas de una computadora son:

La unidad central de procesamiento o CPU, que es la que controla el funcionamiento y realiza las tareas de procesamiento. Su elemento esencial es el microprocesador.

Unidad central de procesamiento (CPU)

Unidad que controla el funcionamiento y realiza las tareas de procesamiento. Su elemento esencial es el microprocesador.

Memoria

Son los dispositivos electrónicos o electromecánicos que almacenan instrucciones, datos o direcciones en forma binaria.

Unidades de entrada y salida

Son los dispositivos mediante los cuales se ingresa o recoge la información.

Buses

Son los canales por los que circula la información en la computadora.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga (2010)	1 Anexos	1-24 180-192
Mano (1986)	1 Anexos	1-4 372-377
Stallings (2006)	1 Anexos	8-26 438-484
Tanenbaum (2000)	1	16-19

Mano, Morris. (1986). *Lógica Digital y diseño de computadores.* México: Prentice Hall.

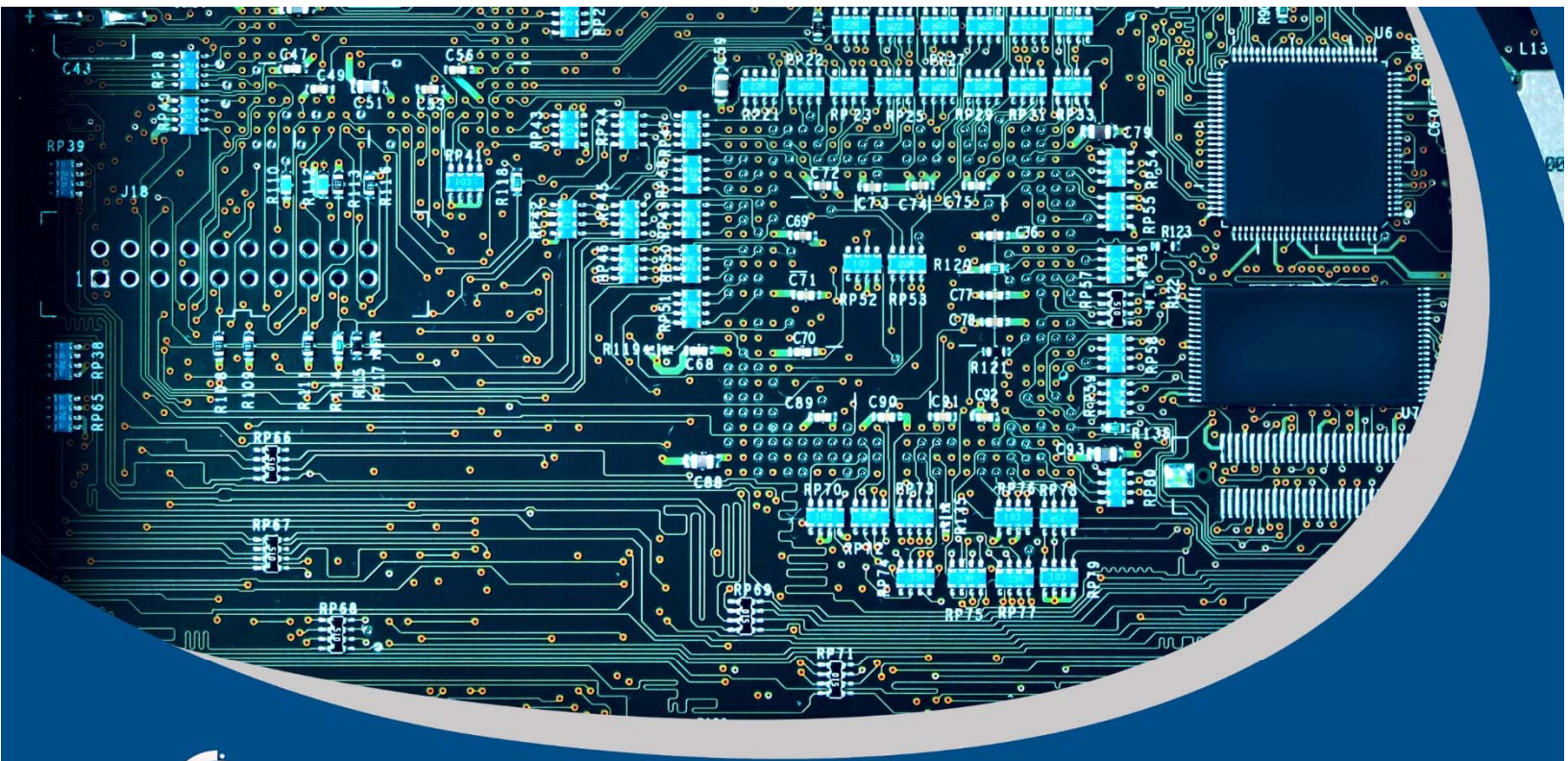
Quiroga, Patricia. (2010). *Arquitectura de computadoras.* México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores.* Madrid: Prentice Hall.

Tanenbaum, Andrew S. (2000). *Organización de computadoras. Un enfoque estructurado*. México: Prentice Hall.

UNIDAD 2

Sistemas de numeración



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno reconocerá los fundamentos teóricos de los sistemas numéricos, las conversiones de bases y operaciones con sistemas numéricos.

TEMARIO DETALLADO
(8 horas)

2. Sistemas de numeración

- 2.1. Sistemas numéricos posicionales
  - 2.1.1. Concepto y ejemplos de sistemas numéricos
- 2.2. Conversión entre bases
  - 2.2.1. Sistema decimal
  - 2.2.2. Sistema binario
  - 2.2.3. Sistema Octal
  - 2.2.4. Sistema hexadecimal
  - 2.2.5. Sistemas de base “n”
- 2.3. Aritmética binaria y en diferentes bases
  - 2.3.1. Operaciones aritméticas
  - 2.3.2. Representación de números enteros y Reales en punto flotante
    - 2.3.2.1. Complementos a la base y a la base disminuida (a r y ar-1)
    - 2.3.2.2. Magnitud y signo
  - 2.3.3. Operaciones Aritméticas con números sin signo
  - 2.3.4. Operaciones aritméticas con números con signo

INTRODUCCIÓN

Para la mayoría de nosotros el sistema numérico base 10 es algo “natural”, sin embargo si se establecen reglas de construcción basadas en otros dígitos, la posibilidad de contar con otras secuencias numéricas y sistemas numéricos, es posible. Las computadoras utilizan el sistema numérico binario. A diferencia del sistema decimal, el binario sólo utiliza dos dígitos: 0 y 1. Entender cómo se realizan las operaciones básicas aritméticas nos ayudará a entender cómo las computadoras procesan uno de los dos tipos de datos con los que las alimentamos, los numéricos.

El adquirir habilidad para el manejo de los sistemas binario, octal y hexadecimal, al igual que cuando se aprende a hablar otra lengua, no sólo significa cambiar la forma de expresar con diferentes señales o símbolos el mismo concepto, idea o entidad, sino adquirir la forma de construcción mental de dichas entidades y sus relaciones. Significa aprender a pensar de manera diferente, con diferente estructura y lógica. El lenguaje no sólo nombra la realidad sino que la ordena, interpreta y finalmente la transforma. Obtener la habilidad para manejar cantidades binarias significa “pensar” o por lo menos estructurar procesos como lo hace la computadora. De esta manera, si nuestra labor profesional está ligada a las computadoras, el entender la forma como interpretan y representan datos es una forma de poder comunicarnos y en consecuencia actuar sobre ellas.

En esta unidad abordamos de manera general qué son los sistemas numéricos. Todos tienen ciertas características que los hacen distinguirse como tales, es decir cumplen con algunas reglas de validación y estructura que los hicieron útiles a las culturas que los crearon. Podemos decir que por convención, es decir por acuerdo entre un grupo de personas, se reconoce a un mismo signo para una misma entidad o idea por todos aceptada. Con esto se crea la representación de ideas mediante símbolos. Los

primeros elementos que le permitieron al hombre identificar “cantidades” diferentes fueron los dedos y de ahí la palabra dígito. La abstracción de la necesidad de medir dio origen a los números, pues estos ya no están asociados necesariamente a las tareas de medición, son abstracciones. Uno de estos sistemas es el binario y es el que emplean las computadoras en sus cálculos. Aunque el sistema numérico que utilizamos generalmente es el decimal, necesitamos comprender las reglas de conversión entre ambos y los sistemas que se relacionan directamente con el binario como son el octal y el hexadecimal.

2.1. Sistemas numéricos posicionales

2.1.1. Concepto y ejemplos de sistemas numéricos

Un sistema numérico es un conjunto de símbolos y reglas de asociación con los que se pueden generar cantidades válidas para el conjunto definido por el sistema.

Cada sistema de numeración se caracteriza por su base, de manera que para la mayoría de nosotros, el sistema numérico base 10 es algo “natural”; sin embargo, no es el único que existe.

El sistema de base n más ampliamente usado para el diseño y construcción de pequeños sistemas digitales hasta una computadora digital es el sistema binario, pero para la programación de dichos sistemas digitales se utilizan los sistemas binario, octal, decimal y hexadecimal.

2.2. Conversiones 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 de su posición.

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

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

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 encuentre más a la izquierda el de "**más significativo**". En los dígitos fraccionales esta consideración sigue siendo válida.

La tabla de equivalencias entre diferentes sistemas de numeración, 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

Equivalencias entre diferentes sistemas de numeración

Entre los sistemas de numeración más utilizados en la informática se encuentran los sistemas de numeración Decimal, Binario, Octal y Hexadecimal.

2.2.1. Sistema decimal

Emplea 10 diferentes signos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9).

2.2.2. Sistema binario

Emplea dos signos (0 y 1)

2.2.3. Sistema Octal

Emplea 8 dígitos (0, 1, 2, 3, 4, 5, 6 y 7)

2.2.4. Sistema hexadecimal

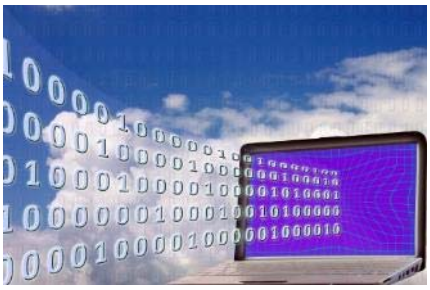
Emplea 15 signos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F)

Para profundizar más sobre el tema Sistemas de numeración, descarga y estudia la presentación *Sistemas Numéricos* **(ANEXO B)**.

2.2.5. Sistemas de base “n”

Es el sistema de numeración considerado.

2.3. Aritmética binaria y en diferentes bases



En una computadora digital, las operaciones aritméticas se realizan en el sistema binario porque el diseño y construcción de circuitos lógicos (ver Unidad 5, temas 5.5 y 5.6) para realizar aritmética binaria es mucho más sencilla que para la aritmética decimal.

2.3.1. Operaciones aritméticas

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. consulta el **(ANEXO C)**. *Aritmética binaria*, en el cual se explica cómo se realiza la suma, resta, multiplicación y división en los sistemas de numeración binaria, octal y hexadecimal.

2.3.2. Representación de números enteros y Reales en punto flotante

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)

2.3.2.1. Complementos a la base y a la base disminuida (a r y ar-1)

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

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

El complemento de 2 de (1001) es

$$(2^4)_{10} - (1001)_2 = (10000)_2 – 1001 = 00111 \text{ (con } p = 4).$$

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 que se va 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 n^p-n^{-q} – N. A continuación presentamos algunos ejemplos.

El complemento de 9 del número $(327)_{10}$ es

$$10^3 - 1 - 327 = 672 \text{ (con } p = 3\text{), y}$$

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

El complemento de 9 del número $(0.173)_{10}$ es

$$1 - 10^3 - 0.173 = 0.826 \text{(con } q = 3\text{), y}$$

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

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

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

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

2.3.2.2. Magnitud y signo

Para la representación de los números binarios con signo debemos considerar dos conceptos: magnitud y signo. Para el caso de la magnitud podemos asociar dos ideas: la magnitud como una propiedad física que es medible (por ejemplo longitud, superficie, volumen, peso tiempo, etc.) y la magnitud como una característica de dimensión o tamaño. Para nuestro caso, como los números binarios no están asociados a magnitudes físicas, emplearemos la magnitud como un concepto de tamaño. De manera práctica podemos decir, de cuántos bits o dígitos binarios está formada una secuencia de caracteres binarios. Por ejemplo:

10100010	es un número binario de 8 bits
111010000111	es un número binario de 12 bits
347FA	es un número hexadecimal de 5 dígitos hexadecimales
3984750	es un número decimal de 7 dígitos decimales

2.3.3. Operaciones Aritméticas con números sin signo

Veamos la operación de suma y resta de números binarios, empecemos por la suma.

Suma binaria sin signo

Para realizar la suma de números binarios, hay que recordar cuatro combinaciones posibles.

0 + 0 = 1

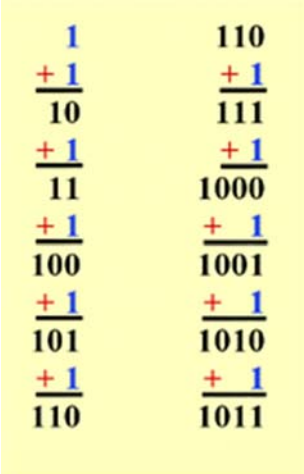
0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

En el caso de 1 + 1 es cero y se arrastra una unidad, que se suma a la izquierda, esto se conoce como bit de acarreo.

Veamos la suma consecutiva de números.



Para realizar la resta de números binarios, hay que recordar cuatro combinaciones posibles.

0 - 0 = 0.

0 - 1 = 1 (con acarreo negativo de 1)

1 - 0 = 1.

1 - 1 = 0.

Veamos un ejemplo de resta

Restamos 17 - 10 = 7

10001

-01010

_____

01111

Fuente
http://www.asifunciona.com/informatica/af_binario/af_binario_5.htm
http://centros.edu.xunta.es/iesmanuelchamosolamas/electricidade/fotos/numeracion.htm

2.3.4. Operaciones aritméticas con 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).

El problema es 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 Interpretación del signo. 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 $(1111)_2$.

Tabla

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

Interpretación del signo

En columna 2 de la Tabla Interpretación del signo 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, etcétera 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 Representación de números en diferentes bases con signo 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 Representación de números en diferentes bases con signo

A partir de la tabla Representación de números en diferentes bases con signo 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.

Operaciones aritméticas con números asignados

En esta sección presentamos las operaciones aritméticas números principalmente con signo negativo, 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)

Complemento a 1

0 0 0 1 0

\+ 1

0 0 0 1 1

Finalmente realizamos la operación de suma

Comprobación

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

Bit de Acarreo

$(17)_{10}$

$+$

$(-29)_{10}$

$(-12)_{10}$

Caso b) Sumando mayor que el sustraendo

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

Solución.

Comprobación

```
  0000  1111
+
  1111  0101
 ------------
 10000  0100
```

$(+15)_{10}$

$+$

$(-11)_{10}$

$(+04)_{10}$

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

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.

$$\begin{array}{r} -13_{10} \\ -11_{10} \\ \hline -24_{10} \end{array}$$

$(+13)_{10} \Rightarrow (0D)_{16}$ $(+11)_{10} \Rightarrow (0B)_{16}$

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.

$$(-19957)_{10}$$
$$+$$
$$(-10999)_{10}$$
$$(-30956)_{10}$$

$(+19957)_{10} \Rightarrow (4DF5)_{16}$ $(+10999)_{10} \Rightarrow (2AF7)_{16}$

Complemento a 15 del número (-19957) 10

F F F F
-
4 D F 5

B 2 0 A
+
1

B 2 0 B

Complemento a 15 del número (-10999) 10

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

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

$$\begin{array}{r} B\ 2\ 0\ B \\ + \\ D\ 5\ 0\ 9 \\ \hline 1\ 8\ 7\ 1\ 4 \end{array}$$

Bit de signo

Finalmente la suma de

$$\begin{array}{r} -19957_{10} \\ + \\ -10999_{10} \\ \hline -30956_{10} \end{array} \qquad \begin{array}{r} B\ 2\ 0\ B_{16} \\ + \\ D\ 5\ 0\ 9_{16} \\ \hline 8\ 7\ 1\ 4_{16} \end{array}$$

Comprobación

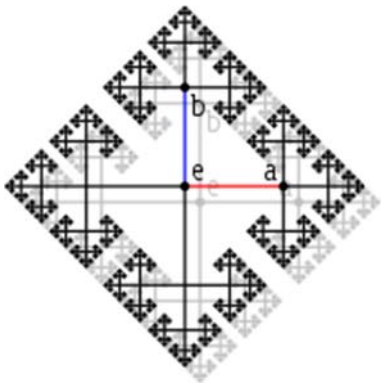
Para comprobar el resultado, obtenemos el valor absoluto del resultado de la suma de la manera siguiente

$$\begin{array}{r} F\ F\ F\ F \\ - \\ 8\ 7\ 1\ 4 \\ \hline 7\ 8\ E\ B \\ + \\ 1 \\ \hline 7\ 8\ E\ C \end{array}$$

RESUMEN

La unidad está dividida en dos temas. La estructura algebraica para la construcción de cualquier número en un sistema base n que se puede aplicar a cualquier sistema numérico. Tenemos habilidad para reconocer de inmediato cualquier cifra en el sistema decimal sin necesidad de estar elaborando la notación extendida, esto se debe a que desde niños hemos estado en contacto con su construcción empleando el número 10 como base. Sin embargo, como se vio en la unidad anterior, la forma de representar cantidades y manejarlas en las computadoras es mediante los números binarios.

Las computadoras utilizan dispositivos electrónicos que pueden mantener dos estados de voltaje, alto y bajo, en consecuencia el sistema binario, aunque ya se había desarrollado desde el siglo XVII, se empezó a aplicar a las computadoras hasta los años cuarentas del siglo pasado. Es importante en nuestro curso entender por lo tanto el funcionamiento de este sistema.



En el primer tema se describió la forma de conversión del sistema decimal al binario y por extensión, del sistema decimal a cualquier sistema de base diferente. El manejo de números binarios no nos es familiar, sin embargo el sistema hexadecimal, derivado fácilmente del binario, nos es más informativo. Le conversión entre estos dos sistemas y el octal es muy sencilla por lo que también se abordaron en este tema. Finalmente se explicó la conversión inversa, cómo pasar una cifra en base n a base diez y específicamente de base 2, 8 y 16 a base 10.

En el segundo tema se trató el fundamento de las operaciones aritméticas en base 10 y desde ahí se explicaron las diferentes operaciones básicas en diversos sistemas, para ello se desarrollaron varios ejemplos en diferentes bases. Las operaciones explicadas fueron suma, resta, multiplicación y división.

Adicionalmente se presentaron los conceptos de números signados, su relación y representación a partir del concepto ‘complemento’. Estos dos conceptos son importantes debido a que la operación de resta en el sistema binario y por lo tanto en las computadoras se realizan empleando el concepto de complemento a la base n y a la base disminuida, n menos uno.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga (2010)	1	1-14 11-16 180-192
Mano (1986)	1	4-10 18 26
Stallings (2006)	Apéndice A	725-731

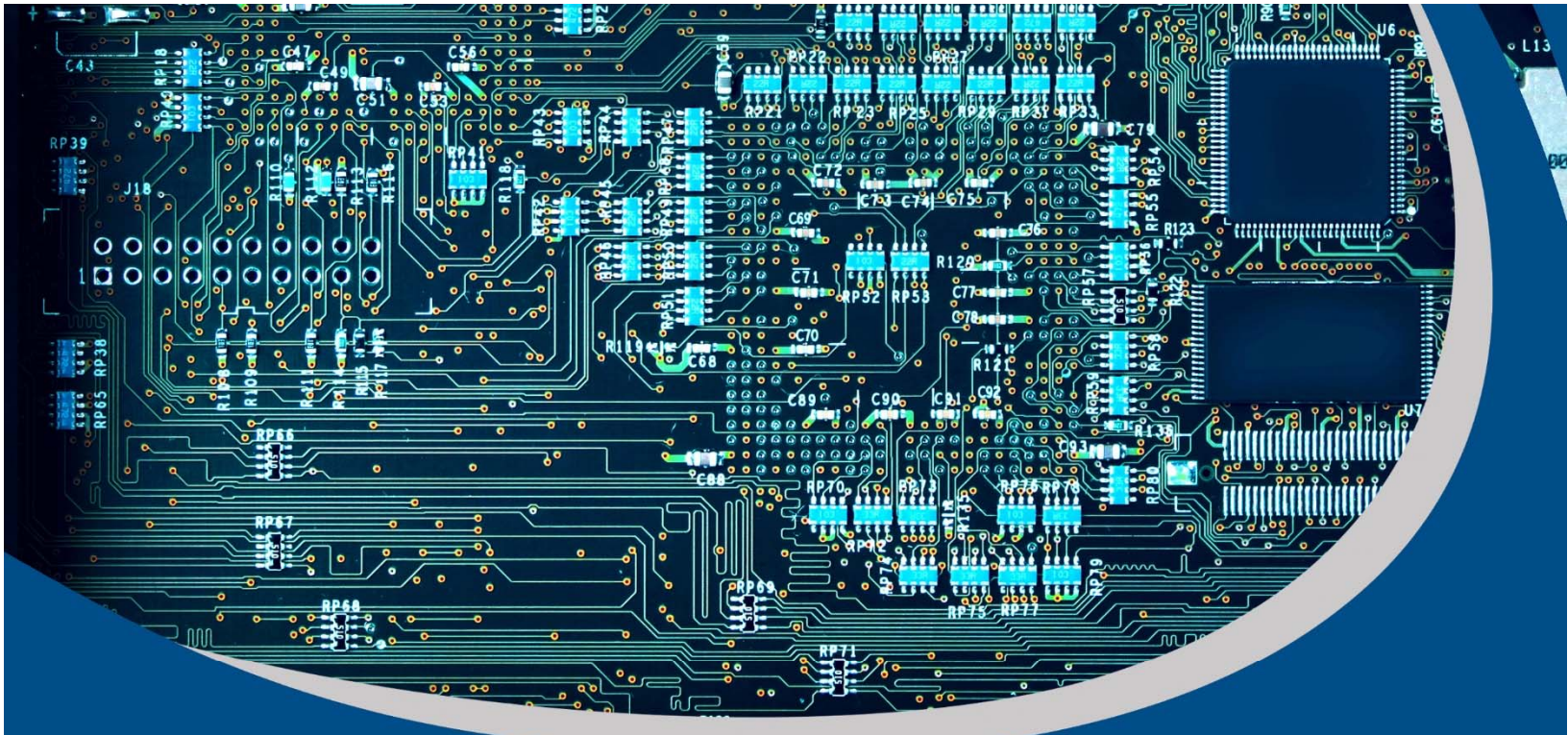
Mano, Morris. (1986). *Lógica Digital y diseño de computadores*. México. Prentice Hall Hispanoamericana.

Quiroga, Patricia. (2010). *Arquitectura de computadoras*. México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores*. Madrid: Prentice Hall.

UNIDAD 3

Códigos



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno podrá realizar representaciones de cantidades en diferentes códigos y secuencias y generar códigos de detección de errores.

TEMARIO DETALLADO
(8 horas)

3. Códigos

- 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

Las computadoras digitales emplean el sistema binario para representar y manipular cualquier información. Lo anterior implica que las señales que se manejan en el mundo real, señales analógicas, tienen que ser representadas en los sistemas informáticos empleando solamente los símbolos uno y cero. Existen diferentes formas de codificar estas señales, sin embargo, podemos especificar que sólo tenemos dos tipos de caracteres en las computadoras, numéricos y alfanuméricos. En el primer caso ya se ha visto en la unidad dos, la forma de manipulación de estos. Sin embargo, también los caracteres numéricos se pueden representar (codificar) en diferentes formas pero manteniendo la estructura y formas de manipulación de un sistema decimal.

En la primera parte de esta unidad se describen los códigos binarios, BCD, X3 y gray. En la segunda se describen las formas de codificación de los caracteres alfanuméricos: ASCII, BCDIC y EBCDIC; finalmente en la tercera se explica el funcionamiento de los códigos de detección de error de paridad.

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.

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" basado solamente en 2 dígitos: 0 y 1, los cuales, dentro de un "sistema", son interpretados de diversas maneras, tales como: SÍ 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 Código Binario 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**

Código Binario

3.1.2. BCD

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 Código BCD 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**

Código BCD

A partir de la tabla Código BCD, 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).

```
   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 Código de Exceso en tres 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

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 escriben 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,

$$\begin{array}{r} 3 \\ +\ 9 \\ \hline 12 \end{array} \qquad \begin{array}{r} 0110 \\ +\ 1100 \\ \hline 1\ \ 0010 \\ + \\ 0011\ \ 0011 \\ \hline 0100\ \ 0101 \end{array}$$

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. 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 más abajo en la parte 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 Código Gray 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

Código Gray

A partir de la tabla Código Gray 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.

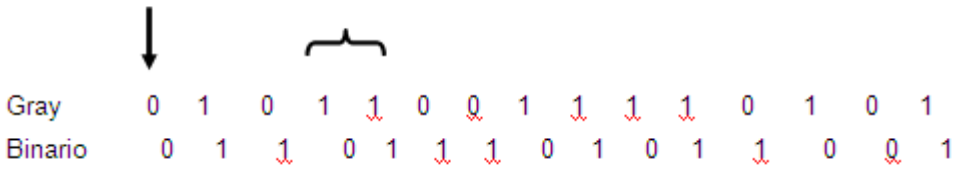
Gray	1	011010111001
Binario	1	101100101110

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

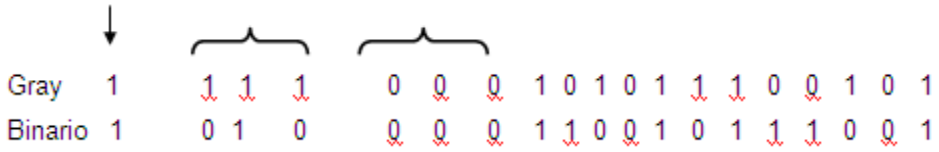
No cambia el bit del binario	Cambia el bit del binario



Primer 1

Ejemplo

Se repite el primer uno	Cambia el bit precedente del binario	Mantiene como estaba al bit del binario



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 que se va 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.

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 Códigos ANSCII y EBCDIC 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

Códigos ANSCII y EBCDIC

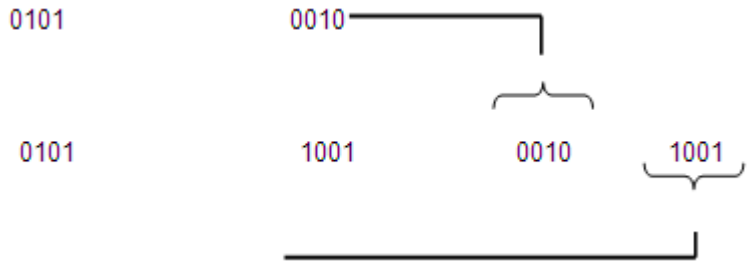
donde:

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).

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 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 la computadora 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 dígito 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.

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.[2]

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 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 Códigos ANSCII y EBCDIC 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

[2] 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.

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:

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

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:

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

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

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 Paridad en el código BCD 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

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 Código Biquinario 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

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 la computadora, 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 Código Biquinario 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.

RESUMEN

La forma de representación de datos en las computadoras se realiza mediante los símbolos cero y uno, que corresponden a los valores que manejan los dispositivos electrónicos con los que están construidas las computadoras digitales. Las reglas de asociación de estos dos valores permiten la generación de estándares de representación denominados códigos. Así, un código es un conjunto de símbolos y reglas de relación para representar información de manera sistemática y estandarizada.

En la teoría de la información, código es la forma que toma la información que se intercambia entre la fuente (el emisor) y el destino (el receptor) de un ciclo de comunicación. Un código implica la comprensión o decodificación del paquete de información que se transfiere, pues además de definir los símbolos por utilizar para la representación de la información, define también las reglas de utilización de dichos símbolos.

Los principales códigos utilizados para definir datos numéricos son el código BCD, X3 y Gray. Este último tiene la ventaja de facilitar la identificación de errores, por ser autocomplementado. Es importante señalar que codificar y convertir una cantidad numérica no tienen el mismo significado. Convertir un número decimal a una base diferente implica que las operaciones aritméticas serán realizadas en el sistema base al que se realiza la conversión, mientras que la codificación representa cada dígito de

una manera diferente (en este caso secuencias binarias para cada dígito) pero manteniendo las estructuras y operaciones de un sistema decimal.

Para representar información alfanumérica en forma binaria, la computadora emplea los códigos ASCII, BCDIC y EBCDIC, los cuales se indicaron en las tablas correspondientes en el desarrollo de la unidad.

Finalmente, debido a que la transferencia de información en un sistema informático es susceptible de errores, se hace necesaria la adición de bits de codificación de error que indiquen la existencia de diferencias entre la información emitida y recibida por las diferentes unidades de una computadora o entre redes de computadoras. Existen diversos códigos de detección y corrección de error, el más común es el de paridad, el cual agrega simplemente un bit a la cadena de bits de la secuencia de información de acuerdo a la regla par o impar.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Mano (1986)	1	16-20
Quiroga (2010)	2	56-72

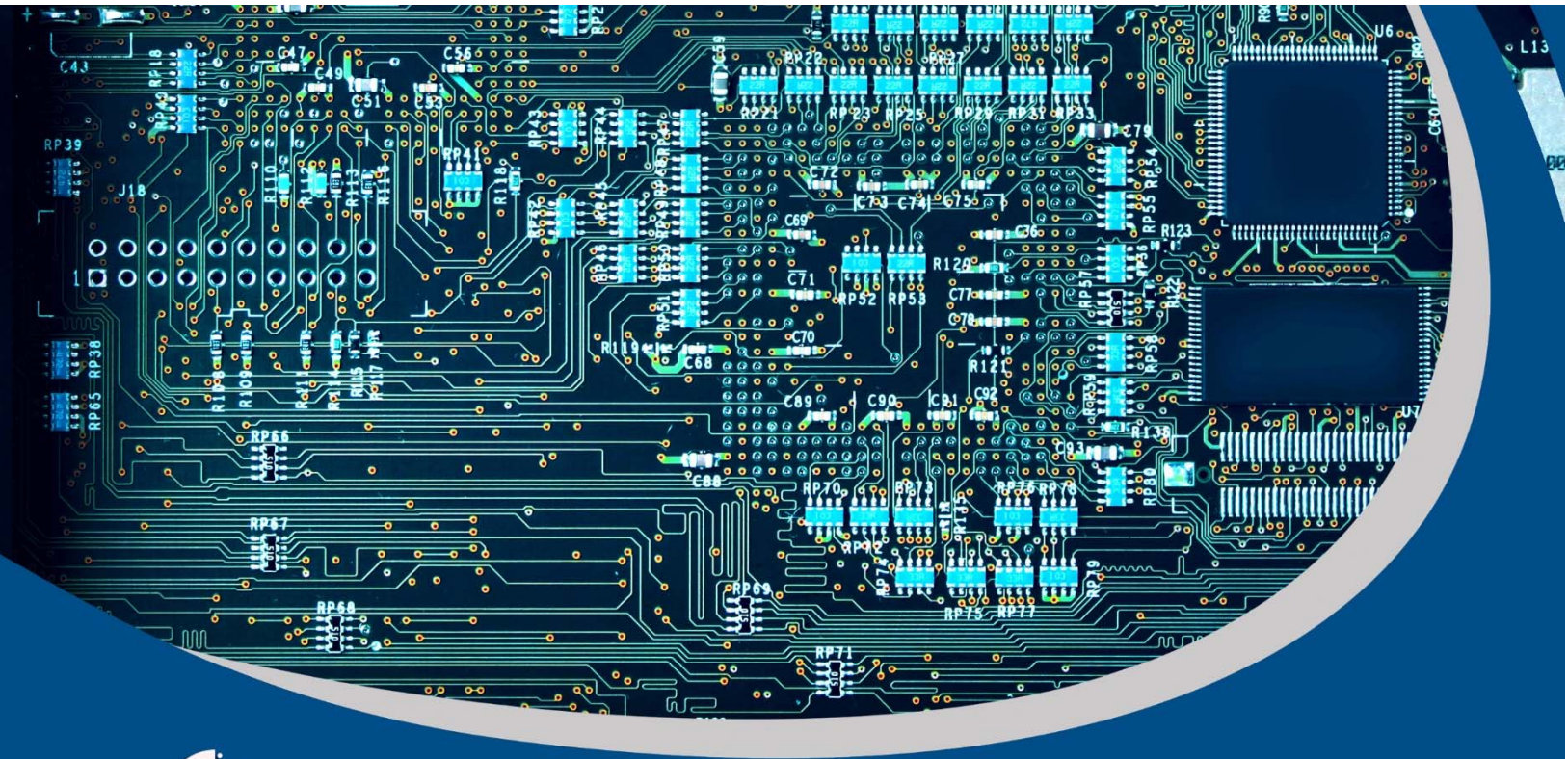
Mano, Morris. (1986). *Lógica Digital y diseño de computadores*. México: Prentice Hall Hispanoamericana.

Quiroga, Patricia. (2010). *Arquitectura de computadoras*. México: Alfaomega.

Licenciatura: Informática

UNIDAD 4

Álgebra de boole



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno explicará los principios fundamentales del álgebra y funciones booleanas y los procesos algebraicos.

TEMARIO DETALLADO
(8 horas)

4. Álgebra de Boole

- 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

La tecnología nos permite construir compuertas digitales a través de transistores y mediante las compuertas diseñamos los circuitos digitales empleados en las computadoras. Sin embargo, el empleo de esta tecnología no determina por sí solo la aparición de las computadoras como procesadores de información, es necesaria la aplicación de principios lógicos y algebraicos que nos permitan manipular, con rigor matemático, variables mediante dispositivos electrónicos. La forma como las computadoras realizan operaciones lógicas es mediante el álgebra de Boole aplicada a los circuitos electrónicos. El álgebra booleana es importante pues permite la sistematización y representación matemática del funcionamiento de los circuitos electrónicos digitales. La sistematización del estudio de los circuitos electrónicos digitales ha tenido tres momentos importantes:

En 1854 George Boole presentó un tratamiento sistemático de la lógica binaria en su libro *Investigación sobre las leyes del pensamiento*.

En 1904 Edward Vermilye Huntington presentó una serie de postulados algebraicos para determinar formalmente los sistemas algebraicos.

En 1938 Claude E. Shannon demostró que los circuitos digitales electrónicos pueden modelarse formalmente utilizando el álgebra de Boole.

Para entender el funcionamiento de las computadoras, es necesario entender los principios, axiomas, teoremas y postulados del álgebra que nos interesa. El presente capítulo está dividido en tres temas: principios de electrónica binaria y álgebra booleana; propiedades fundamentales y tercero, técnicas de minimización de funciones. En el primero se establecen los elementos de funcionamiento de circuitos digitales; en el segundo se establecen formalmente los axiomas y postulados que le dan forma y estructura matemática al álgebra de Boole. En el último tema, se presentan las dos principales formas de minimizar funciones booleanas que son manipulación algebraica y mapas de Karnaugh.

Es importante aclarar que múltiples problemas y procesos del funcionamiento de circuitos digitales se pueden modelar mediante estas funciones y que para su diseño eficiente, estas deben representarse en muchas ocasiones en su forma mínima, por lo el proceso de minimización adquiere relevancia. En el último tema abordaremos estas formas de minimización y otras formas de representación de funciones booleanas.

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 unidad 5) y en el diseño de circuitos secuenciales basados en flip-flops, (ver unidad 6).

4.1.1. Lógica binaria

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	La Lógica positiva define al “0” lógico como falso y al “1” como verdadero.
Lógica Negativa	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 Teoremas de Álgebra de Boole

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

Teoremas de Álgebra 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 3. (Técnicas de minimización

de funciones), a continuación daremos una breve explicación del uso de las leyes De Morgan.

Las leyes de De Morgan son:

1.) $\overline{A + B + C + \ldots} = \overline{A}\,\overline{B}\,\overline{C}\ldots$

2.) $\overline{A\,B\,C\ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$

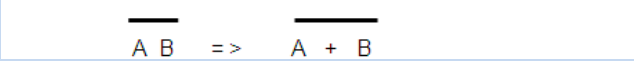
Para poder utilizar de manera correcta las leyes de De Morgan se debe aplicar los siguientes pasos:

Se intercambia el operador OR (+) por el operador AND () o si es el caso intercambiar el operador AND () por el operador OR (+).
Se niegan cada una de las variables
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.

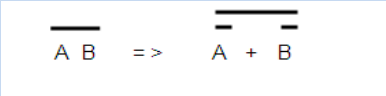
$$\overline{A\,B} = \overline{A} + \overline{B}$$

Intercambio del operador AND () por el operador (+)

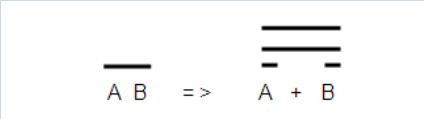


La negación del término en este paso se sigue conservando debido a que únicamente se intercambio el operador.

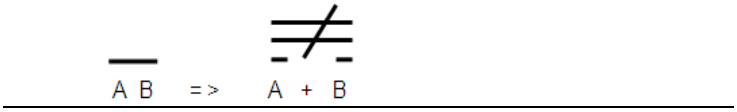
Se niega cada una de las variables



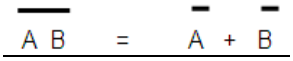
Se niega todo el término



Aplicando la propiedad de involución (ver tabla 4.1) al resultado anterior



con lo cual se obtiene el resultado



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 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 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”.

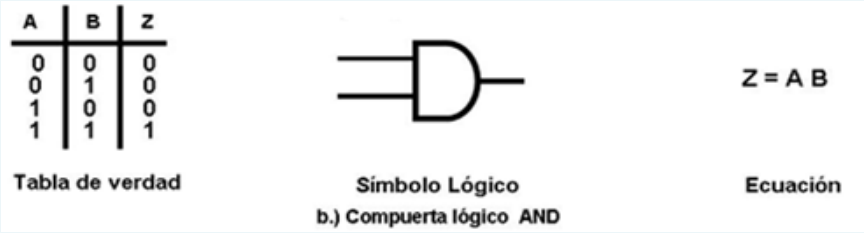


Figura 1a

En la tabla de verdad de la figura Compuertas básicas, el dígito binario 1 representa un nivel alto de voltaje, y el dígito binario 0 un nivel bajo de voltaje.

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Tabla de verdad

Símbolo Lógico

a.) Compuerta lógico OR

$Z = A + B$

Ecuación

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Tabla de verdad

Símbolo Lógico

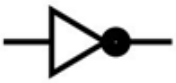
b.) Compuerta lógico AND

$Z = A\,B$

Ecuación

A	Z
0	1
1	0

Tabla de verdad



Símbolo Lógico

c.) Compuerta lógico AND

$Z = \overline{A}$

Ecuación

Figura Compuertas básicas

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 1b** muestra el símbolo lógico, su tabla de verdad y su ecuación característica de dicha compuerta.

Figura 1b

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 1c) presenta el símbolo lógico, la tabla de verdad y la ecuación característica de la compuerta NOT.

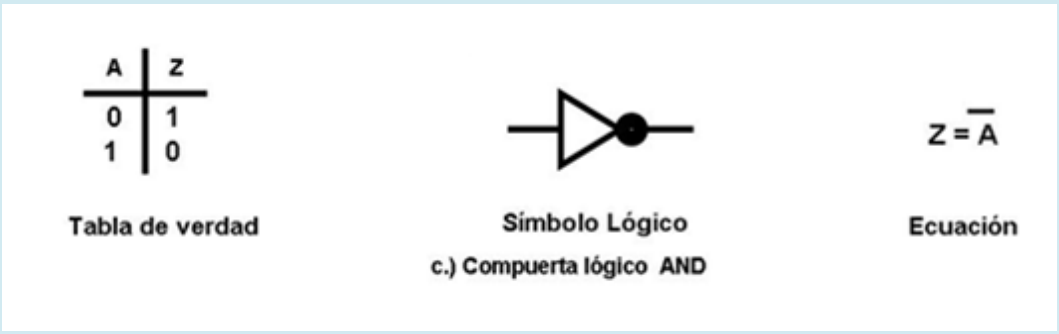


Figura 1c.

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 Compuertas complementarias. Las razones de la popularidad de las puertas inversoras (NOR y NAND) son:

a) Baratas

b) Rápidas, y

c) Disipan menos potencia

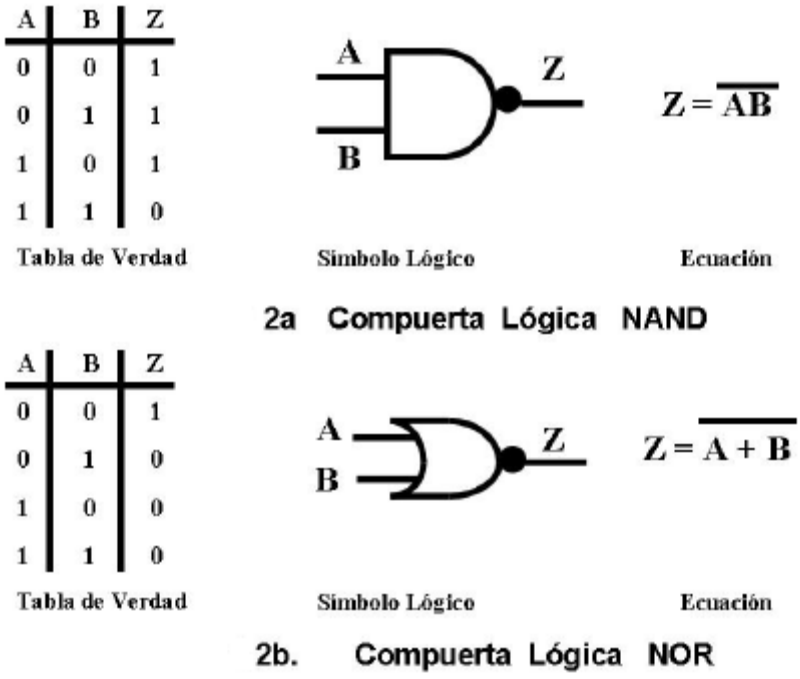
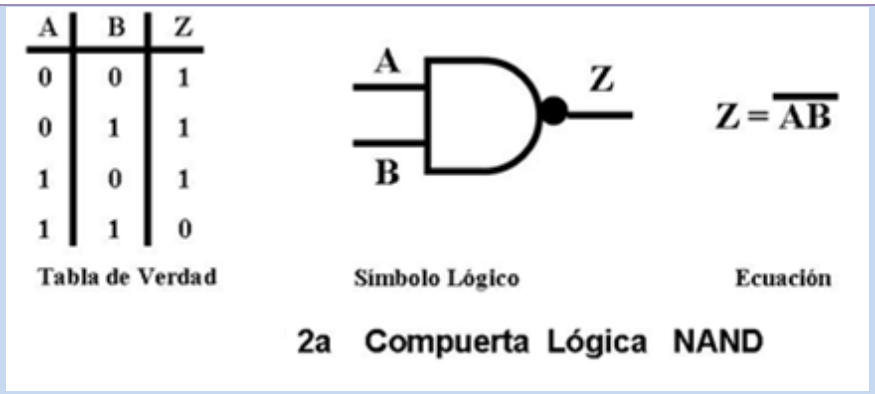


Figura Compuertas Complementarias

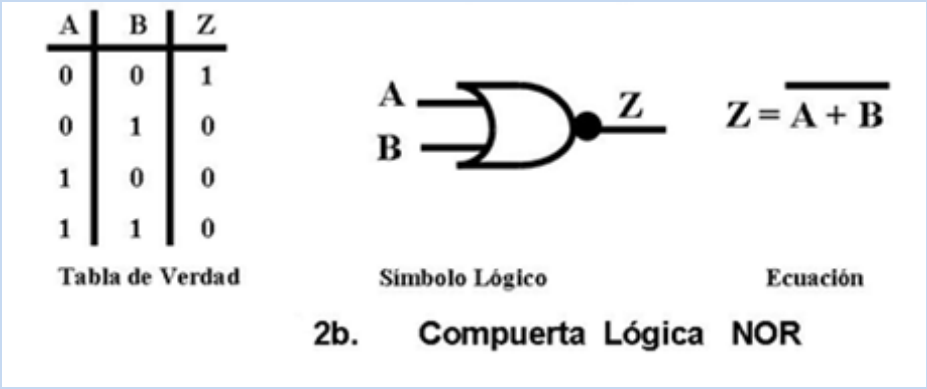
Compuerta NAND	La compuerta NAND es equivalente a una compuerta AND seguida de una compuerta NOT, tal como se muestra en la figura 2a). El funcionamiento de esta compuerta es el siguiente: La salida presenta un nivel bajo solamente si todas las entradas están en nivel alto (“1”). La salida tiene un nivel alto si cualquiera de las entradas está en nivel bajo “0”).



2a Compuerta Lógica NAND

Compuerta NOR

La compuerta NOR es equivalente a una compuerta OR seguida de una compuerta NOT, tal como se muestra en la figura 2b.

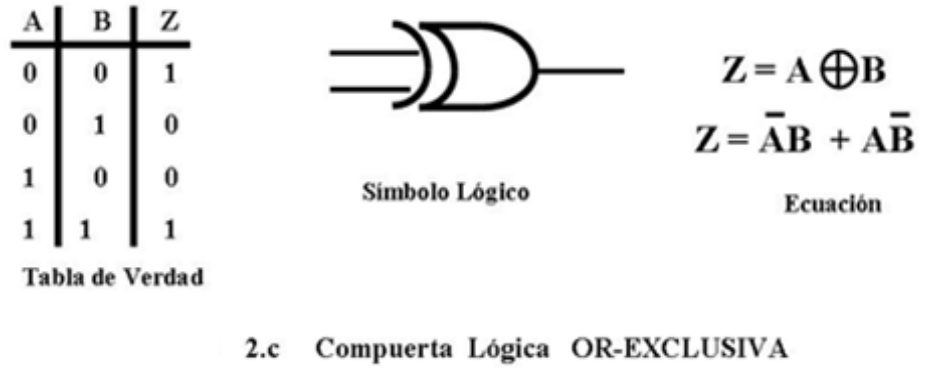
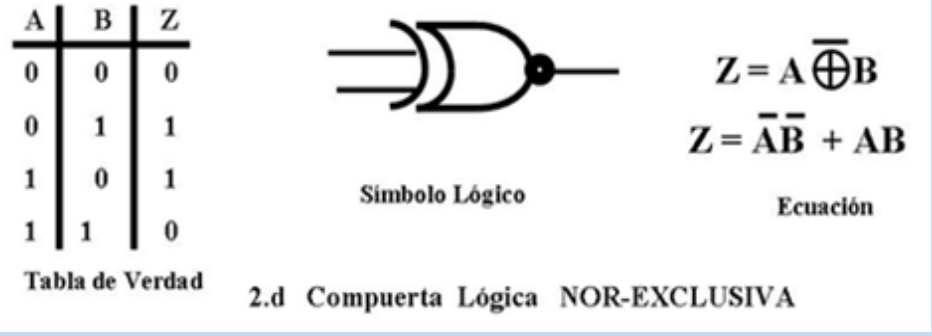


2b. Compuerta Lógica NOR

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 2b. se presenta el símbolo lógico, tabla de verdad y la ecuación característica de la compuerta NOR.

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 2c. muestra el símbolo lógico, tabla de verdad y la ecuación característica.

	 2.c Compuerta Lógica OR-EXCLUSIVA
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 2d muestra el símbolo lógico, tabla de verdad y ecuación característica.  2.d Compuerta Lógica NOR-EXCLUSIVA

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.

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 Algebre 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

$$f(A,B,C) = \underset{I}{\overline{AB + C}} + \underset{II}{\overline{\bar{A}C\bar{B}}} + \underset{III}{AC(B + BA)}$$

Factorizando el término III y utilizando el teorema de complemento

$$f(A,B,C) = \underset{I}{\overline{AB + C}} + \underset{II}{\overline{\bar{A}C\bar{B}}} + \underset{III}{AC(B(1 + \cancel{A}))}$$

y ordenando (propiedad conmutativa) la ecuación

$$f(A,B,C) = \overline{AB + C} + \overline{\bar{A}\bar{B}C} + ABC$$

Aplicando la ley de De Morgan a los términos I y II de la expresión anterior

$$f(A,B,C) = \overline{\overline{AB} + \bar{C}} + \overline{\bar{A}\bar{B} + C} + ABC$$

Con lo cual obtenemos la expresión

$$f(A,B,C) = (\overline{AB} + \bar{C}) + (\overline{\bar{A}\bar{B}} + \bar{C}) + ABC$$

A la expresión anterior aplicamos la ley de De Morgan a los términos I y II

$$f(A,B,C) = \overline{(\bar{A} + \bar{B})}\,\bar{C} + \overline{(A + B + \bar{C})} + ABC$$

Se obtiene la siguiente expresión

$$f(A,B,C) = (\bar{A} + \bar{B})\,\bar{C} + (A + B + \bar{C}) + ABC$$

Utilizando la propiedad distributiva

$$f(A,B,C) = (\bar{A}\bar{C} + \bar{B}\bar{C}) + (A + B + \bar{C}) + ABC$$

Factorizando y utilizando el propiedad de complemento, se obtiene

$$f(A,B,C) = \bar{A}\,\bar{C} + C(1 + B) + A + B + ABC$$

$$f(A,B,C) = \bar{C}(A + 1) + A(BC + 1) + B$$

$$f(A,B,C) = A + B + \bar{C}$$

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 Mapa de Karnaugh para 2 variables. 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.

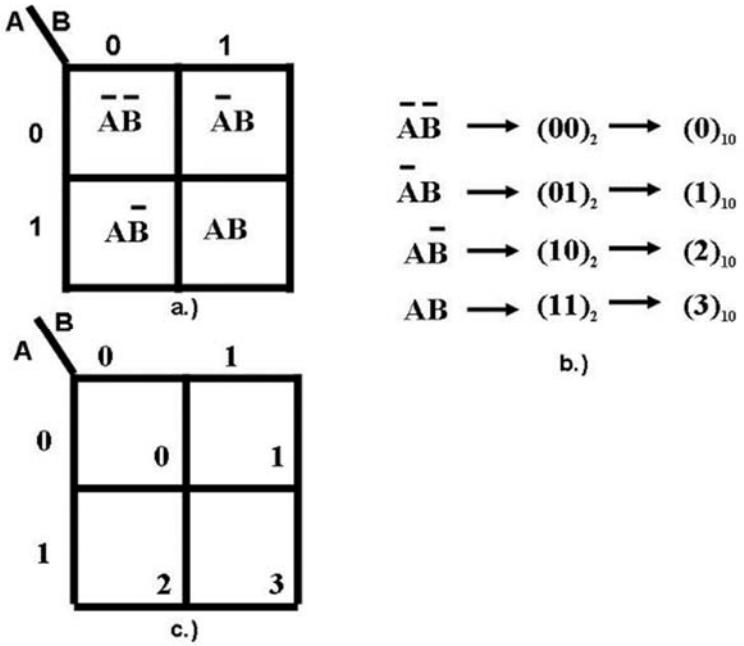


Figura Mapa de Karnaugh para 2 variables

La figura Mapa de Karnaugh para 2 variables 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 Mapa de Karnaugh para 2 variables a) La figura Mapa de Karnaugh para 2 variables b 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 Mapas de Karnaugh para 3 variables), cuatro (ver figura Mapas de Karnaugh para 4 variables) 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.

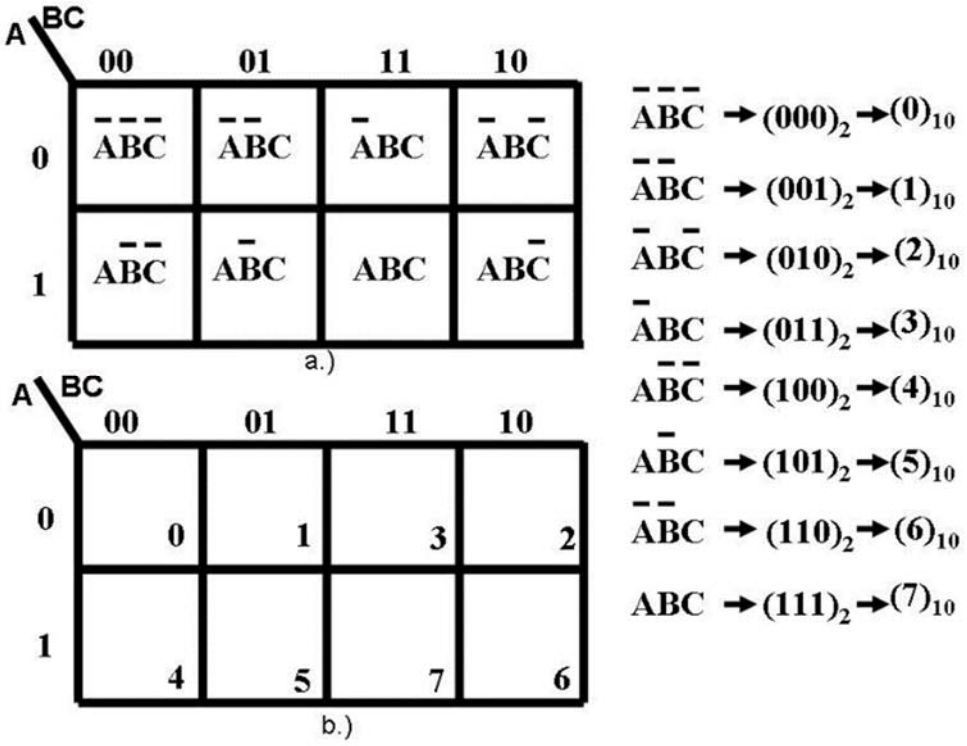


Figura Mapas de Karnaugh para 3 variables

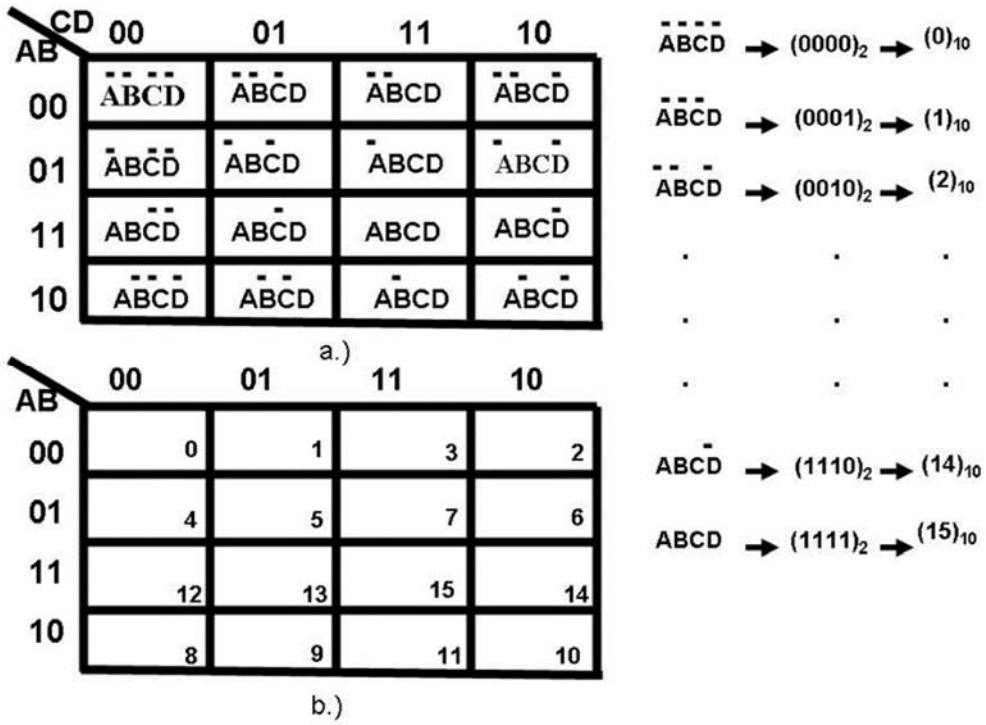


Figura Mapas de Karnaugh para 4 variables

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 (2n) celdas contiguas, si tuviera cuatro variables de entrada (n = 4) se forma o construye un Mapa de Karnaugh de 16 celdas (24 = 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.

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 Mapas de Karnaugh para 3 variables satisfacen las condiciones requeridas para aplicar la propiedad de complemento del álgebra de Boole. Dado que en el mapa Karnaugh de la figura Mapas de Karnaugh para 3 variables 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 Mapas de Karnaugh para 3 variables.

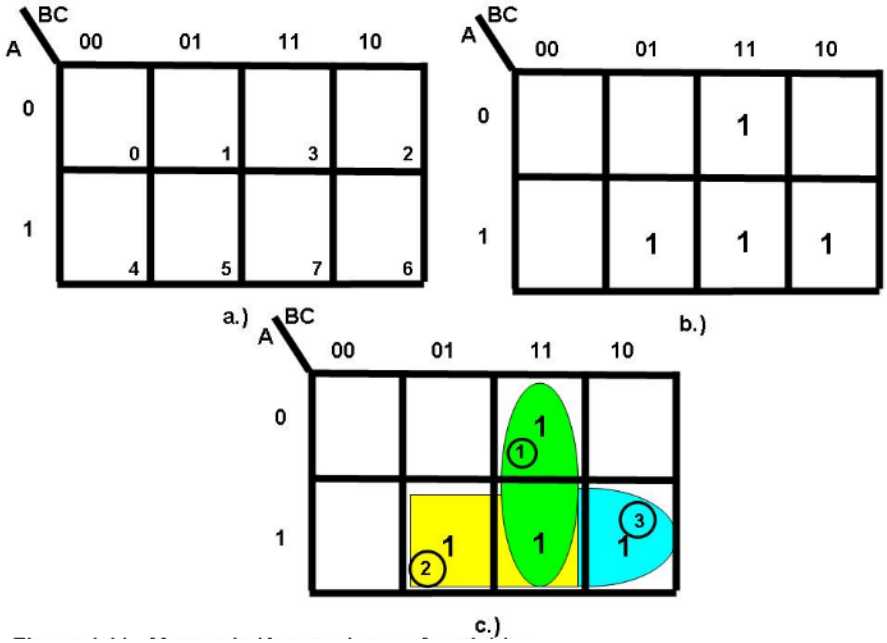


Figura Mapas de Karnaugh para 3 variables a-c

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 Mapas de Karnaugh para 3 variables 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 Mapas de Karnaugh para 3 variables 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 Mapas de Karnaugh para 3 variables 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 Mapas de Karnaugh para 3 variables 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 Circuito lógico.

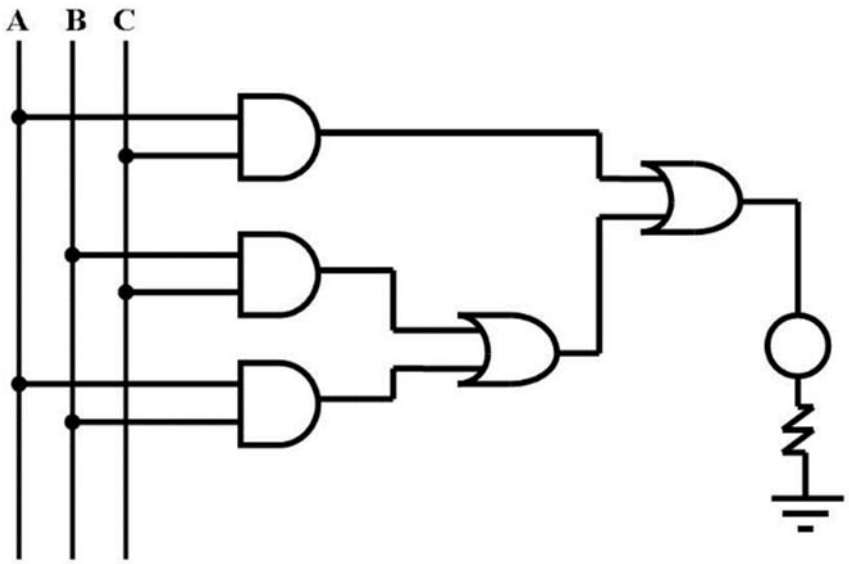


Figura Circuito lógico

Ejemplo: Utilizando los mapas de Karnaugh reduce la siguiente función

Solución:

$$f(A,B,C) = \bar{A}\,\bar{B}\,\bar{C} + A\,\bar{B}\,\bar{C} + \bar{A}\,B\,\bar{C} + A\,B\,\bar{C} + \bar{A}\,B\,C + A\,\bar{B}\,C$$

a partir de la función tenemos los términos y su equivalencia en binario y decimal.

$\bar{A}\,\bar{B}\,\bar{C} = (000)2 = (0)10$ $A\,\bar{B}\,\bar{C} = (001)2 = (1)10$ $\bar{A}\,B\,\bar{C} = (010)2 = (2)10$

$A\,B\,\bar{C} = (100)2 = (4)10$ $\bar{A}\,\bar{B}\,C = (101)2 = (5)10$ $\bar{A}\,B\,C = (110)2 = (6)10$

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 por 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 Mapas de Karnaugh para 3 variables

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 1’s. El mapa de Karnaugh con los “1”s en sus celdas es el siguiente

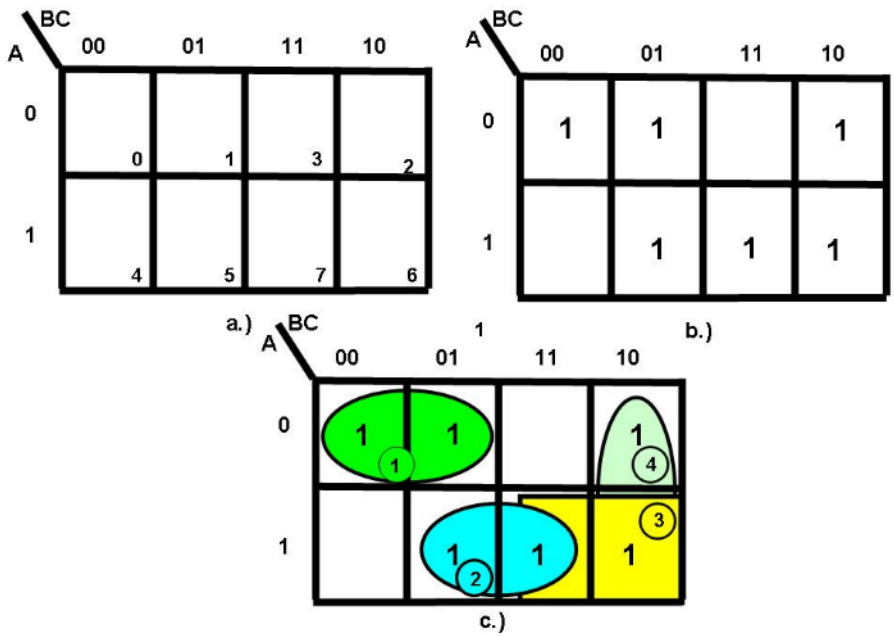


Figura Mapas de Karnaugh para 3 variables

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 Mapas de Karnaugh para 3 variables 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 Mapas de Karnaugh para 3 variables 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 **Circuito lógico**.

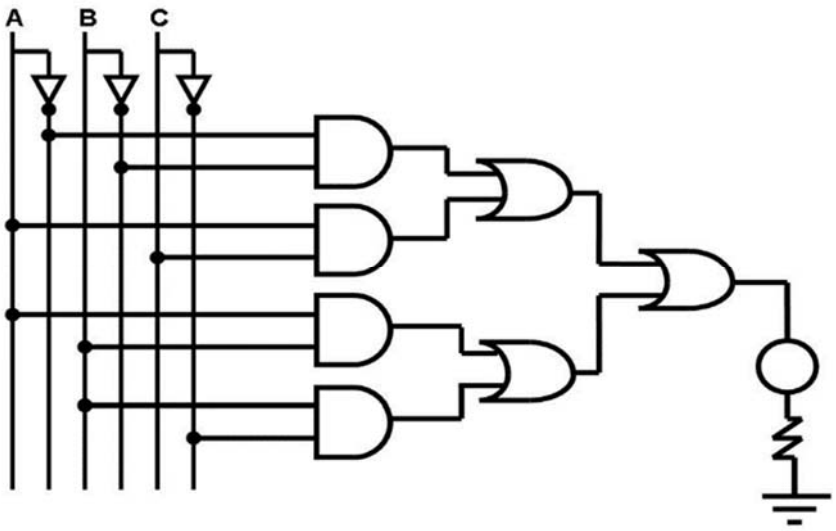
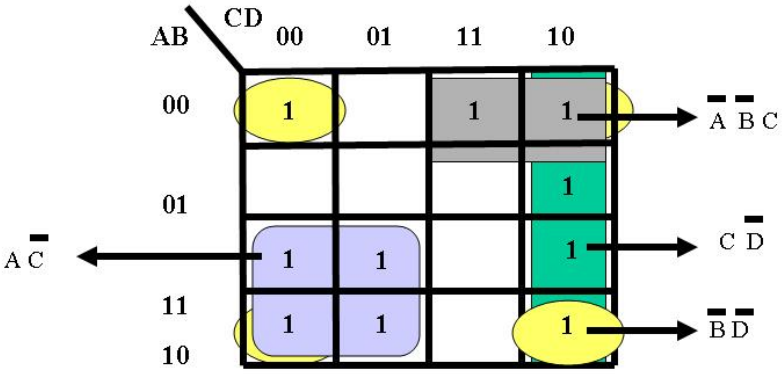


Figura Circuito lógico

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 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)= (2,5,6,7,8,9,10,12,13,14,18,21,22,23,24,25,26,18,19,30)

Solución

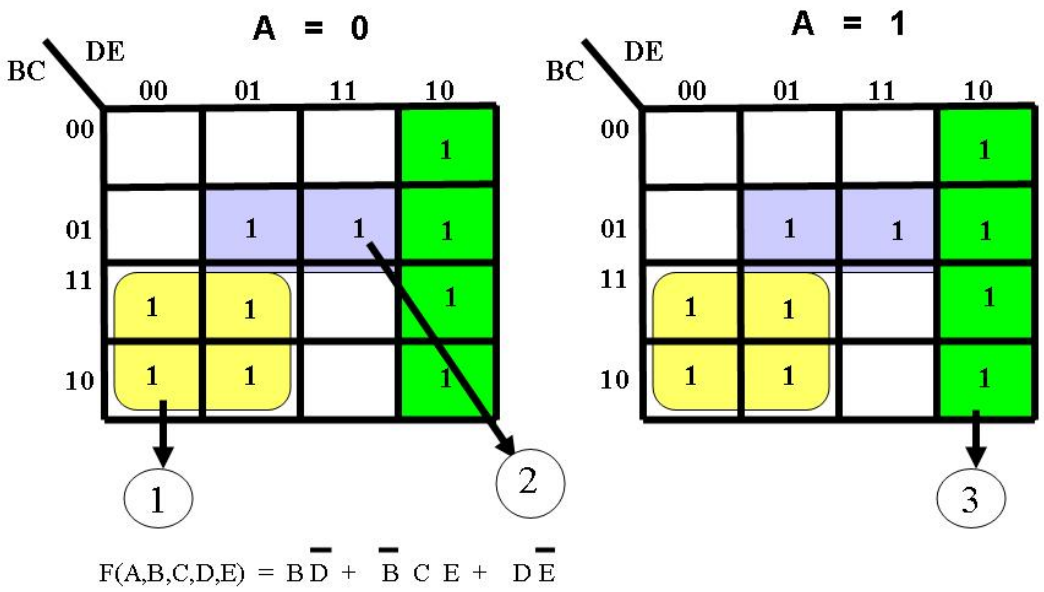


Figura Mapa de Karnaugh para 5 variables

RESUMEN

Inicialmente se presentaron los elementos y axiomas del álgebra de Boole:

Para el álgebra booleana, el conjunto de valores es el conjunto que contiene los elementos cero y uno. Las operaciones definidas son AND, OR y NOT.

OR	AND	NOT
• El operador OR (O) designado también como + es la representa la operación de “suma binaria” no es una suma en el sentido aritmético, sino lógico. C=A + B significa que la variable C será válida cuando alguna de las dos variables A o B sean válidas.	• El operador AND (Y) designado también como es la representación de la multiplicación “binaria” lógica. C es válida cuando las dos variables A y B (ambas) sean válidas.	• El operador lógico NOT (negación) significa que B es igual a A negada. O bien que cuando A es falsa, B es cierta.

Los axiomas del álgebra booleana son:

- Cerradura. Para los operadores binarios AND y OR
- La ley asociativa la cual no se establece en los postulados de Huntington, sin embargo si se cumple en el álgebra booleana.
- Ley conmutativa.
- La ley distributiva de + sobre. no se cumple en el álgebra ordinaria y sí en la booleana.
- El álgebra de Boole no posee elementos inversos aditivos o multiplicativos, por lo que no existe la operación de resta o multiplicación.

- Existencia de elementos identidad e inverso, este último define los elementos llamados complementos, los cuales no existen en el álgebra ordinaria.
- Los elementos del álgebra ordinaria están dentro del conjunto de los números reales, mientras que los elementos del álgebra booleana sólo son el uno y el cero.

Los principales teoremas del álgebra booleana son:

- Idempotencia, de absorción, leyes de De Morgan, teorema de adyacencia y teorema de dualidad. Estos teoremas nos permiten la manipulación de funciones, por ejemplo para encontrar funciones complementos.
- Las funciones booleanas se pueden representar de varias formas: tablas de verdad, canónica, normalizada, mínima, como suma de productos y como producto de sumas. La manipulación algebraica nos permite transformar una presentación en otra de acuerdo a las condiciones del problema. Sin embargo el manejo algebraico siempre representa un proceso laborioso y a veces complicado. Para minimizar funciones se pueden emplear los mapas de Karnaugh que son una aplicación sistemática del teorema de adyacencia a partir de una representación gráfica de funciones basada en los diagramas de Venn. De esta manera la minimización de funciones es una tarea más sencilla. Es importante entender a los mapas de Karnaugh como una forma más de representar funciones boolenas.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga (2010)	5	93-102
Mano (1986)	1	26-32
Stallings (2006)	Apéndice B	733-737

Mano, Morris. (1986). *Lógica Digital y diseño de computadores*. México: Prentice Hall Hispanoamericana.

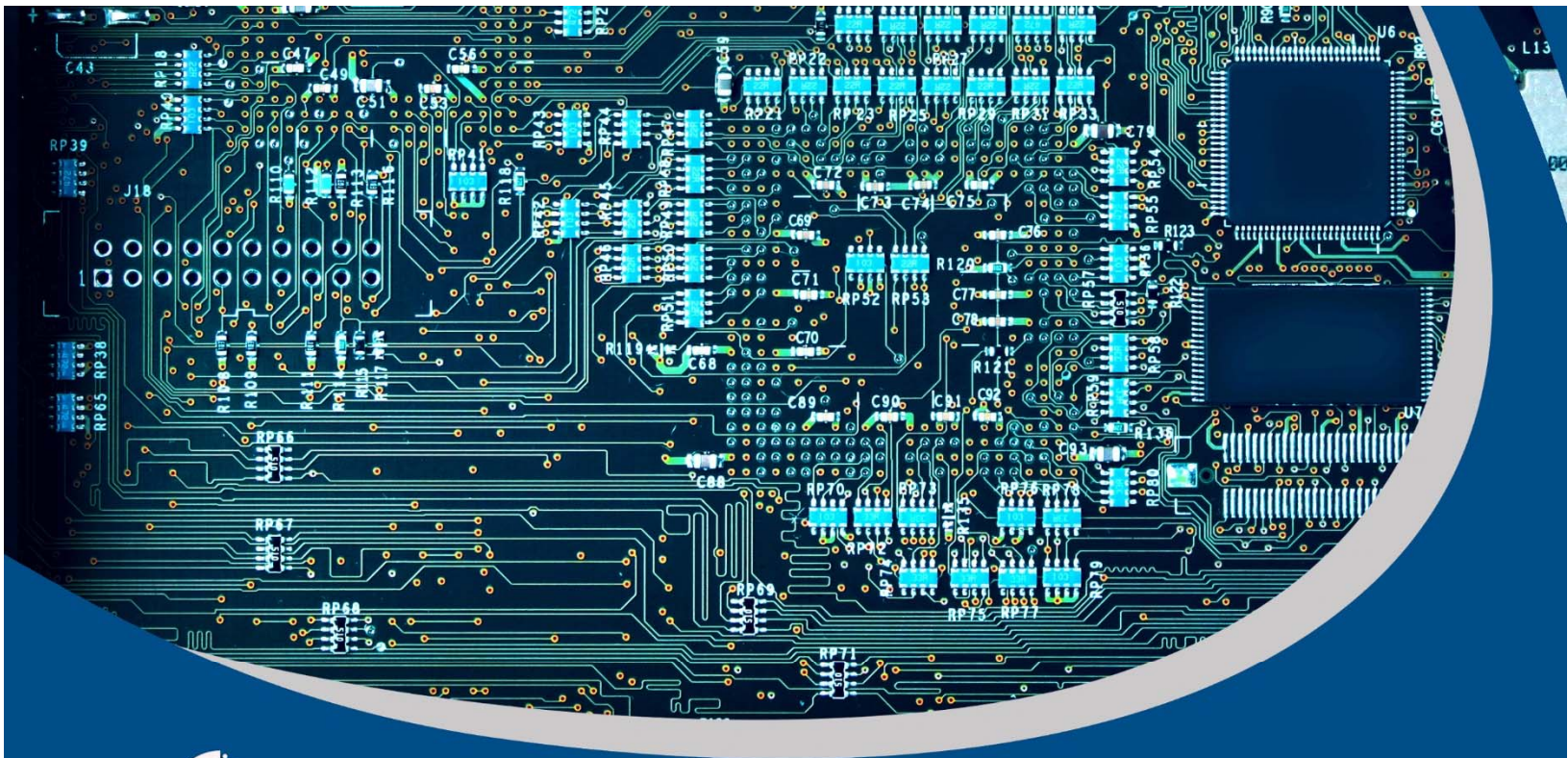
Quiroga, Patricia. (2010). *Arquitectura de computadoras*. México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores*. Madrid: Prentice Hall.

Licenciatura: Informática

UNIDAD 5

Circuitos combinatorios o combinacionales



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno podrá reconocer el funcionamiento y la construcción de sumadores, decodificadores y multiplexores a partir de compuertas básicas, diseñar circuitos combinacionales mediante compuertas digitales y deducirá la expresión algebraica a partir de un circuito digital.

TEMARIO DETALLADO
(10 horas)

5. Circuitos combinatorios o combinacionales

5.1. Multiplexores
5.2. Demultiplexores
5.3. Codificadores
5.4. Decodificadores
5.5. Medio Sumador
5.6. Sumador completo
5.7. Restadores
5.8. Comparadores.

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 después de cualquier cambio en las entradas. La figura Diagrama en bloques de una unidad lógica combinacional, ilustra un modelo de unidad lógica combinacional.

Esta unidad combinacional recibe un conjunto de entradas i0,…,in y produce un conjunto de salidas f0,…,fm, 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 Unidad 6).

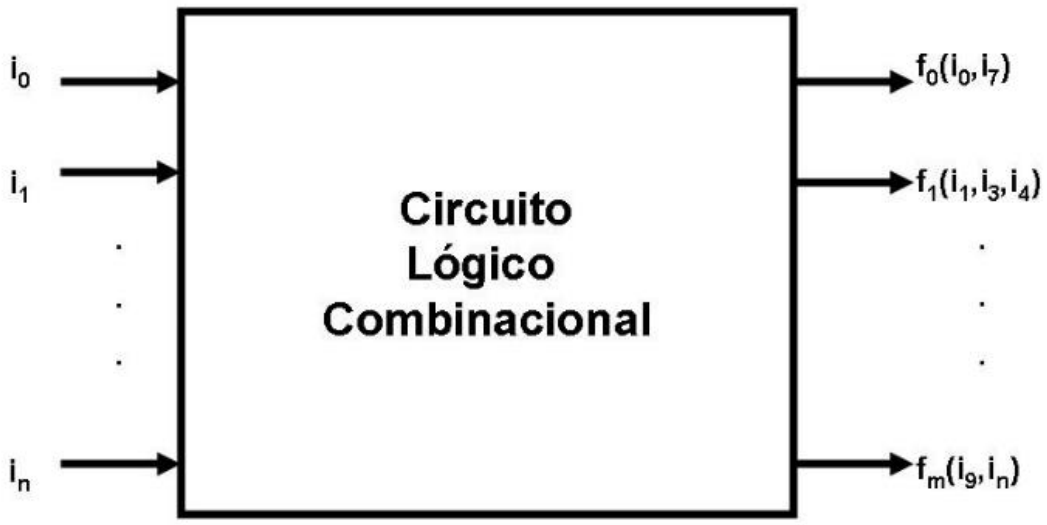


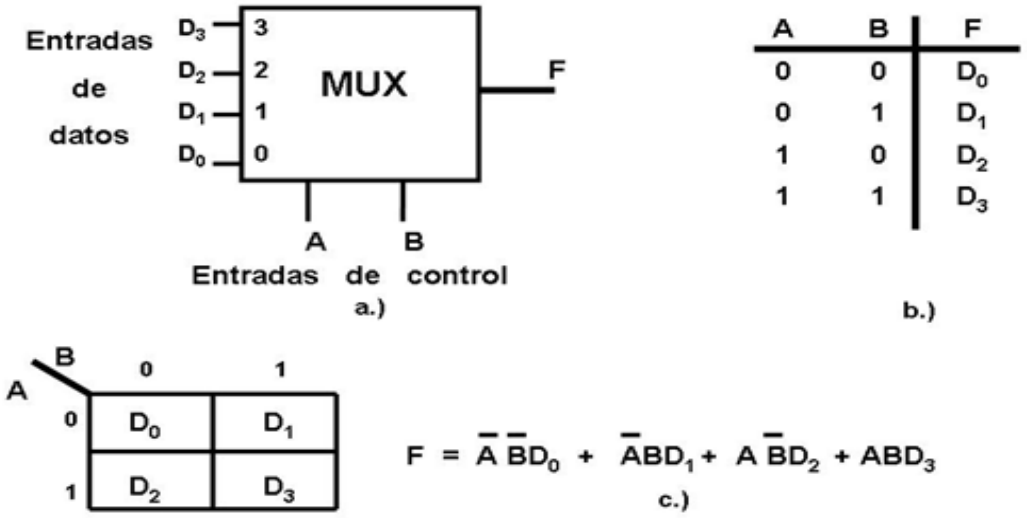
Diagrama en bloques de una unidad lógica combinacional

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.

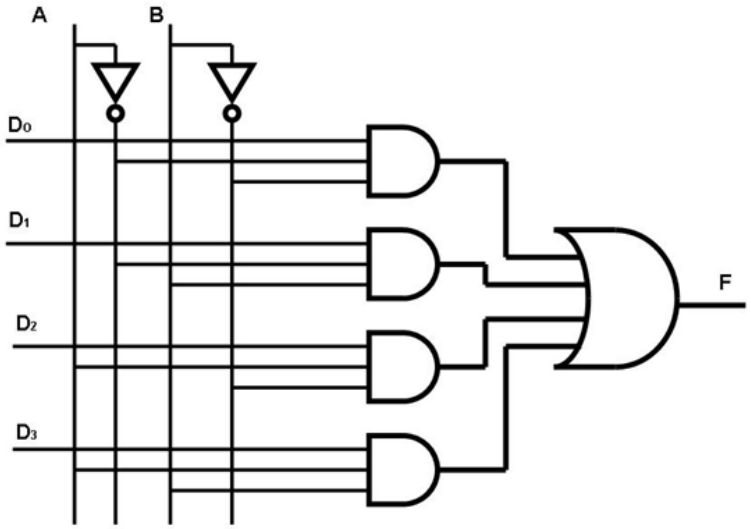
5.1. Multiplexores

Un circuito multiplexor (MUX) es un elemento que conecta una cantidad dada de entradas a una única salida. La figura Multiplexor 4 entradas 1 salida 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 Multiplexor 4 entradas 1 salida. b.) Tabla de Verdad. En la figura Multiplexor 4 entradas 1 salida. c.) Función lógica se muestra la obtención de la función lógica del multiplexor a partir de su tabla de verdad y en la figura Multiplexor 4 entradas 1 salida. d.) Diagrama lógico se presenta el diagrama lógico del multiplexor.



Multiplexor de 4 entradas y 1 salida.

a.) Diagrama a bloques, b.) Tabla de Verdad y c.) Función lógica

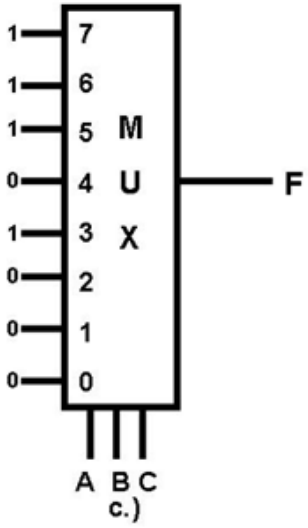


Multiplexor de 4 entradas y 1 salida
d.) Diagrama lógico

Una aplicación de los multiplexores es la implementación de funciones lógicas como se muestra en la figura Implementación de una función utilizando un multiplexor de 8 entradas. 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 por 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.

$F = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

a.)



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

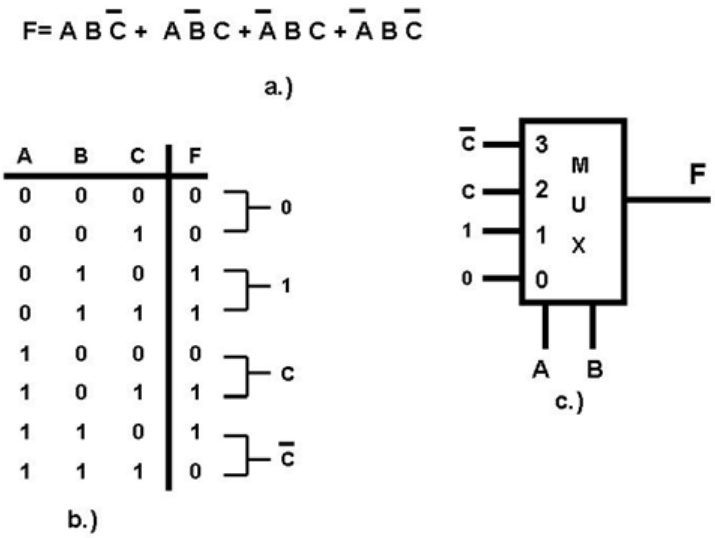
b.)

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 Implementación de una función utilizando un multiplexor de 4 entradas de datos. La figura Implementación de una función utilizando un multiplexor de 4 entradas de datos b) Tabla de verdad ilustra la tabla de verdad de tres variables de la función lógica por implementar (ver, figura Implementación de una función utilizando un multiplexor de 4 entradas de datos a) Función por implementar) 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, C} 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, B = 1, la función F = C, por lo tanto, la entrada de datos 11

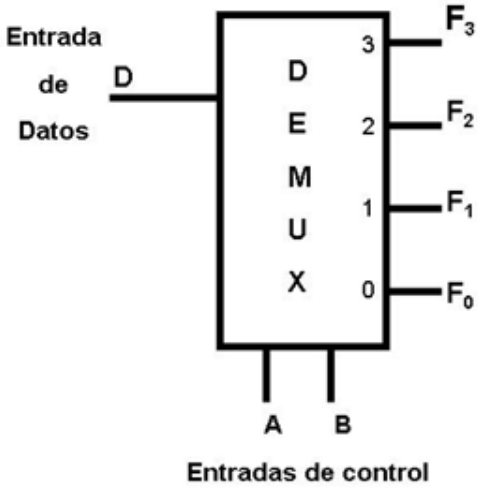
adopta el valor de C. De esta manera, se puede implementar una función de tres variables usando un multiplexor con cuatro entradas de datos y dos entradas de control.



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 Demultiplexor de 2x4 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 Fi salidas de acuerdo con los valores que adopten sus entradas de control. La figura Demultiplexor de 2x4 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

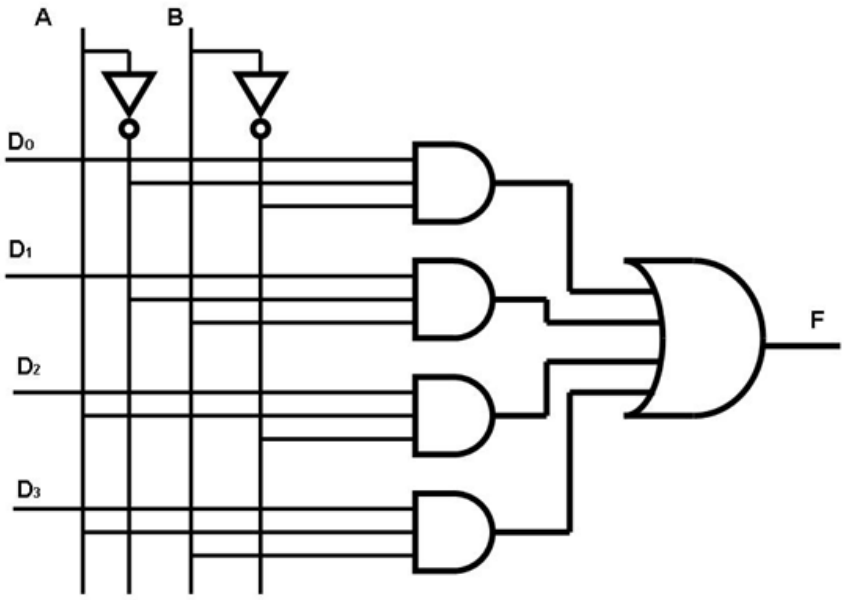
b.)

$F_0 = D\,\bar{A}\,\bar{B}$ $F_1 = D\,\bar{A}\,B$ $F_2 = D\,A\,\bar{B}$ $F_3 = D\,A\,B$

c.)

Demultiplexor de 2x4.

a.) Diagrama en bloques, b.) Tabla de verdad y c.) Las funciones de salida.



Multiplexor de 4 entradas y 1 salida

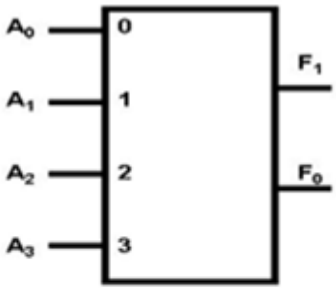
d.) Diagrama lógico

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 Codificador de prioridad de 4 a 2. El esquema de prioridades impuesto sobre las entradas hace que A_i tenga una prioridad mayor que A_{i+1}. La salida de dos bits adopta los valores 0_{10}, 1_{10},2_{10} u 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 A_0 ($F_0 = 0$ y $F_1 = 0$).

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 Codificador de prioridad de 4 a 2. c) Función de verdad ilustra el diagrama lógico para un codificador de prioridad de 4 entradas y 2 salidas.



a.)

$$F_0 = \overline{A}_0\,\overline{A}_1\,A_3 + \overline{A}_0\,\overline{A}_1\,A_2$$

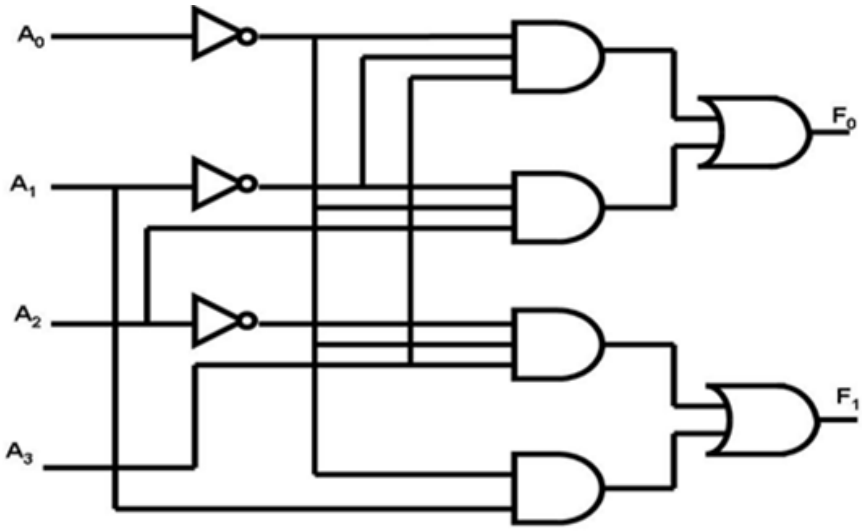
$$F_1 = \overline{A}_0\,\overline{A}_2\,A_3 + \overline{A}0\,A_1$$

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.)

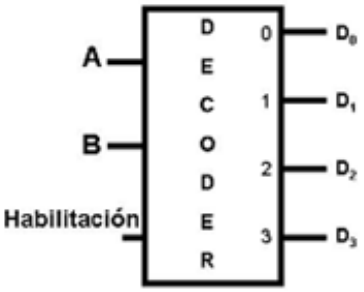
Codificador de prioridad de 4 a 2.
a.) Diagrama en bloques, b.) Tabla de verdad, c.) Funciones de salida.



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 Decodificador 2 a 4 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 Decodificador 2 a 4 c) Funciones de salida. 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.)

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

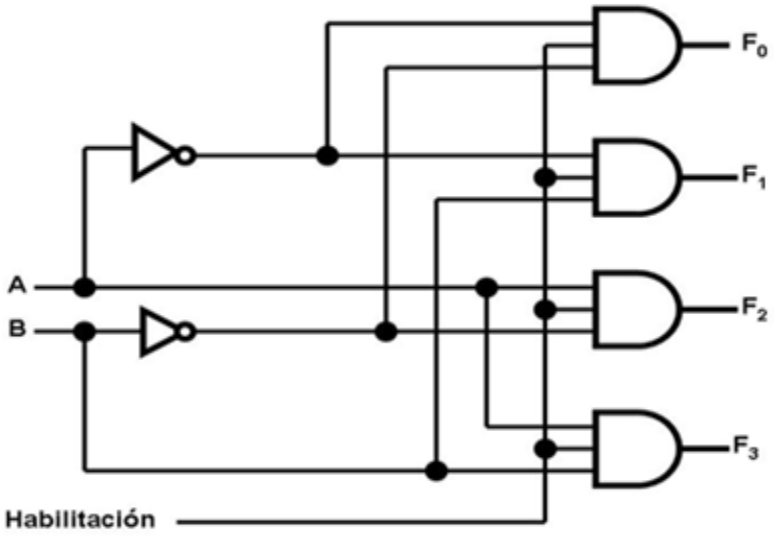
b.)

$D_0 = \overline{A}\,\overline{B}$ $D_1 = \overline{A}B$

$D_2 = A\overline{B}$ $D_3 = AB$

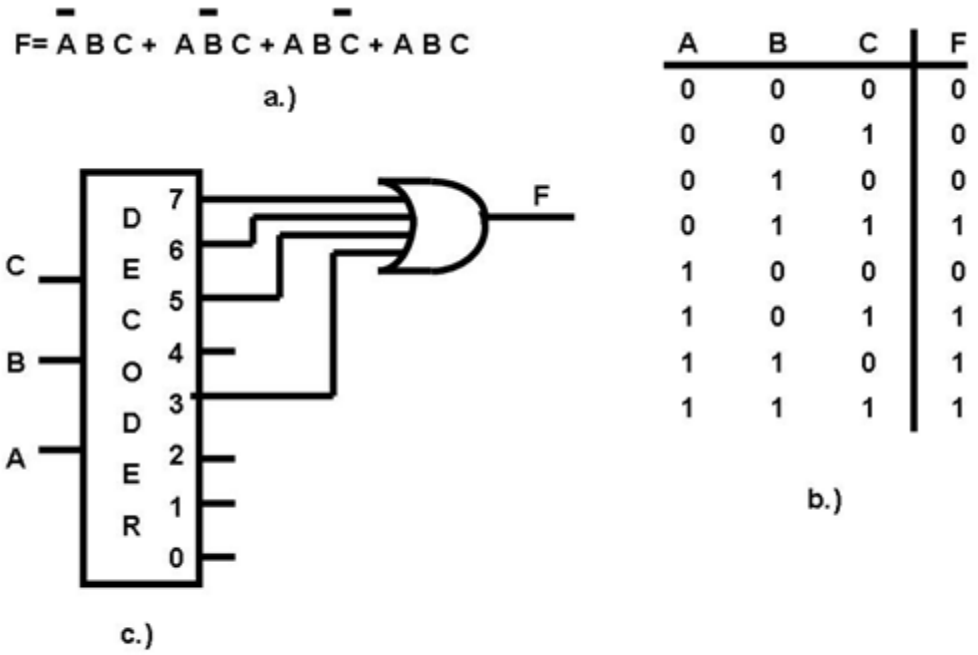
c.)

Decodificador 2 a 4. a.)Diagrama a bloques, b) Tabla de verdad y c.) funciones de salida.



decodificador 2 a 4
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 Implementación de una función utilizando un decodificador 3 a 8 se puede ver la implementación de la función con un decodificador de 3 a 8. Las salidas no utilizadas se dejan desconectadas.



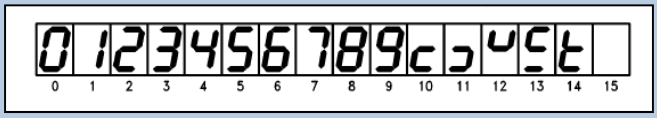
Implementación de una función utilizando un decodificador 3 a 8.
a.) Función a implementar, b.) Tabla de verdad y c.) Diagrama Lógico.

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:

Se enuncia el problema

Diseñe un decodificador BDC a siete segmentos utilizando compuertas básicas



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.

Se le asigna letras a las variables de entrada y a las funciones de salida.

Entradas => A, B, C, y D

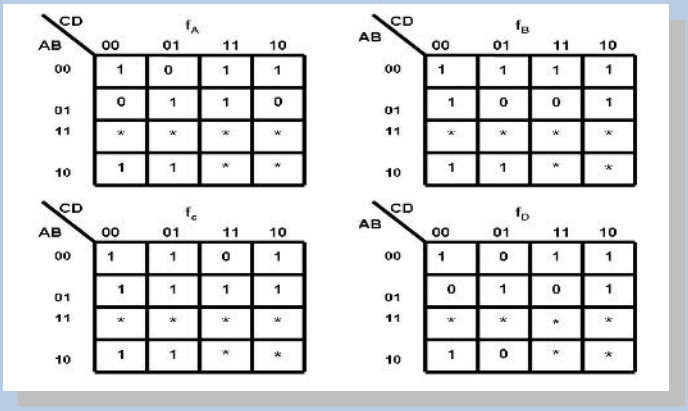
Salidas => f_a, f_b, f_c, f_d, f_e, f_f, y f_g.

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

Se obtiene la función de Boole simplificada, en este caso utilizamos el método de Karnaugh a cada una de las salidas.

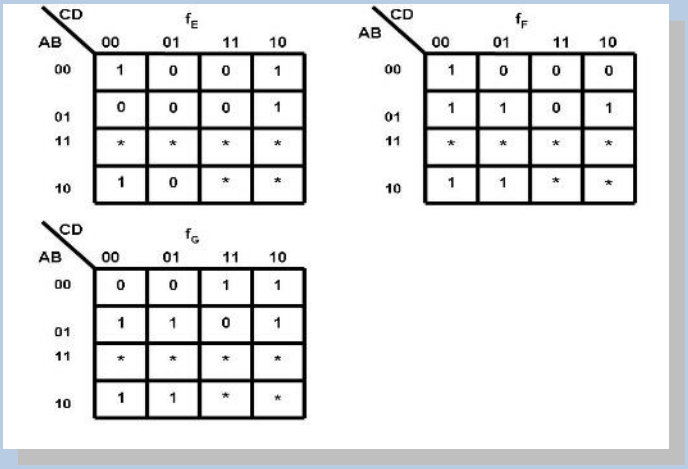


f_A	CD 00	CD 01	CD 11	CD 10
AB 00	1	0	1	1
AB 01	0	1	1	0
AB 11	*	*	*	*
AB 10	1	1	*	*

f_B	CD 00	CD 01	CD 11	CD 10
AB 00	1	1	1	1
AB 01	1	0	0	1
AB 11	*	*	*	*
AB 10	1	1	*	*

f_C	CD 00	CD 01	CD 11	CD 10
AB 00	1	1	0	1
AB 01	1	1	1	1
AB 11	*	*	*	*
AB 10	1	1	*	*

f_D	CD 00	CD 01	CD 11	CD 10
AB 00	1	0	1	1
AB 01	0	1	0	1
AB 11	*	*	*	*
AB 10	1	0	*	*

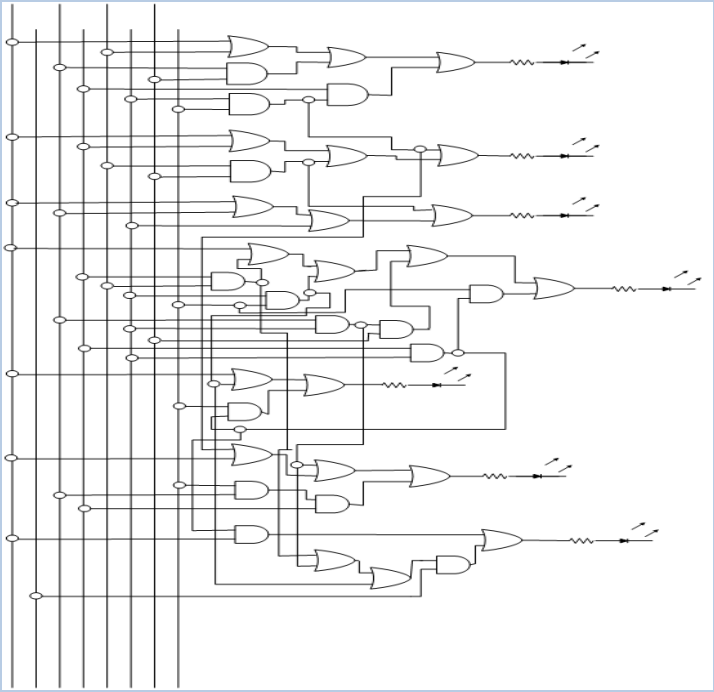


f_E	CD 00	CD 01	CD 11	CD 10
AB 00	1	0	0	1
AB 01	0	0	0	1
AB 11	*	*	*	*
AB 10	1	0	*	*

f_F	CD 00	CD 01	CD 11	CD 10
AB 00	1	0	0	0
AB 01	1	1	0	1
AB 11	*	*	*	*
AB 10	1	1	*	*

f_G	CD 00	CD 01	CD 11	CD 10
AB 00	0	0	1	1
AB 01	1	1	0	1
AB 11	*	*	*	*
AB 10	1	1	*	*

Se dibuja el diagrama lógico del decodificador 7 segmentos



5.5. 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 22 = 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 Karnaugh, como se muestra en la figura Obtención de la ecuación de S_0 y C_0 utilizando mapas de Karnaugh.

$$S_0 = A_0\bar{B}_0 + \bar{A}_0B_0$$

$$S_0 = A_0 \oplus B_0$$

$$S_0 = A_0B_0$$

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:

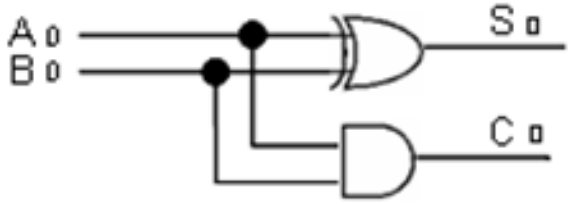


Diagrama lógico: Medio sumador

5.6. Sumador completo

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

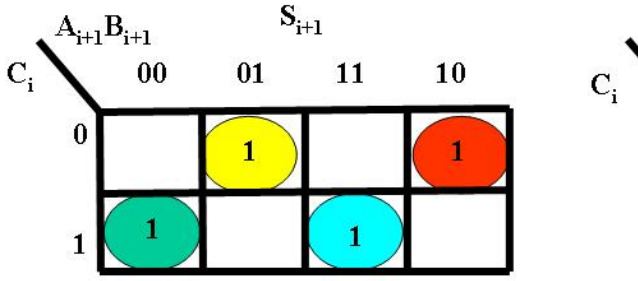
C_{i+1} es el bit de acarreo al momento de sumar tres bits.

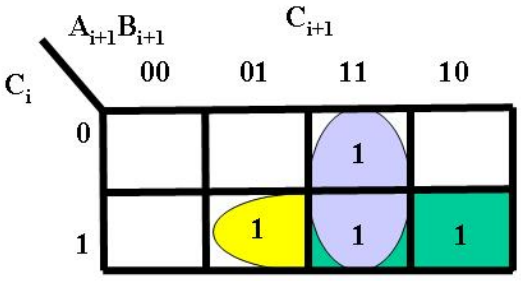
A partir de estos resultados obtenemos la tabla de verdad del sumador completo, la cual presentamos a continuación.

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

Tabla de verdad **Sumador completo**

A partir de la tabla de verdad, podemos encontrar la ecuación de salida para el resultado S_{i+1} de la suma de tres bits, así como la ecuación de salida del bit de acarreo C_{i+1} utilizando Mapas de Karnaugh, como se muestra en la figura Obtención de las ecuaciones de S_{i+1} y C_{i+1} empleando mapas de Karnaugh.





$$S_{i+1} = \overline{C}_{i+1}A_{i+1}\overline{B}_{i+1} + \overline{C}_{i+1}\overline{A}_{i+1}B_{i+1} + C_{i+1}A_{i+1}B_{i+1} + C_{i+1}\overline{A}_{i+1}\overline{B}_{i+1}$$

$$C_{i+1} = C_i B_{i+1} + A_{i+1}B_{i+1} + C_{i+1}A_{i+1}$$

$$S_{i+1} = \overline{C}_{i+1}(A_{i+1}\overline{B}_{i+1} + \overline{A}_{i+1}B_{i+1}) + C_{i+1}(A_{i+1}B_{i+1} + \overline{A}_{i+1}\overline{B}_{i+1})$$

$$C_{i+1} = C_i(A_{i+1} + B_{i+1}) + A_{i+1}B_{i+1}$$

$$S_{i+1} = \overline{C}_{i+1}(A_{i+1} \oplus B_{i+1}) + C_{i+1}\overline{(A_{i+1} \oplus B_{i+1})}$$

$$S_{i+1} = C_{i+1} \oplus (A_{i+1} \oplus B_{i+1})$$

La implementación (diagrama lógico) de la ecuación del sumador completo para S_{i+1} y C_{i+1} nos quedaría de la siguiente forma:

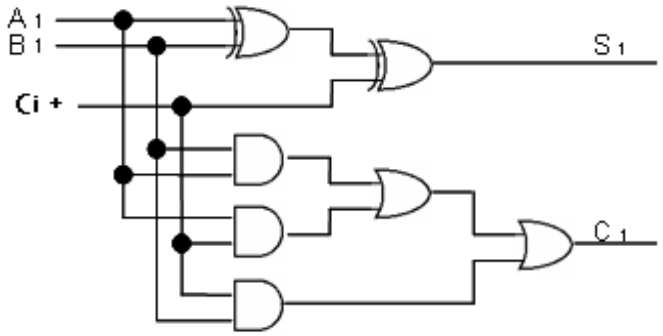
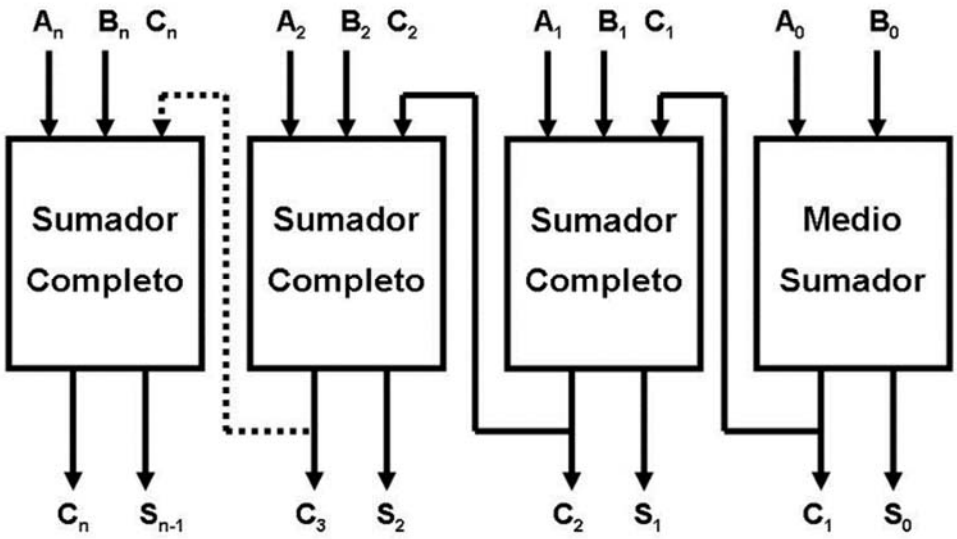


Diagrama lógico: Sumador completo

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 Sumador de n-bits implementados con n-1 sumadores completos.



Sumador de n-bits implementados con n-1 sumadores completos

A partir del *diagrama a bloques del sumador de 4 bits* (ver figura Sumador de n-bits implementados con n-1 sumadores completos) se construye el diagrama lógico el cual se

presenta en la figura Diagrama lógico de un Sumador de 4 bits en cascada y su respectivo diagrama eléctrico en la figura Diagrama eléctrico de un sumador de 4 bits en cascada.

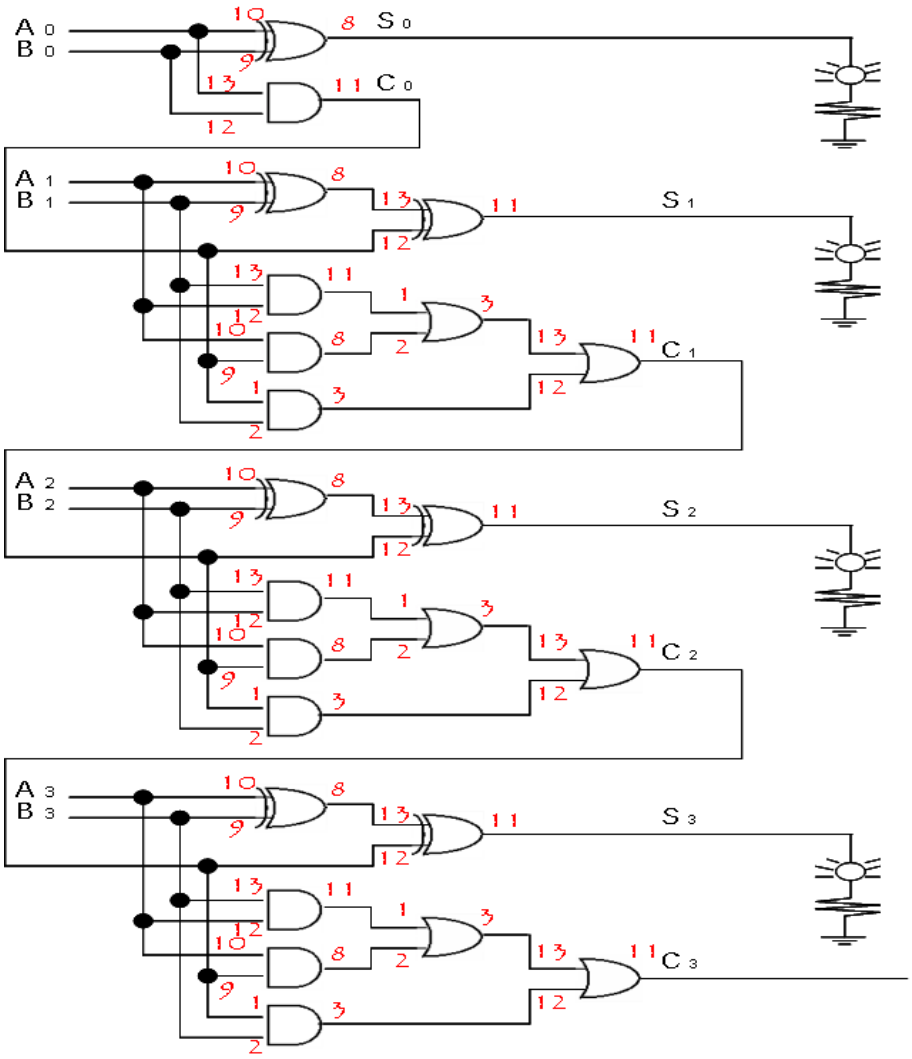


Diagrama eléctrico: Sumador de 4 bits

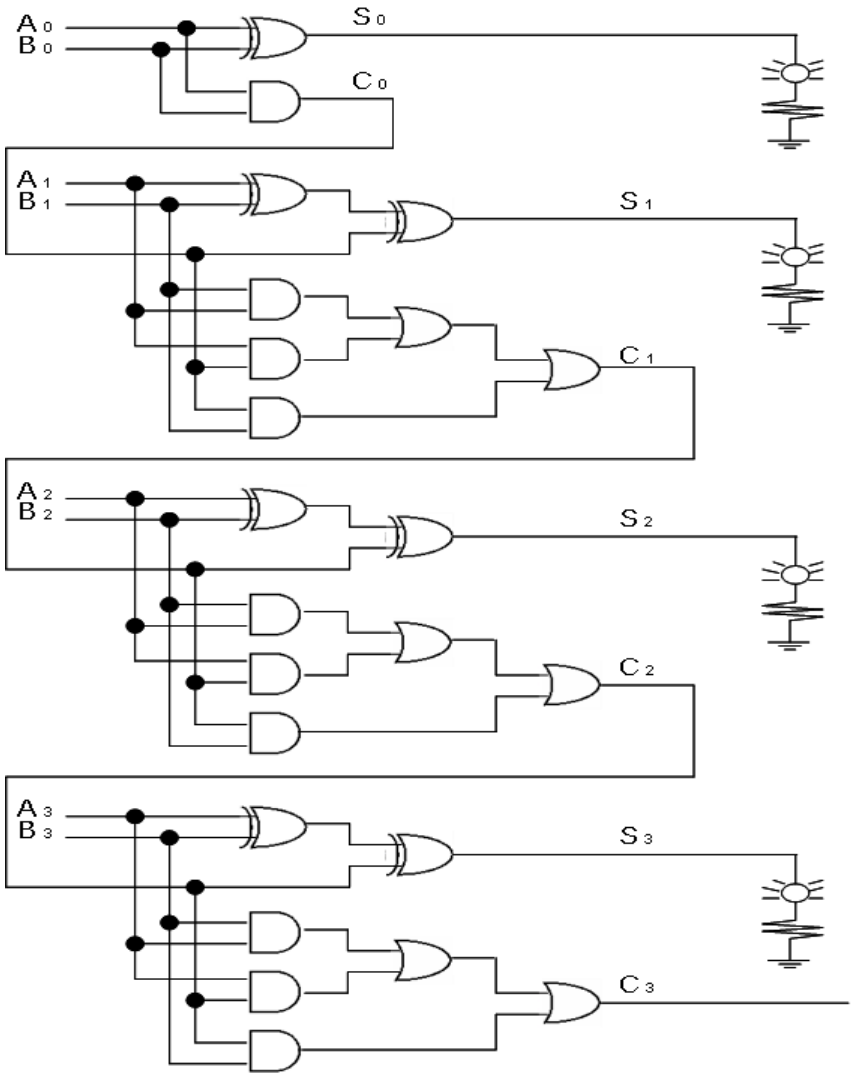


Diagrama lógico: Sumador de 4 bits

5.7. Restadores

El caso de los restadores en cuanto a su diseño mediante dispositivos digitales (compuertas binarias) es semejante al diseño de sumadores. De la misma manera como se crea un sumador de dos bits con acarreo, se puede generar un restador de dos bits con un bit llamado borrow. Para el caso de un restador binario de un digito o medio restador tenemos la siguiente tabla:

0 – 0 = 0
1 – 1 = 0
1 – 0 = 1
0 – 1 = 1

(En este caso se considera un “acarreo” o como lo conocíamos de nuestra aritmética básica “se toma prestado uno”) El concepto de *borrow* es semejante al de acarreo de la suma. Este caso de un restador bit a bit se llama semi restador y análogamente al caso de la suma solo se considera el acarreo como un dígito que indica que el signo del resultado es negativo o positivo.

Tenemos por lo tanto la siguiente tabla para la resta binaria de un dígito:

Minuendo	Sustraendo	Resultado	*Borrow*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

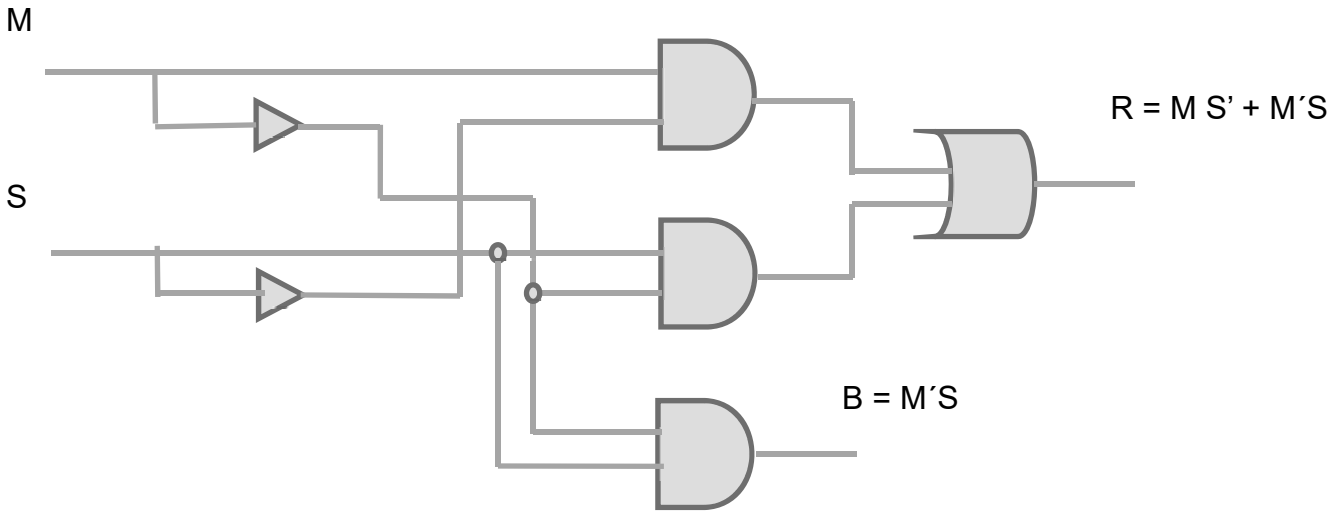
Funciones:

Mediante mapas K obtenemos las funciones asociadas:

S \ M	0	1
0	0	1
1	1	0

S \ M	0	1
0	0	0
1	1	0

La construcción a partir de compuertas AND y OR es:



Ejercicio

Obtén mediante mapas de Karnaugh las funciones para un restador de dos bits.

5.8. Comparadores

Comparadores

Un tipo especial de circuitos lógicos combinacionales son los comparadores. Estos circuitos toman dos valores binarios y al compararlos determinan la salida o salidas dependiendo de las condiciones planteadas para el problema. Por ejemplo si comparamos dos dígitos binarios de un solo bit A y B, podemos tener tres opciones posibles:

A > B

A < B

A = B

En una tabla podemos observar varias combinaciones:

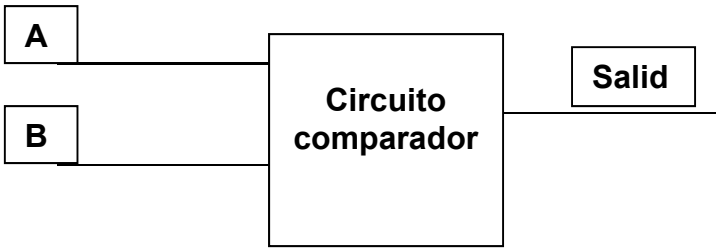
A	B	A > B	A < B	A = B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Si tenemos un sistema con dos entradas de un bit cada una y un bit de salida podemos también tener tres condiciones:

La salida es válida si la variable A es mayor o igual a B = La salida es válida si B es menor que A.

La salida es válida si la variable B es mayor o igual a A = La salida es válida si A es menor que B

La salida de la variable A es igual a B



Consideremos un caso: Diseñar un comparador de dos entradas de un bit cada una y una salida, la cual será válida solamente si A > B.

Tenemos la siguiente tabla:

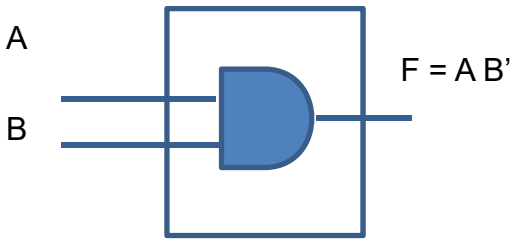
A	B	Salida F
0	0	0 (Falso)
0	1	0 (Falso)
1	0	1 (Cierto)
1	1	0 (Falso)

Para encontrar la función de salida tenemos:

F = A B’

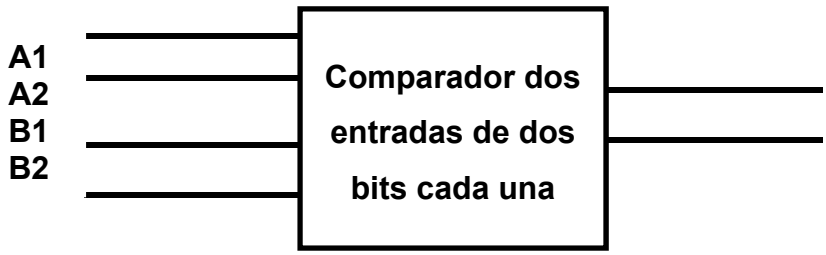
F = A’ B’ + A’ B + A B = A’ + B

Y el circuito para F construido con compuertas será:



El comportamiento de este circuito responderá en la salida F válida solo cuando A sea mayor que B.

En este caso el comparador solo tiene dos entradas de un bit, sin embargo se pueden diseñar comparadores para entradas con más de un bit, por ejemplo un comparador de dos entradas de dos bits tenemos lo siguiente:

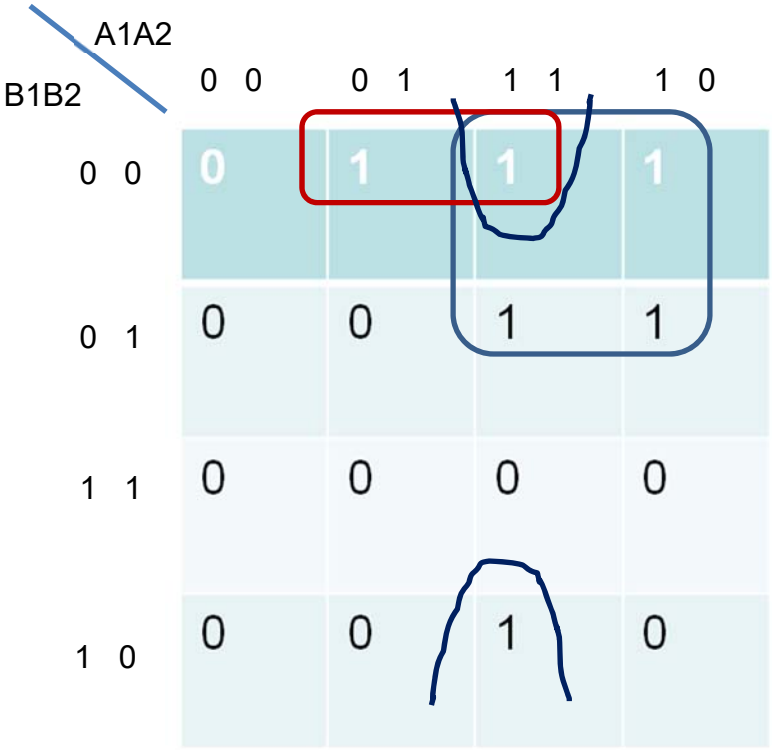


La tabla de verdad depende de la o las condiciones establecidas. Por ejemplo:

Determinar la función de salida para un comparador de dos entradas de dos bits cada una, en el cual la salida sea válida si A es mayor que B. En esta tabla la entrada A está formada por los bits A1 y A2 y la entrada B está formada por los bits B1 y B2.

A	B	Salida
A1 A2	B1 B2	C
0 0	0 0	0
0 0	0 1	0
0 0	1 0	0
0 0	1 1	0
0 1	0 0	1
0 1	0 1	0
0 1	1 0	0
0 1	1 1	0
1 0	0 0	1
1 0	0 1	1
1 0	1 0	0
1 0	1 1	0
1 1	0 0	1
1 1	0 1	1
1 1	1 0	1
1 1	1 1	0

Para determinar la función de salida utilizamos un mapa de Karnaugh:



A partir de este mapa obtenemos la función F de salida.

$$F = A1\ B1' + A2\ B1'\ B' + A1\ A2\ B2'$$

RESUMEN

Las principales características en la construcción de circuitos electrónicos son: funciones que realizan, tecnología utilizada y escala o cantidad de transistores integrados en una pastilla. En cuanto a la función que realizan tenemos:

Compuertas básicas: AND, OR, NOT, NAND, NOR Y EXOR.

Funciones	Operaciones Booleanas básicas, decodificadores, multiplexores, sumadores.
Integración de circuitos	Pequeña escala de integración (SSI). Alrededor de 12 compuertas. Mediana escala de integración (MSI). Entre 12 y 100 compuertas. Gran escala de integración (LSI). De 100 a 1000 compuertas. Muy grande escala de integración. (VLSI). Más de 1000 compuertas.

Los circuitos SSI se utilizan para la construcción de compuertas básicas encapsuladas. Los circuitos MSI se emplean en sumadores, multiplexores y decodificadores. Los circuitos LSI son los que pueden almacenar grandes cantidades de información o bien realizar procesos completos, se utilizan para construir memorias y arreglos lógicos programables y los primeros procesadores en los años 70. Los circuitos VLSI se utilizan actualmente para la construcción de microprocesadores.

En esta unidad se usaron conceptos desarrollados en unidades anteriores para el diseño de circuitos básicos utilizados en la construcción de computadoras tales como

sumadores, comparadores y convertidores de código. Dichos circuitos junto con los que desarrollarán la unidad de lógica secuencial permitirán comprender el funcionamiento de un microprocesador al integrar los conceptos de registros y contadores. La parte operativa, que realiza las funciones algebraicas y lógicas en una computadora se llama Unidad Aritmética Lógica y está construida por los elementos que hemos revisado.

La construcción de estos bloques funcionales se realiza mediante una metodología adecuada que nos permite llegar desde el enunciado del problema, especificando sus requerimientos, hasta la construcción del circuito.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga	5, 6	102-104, 118-127
Mano	5	160-179 y 180-188
Stallings	Apéndice B	748-758

Mano, Morris. (1986). *Lógica Digital y diseño de computadores.* México: Prentice Hall Hispanoamericana.

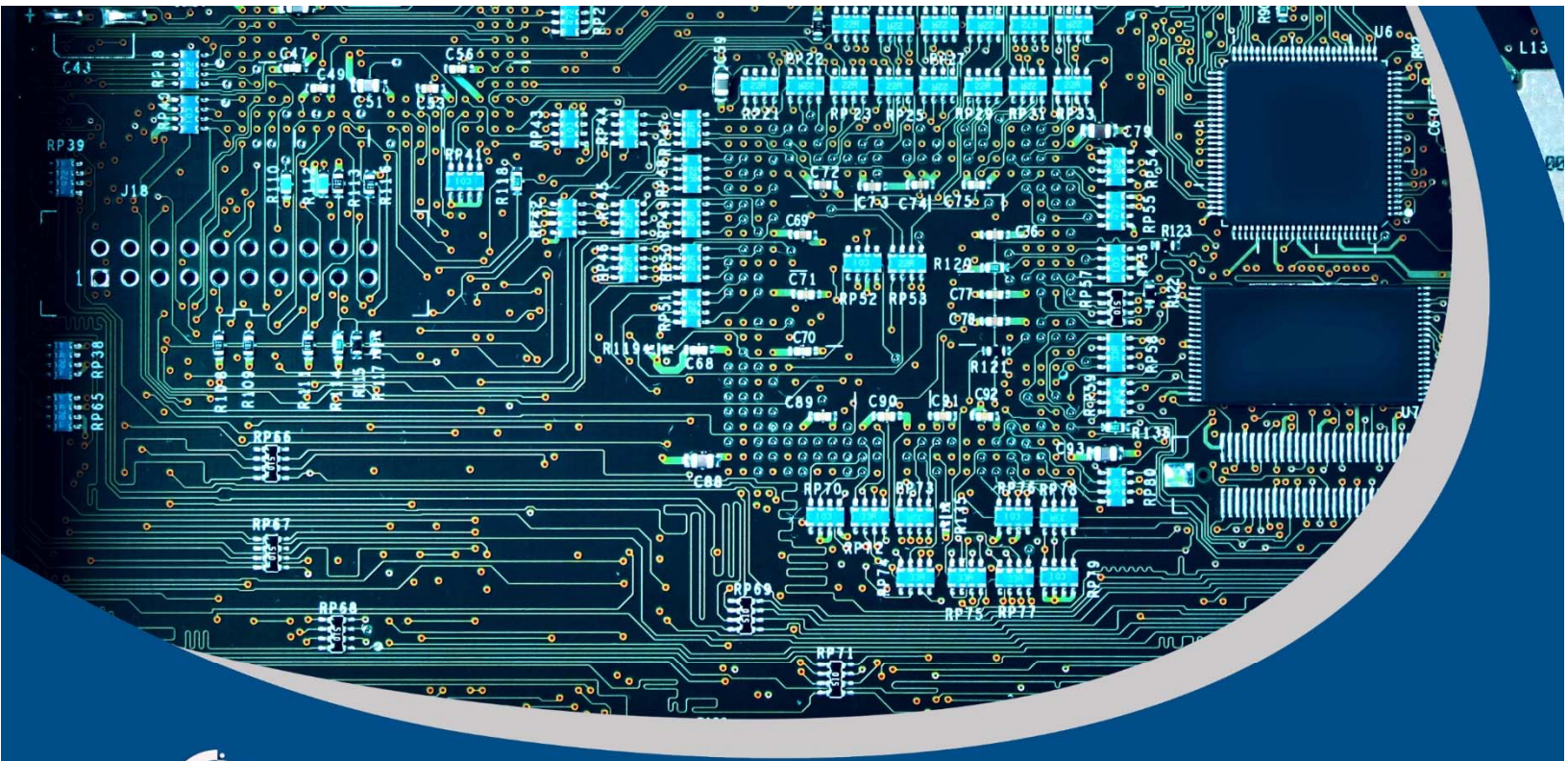
Quiroga, Patricia. (2010). *Arquitectura de computadoras.* México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores.* Madrid: Prentice Hall.

Licenciatura: Informática

UNIDAD 6

Circuitos secuenciales



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno identificará los circuitos, los diferentes tipos de flip-flops, temporizadores y contadores.

TEMARIO DETALLADO
(10 horas)

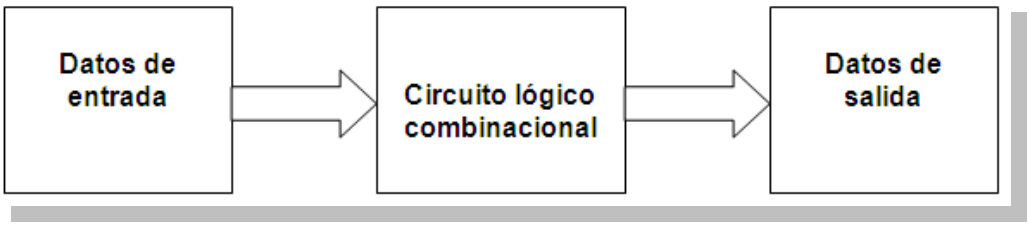
6. Circuitos secuenciales

- 6.1. Circuitos síncronos
- 6.2. Circuitos asíncronos
- 6.3. Flip-Flops (JK, SR, T, D)
- 6.4. Registradores de corrimiento
- 6.5. Temporizadores
- 6.6. Contadores

INTRODUCCIÓN

En las unidades anteriores hemos manejado los elementos básicos que conforman un sistema digital. Por un lado el manejo binario de la información mediante códigos, sistemas numéricos y el álgebra booleana; por el otro los dispositivos electrónicos digitales que procesan la información de acuerdo con las normas de los elementos conceptuales anteriores: compuertas, decodificadores, multiplexores y sumadores.

Hasta ahora, hemos considerado las salidas de los sistemas digitales dependientes únicamente de las entradas. Es decir tenemos una serie de valores de salida en función de los valores de entrada. Adicionalmente los circuitos que hemos manejado solamente procesan la información pero no la almacenan para futuras aplicaciones. Esto representa una limitante pues cada vez que se procesa una serie de datos, es necesario "programar" el circuito para una tarea específica. Por un lado las salidas de los circuitos vistos no se pueden guardar para su posterior utilización y por el otro los circuitos tienen que ser alimentados (tanto de instrucciones como de datos) cada vez que se realiza un proceso. Elaborando un modelo de los circuitos vistos hasta ahora tendríamos lo siguiente:



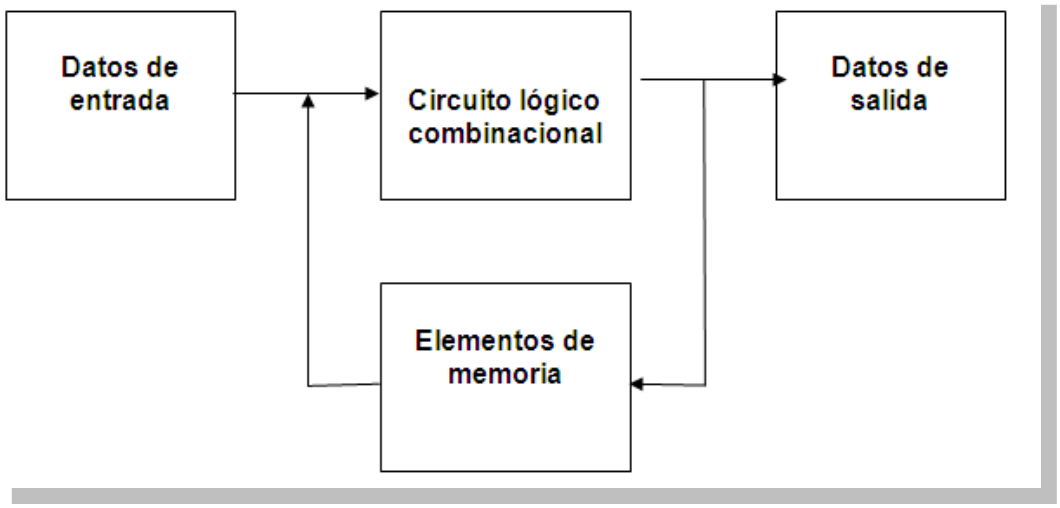
Otra característica de este modelo de procesamiento es que las entradas pueden ser modificadas en cualquier momento y como resultado, las salidas del circuito cambian inmediatamente teniendo un retraso determinado por los tiempos de retardo de cada

nivel de implementación digital. Es decir, la variable tiempo no está siendo controlada en los procesos.

En la presente unidad incluimos dos conceptos en el análisis digital: tiempo y realimentación, lo que implica integrar elementos de memoria en nuestros circuitos capaces de almacenar información binaria. En un sistema informático implica la posibilidad de almacenar información tanto de tareas de procesos como de resultados de los mismos, es decir instrucciones y salidas de procesos que pueden ser realimentados a los sistemas.

A nivel microcomponentes, la construcción de elementos de almacenamiento digital se realiza mediante arreglos de compuertas básicas que generan una categoría diferente de dispositivos: los flip fops. Mediante estos dispositivos construiremos dos tipos de circuitos importantes de una microcomputadora: los registros y los contadores. Los primeros almacenan información como una secuencia de varios bits; los segundos nos permiten sincronizar varios procesos de diversas unidades que funcionan a diferentes frecuencias de operación.

Aquí incluimos una metodología mediante la cual podemos diseñar circuitos digitales a los cuales se integran elementos de memoria o realimentación. Esta metodología utiliza varias formas de representación de la información de las variables de entrada, los estados del circuito y las salidas del mismo como son los diagramas de estado, las tablas de estado y las cartas de estado de máquina. Consecuentemente, al incluir elementos de realimentación, el modelo de proceso de información será el siguiente.


Datos de entrada
Circuito lógico combinacional
Datos de salida
Elementos de memoria

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	•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). (López, 2006, p. 2)

6.2. Circuitos asíncronos

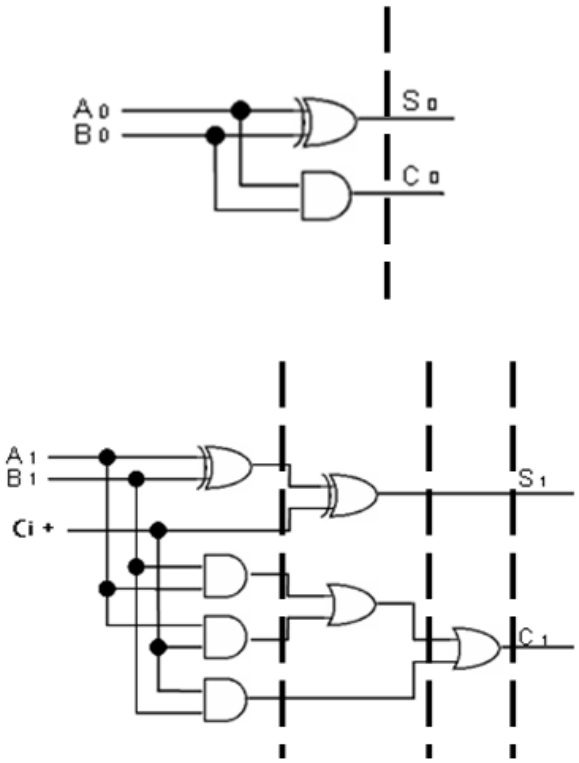
Los circuitos secuenciales asíncronos son circuitos digitales donde los cambios de estado ocurren al cambiar las señales de entrada, afectados por los retardos asociados a las compuertas lógicas utilizadas en su implementación, es decir, estos circuitos no usan elementos especiales de memoria, aunque sí utilizan líneas de realimentación. Los retrasos en el tiempo de respuesta no están bajo el adecuado control del diseño, lo cual puede afectar su funcionamiento, pues estos retardos no son idénticos en cada compuerta lógica.

Los circuitos secuenciales síncronos son los que se emplean en el diseño de elementos de computadoras y procesadores, específicamente en el diseño de registros y contadores.

Circuitos secuenciales asíncronos

Aunque no se trata de un circuito específicamente secuencial, un circuito sumador de dos palabras de 4 bits, cada una ilustra el efecto de los retrasos en el tiempo y cómo se puede diseñar reduciendo los niveles de implementación y por lo tanto los retrasos en el tiempo. El circuito lógico es un sumador con acarreo (*carry look ahead*). Al estar construido con sumadores completos en serie, la realimentación se realiza de un sumador al siguiente mediante la señal de acarreo.

En los siguientes diagramas se muestran los circuitos para un sumador medio y un sumador completo.



Para el medio sumador observamos un nivel de retraso debido a que solo manejamos dos compuertas y no están conectadas secuencialmente. Para los sumadores completos observamos tres niveles de diseño lo que representa tres retrasos. Sin embargo, en el sumador completo, si utilizamos una compuerta OR de tres entradas, los niveles se reducen a dos. El sumador completo realiza la suma de un solo bit con acarreo de entrada y de salida.

Para implementar un sumador de dos palabras de cuatro bits utilizando sumadores completos, tenemos lo siguiente:

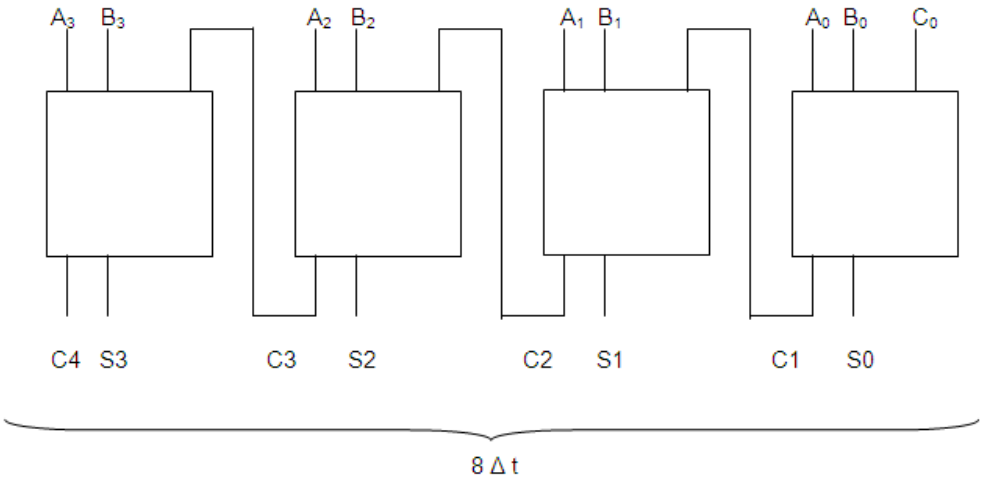
$A_3\ A_2\ A_1\ A_0 \leftarrow C_0$

$B_3\ B_2\ B_1\ B_0$

———————

$C_4 \leftarrow S_3\ S_2\ S_1\ S_0$

El diagrama de bloques utilizando el acarreo de salida como entrada al bloque siguiente es:



Dado que cada sumador completo tiene un retraso de 2 Δt para los cuatro sumadores tenemos 8 Δt. Si consideramos que el primer bloque puede ser un medio sumador, solo requerimos 7 Δt. De manera general se requieren 2N-1 Δt para realizar la suma de N bits. El problema que se presenta en el circuito es que cada bloque tiene que esperar el acarreo de la etapa previa para poder realizar la suma de dos bits. Si podemos determinar cuánto vale el acarreo de salida previamente, podemos realizar la suma de parejas de bits al mismo tiempo.

El circuito que realiza esta operación se denomina Sumador con *Carry Look Ahead* y genera todos los acarreos previamente y después realiza la suma de cada pareja de bits.

Para el Sumador completo:

$Si = Ai \oplus Bi \oplus Ci$

$Ci+1 = Ai\ Bi + (Ai \oplus Bi)\ Ci$

$Ci+1 = \underbrace{Ai\ Bi}_{Gi} + \underbrace{(Ai \oplus Bi)}_{Pi}\ Ci$

Donde Gi es la función generadora y Pi es la función propagadora.

Por lo que tenemos:

$$Ci+1 = Gi + Pi\ Ci$$

Por lo que podemos prever cuánto vale el acarreo para cada etapa.

i = 0

$Ci = G0 + P0\ C0 = Go$ dado que C0 es cero → $C1 = G0$

Para i = 1

$C2 = G1 + P1C1 = G1 + P1G0$

Para i=2

$C3 = G2 + P2C2 = G2 + P2\ (G1+P1G0) = G2 + P2G1 + P2P1G0$

Para i=3

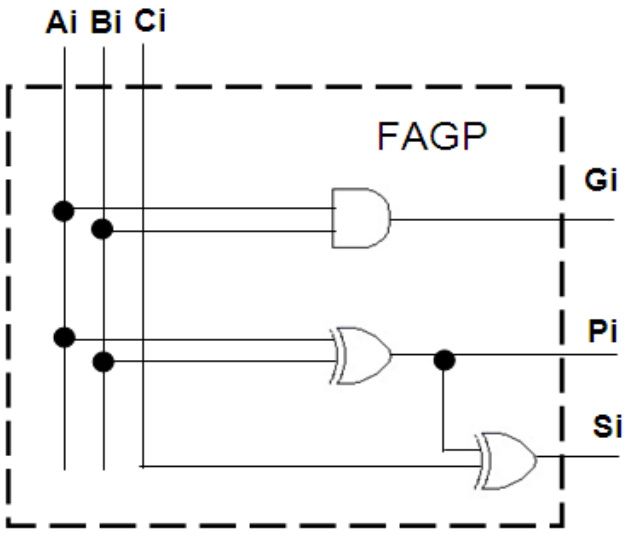
C4 = G3 + P3C3= G3 + P3 (G2 + P2G1 + P2P1G0) = G3 + P3G2+ P3P2G1+ P3P2P1G0

En la tabla siguiente se muestran los valores para los primeros valores de i:

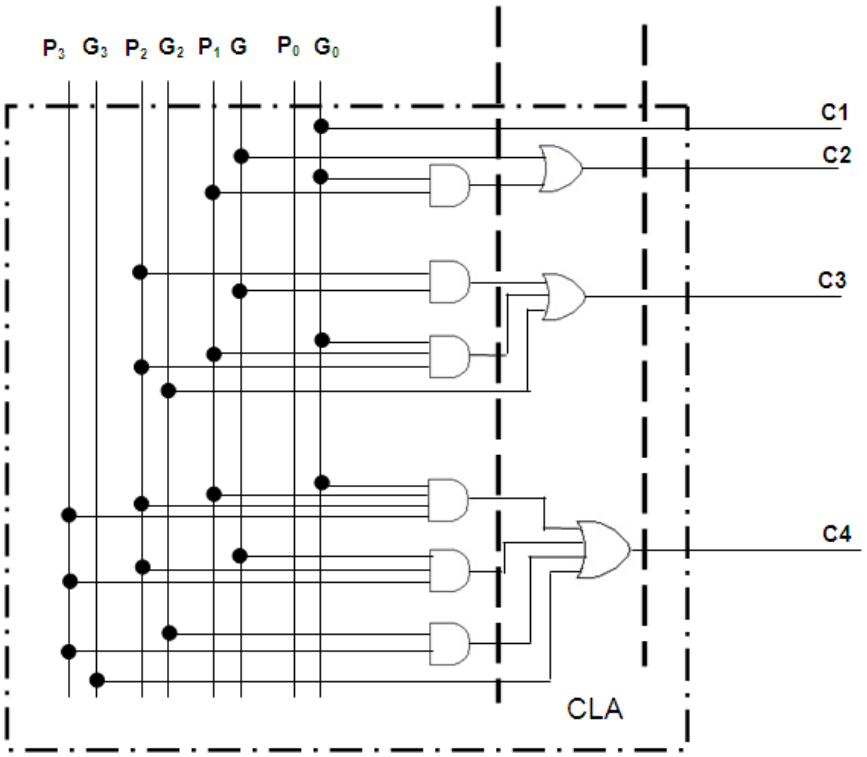
Valor de i	C	Valor del acarreo
0	C1	G0
1	C2	G1 + P1G0
2	C3	G2 + P2G1 + P2P1G0
3	C4	G3 + P3G2 + P3P2G1 + P3P2P1G0
4	C5	G4 + P4G3 + P4P3G2 + P4P3P2G1+ P4P3P2P1G0

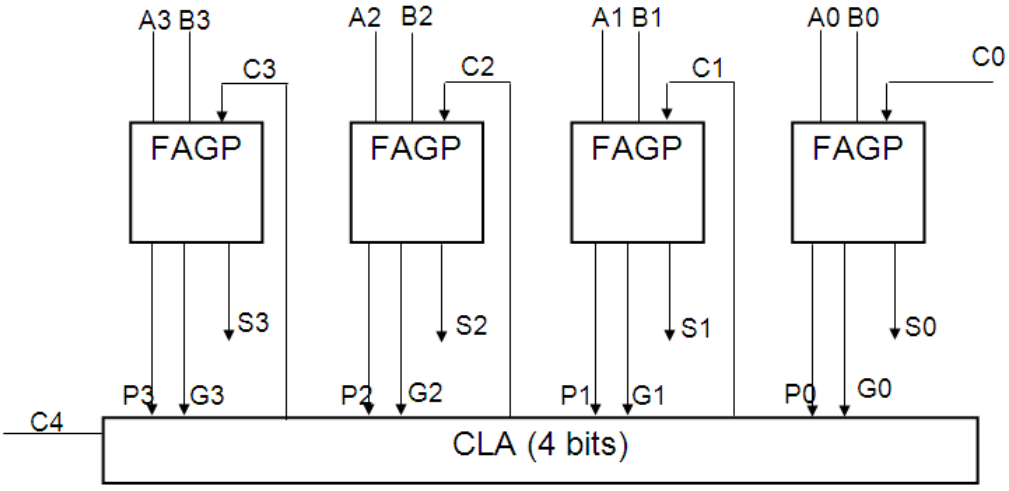
Para un sumador de 4 bits, utilizamos hasta i=3. Para el diseño implementamos, por un lado, las ecuaciones 1 en lo que se llama un sumador completo generador propagador (SCGP) y, por el otro, las ecuaciones 2 llamadas *Carry Look Ahead* (CLA), tenemos:

Circuito para el generador propagador



Circuito para el Carry Look Ahead (CLA). Para este circuito, implementado con dos niveles de compuertas, solo tenemos 2 Δ t en la generación de los acarreos.





6.3. Flip-Flops (JK, SR, T, D)

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 realimentación (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:

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

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.

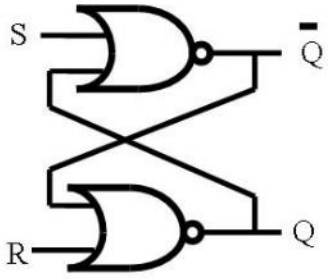
- Para que un flip-flop cambie de estado, es necesario introducir una señal de entrada.

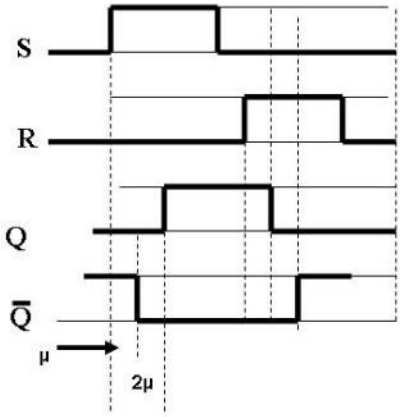
- El flip-flop tiene dos salidas, $\overline{\mathbf{Q}}$ y Q, las cuales son siempre complementarias.

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 Q, y puede estar construido a partir de dos puertas NOR unidas por una retroalimentación, (ver figura Circuito Flip-Flop S-R a base de compuertas NOR), o por dos compuertas NAND también unidas por una retroalimentación, (ver figura Flip-Flop SR a base de compuertas NAND.)





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 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, este tipo de flip-flop puede funcionar como una memoria de 1 bit. A partir de la figura Circuito Flip-Flop S-R a base de compuertas NOR, 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.

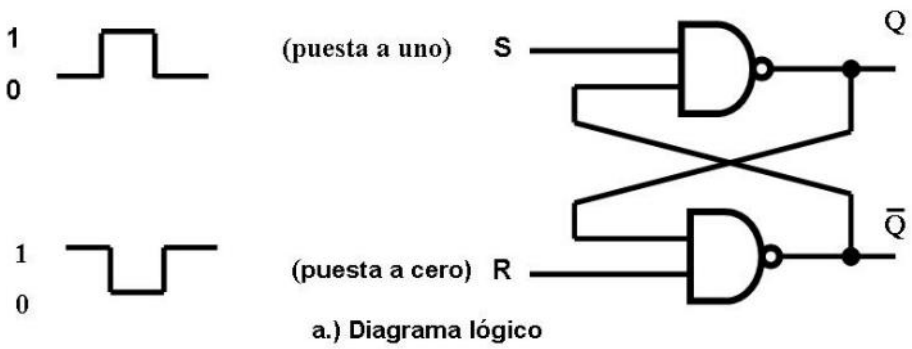
Después de cierto tiempo de retardo (μ, la salida de la puerta NOR inferior será $\overline{\mathbf{Q}}$=0).

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 Circuito Flip-Flop S-R a base de compuertas NOR, 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).

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 *Flip-Flop SR a base de compuertas NAND*.



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 Flip-Flop SR a base de compuertas NAND

En la figura Flip-Flop SR a base de compuertas NAND, 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 Implementación de Flip-Flop SR a partir de diversas compuertas básicas. 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.

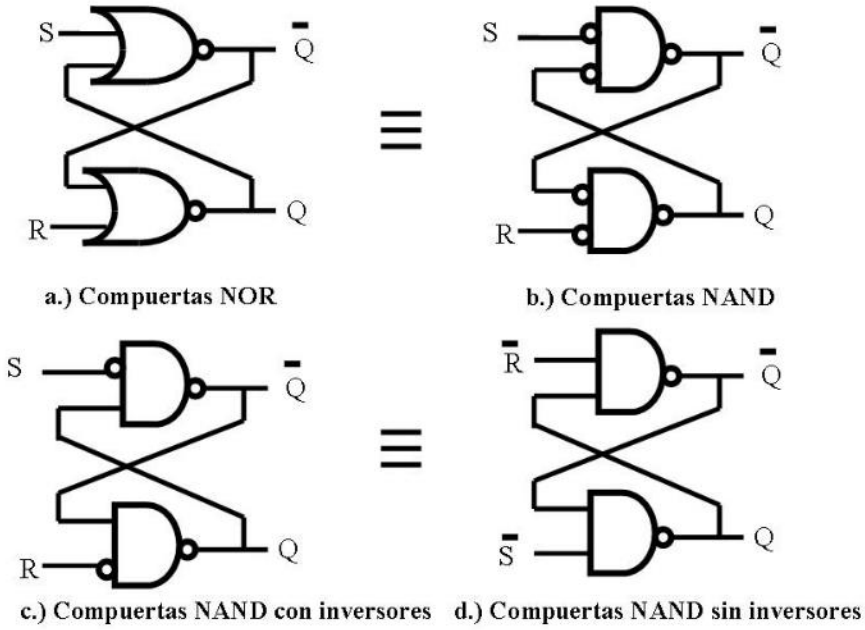


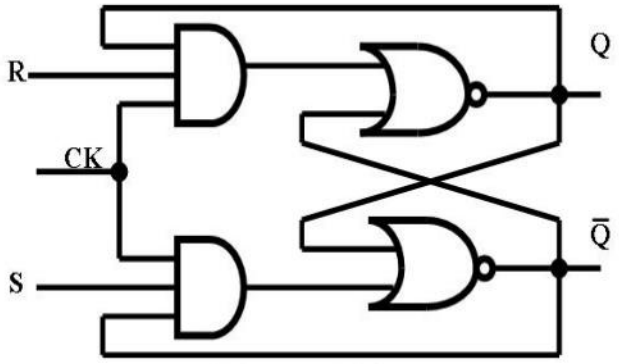
Figura Flip-flop SR

Implementación de Flip-Flop SR a partir de diversas compuertas básicas

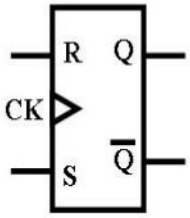
Existen otros tipos de flip-flops (RS, JK, T y D) a los cuales se les conoce como *flip-flop temporizados* y los cuales son muy utilizados en el diseño e implementación de circuitos secuenciales, veámoslos:

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 Flip Flop RS Temporizado 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



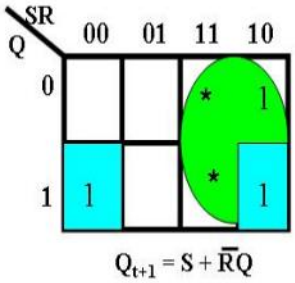
b.) Símbolo Lógico

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

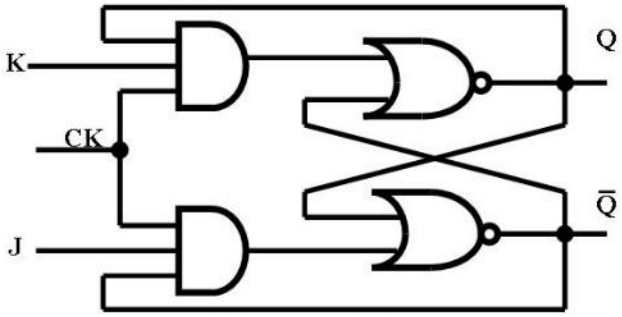


e.) Ecuación Característica

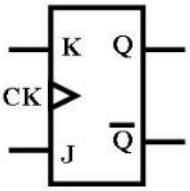
Flip Flop RS Temporizado

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 Flip-Flop JK Temporizado 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



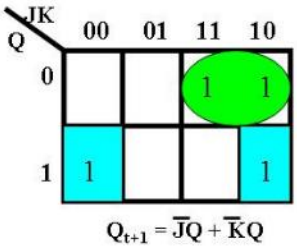
b.) Símbolo Lógico

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

Flip-Flop JK Temporizado

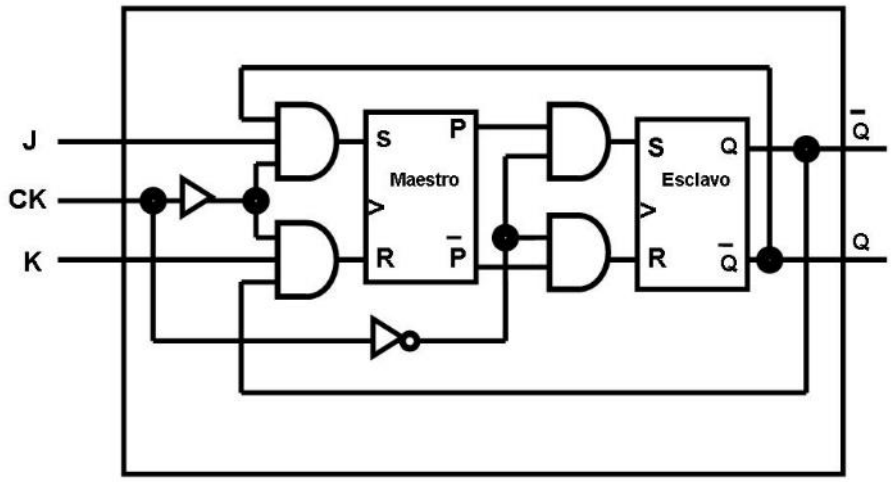
Las entradas JK solo realizan la función de puesta a 1, causando que la salida sea 1; la entrada K solo realiza la función de puesta a cero, provocando que la salida sea 0. Cuando J y K son 1, la función realizada se denomina función de conmutación: la salida se invierte.

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 Flip-Flop Maestro-Esclavo JK. 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.

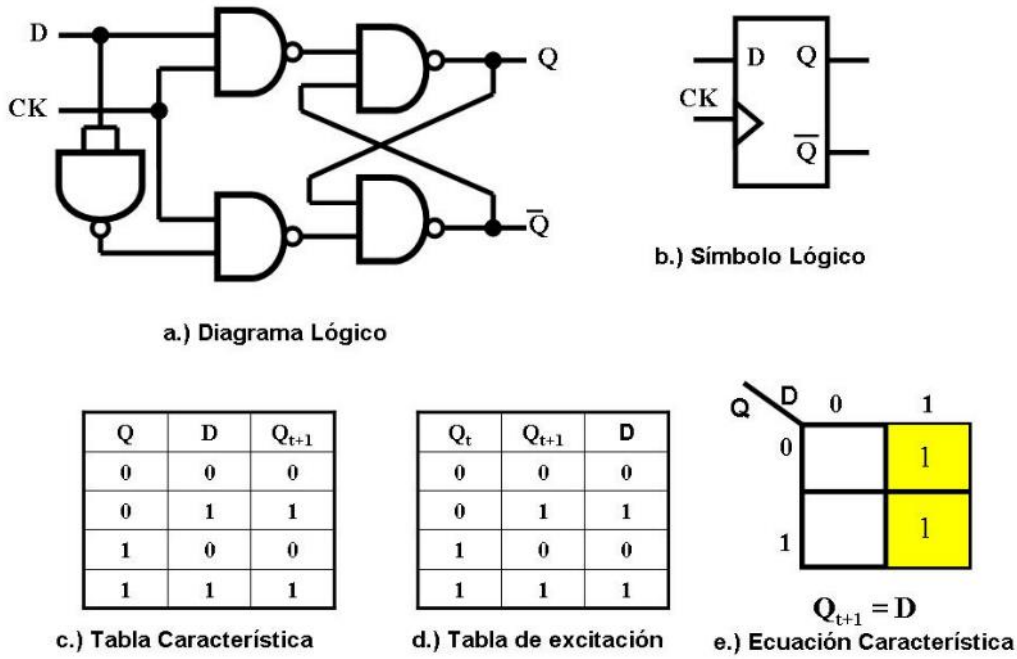
Se puede resolver el problema de la “captura de unos” por medio de la construcción de flip-flops activados por flanco, en los que el estado de la entrada se analiza solo en las transiciones del reloj (de alto a bajo) si el circuito se activa por flanco negativo o de bajo a alto, se trata de un flip-flop activado por flanco positivo, instantes en los cuales las entradas deberían estar estables.



Flip-Flop Maestro-Esclavo JK

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 Flip-Flop D Temporizado 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

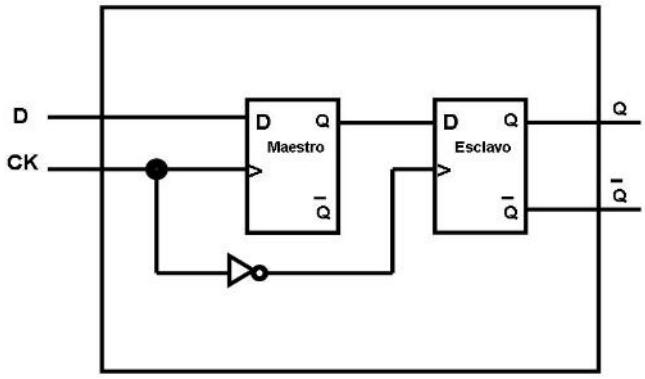
Flip-Flop D Temporizado

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 Flip-Flop Maestro-Esclavo D. 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 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 Flio-Flop Maestro-esclavo D, la transición de la salida se produce en el flanco negativo de la señal de sincronismo.

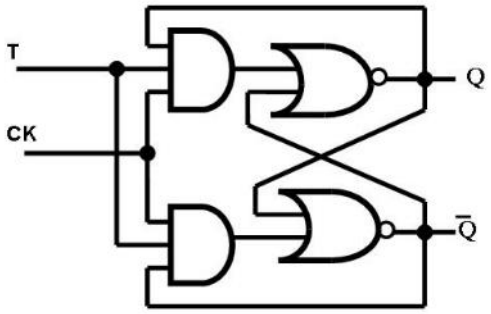
Un flip-flop activado por nivel puede cambiar sus estados en forma continua cuando la señal de reloj está en su estado activo (alto o bajo, según como se haya diseñado el flip-flop). Un flip-flop activado por flanco solo cambia en una transición creciente o decreciente de la señal de reloj.



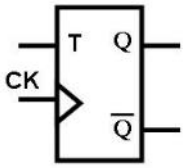
Flio-Flop Maestro-esclavo D

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 Flip-Flop T Temporizado). En dicha figura se muestra su símbolo lógico, tabla característica, tabla de excitación y ecuación característica.



a.) Diagrama Lógico



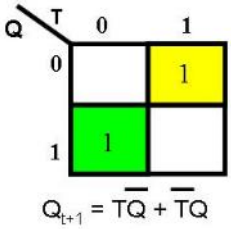
b.) Símbolo Lógico

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

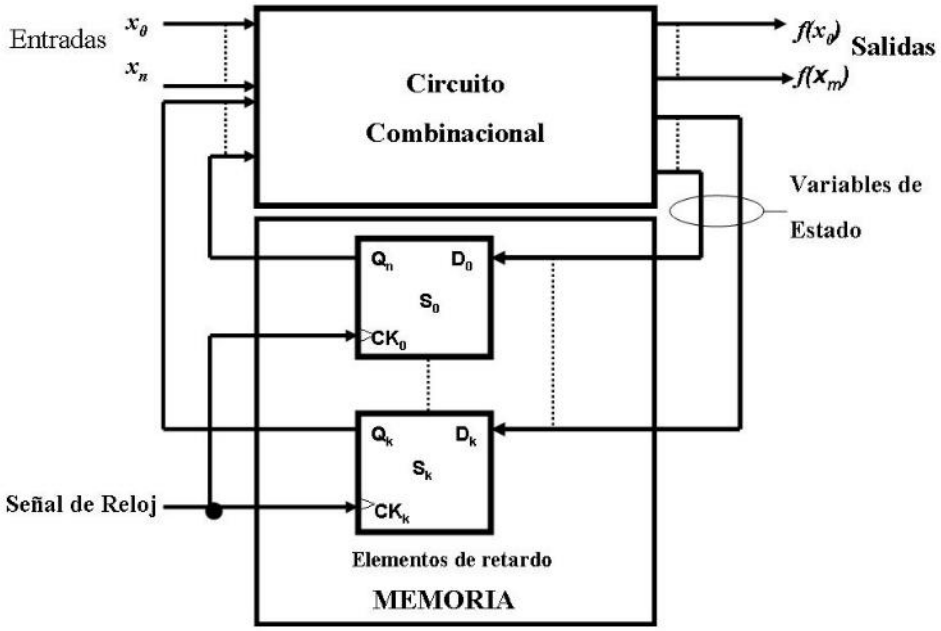


$Q_{t+1} = \overline{T}Q + T\overline{Q}$

e.) Ecuación Característica

Diseño de un Circuito Secuencial

Haciendo nuevamente referencia al circuito secuencial de la figura Modelo clásico de un circuito secuencial 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 temaTemporizadores.



Modelo clásico de un circuito secuencial

El procedimiento para diseñar un circuito secuencial síncrono es el siguiente:

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.)
Obtención tabla de estado	De la información recabada del punto anterior, se obtiene la tabla de estado.
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.
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.
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
Elección del flip-flop	Se selecciona el tipo de flip-flops que se va a utilizar en el circuito secuencial.
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).
Obtención de las funciones de salida	Usando cualquier método de simplificación (por ejemplo, mapas de Karnaugh o álgebra de Boole)

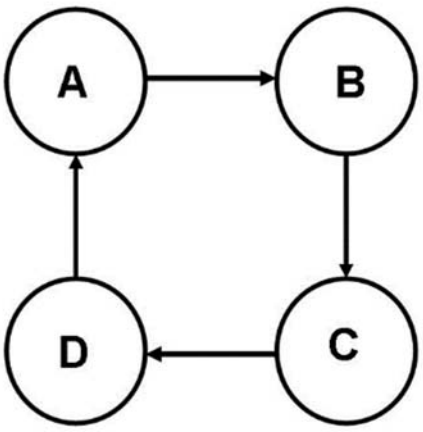
	se deducen las funciones de salida del circuito n flip-flops.
Dibujar el diagrama lógico	Se dibuja el diagrama lógico (y se comprueba el circuito secuencial).
Dibujar el diagrama eléctrico (opcional)	Se dibuja y se alambra el diagrama eléctrico.

A continuación se muestra 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 Ejemplo 1. Diagrama de Estados)



Ejemplo 1. Diagrama de Estados

2. Obtención tabla de estado

Estado Presente	Estado Futuro
A	B
B	C
C	D
D	A

Tabla de estado

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 Q_0 Q_1	Estado Futuro Q_0 Q_1
0 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0

8. Obtención de las funciones de salida

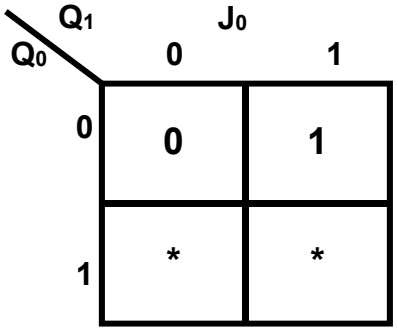
Usando cualquier método de simplificación (por ejemplo, mapas de Karnaugh o álgebra de Boole) se deducen las funciones de salida del circuito de los n flip-flops.

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:

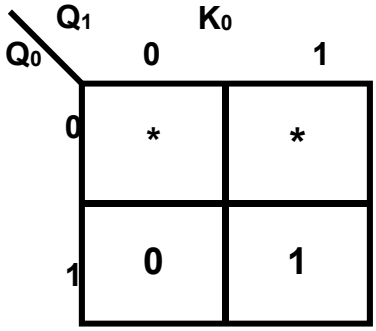
Q_t	Q_{t+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

Tabla de excitación

Estado Presente Q_0 Q_1	Estado Futuro Q_0 Q_1
1 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0



$J_0 = Q_1$



$K_0 = Q_1$

Q_0 \ Q_1	0	1
0	1	*
1	1	*

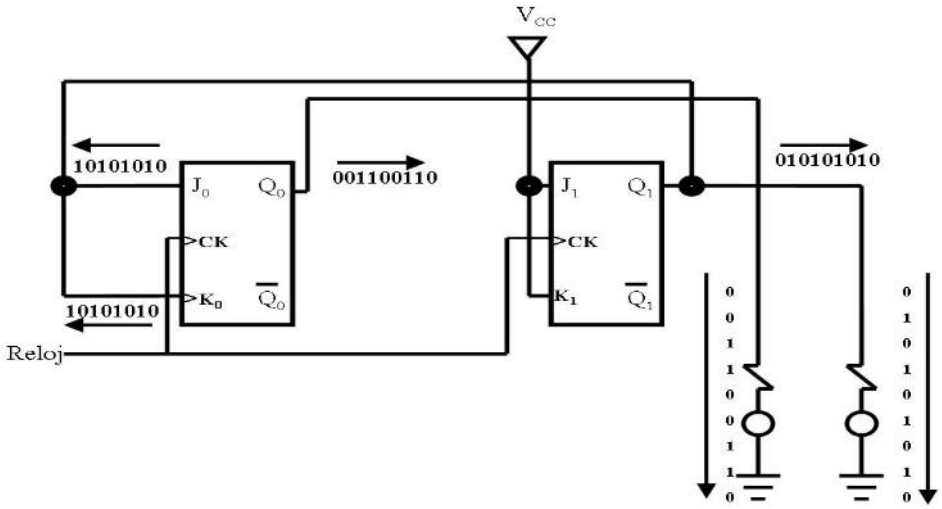
$J_1 = 1$

Q_0 \ Q_1	0	1
0	*	1
1	*	1

$K_1 = 1$

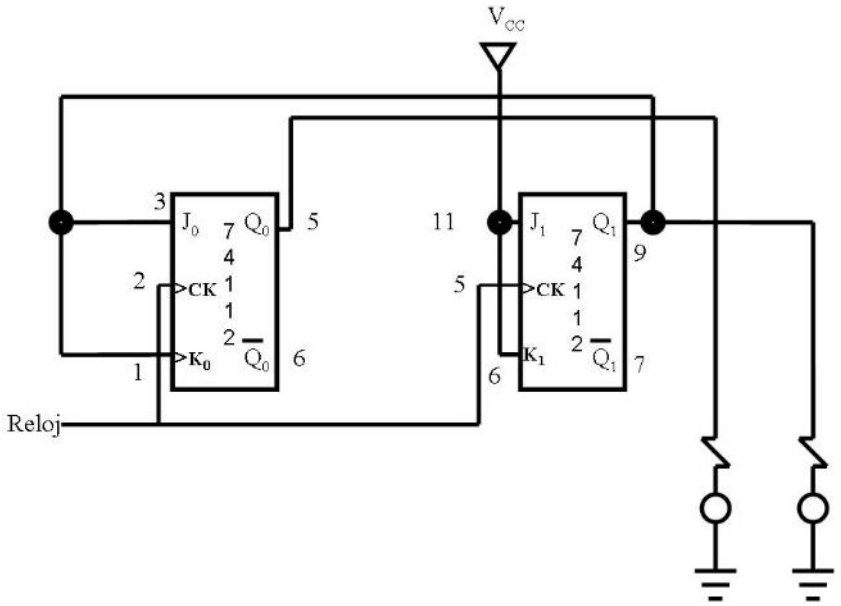
9) Dibujar el diagrama lógico

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.

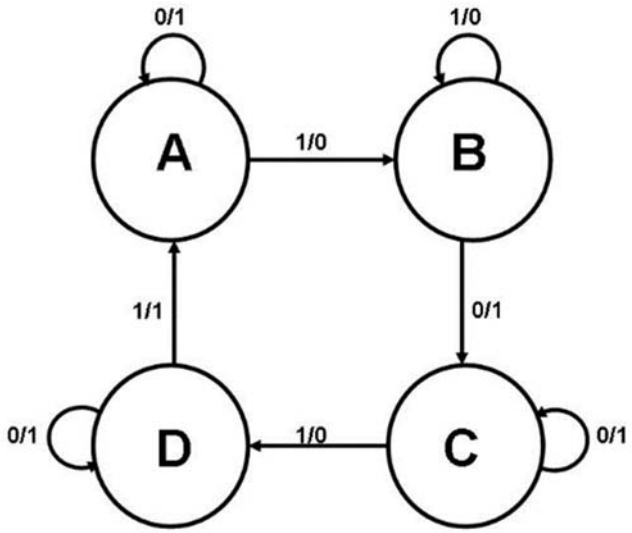


Figura 6.10. Ejemplo 2. Diagrama de Estados

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
C	C	D	1	0
D	D	A	1	1

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 que se va 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 que se va 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
1	0	1	0	1	1	1	0
1	1	1	1	0	0	1	1

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
1	0	1	0	1	1	1	0
1	1	1	1	0	0	1	1

Tabla de estados

Q_1Q_0 / X	00	01	11	10
0	*	1	*	*
1	*	0	*	*

J_0

$J_0 = \overline{X}$

Q_0Q_1 / X	00	01	11	01
0	1	*	0	0
1	1	*	1	0

K_0

$K_0 = \overline{Q_0} + XQ_1$

Q_1Q_0 / X	00	01	11	10
0	0	*	*	0
1	1	*	*	1

J_1

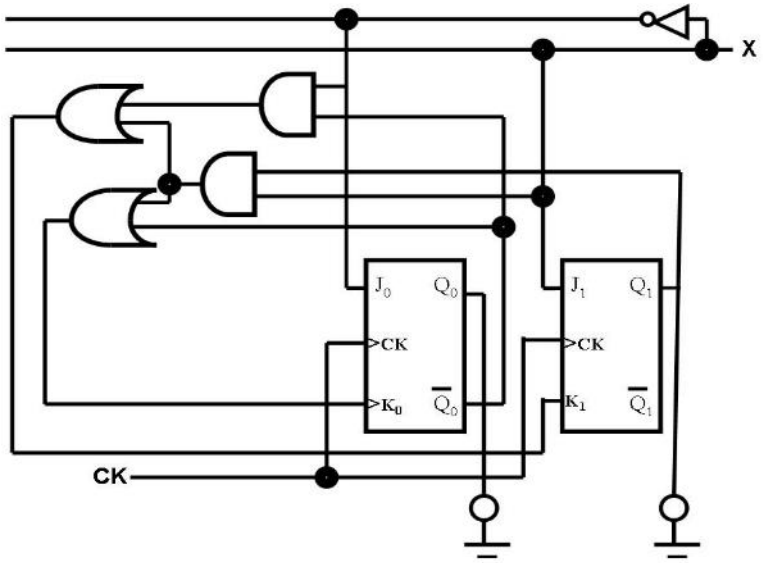
$J_1 = X$

Q_0Q_1 / X	00	01	11	10
0	*	1	0	*
1	*	0	1	*

K_1

$K_1 = \overline{X}\ \overline{Q_0} + X Q_1$

9. Dibujar el diagrama lógico



6.4. Registradores de corrimiento

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

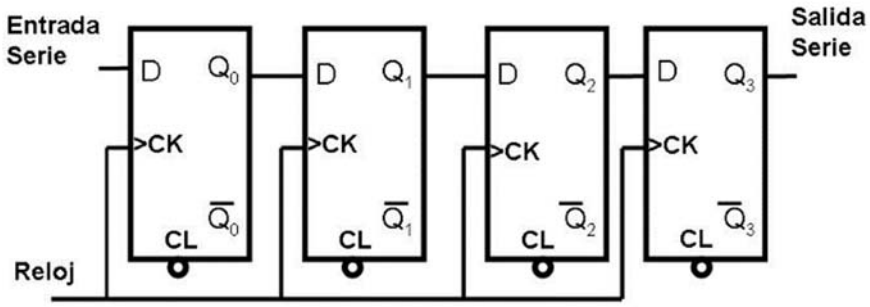
- Registro de corrimiento
- Registro en paralelo
- Registro universal

Registro de corrimiento

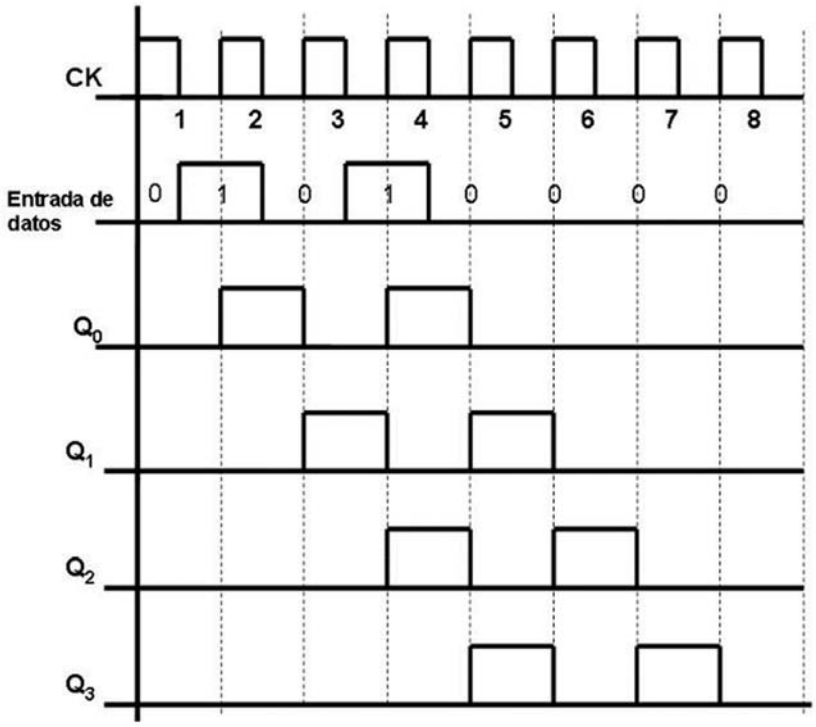
Un registro de corrimiento acepta y/o transfiere información vía serie. Un registro se puede construir utilizando alguno de los diferentes tipos de flip-flops, por ejemplo RS, JK, T y el D. En esta sección mostramos un registro de corrimiento (entrada serie-salida serie) de 4 bits utilizando flip-flop tipo D, (ver Figura Registro de desplazamiento de 4 bits). En la Figura Registro de corrimiento (entrada serie-salida serie) 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 (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.



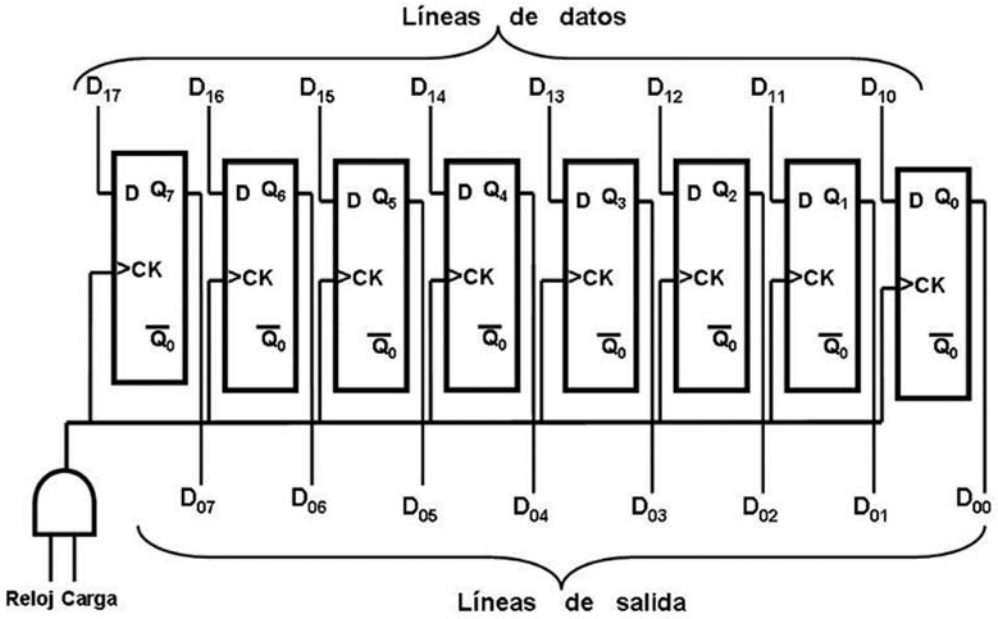
Registro de desplazamiento de 4 bits



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 Registro en paralelo de 8 bits. 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 D_{10} a la D_{17}.



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 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:

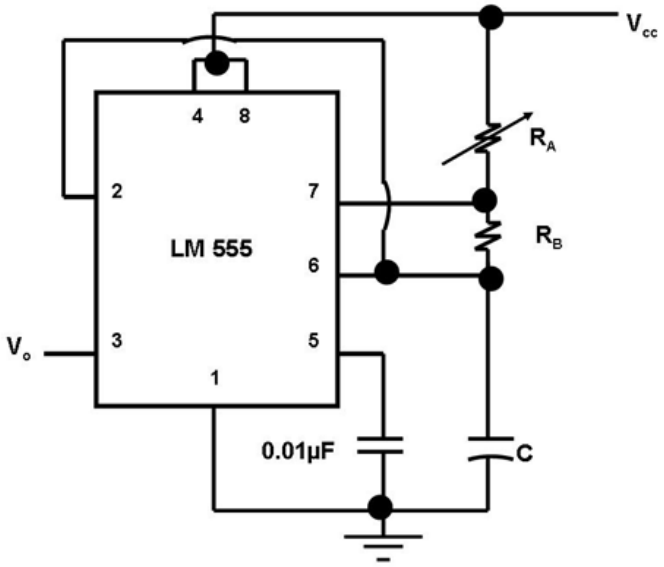
- Monoestable
- Biestable, y
- Astable

Los circuitos multivibradores monoestables encuentran vasta 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 C.I. LM555



C.I. LM555 (Configuración Astable)

Para generar un oscilador de onda cuadra que tenga un funcionamiento de 1 [Hz] a 20 [Hz], utilizando el C.I. LM555 (en configuración astable) se realiza de la siguiente manera.

Procedimiento

Se tiene que	F1=1 Hz	=	T1=1 /f1	=	1 seg
	F2= 20 Hz	=	T2=1 /f2	=	o.o5 seg

$T = t_1+t_2$	(1)
$t_1 = (0.693)\ C\ (R_A+R_a)$	(2)
$t_2 = (0.693)\ C\ (R_a)$	(3)
$T = (0.693)\ C\ (R_A+2R_B)$	(4)

Se propone que C=200[μF] (Capacitor electrolítico de valor comercial)

Despejando R_B de la ec. (3)

$$R_B = \frac{0.05}{(0.693)(220\times10^{-6})} = 327.90 \text{ [ohm]}$$

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 2^n-1. En la figura Contador binario 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 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.

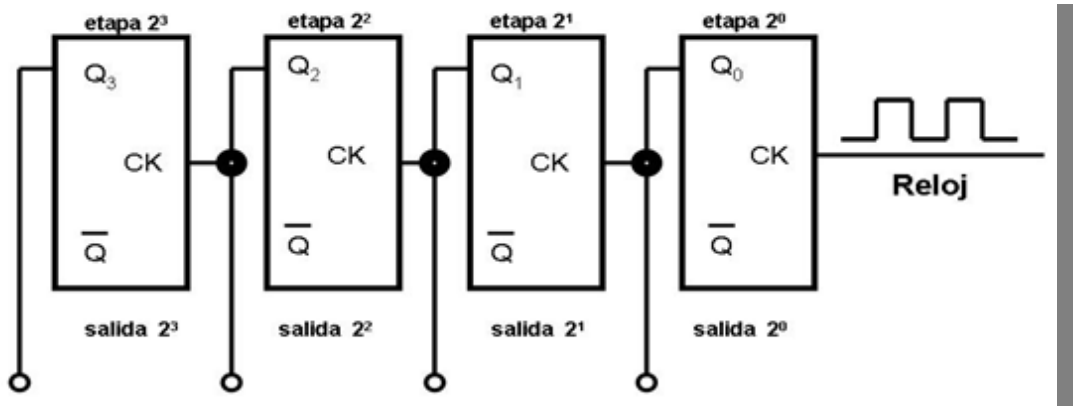


Diagrama a bloques de un Contador Binario

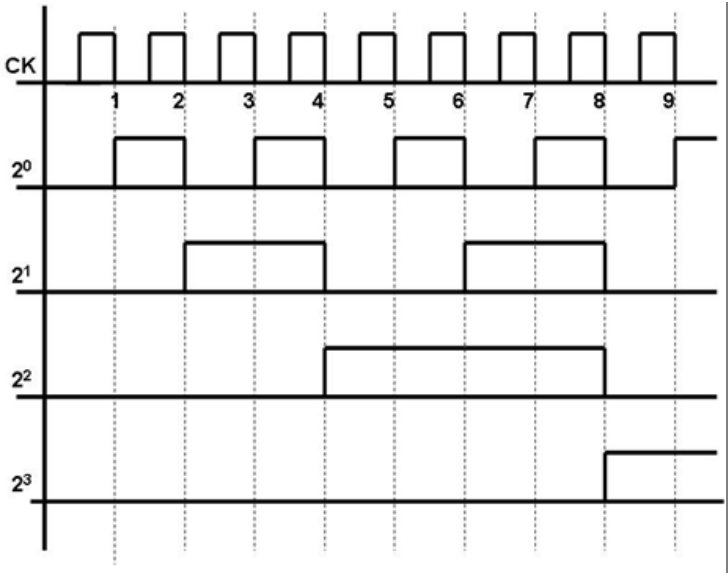
Utilizando lógica positiva, resulta que el flip-flop terminará cuando la entrada cambie de 1 a 0. Puesto que el primer impulso de reloj, aplicado a la entrada, cambia la salida de la etapa 2^0 de 0 a 1, la etapa 2^1 no se terminará. Solamente cambia de estado la etapa 2^0. La entrada del segundo impulso 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 Contador binario de cuatro etapas).

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	

Contador binario de cuatro etapas

A partir de la tabla Contador binario de cuatro etapas 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 Formas de onda. 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 Contador binario 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.



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 el objetivo 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 Diagrama a bloques).

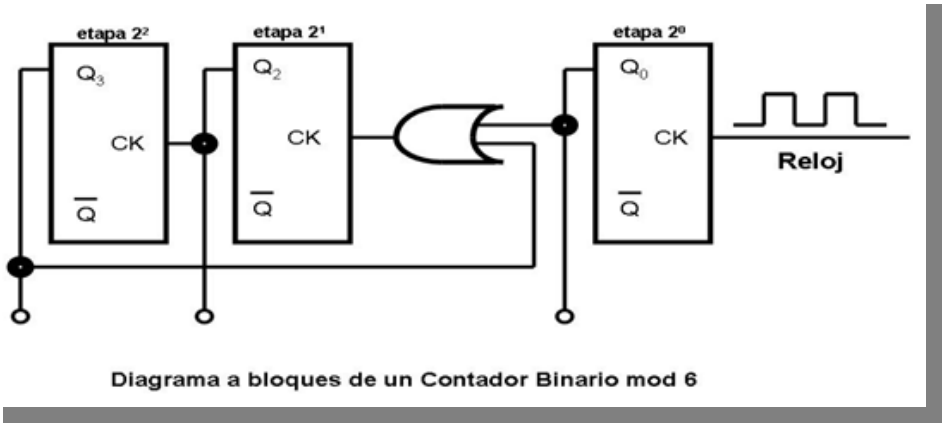
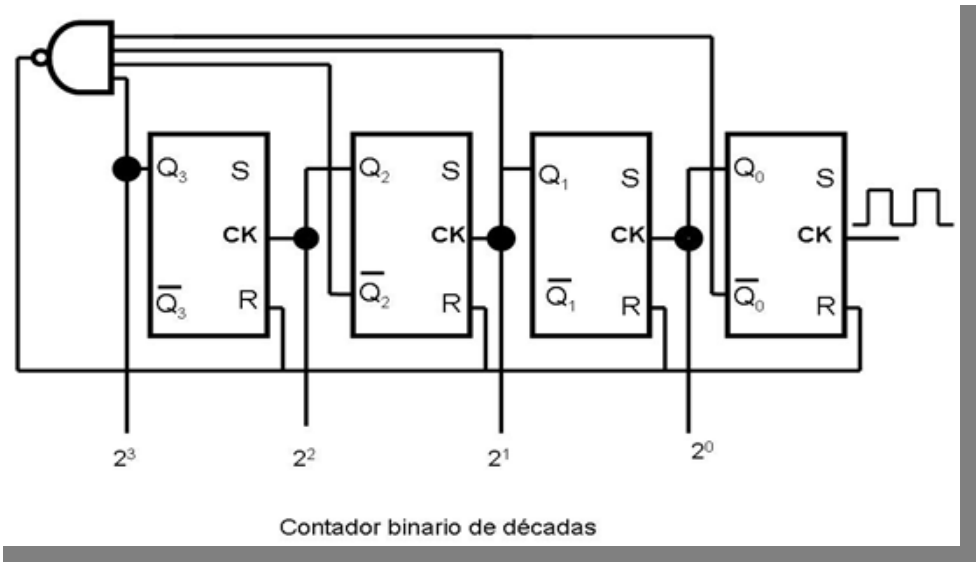


Diagrama a bloques de un Contador Binario mod 6

En la figura Diagrama a bloques se muestra un circuito para el conteo. Un contador para el conteo de 6 se suele llamar “módulo 6” (generalmente se abrevia mod 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 Contador binario de décadas y su tabla de recuento es la tabla Contador binario decimal.



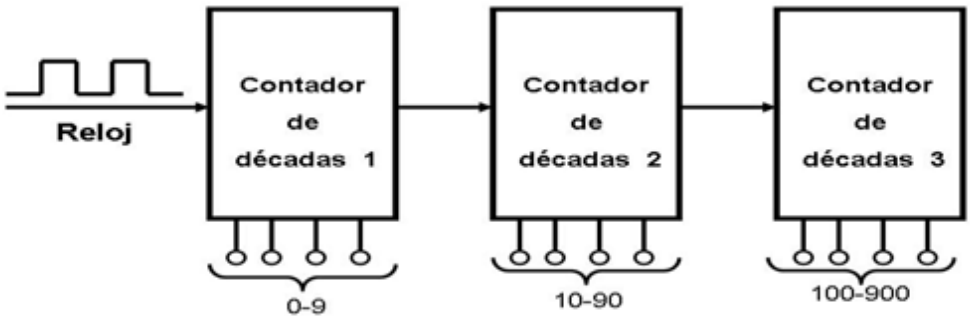
Contador binario de décadas

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 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

Contador binario decimal

Para conseguir un contador en escala de 10 pueden asociarse varios contadores de décadas (ver figura Contador de décadas).



Contador de décadas para conteo en la escala de 10

RESUMEN

Los circuitos secuenciales incorporan un conjunto de dispositivos electrónicos capaces de almacenar datos de manera indefinida (para las memorias de tipo RAM, mientras se mantenga la energía), estos circuitos son las memorias y funcionan como elementos de realimentación para un circuito combinacional o procesador de datos. Al hablar de elementos de memoria, se incorpora también una variable que no se consideraba en los circuitos combinacionales, el tiempo. Los circuitos secuenciales se clasifican en síncronos y asíncronos.

En un circuito secuencial asíncrono, los cambios de estado ocurren por los retardos asociados con las compuertas lógicas utilizadas en su implementación, es decir, estos circuitos no usan elementos especiales de memoria, pues se sirven de los retardos debidos a los tiempos de respuesta de las compuertas lógicas utilizadas. Debido a que los retardos están fuera del control del diseño ocasionan problemas de funcionamiento y además no son idénticos en cada compuerta lógica. Estos retrasos están determinados por los niveles de implementación, los tipos de tecnología empleada y el retraso de cada dispositivo. Por ejemplo, para un sumador completo diseñado con compuertas, se pueden tener al menos dos niveles de diseño para un bit, sin embargo al utilizar sumadores en cascada los retrasos son acumulativos.

Los circuitos secuenciales síncronos cambian de estado en los ciclos marcados por una señal de entrada oscilatoria de onda cuadrada denominada reloj. Con esto se pueden evitar los problemas que tienen los circuitos asíncronos originados por cambios de estado no uniformes en todo el circuito. Adicionalmente, los circuitos síncronos, emplean elementos de memoria llamados flip flops, los cuales son implementados mediante compuertas digitales como NAND y NOR e incorporan la realimentación de señales en su diseño.

Los flip flops almacenan un bit de información y pueden ser de cuatro tipos: SR, JK, D y T. Todos los los flip flops cuentan con dos salidas Q y Q' y las entradas dependen del tipo. Los SR tienen dos entradas S y R. Al tener la señal S alta, la salida Q se enciende (set), es decir el estado del flip flop es uno. Si la señal R es alta, el estado Q va a ser cero (reset). Para las entradas S=0 y R=0 el flip flop mantiene el valor que tenía y para los valores S=1 y R=1 no está definido por lo que no se usa.

Los flip flops tipo JK operan de manera semejante a los SR, la diferencia es que sí está definido el uso para J=1 y K=1, en este caso la salida Qt+1es Q', es decir cambia de estado.

Los flip flops D cambian de estado siguiendo al valor de la entrada D, mientras que los T, cambian de estado para cuando el valor de T es uno; si el valor de T=0, el valor del flip flop se mantiene.

Mediante arreglos de flip flops y lógica combinacional, podemos diseñar y construir registros, los cuales son elementos de memoria que pueden almacenar varios bits en forma de palabras. También mediante arreglos de flip flops podemos construir circuitos contadores que nos permiten sincronizar dispositivos que trabajan a diferentes frecuencias.

Finalmente se presentó una metodología para el diseño de circuitos lógicos secuenciales que nos permiten la construcción de los mismos desde el enunciado del problema.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga (2010)	6 y 8	124-127 131 136-142 184-188 251-259
Mano (1986)	6 y 7	208-210 210-216 265-281
Stallings (2006)	Apéndice B 12	758-762 764-767 440-446

Mano, Morris. (1986). *Lógica Digital y diseño de computadores*. México: Prentice Hall Hispanoamericana.

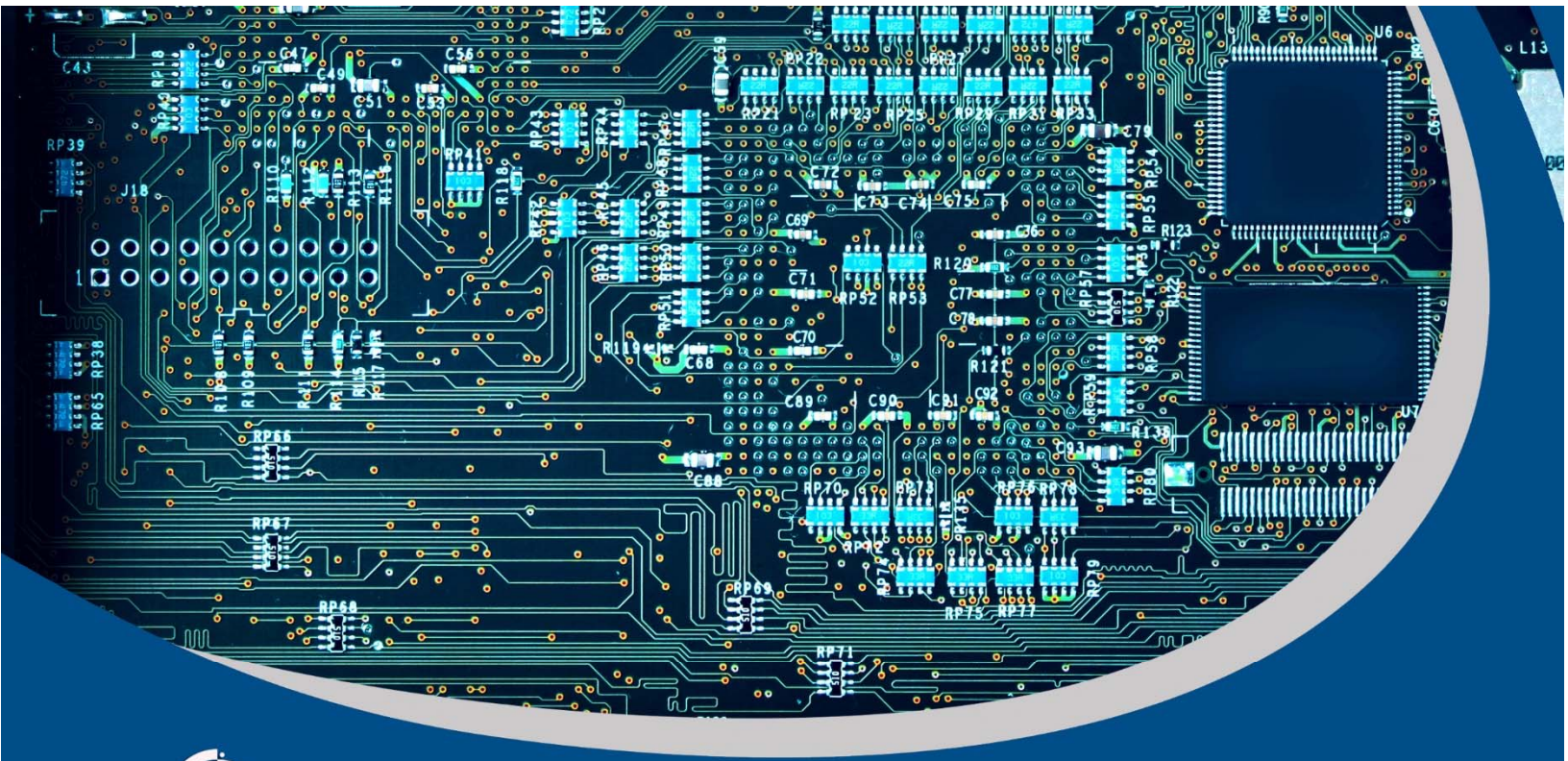
Quiroga, Patricia. (2010). *Arquitectura de computadoras*. México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores*. Madrid: Prentice Hall

Licenciatura: Informática

UNIDAD 7

Memorias



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 actualizado 2016

OBJETIVO PARTICULAR

El alumno conocerá las características de los tipos, ciclos y organización de las diferentes memorias de una computadora.

TEMARIO DETALLADO
(8 horas)

7. Memorias

- 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. Refrescamiento
- 7.3. Mapa de memoria
  - 7.3.1. Organización de memoria
  - 7.3.2. Tendencias tecnológicas de memorias (holograma, SSD, FLASH)
- 7.4. Memoria caché
- 7.5. Memoria virtual

INTRODUCCIÓN

Dentro del modelo de von Neumann la memoria fue el elemento que permitió la construcción de las computadoras como ahora las conocemos. Su desarrollo ha tenido una serie de transformaciones y cambios en las directivas de su construcción. Hasta antes de que von Neumann planteara la utilización de memorias para almacenar instrucciones, las computadoras requerían introducir tanto datos como programas cada vez que se necesitaba un proceso diferente.

El almacenamiento de datos en las computadoras utiliza varios medios, superficies magnéticas, núcleos magnéticos y semiconductores electrónicos. Las primeras son la base de discos y cintas magnéticas, mientras que los electrónicos conforman las memorias de tipo RAM y ROM. La diferencia entre las unidades de almacenamiento (que se verán en detalle en la unidad 8) como discos y cintas magnéticas y las memorias es que las primeras guardan la información aunque la computadora esté apagada, que son más baratas y que pueden almacenar grandes cantidades de información, aunque tienen la desventaja de que son más lentas en los procesos de lectura y escritura.

Los registros que manejamos anteriormente se utilizan para conformar un tipo de unidades más grandes de información llamadas memorias. Los registros de una computadora digital son de dos tipos: operacionales y de almacenamiento. Los primeros incluyen circuitos lógicos combinacionales que, además de almacenar la información, permiten realizar alguna modificación a los mismos, mientras que los de almacenamiento solamente retienen la información sin transformarla. La mayoría de los registros utilizados en las computadoras son de este tipo.

Los registros son construidos mediante celdas binarias (flip flops), las memorias están formadas por series de registros. Para que las memorias puedan operar, deben cumplir con las siguientes propiedades básicas: representación binaria de datos, tamaño pequeño, costo de almacenamiento por bit reducido y el tiempo de acceso a los datos (lectura o escritura) debe ser rápido.

En esta unidad estudiaremos los diferentes tipos de unidades de memoria presentes en las computadoras: RAM (y sus variantes), ROM, caché, así como sus características y la estructura de uso dentro de la computadora, llamada mapa de memorias

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. 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 (CI) 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 en relación con el 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	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	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 actualizar (o recargar de "*refresh*") 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 actualizar 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.

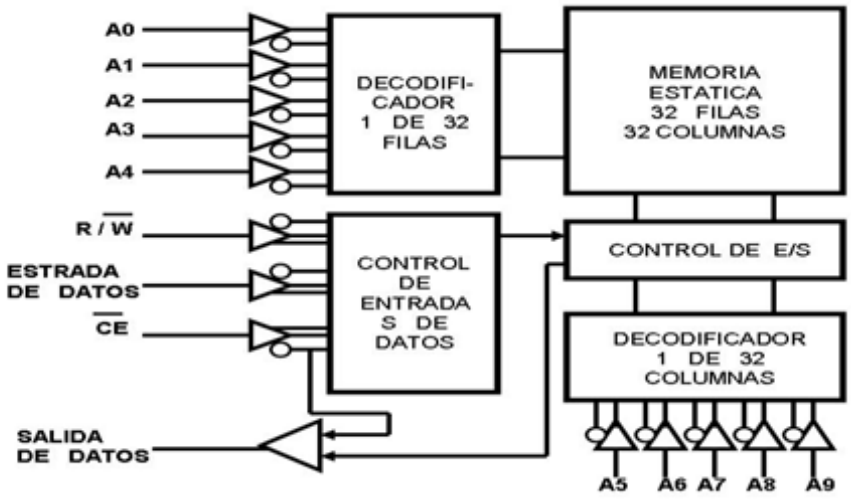
	Debido a que las localidades de memoria estáticas requieren de más componentes que las localidades de memoria dinámicas, las memorias dinámicas siempre están un paso adelante de las estáticas en cuanto a la densidad de las memorias. Actualmente en el mercado se encuentran memorias dinámicas de 4 Gbytes y estáticas de 512 Kbytes. Las memorias dinámicas son más baratas que las estáticas.

Organización de las memorias RAM

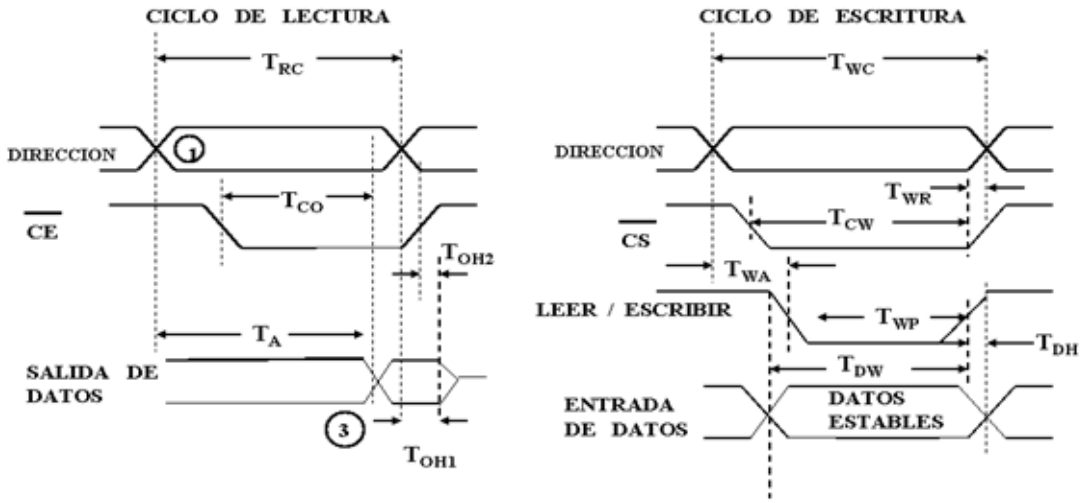
Las memorias RAM tienen líneas de dirección, líneas de datos, líneas para la alimentación de la energía, una o más líneas para habilitar el integrado (CS ($\overline{CS}$ o $\overline{CE}$) y líneas de control para indicar la dirección del flujo de datos (Lectura o Escritura).

Una memoria RAM consta de dos bloques funcionales: Arreglo de localidades de memoria y los circuitos internos de interfaz, ver figuras 1a. y 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.

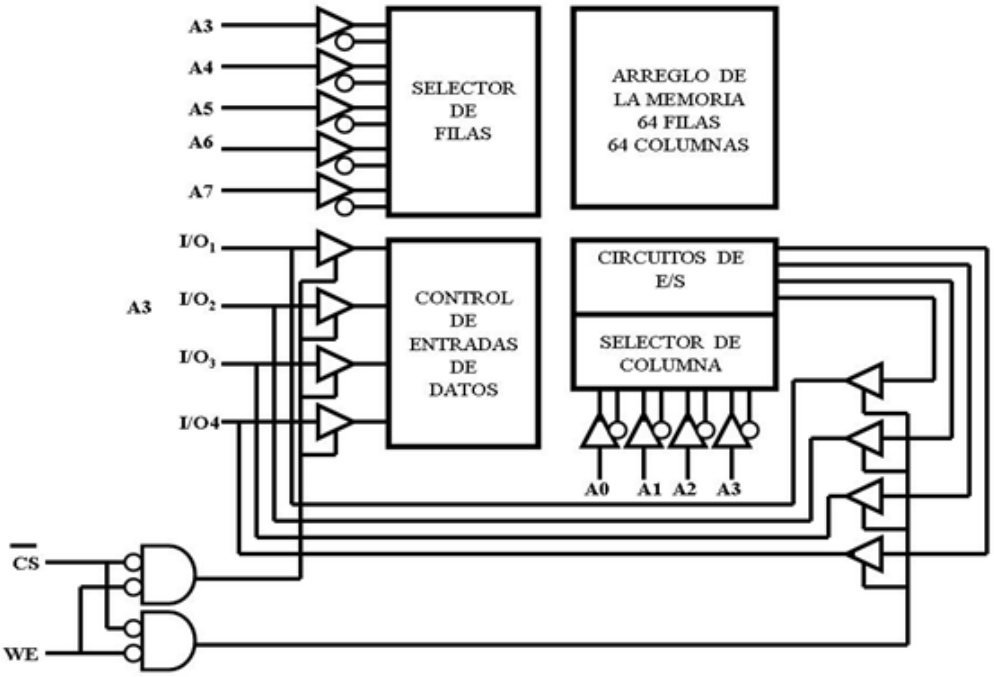
Los circuitos internos, una vez seleccionado el integrado y la localidad de la memoria, determinan si se va a recibir un dato (escribir) o si se va a enviar (leer).



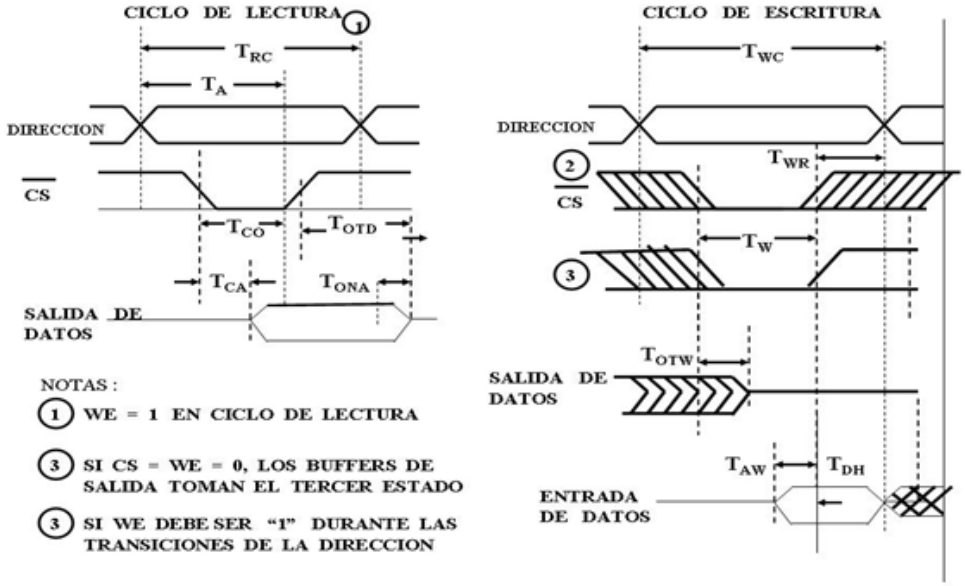
1a) Diagrama interno de la Memoria RAM 2102



1b Diagramas de tiempo de la memoria RAM 2102



2a) Memoria RAM 2114 - Diagrama a Bloques -



NOTAS :

1. WE = 1 EN CICLO DE LECTURA
3. SI CS = WE = 0, LOS BUFFERS DE SALIDA TOMAN EL TERCER ESTADO
3. SI WE DEBE SER "1" DURANTE LAS TRANSICIONES DE LA DIRECCION

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:

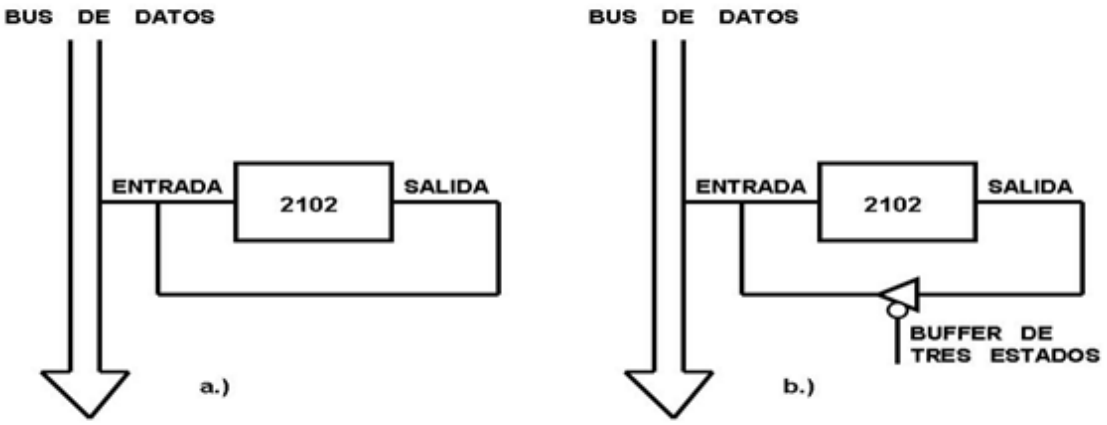
1. E/S separadas sin líneas OD.
2. E/S comunes sin línea OD.

3. E/S separadas con línea para deshabilitar salidas (OD, Output Disable).
4. E/S comunes con la línea para deshabilitar salidas (OD).

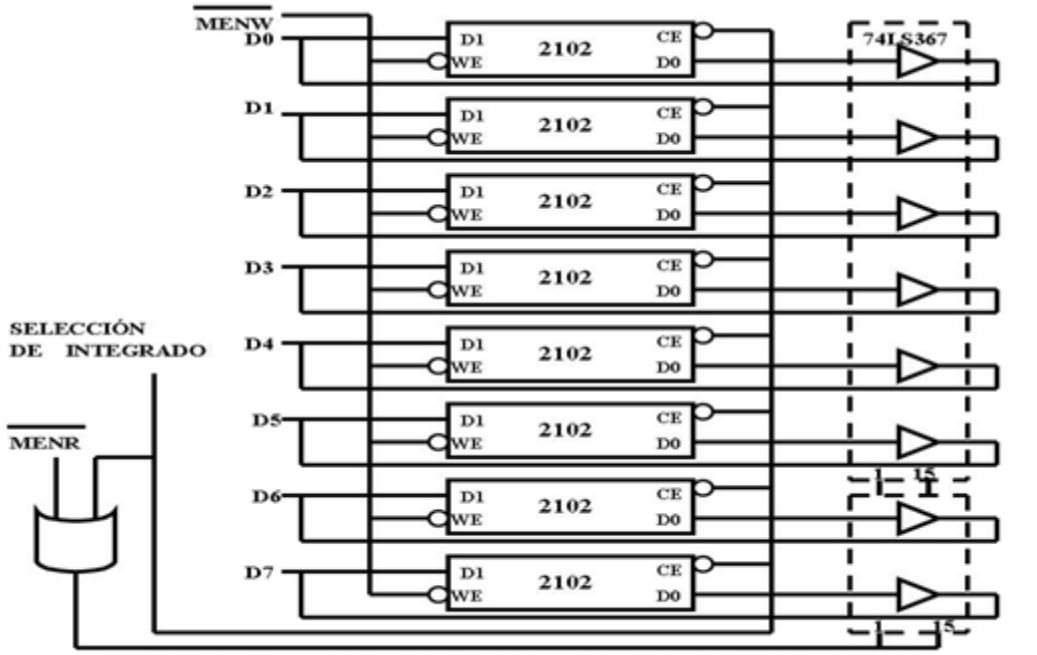
Memorias con E/S separadas

Un ejemplo de memoria con entrada y salida separadas es el CI RAM 2102 de Intel, ver figura, donde 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 3b. La figura 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.



3. Conexión de un 2102 al Bus de Datos



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 $\overline{MENR}$ se combinan para activar al buffer 74LS367. Durante una operación de Lectura (Leer de Memoria) la entrada **3** ($R/\overline{W}$) tiene nivel 1, al conectar esta entrada a la señal $\overline{MENW}$ se cumple este requisito. Cuando la 2102 ha sido seleccionada y se activa la línea $\overline{MENR}$, 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.

Memorias con E/S comunes

Un ejemplo de memoria de entrada y salida comunes es la RAM 2114, ver figura 2a., donde 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 $\overline{WE}$ para indicar una operación de leer ($\overline{WE}$ = 1) o una de escribir ($\overline{WE}$= 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.

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 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 de escritura (escribir) la línea OD debe tomar nivel 1 para poner las líneas de Salida de Datos en el tercer estado.

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 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 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.

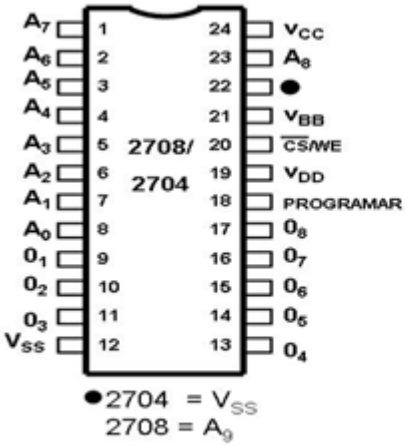
En las especificaciones del fabricante se indican las características para programar y borrar las memorias EPROM. Una buena costumbre es la de tapar las ventanas para prevenir las exposiciones accidentales de luz ultravioleta que pudieran alterar el contenido de las celdas. Las memorias EPROM más populares se muestran en la siguiente Tabla:

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 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”.

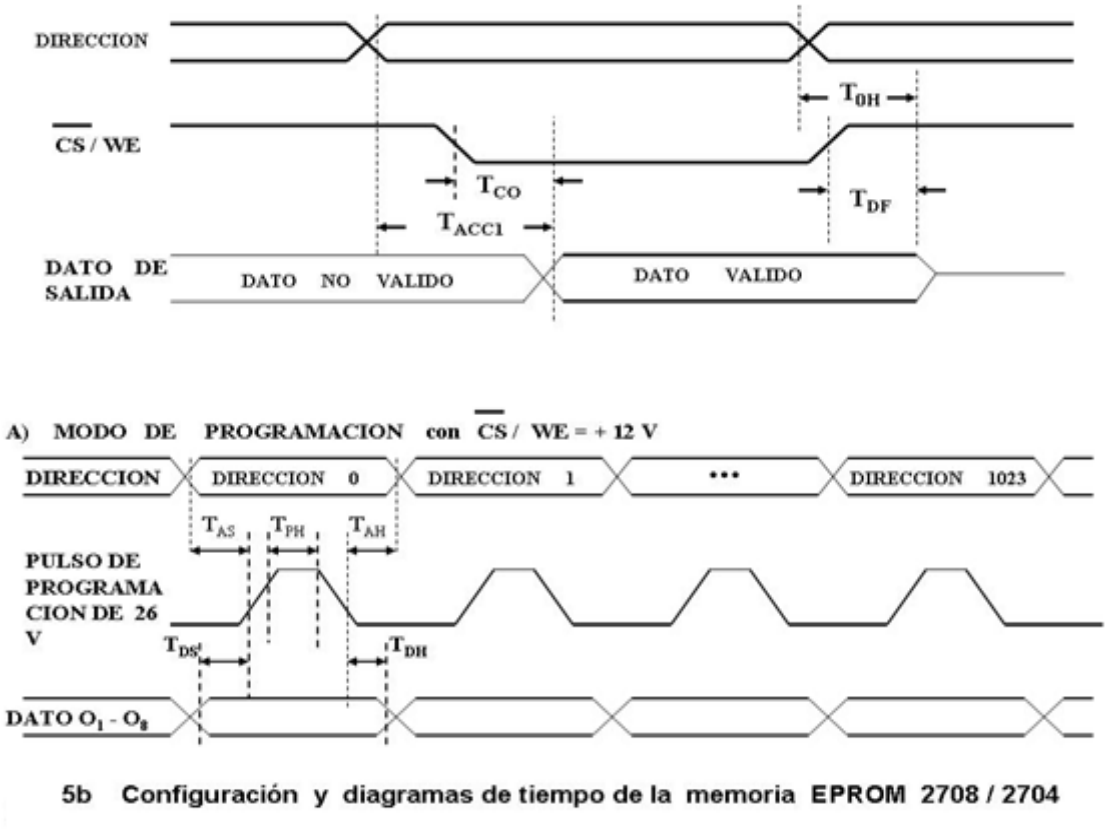
Los C.I. 2708 y 2704 son dos EPROM de 8192 bits (1024 x 8) y de 4096 bits (512 x 8) respectivamente. Los dos están fabricados con técnica MOS con canal N de silicio en integrados de 24 “pins”, ver figura 5a. Estos integrados tienen una ventana transparente por los que se accede directamente el circuito 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

5a) Descripción de terminales y tiempos del EPROM 2708/2704



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 A0 - A9
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

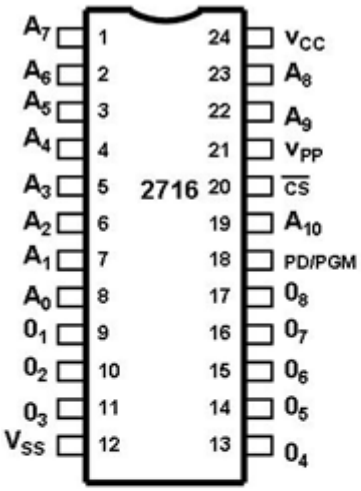
Donde

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 que 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 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

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 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.

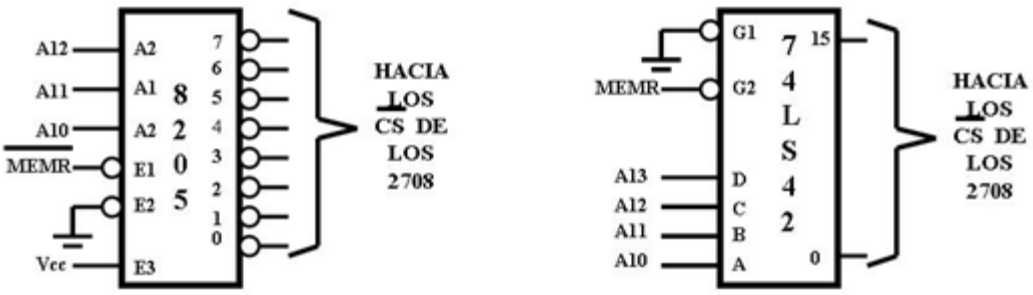
Arquitectura de memorias ROM

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.

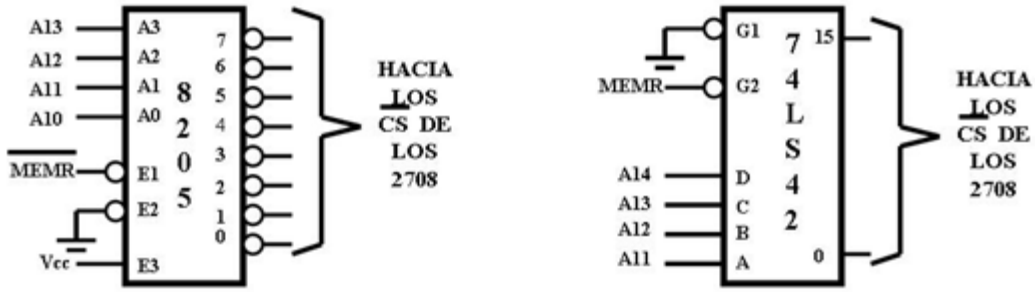
Direccionamiento no-absoluto en memorias ROM

Las figuras 7a. y 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 7a. es de 0 a 1FFFH y el de la figura 7b. es de 0 a 3FFFH. Los dos decodificadores se habilitan únicamente en ciclos de Lectura de Memoria, MEMR=0

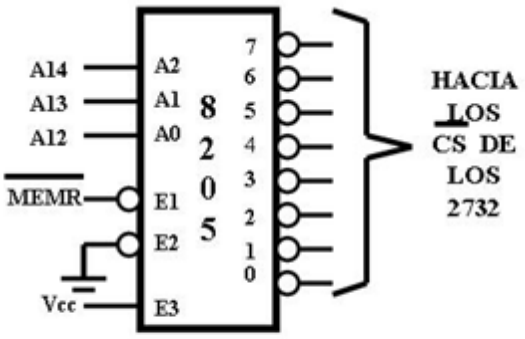
Los decodificadores de la figura 8. permiten seleccionar 8 (el 8205) y 16 (el 74LS42) memorias ROM de 2K (2716). El decodificador de la figura 9. permite seleccionar 8 memorias ROM de 4K (2732). Todos estos decodificadores proporcionan direccionamiento "no-absoluto".



7. Direccionamiento no absoluto con decodificador para a.) 8K y b.) 16K con EPROM's 2708



8. Direccionamiento no absoluto con decodificador para a.) 16 K y b.) 32 K con EPROM's 2716

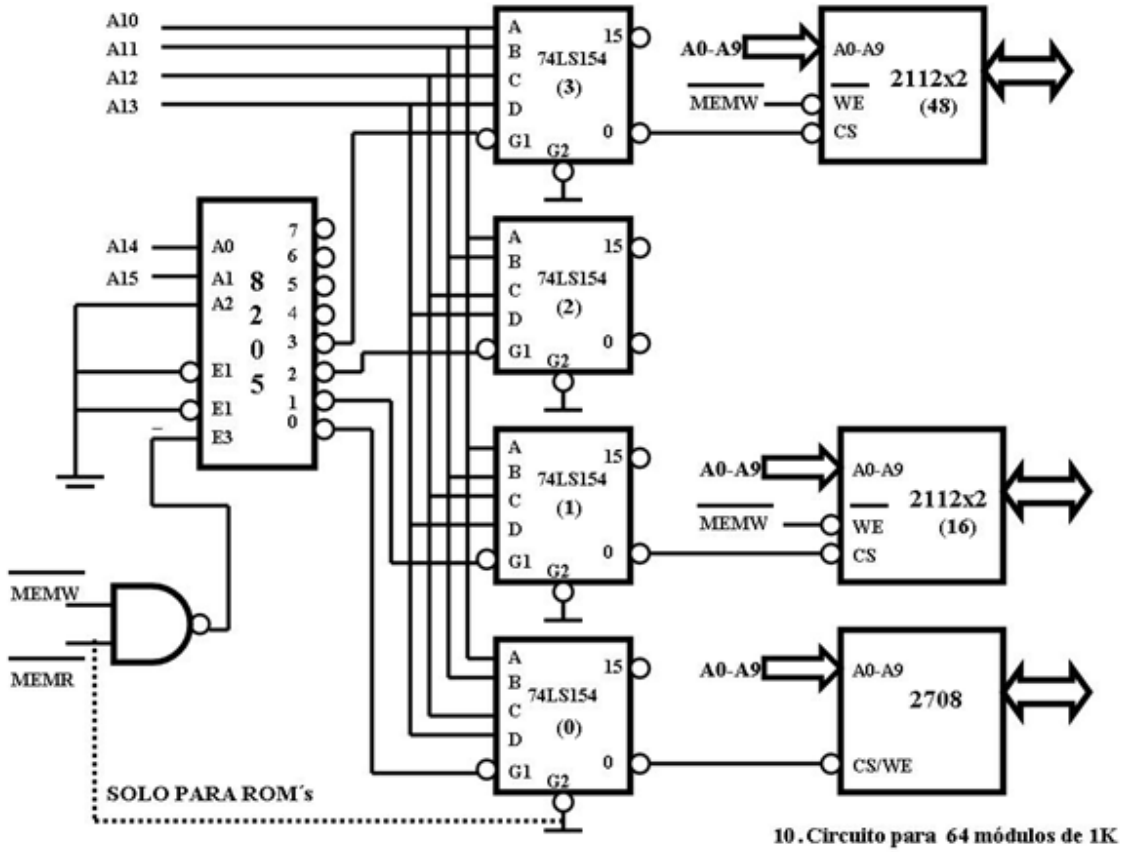


9. Direccionamiento no absoluto con decodificador para 32K con EPROM's 2732.

Direccionamiento absoluto en memorias ROM

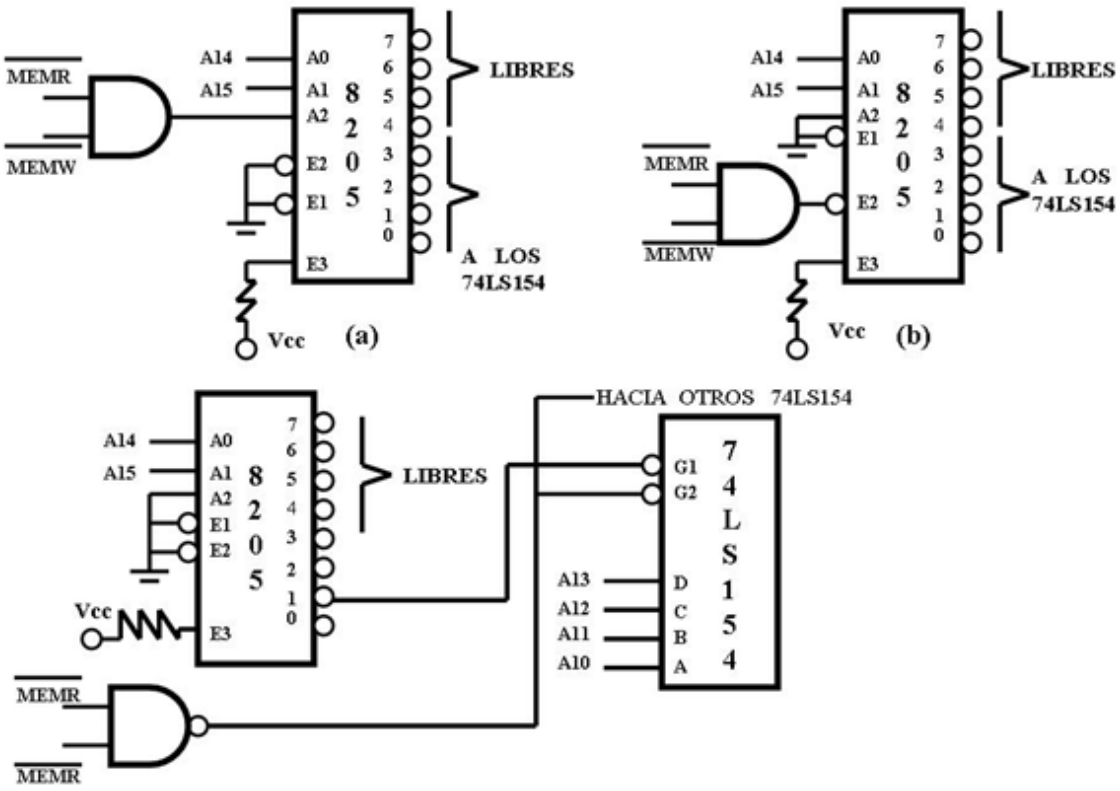
En el direccionamiento "absoluto" se debe usar las 16 líneas de dirección. Generalmente las memorias de las microcomputadoras utilizan memorias RAM y ROM (PROM o EPROM), y las ROM tienen las direcciones más bajas, comenzando con 0. Esto debido a que al encender el sistema de energía de la microcomputadora 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 10. Circuito para 64 módulos de 1 k 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 $\overline{\boldsymbol{MEMR}}$; para que las memorias conectadas a las salidas del 74LS154 se debe conectarlas a las entradas $\overline{\boldsymbol{CS}}$ 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.



10. Circuito para 64 módulos de 1K

Los circuitos de las figuras 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 $\overline{\boldsymbol{MEMR}}$, para que los 74LS154 se habiliten en ciclos de leer memoria.



11. Variantes de los decodificadores de la figura 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:

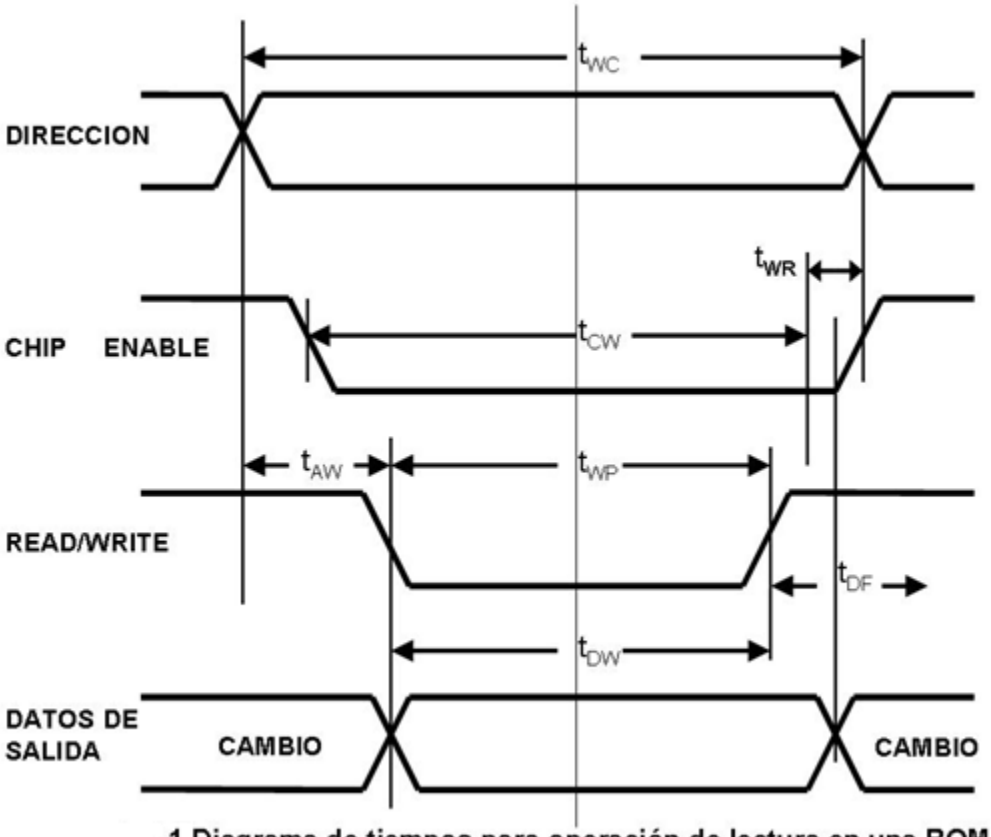
1. Se aplica una dirección a las entradas de dirección de la memoria ROM.
2. Se selecciona el circuito de la ROM activando sus entradas de selección de circuito (Chip Select CS o Chip Enable CE).
3. El contenido de la localidad de memoria seleccionada aparece en las salidas de datos de la ROM, después 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, 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 t_{CO} o tiempo de acceso de CS (Chip Select). Los parámetros, t_A y t_{CO}, se muestran en el diagrama de tiempos de la figura 1. Diagrama de tiempos par operación de lectura de ROM. 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, t_{CO} es generalmente menor que t_A. Esto se debe a que la lógica de selección (Chip Select) está conectada directamente a los buffers de salida.



1. 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), 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) 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 t_{OH} indica cuánto tiempo es todavía válido el dato de salida después de que la dirección ha cambiado. El 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), 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 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.

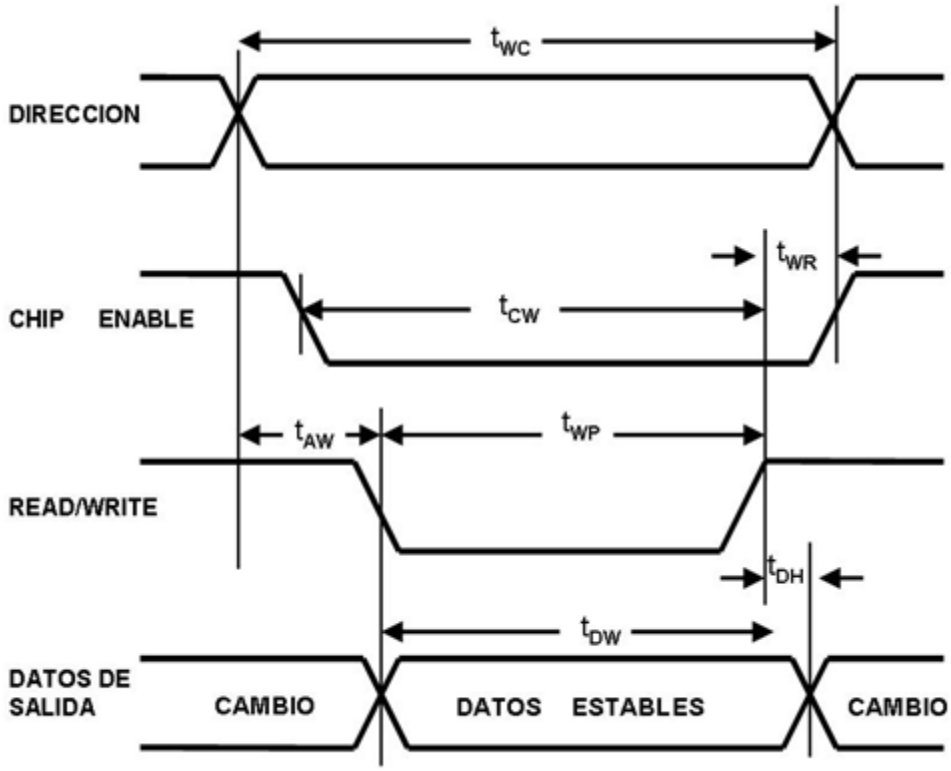
2. Se aplica el circuito de la RAM activando sus entradas de habilitación de chip (chip enable).

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 2 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 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.



2. 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), 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), 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), t_{WR}, indica cuánto tiempo debe 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), 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. Refrescamiento

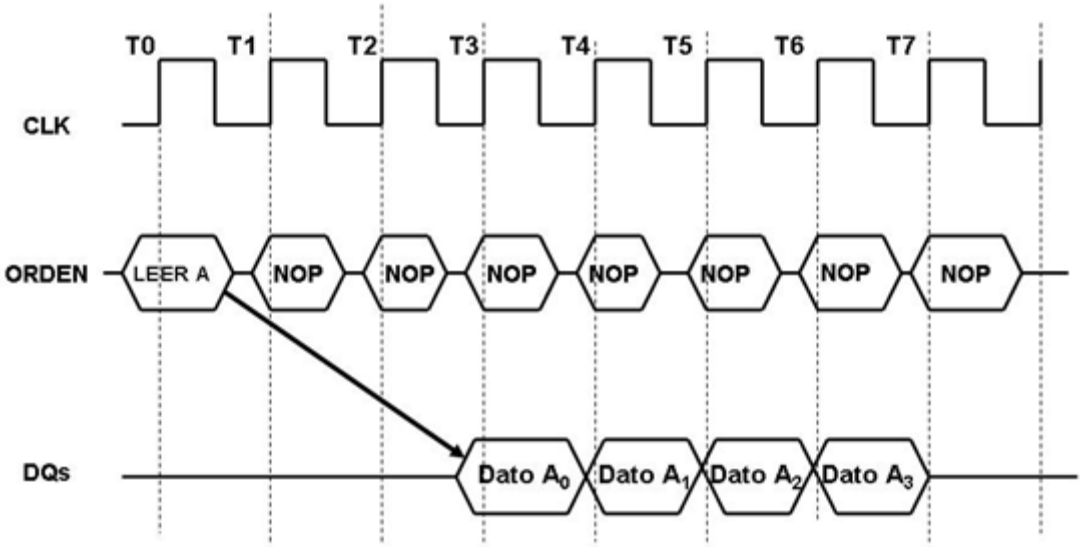
Las memorias RAM se pueden clasificar en memorias estáticas y dinámicas. Las memorias RAM dinámicas tienen la propiedad de que los datos almacenados decaen o se desvanecen espontáneamente y deben ser actualizados a intervalos regulares debido a que están fabricadas con la tecnología CMOS. Además los 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 3. Temporizador de una lectura… 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.

[**Nota**: es en Centroamérica donde ‘refreshment’ se ha traducido como ‘refrescamiento’, pero la ‘actualización’ (o ‘renovación’) son traducciones preferibles.]

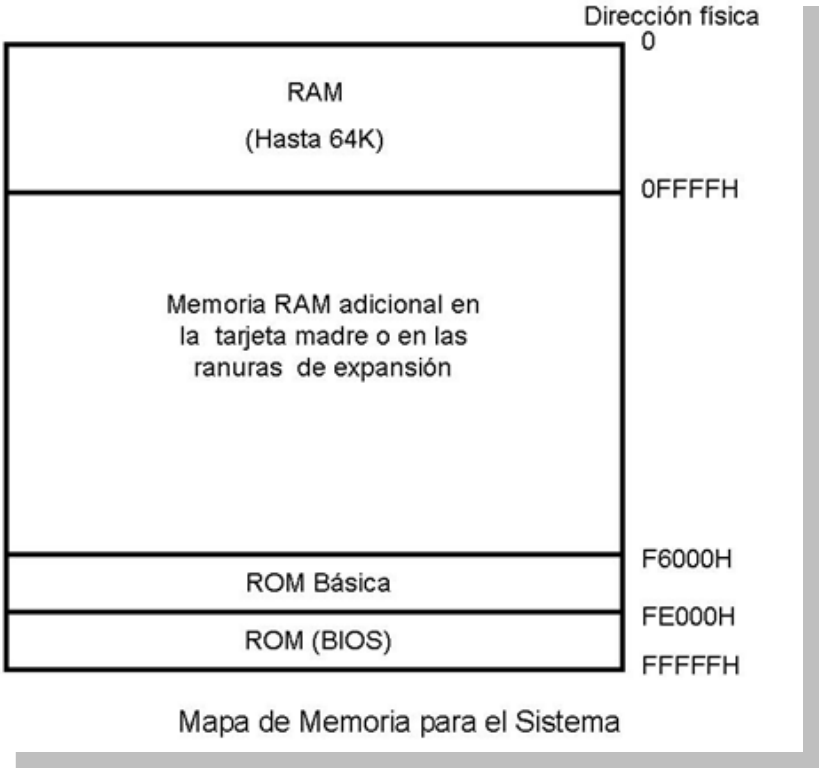


3 Temporización de una lectura de SDRAM (longitud de ráfaga =4, latencia de $\overline{CAS}$ = 2)

7.3. Mapa de memoria

Una computadora digital organiza su memoria RAM y memoria ROM en un Mapa de Memoria, 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), etcétera.

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 *Mapa de memoria para el sistema*.



Mapa de Memoria para el Sistema

En la figura *Mapa de memoria para el sistema,* 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.

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.

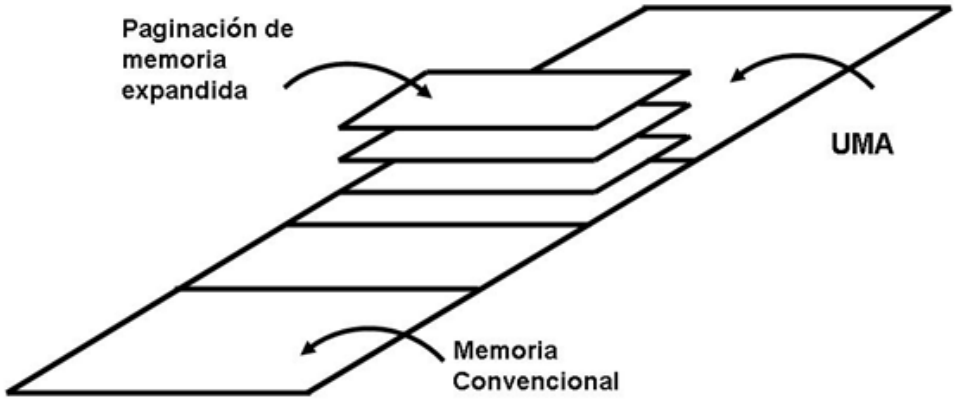
Memoria expandida

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 1. Memoria expandida) 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.

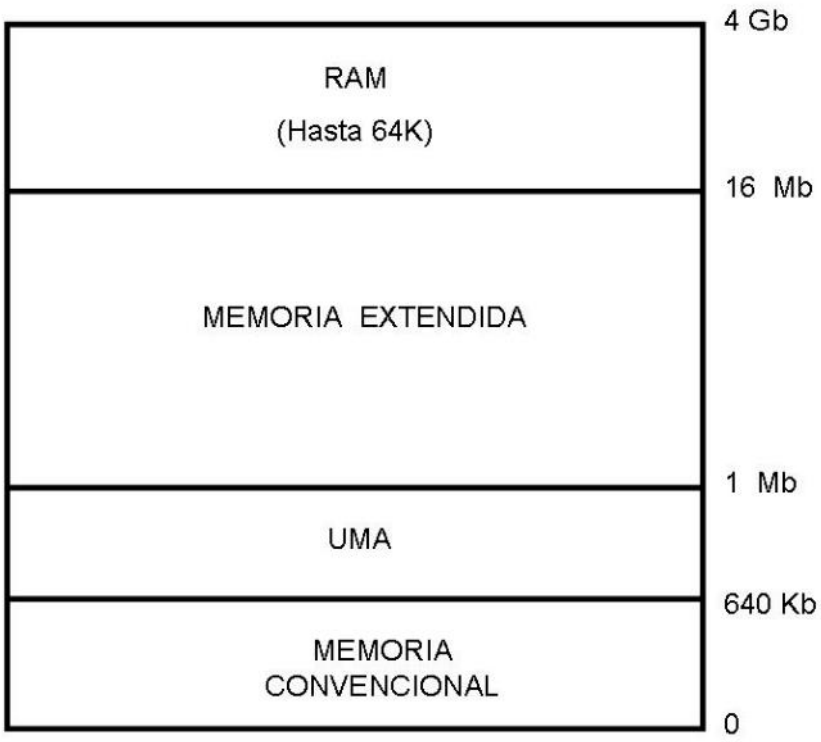
Como solamente se puede trabajar con 64K de información a la vez, es necesario copiar continuamente datos desde la memoria expandida (más de 1Mb) a la memoria superior y viceversa. Otra solución más rápida y eficiente es la llamada memoria extendida.



1. 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 2. Memoria extendida 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.



2. Memoria extentida

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

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

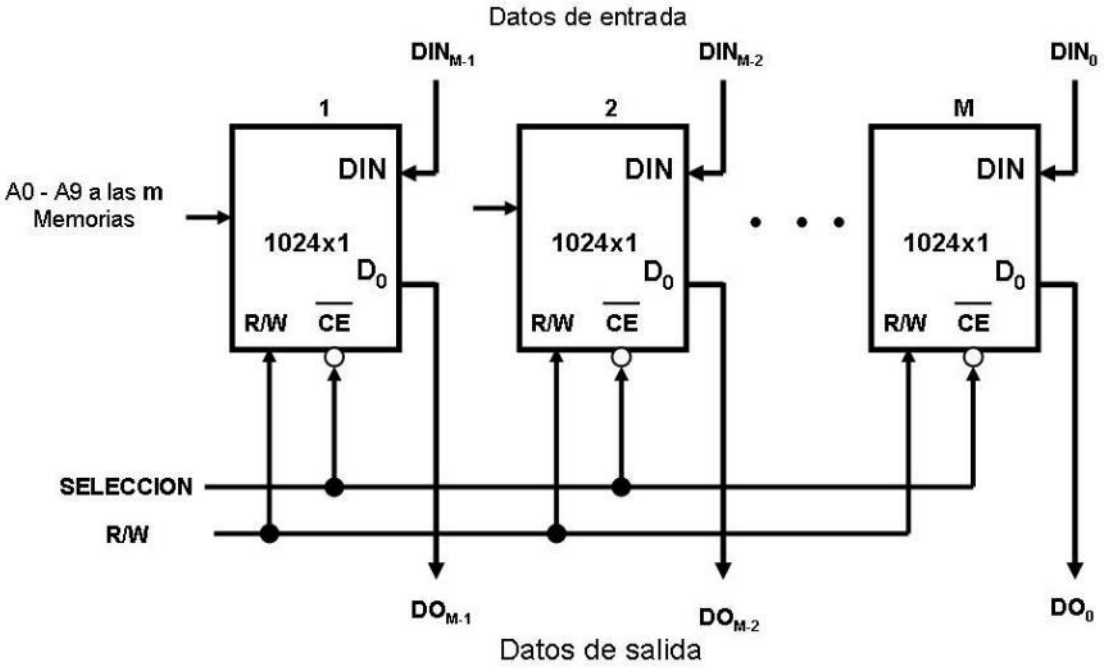
Son los primeros 640 Kb de memoria extendida. Esto es únicamente válido para Computadoras Personales que cuenten con memoria extendida.

7.3.1. 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 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 Memoria se 1024 x m bits formado a partir de circuitos de memoria de 1024 x 1.

Memoria se 1024 x m bits formado a partir de circuitos de memoria de 1024 x 1

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.

7.3.2. Tendencias tecnológicas de memorias (holograma, SSD, FLASH)

Demanda de aplicaciones

La memoria es un componente crítico en cualquier sistema de cómputo. En la memoria de trabajo se almacenan temporalmente tanto instrucciones, como direccionamientos y datos. Adicionalmente de acuerdo al manejo de los mapas de memoria donde se reservan ciertas direcciones para aplicaciones especificas de control, se establecen formas para hacer más eficiente el uso de memoria. Sin embargo éstas tienen sus límites operativos.

El incremento de servidores virtuales y bases de datos virtuales requiere de capacidad de almacenamiento adicional para los equipos que operan en estas condiciones. Por ejemplo para aplicaciones de e-commmerce o de servicios donde se operan una gran cantidad de transacciones, el tamaño insuficiente de memoria puede redundar en un funcionamiento lento del procesador, debido a la cantidad de hilos o procesos en ejecución y por la asignación de memoria para cada uno de ellos. Recordemos que si los requerimientos de memoria RAM rebasan la memoria instalada, se comenzará a utilizar memoria swap o hacer una paginación con memoria virtual en disco que es miles de veces más lenta que la RAM. Si los servidores rebasan un límite de operación de memoria RAM para usar memoria virtual, los servicios se vuelven lentos pudiendo llegar inclusive a dejar de operar. Por lo tanto el tamaño de la memoria para estos casos determina la cantidad de máquinas virtuales para un mismo equipo. Por ejemplo, podemos observar en las siguientes gráficas cómo cambia la cantidad de instrucciones

procesadas por minuto y la cantidad de máquinas virtuales soportadas dependiendo de la cantidad de memoria RAM instalada.

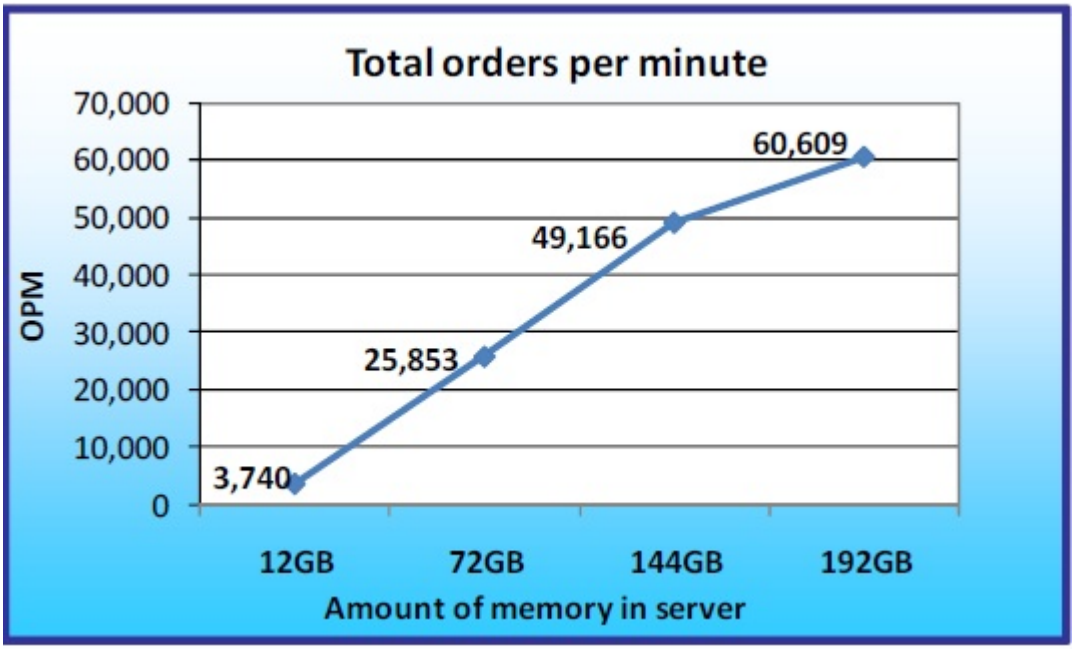


Imagen de la cantidad de instrucciones por minuto para tamaño de memoria variable (www.kingston.com)

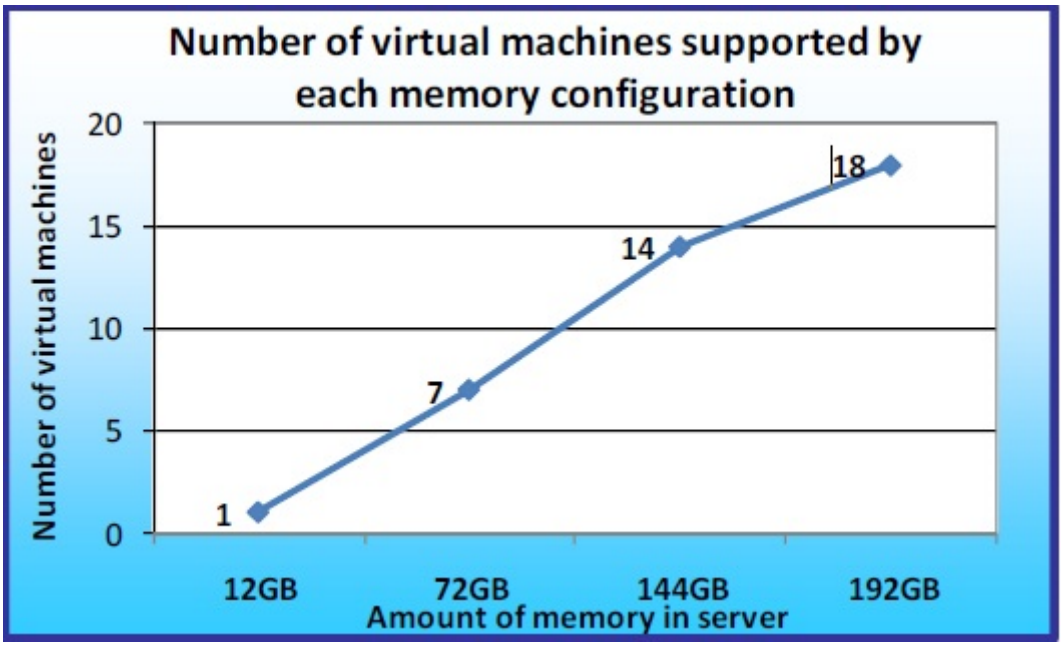


Imagen de la cantidad de máquinas virtuales dependiendo de la cantidad de memoria (www.kingston.com)

Para el caso de procesadores, el tamaño de memoria requerida está determinado por la cantidad de máquinas virtuales y por la cantidad de transacciones. Lo anterior establece una correspondencia entre los requerimientos de servicios en la Web y la cantidad de memoria instalada. Si partimos de este hecho, estaremos en una situación donde los requerimientos de memoria crecerán para satisfacer estas demandas crecientes. Una breve revisión del desarrollo de memorias nos puede dar la pauta para establecer tendencias en un futuro de las memorias RAM.

1970 Memorias de 64 bits de silicio.
1989 memorias modulares de chips montadas en un circuito impreso.
1999 memoria value RAM 1 GB genera estándares que permiten cambiar memorias sin cambiar circuitos.
2002 Aplicaciones multimedia DDR2. Memorias refrigeradas.

Por otro lado, la necesidad de cantidades mayores de memoria para aplicaciones gráficas sobre todo fotografías, audio y video como apreciamos en la tabla de almacenamiento de abajo, nos lleva a pensar en el uso de tecnologías que tiendan a considerar los siguientes aspectos en la fabricación de memorias:

- Cantidades mayores de almacenamiento.
- Tamaño reducido de los componentes y circuitos.
- Menor consumo de potencia.
- Tiempos menores de acceso (lectura y escritura)

Tamaños de memorias USB disponibles.

Almacenamiento Capacidad	Fotos**						Grabación de video***				Utilización del almacenamiento				
	6MP	8MP	10MP	12MP	18MP	24MP	9 Mbps (min.)	12 Mbps (min.)	24 Mbps (min.)	48 Mbps (min.)	Películas HD .MKV	.MP3 de música	Películas .MP4 comprimidas	Juegos para computadora	Copias de seguridad
2GB*	1,000	750	600	500	333	250	27	20	10	5	X	519	1	X	X
4GB*	2,000	1,500	1,200	1,000	667	500	53	40	20	10	X	1,038	3	X	X
8GB*	4,000	3,000	2,400	2,000	1,333	1,000	107	80	40	20	X	2,076	7	X	X
16GB*	8,000	6,000	4,800	4,000	2,667	2,000	213	160	80	40	1	4,151	14	X	X
32GB*	16,000	12,000	9,600	8,000	5,333	4,000	427	320	160	80	2	8,302	28	1	1
64GB*	32,000	24,000	19,200	16,000	10,667	8,000	853	640	320	160	5	16,605	56	3	2
128GB*	64,000	48,000	38,400	32,000	21,333	16,000	1,707	1,280	640	320	10	33,209	112	6	4
256GB*	128,000	96,000	76,800	64,000	42,667	32,000	3,413	2,560	1,280	640	21	66,418	224	12	9

Tabla de capacidad de almacenamiento de memorias.
(http://www.kingston.com/es/usb/storage_chart)

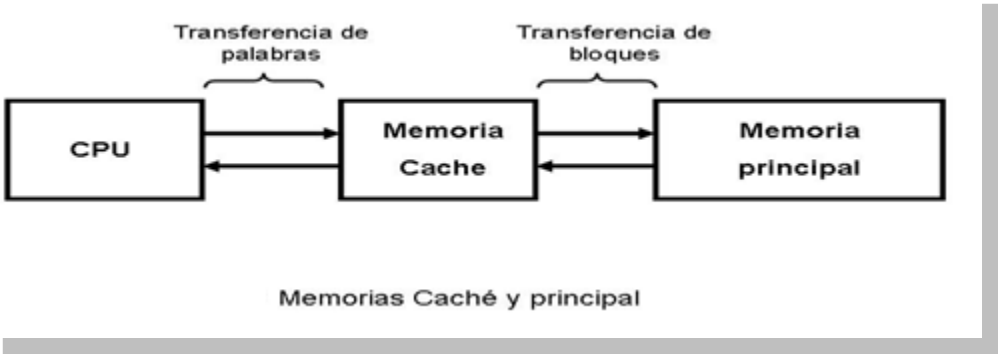
Como resultado de estas cuatro condiciones podemos encontrar en el mercado memorias por ejemplo del tipo Hyper X LoVo, cuyas características principales son: alto rendimiento y bajo consumo de energía. Estas memorias utilizan un difusor de calor, por lo que los requerimientos de enfriamiento reducen el consumo de energía de refrigeración.

7.4. Memoria caché

La memoria caché 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 caché razonablemente pequeño puede generar mejorías significativas en el rendimiento. Dado que la memoria caché 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 caché está situada entre el microprocesador y la memoria principal.

Funcionamiento

El objetivo de la memoria caché 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 Memorias Caché y principal. Hay una memoria principal relativamente grande y más lenta, junto con una memoria caché más pequeña y rápida. La memoria caché 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 caché. 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 caché 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 caché 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.



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 caché.

Finalmente, el propósito de la memoria caché 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 caché determina la frecuencia con que se utilizan los datos, transfiere los que se usan a menudo a la memoria caché y los elimina cuando identifica datos de uso menos constante. Los datos en la memoria caché 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 caché

A continuación se resumen los aspectos de diseño de la memoria caché y el cual se divide en las siguientes categorías:

- Tamaño de la caché
- Tamaño del bloque

Un **tamaño de la memoria caché** razonablemente pequeño puede tener un impacto significativo en el rendimiento. Otro aspecto relacionado con la capacidad de la caché es el tamaño del bloque: el cual es la unidad de datos que se intercambia entre la caché y la memoria principal.

Según el tamaño del bloque se incrementa desde muy pequeño 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 caché 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 caché 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 cien 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 y desventajas

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.

RESUMEN

En esta unidad se revisaron inicialmente los tipos de memorias de semiconductores entendiéndolas como arreglos de celdas de información. En las unidades anteriores vimos cómo se integran registros y contadores a partir de flip flops y circuitos lógicos combinacionales. Aquí se integraron los bancos de memorias a partir de arreglos de flip flops para el caso de memorias de semiconductores. Se explicó también la diferencia entre las memorias dinámicas y estáticas y cómo afectan la arquitectura de las computadoras al tener que incluir en el caso de las memorias dinámicas circuitos de actualización de la información. En el tema dos, se explicó el ciclo de memoria tanto para la escritura como para la lectura. En el tema tres, se revisó la forma de direccionar las celdas de memoria a partir de localidades y direcciones y cómo se construye un mapa de memorias de manera general y también de manera particular el almacenamiento de la información en una computadora atendiendo al tipo de procesos y de información. En el tema cuatro, se describió una memoria caché; los tipos y el uso que se le da en una computadora. Finalmente en el tema cinco se describió el uso de memoria virtual en disco duro y cómo es su manejo a partir del concepto de paginación, lo que le proporciona a la computadora una memoria adicional de trabajo a la memoria RAM instalada, con sus correspondientes limitaciones al estar apuntando a espacios de almacenamiento en disco duro, pero que para efectos de procesos se maneja como memoria virtual.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga (2010)	7 9	166 207-217 236-243
Mano (1986)	5 7	188-195 306-312
Stallings (2006)	4 7 8	104-109 150-172 272-288
Tanenbaum (2000)	3 6	141-154 406-421 404-429

Mano, Morris. (1986). *Lógica Digital y diseño de computadores*. México. Prentice Hall Hispanoamericana.

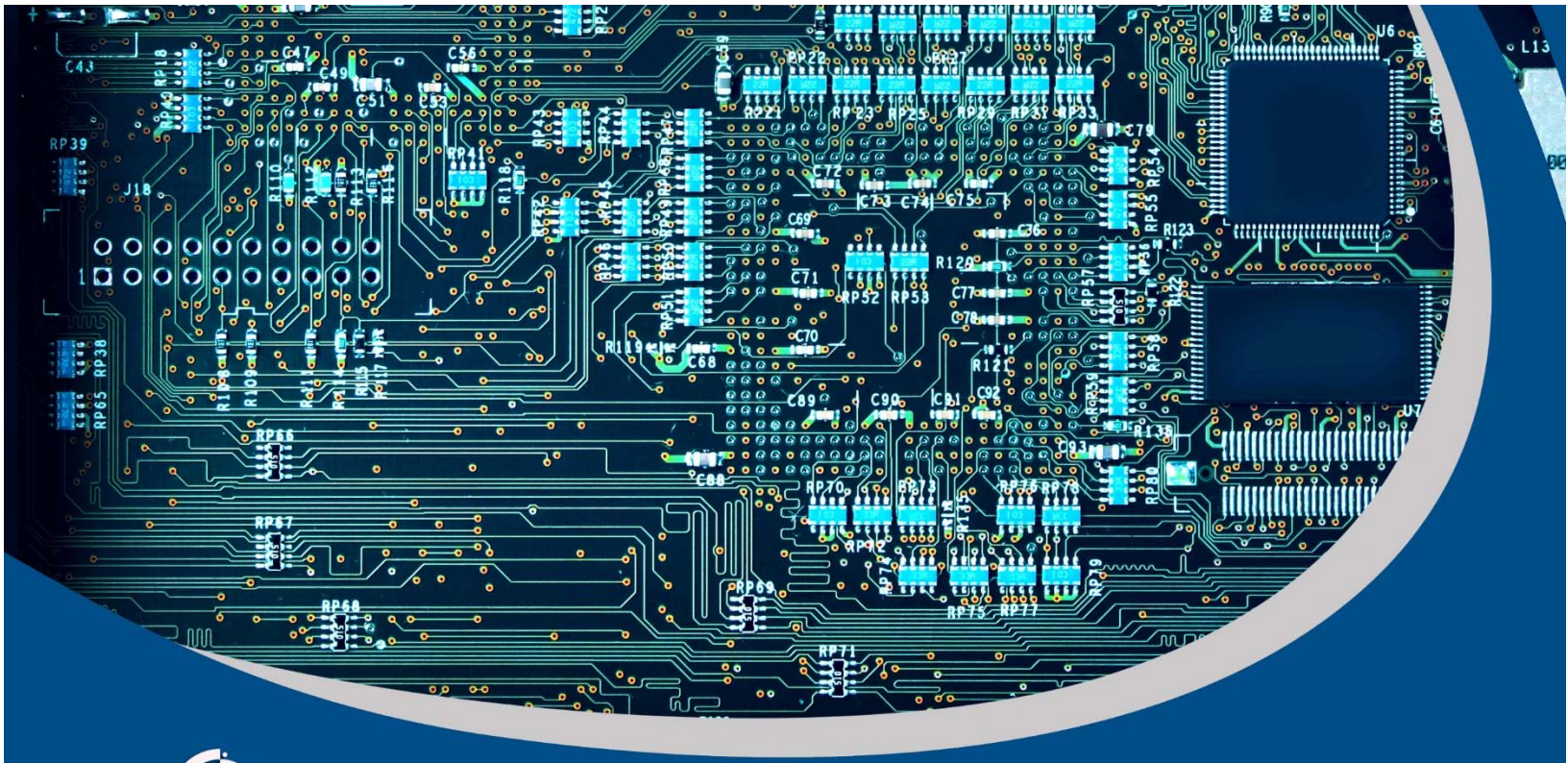
Quiroga, Patricia. (2010). *Arquitectura de computadoras*. México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores*. Madrid: Prentice Hall.

Tannenbaum, Andrew S. (2000). *Organización de computadoras. Un enfoque estructurado*. México: Prentice Hall.

UNIDAD 8

Unidades funcionales



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

OBJETIVO PARTICULAR

El alumno reconocerá el funcionamiento del sistema básico de entradas y salidas de una computadora, sus interrupciones, la configuración de sus puertos de comunicación y las características del almacenamiento de datos.

TEMARIO DETALLADO
(6 horas)

8. Unidades funcionales

- 8.1. Arquitectura de una PC con el esquema de von Neumann
  - 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

En esta unidad revisaremos las diversas unidades que intervienen en el funcionamiento de una computadora. Estas unidades están integradas con circuitos digitales que operan bajo los principios de la lógica digital y utilizan el código binario para el procesamiento de la información. La carga de trabajo en una computadora la realiza el microprocesador y de acuerdo a los avances tecnológicos, en modernas computadoras de escritorio existe la tendencia a integrar varios procesadores o núcleos que realizan procesos paralelos. Estos procesadores transforman secuencias binarias de datos y nos entregan datos procesados mediante puertos conectados a los dispositivos de entrada y salida. Sin embargo para poder llevar a cabo esta tarea se requieren de diversos dispositivos y tecnologías que interconectan los procesos para el flujo de los datos y para poder ingresar y presentar la información procesada. El flujo de datos en una computadora se realiza mediante los buses y los programas que los controlan.

Así mismo se requieren de “caminos cortos” que aceleren las transferencias de información desde un dispositivo a otro, generalmente la memoria, sin intervención del microprocesador, a estos canales de acceso rápido se les denomina Acceso Directo a Memoria (DMA por sus siglas en inglés).

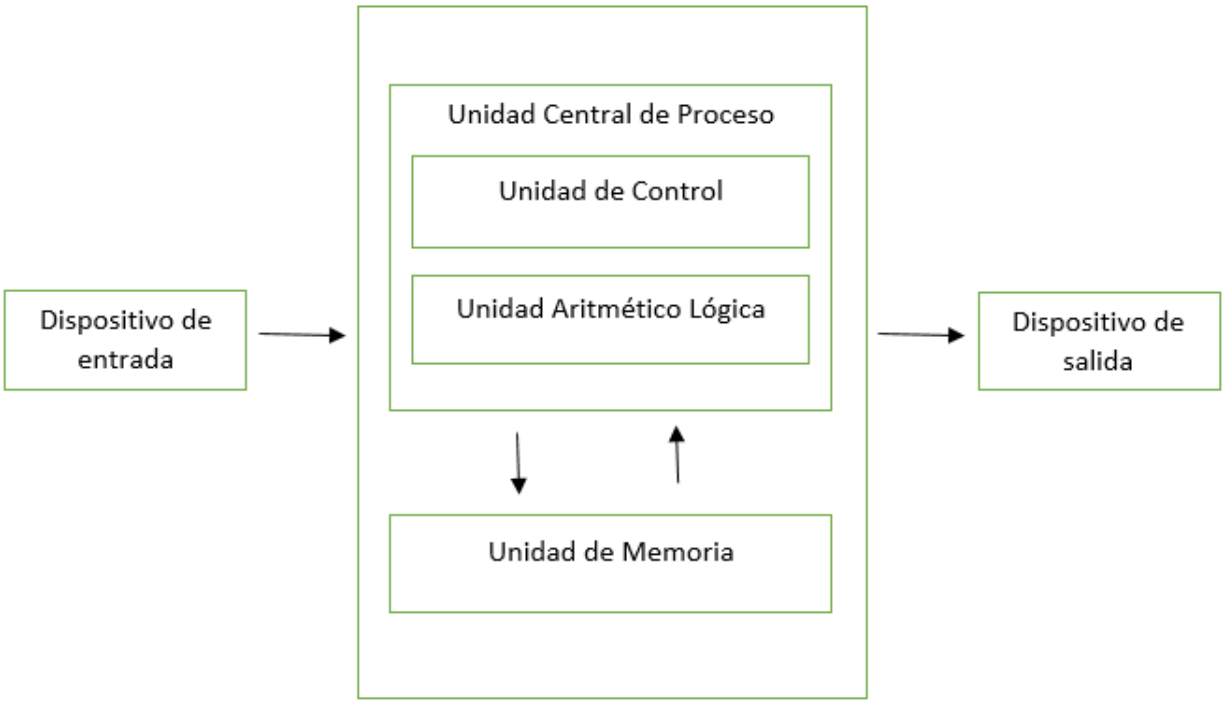
Finalmente estudiaremos cómo es el proceso de encendido de una computadora, qué sucede desde que encendemos el botón de la computadora hasta que se despliega la pantalla con las amigables ventanas de control o el cursor en caso de una máquina con interfaz en modo texto.

8.1. Arquitectura de una PC bajo el esquema de von Neumann

La arquitectura von Neumann, es una arquitectura desarrollada por el físico y matemático John von Neumann en 1945. Von Neumann describe una arquitectura que consta de los siguientes componentes:

- Una unidad de procesamiento (que contiene una unidad aritmético lógica y registros del procesador).
- Una unidad de control. (Que contiene un registro de instrucciones y contador de programa).
- Memoria principal para almacenar datos e instrucciones.
- Memoria secundaria.
- Dispositivos de entrada.
- Dispositivos de salida.

El diseño de una arquitectura von Neumann es el que actualmente utilizan las computadoras modernas.



Elaboración propia.

Una máquina von Neumann, al igual que prácticamente todas las computadoras modernas, consta de cuatro componentes principales:

1. Dispositivo de operación (DO), que ejecuta instrucciones de un conjunto de instrucciones, sobre la información almacenada.
2. Unidad de control (UC), que organiza la implementación consistente de algoritmos de decodificación de instrucciones que provienen de la memoria del dispositivo. Por lo general, el DO y la UC conforman una estructura llamada Unidad Central de Proceso, el cual en una computadora es el microprocesador.

3. Memoria del dispositivo, es un conjunto de celdas con identificadores únicos, que contienen instrucciones y datos, esto estaría representado por la memoria RAM.
4. Dispositivo de Entrada/Salida, que permite la interacción con la computadora, un dispositivo de entrada sería un teclado, un micrófono o un ratón, y un dispositivo de salida sería un monitor o una bocina.

Fuente:
https://www.genbetadev.com/actualidad/como-funciona-la-computacion-actual-funcionaiento-de-la-arquitectura-de-von-neumann (Fecha de consulta: 19/05/2017).

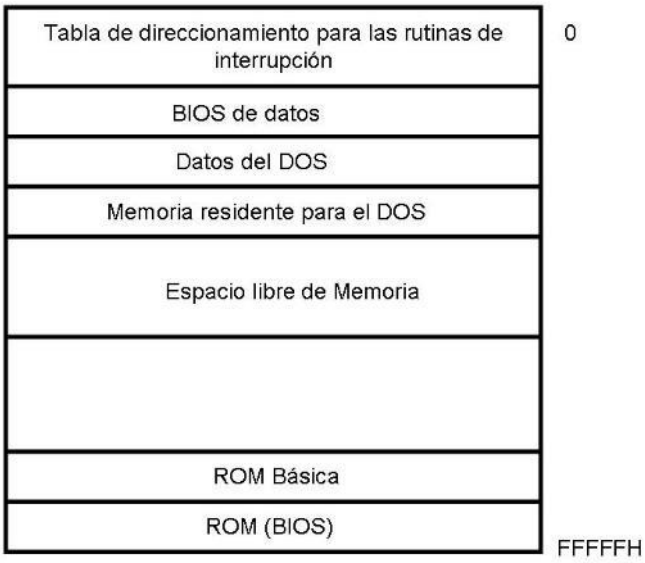
8.1.1. El BIOS

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, etcétera.

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 Ubicación del BIOS en el Mapa de Memoria.



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 Direcciones de entrada/salida.

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		

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

Ruta 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.

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.

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. (Wikipedia: Interrupción)

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 la microcomputadora no tiene que estar preguntando si un periférico tiene datos o está lista para recibir datos, es el periférico con la interrupción quien 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) 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 Instrucciones RST 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 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 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 Interrupt). 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
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. (¿Que son los canales DMA?)

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 siguiente figura Diagrama a bloques del 8237 de Intel 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 datos DB0- DB7 se convierten en las líneas de dirección A8-A15 durante la transferencia por DMA.

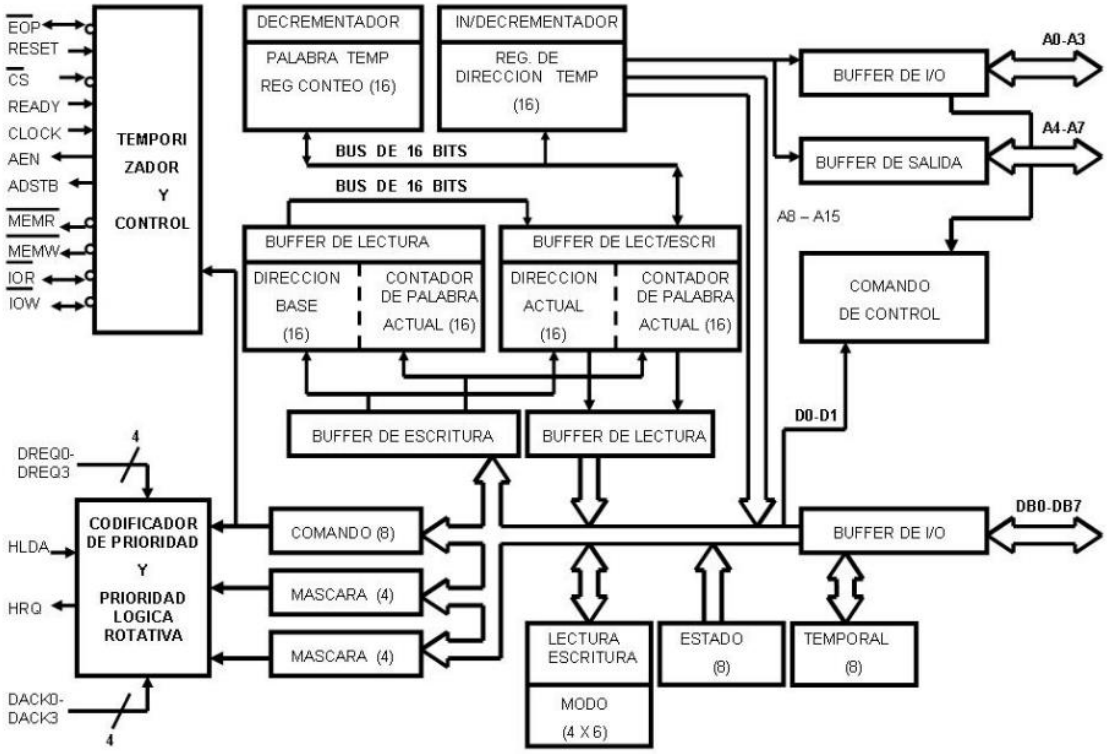


Figura Diagrama a bloques del 8237 de Intel

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. (Jiménez, 2006)

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 por 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 disminuye (dependiendo de si el bit 5 de la palabra de modo del canal es 0 ó 1), y la cuenta actual disminuye. 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 disminuya a 0 los registros actuales se pueden recargar automáticamente si el bit 4 del registro de modo del canal lo 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 cuáles 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 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 actualización 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 interfazs 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 Direcciones del DMA 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. (Véase, Jiménez, 2006)

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
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. 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.

Interfaz programable de E/S en paralelo 8255

La interfaz 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 interfaz 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 Diagrama lógico de la 8255.

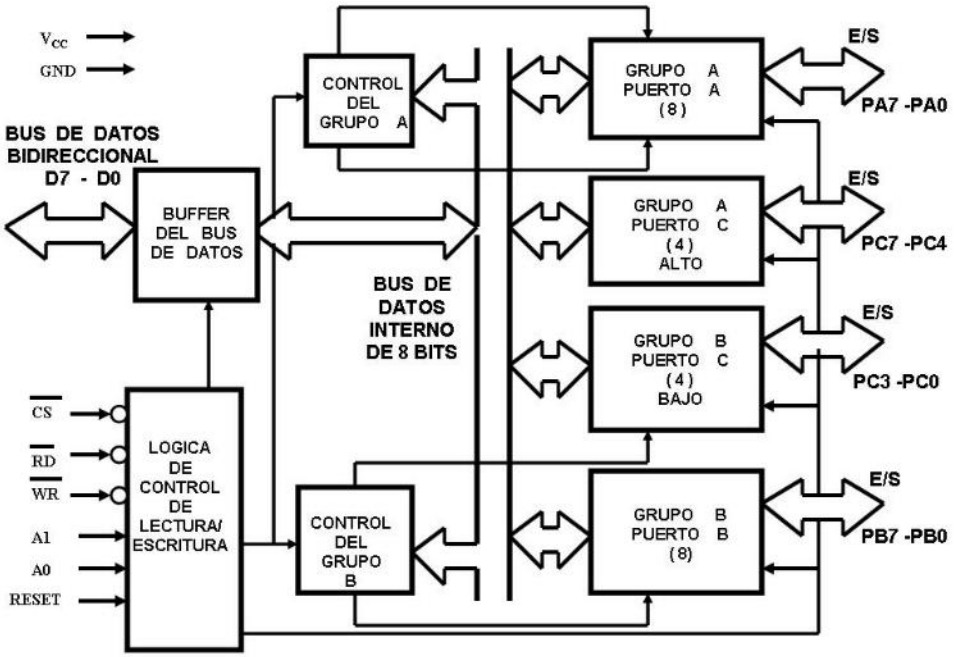


Figura. Diagrama lógico de la 8255

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).

$\overline{CS}$ Selector de Circuito Integrado

Un nivel bajo en esta entrada habilita la comunicación entre el 8255 y el microprocesador 8085A.

$\overline{RD}$ 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.

$\overline{WR}$ 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 $\overline{RD}$y $\overline{WR}$, 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 Operación Básica del C. I. 8255. 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. 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, proporcionando la configuración funcional del C.I. 8255, como se muestra en la siguiente figura Formato de definición del modo. 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)

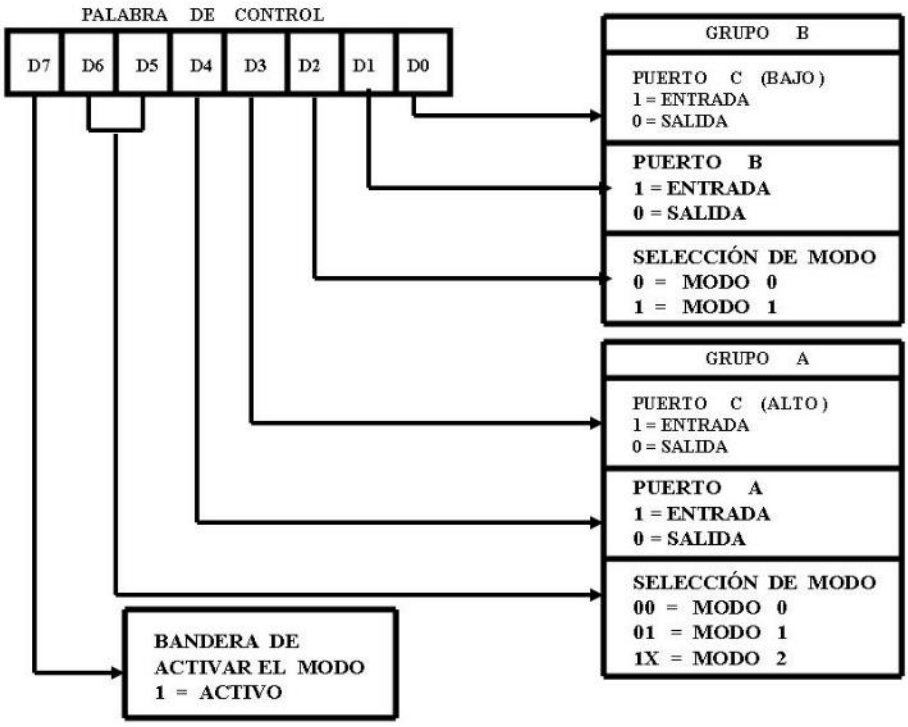


Figura Formato de definición del modo

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 anterior figura Formato de definición del modo 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 Definiciones de modo básico ilustra las definiciones de los puertos con los tres modos de operación.

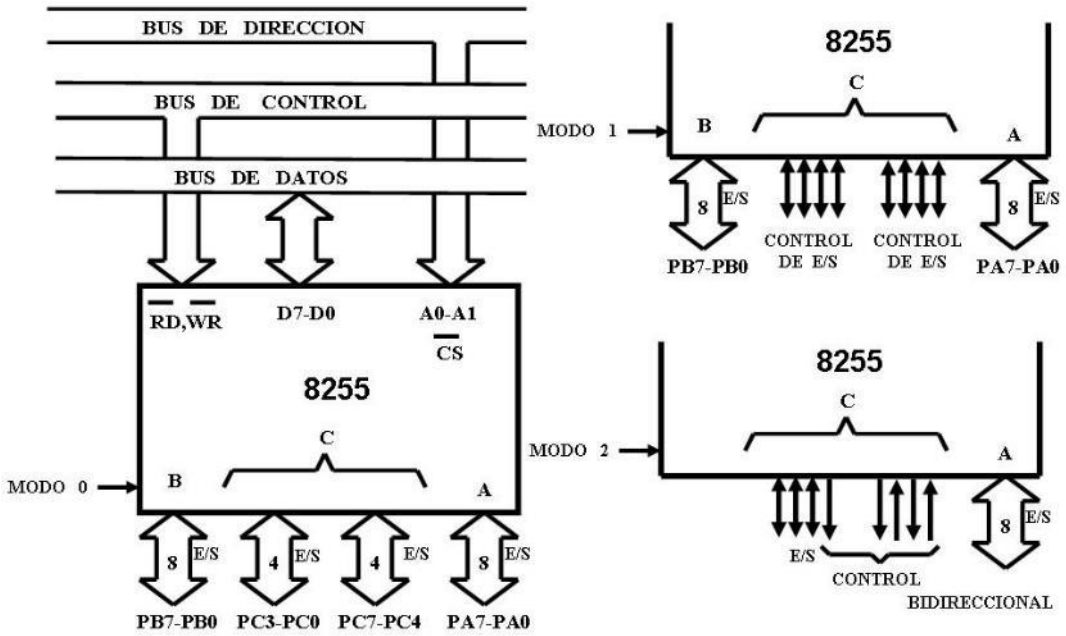


Figura. Definiciones de modo básico

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 Palabra de control y configuración en el Modo 2 ilustra la palabra de control y la configuración del puerto A en el modo 2.

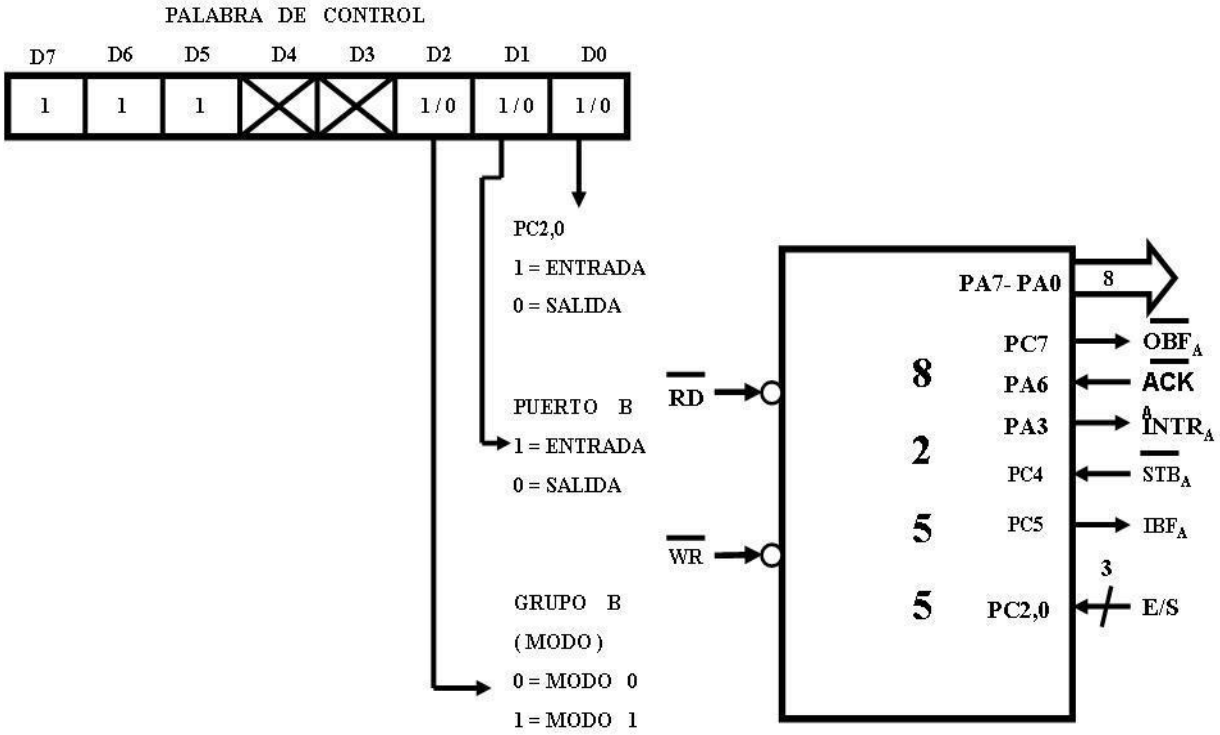


Figura. Palabra de control y configuración en el Modo 2

El modo 2 no utiliza las terminales PC0, PC1 y PC2, por lo que el puerto B se puede programar en el modo 0 o en el modo 1, la tabla Combinaciones con el modo 2 ilustra las combinaciones con el modo 2.

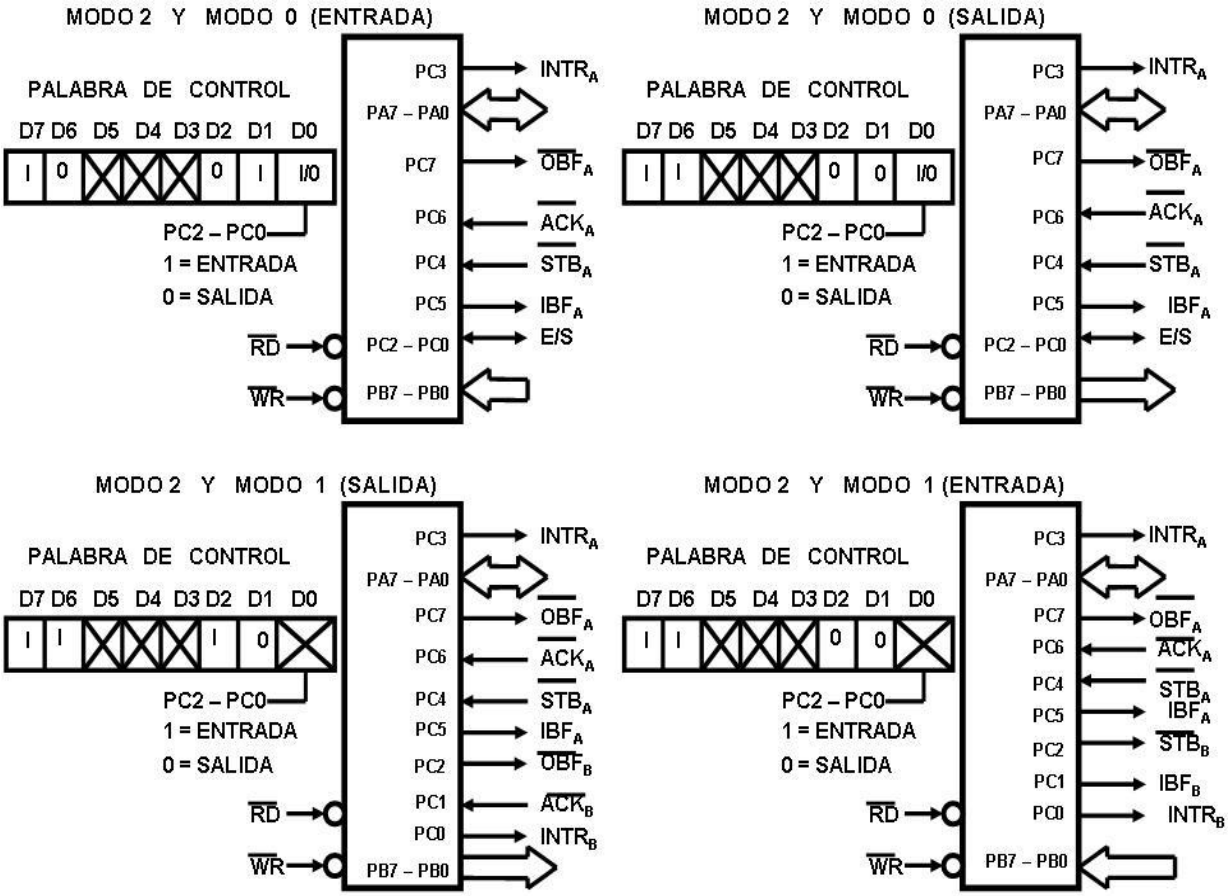


Tabla Combinaciones con el modo 2

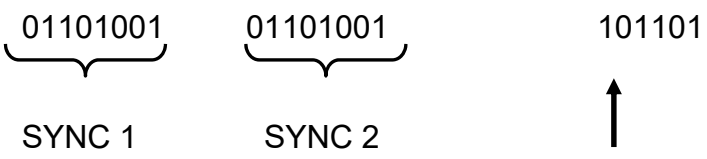
Puerto serial (Protocolo de Entrada y Salida Serie)

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

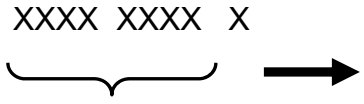
La característica principal de la transferencia de datos en serie en forma síncrona es que la corriente de datos debe ser continua. Habiendo establecido una vez la velocidad de baudios de transferencia de datos serie, el dispositivo transmisor debe transmitir un bit de datos en cada pulso de reloj, y 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 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:



8 bits de datos

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

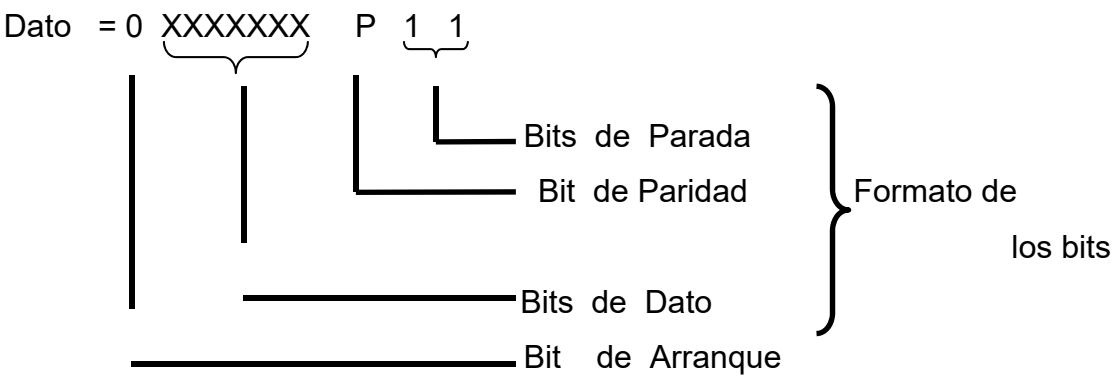
los datos serie que recibe tratando de encontrar en el flujo de datos serie el patrón estándar SYNC. Si su protocolo se llama por un solo carácter SYNC, entonces el dispositivo receptor comenzará a interpretar los datos en cuanto encuentre un carácter SYNC. El protocolo también puede esperar por dos 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. 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.

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. (Mateos, 1998)

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 Transferencia de datos en forma Serial. 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.

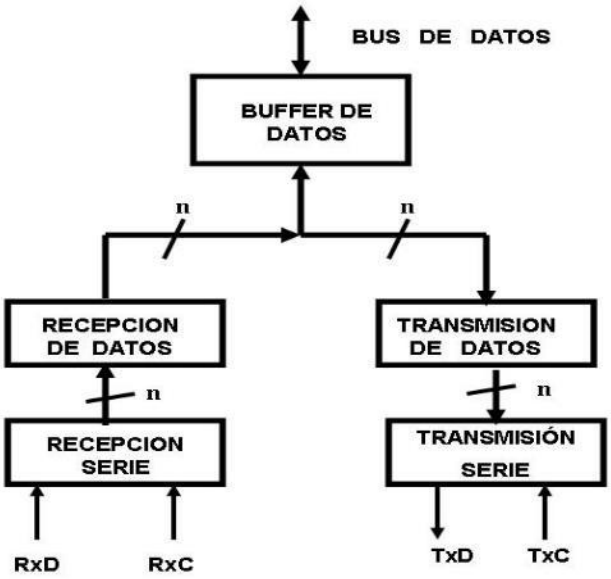


Figura. Transferencia de datos en forma Serial

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 Transferencia de datos en forma Serial.

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 Transferencia de datos en forma Serial.

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 interfaz de E/S serie

Dadas las muchas opciones de la interfaz 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 Parámetros de E/S serie se identifican los parámetros que se deben seleccionar bajo control del programa para cada tipo; además, 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. Parámetros de E/S serie

Condiciones de error de E/S serie

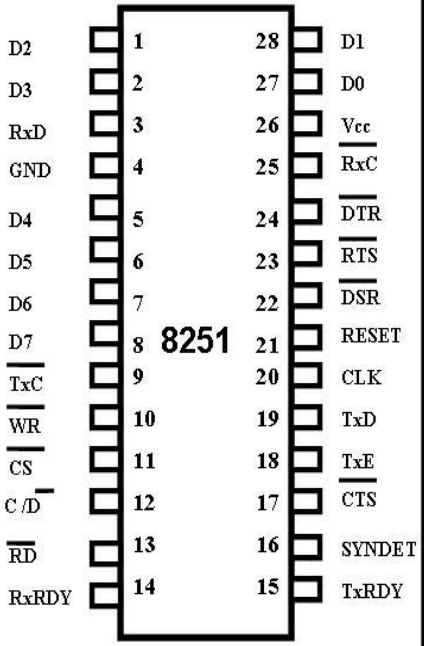
Las señales de entrada cuyos niveles deben ser posibles de leer son:

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 interfaz de un dispositivo de E/S serie aborte las operaciones. El error se reporta en el "Registro de Estados" y las operaciones continúan.

Interfaz 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 las siguientes figuras:



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. Configuración de CI 8251

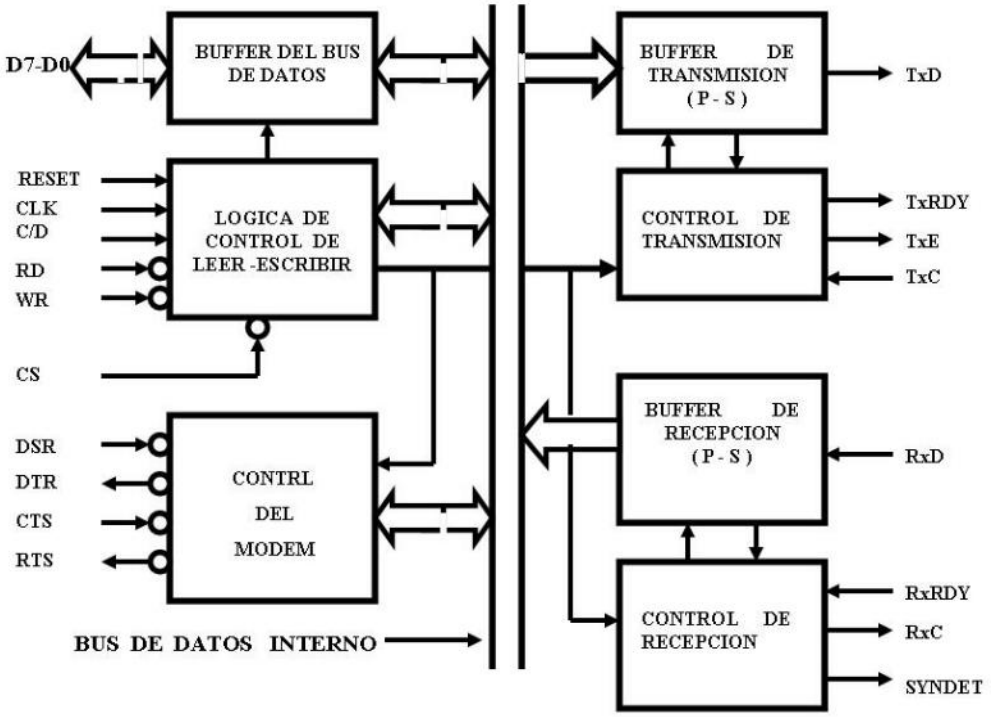


Figura 8 Configuración del CI 8251 (Continuación)

Las señales se pueden dividir en cuatro categorías:

1. Control e interfaz con la CPU
2. Entrada serie
3. Salida serie
4. Control del módem

La figura Interfaz del Puerto Programable Serial a) 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.

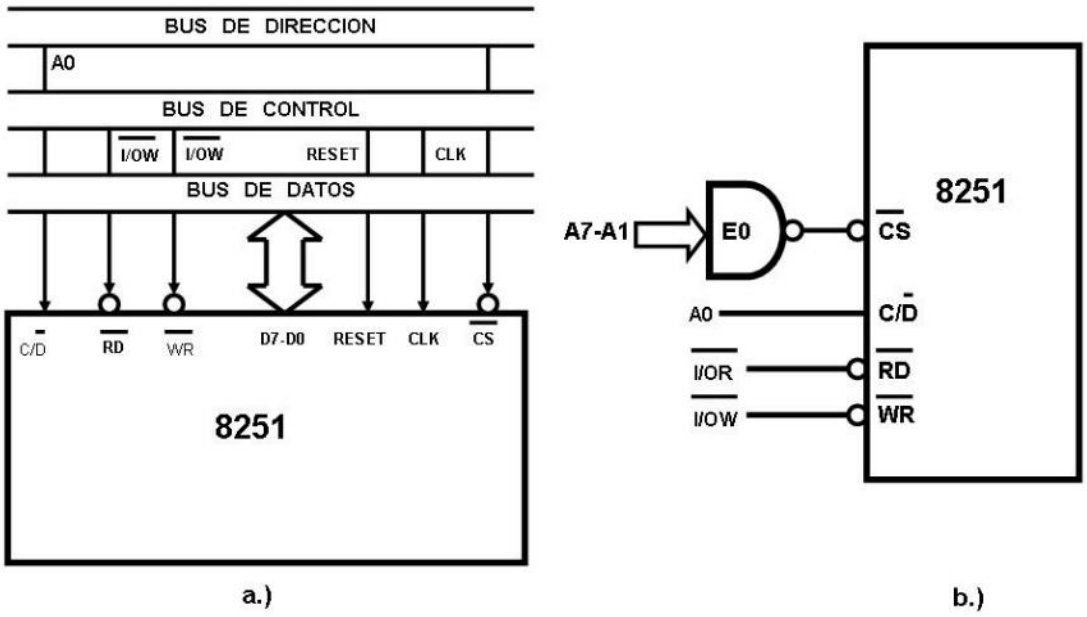


Figura Interfaz del Puerto Programable Serial

a) Conexión del 8251 al Bus del Sistema 8085
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 Selección del tipo de transferencia muestra las combinaciones de estas cuatro señales y la anterior figura Interfaz del Puerto Programable Serial ilustra la configuración para la selección de la interfaz. 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 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 siguiente figura Diagrama del flujo de datos. 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, $\overline{\mathbf{CS}}$ 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.

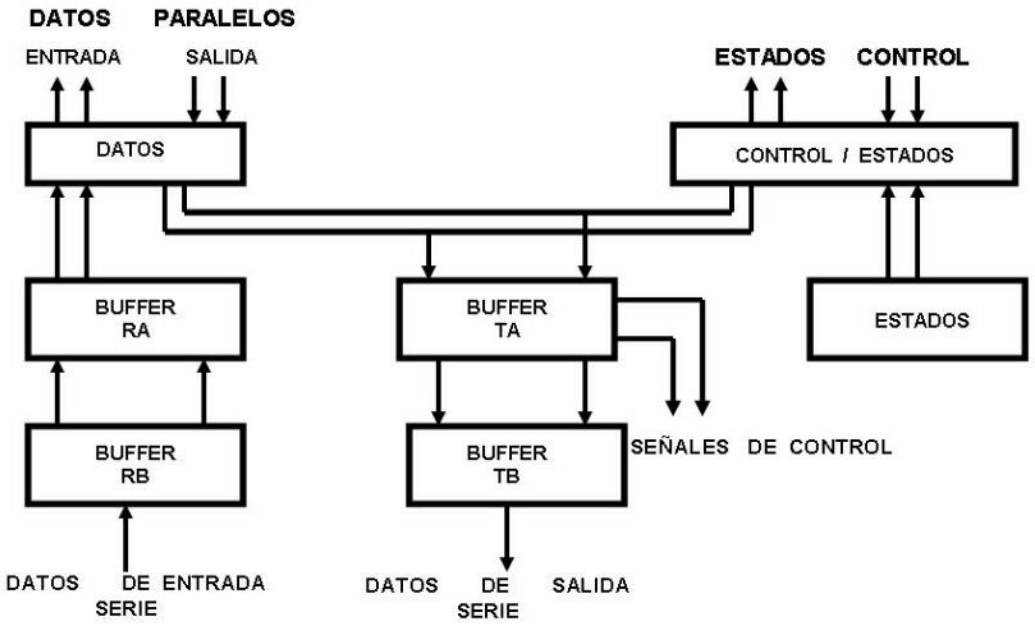


Figura. Diagrama del flujo de datos

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-overrun).

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 Formato para Lectura de los Estados, del Registro de Estado se pone siempre que el buffer TA esté vacío.

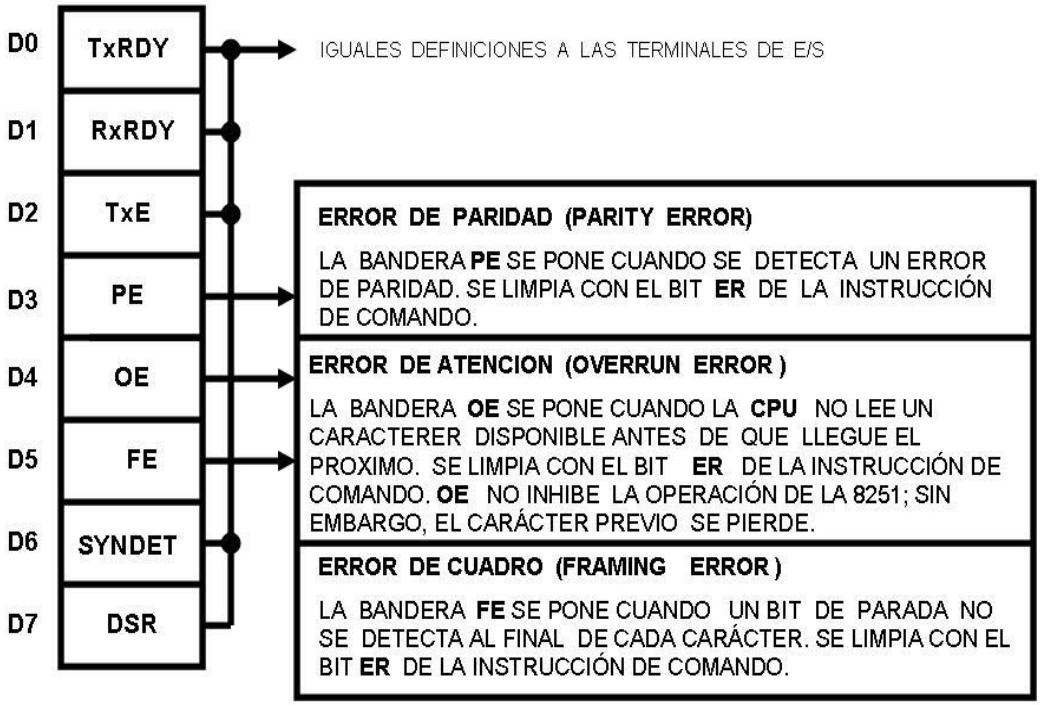


Figura Formato para Lectura de los Estados

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 Diagrama de tiempos de transmisión asíncrona ilustra el diagrama de tiempos durante la transmisión asíncrona de datos serie.

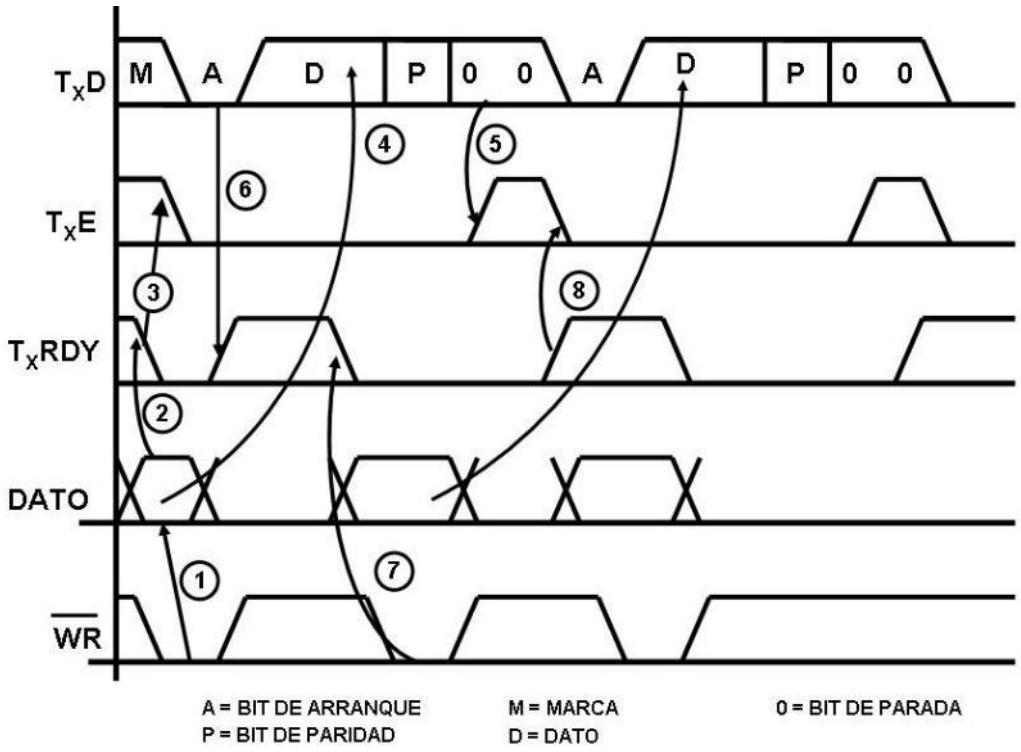


Figura. Diagrama de tiempos de transmisión asíncrona

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 si 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 Diagrama de tiempos de recepción asíncrona 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 verificar 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.

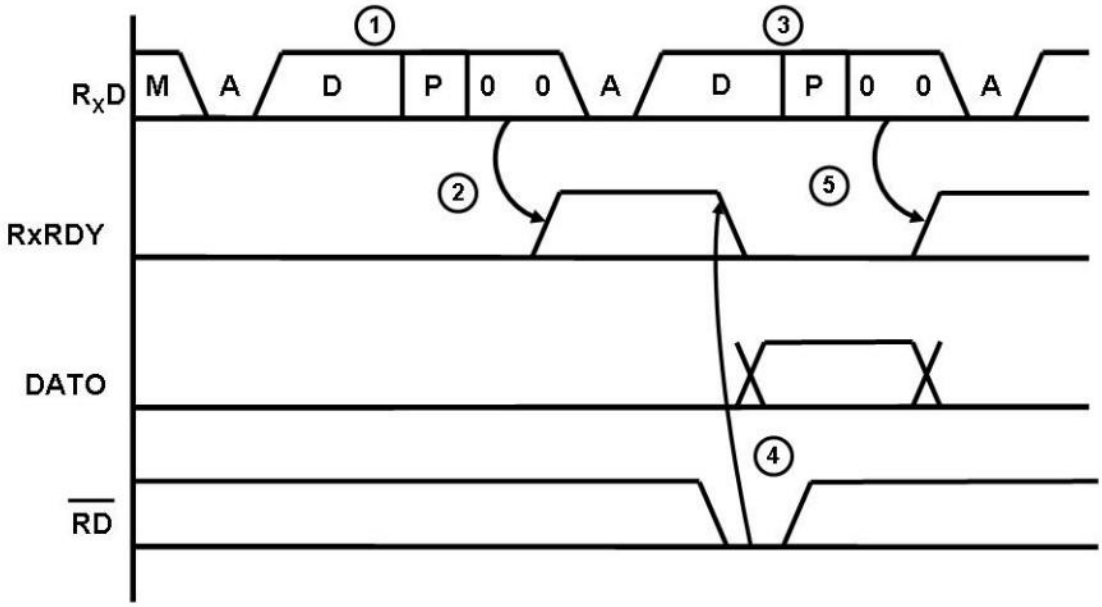


Figura Diagrama de tiempos de recepción asíncrona

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 a 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 Formato de Instrucción de Comando., 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.

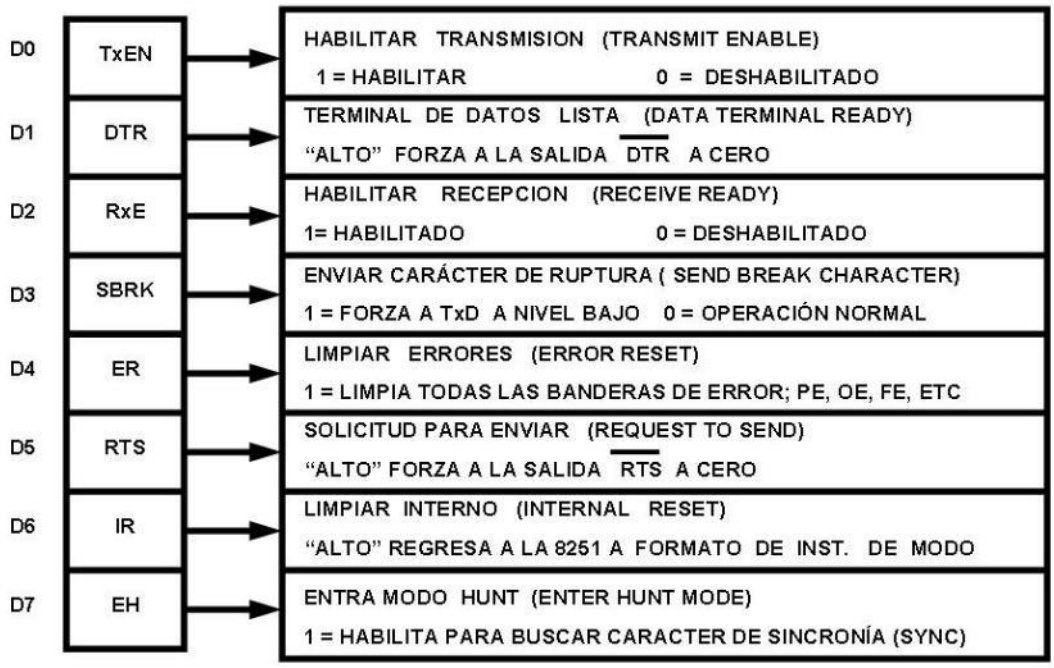


Figura Formato de Instrucción de Comando

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 (ver la anterior figura Formato de Instrucción de Comando) 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 siguiente figura Secuencia de la programación de las instrucciones ilustra la secuencia de la instrucción de Modo y de Comando.

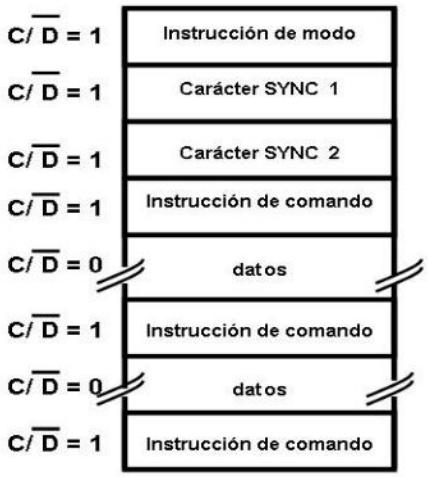


Figura Secuencia de la programación de las instrucciones

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 OVERRUN 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 Formato de la Instrucción de Modo Asíncrono 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 $\overline{\mathbf{RxC}}$.

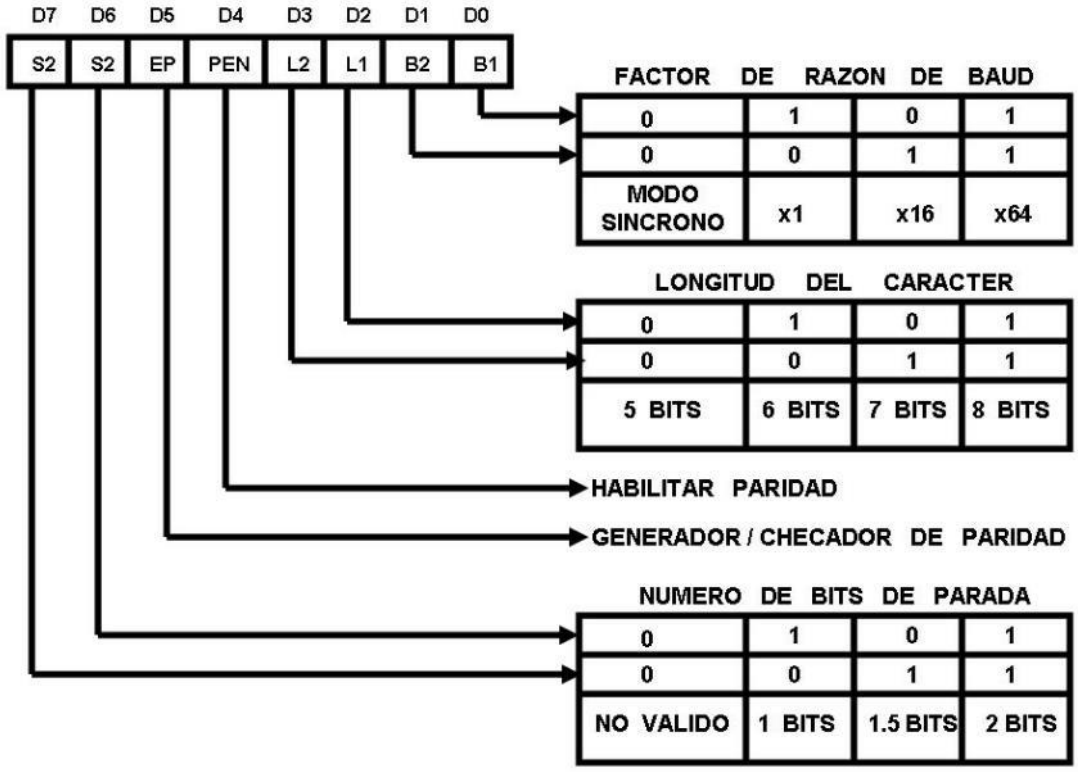


Figura Formato de la Instrucción de Modo Asíncrono

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 siguiente figura Modo Asíncrono 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.

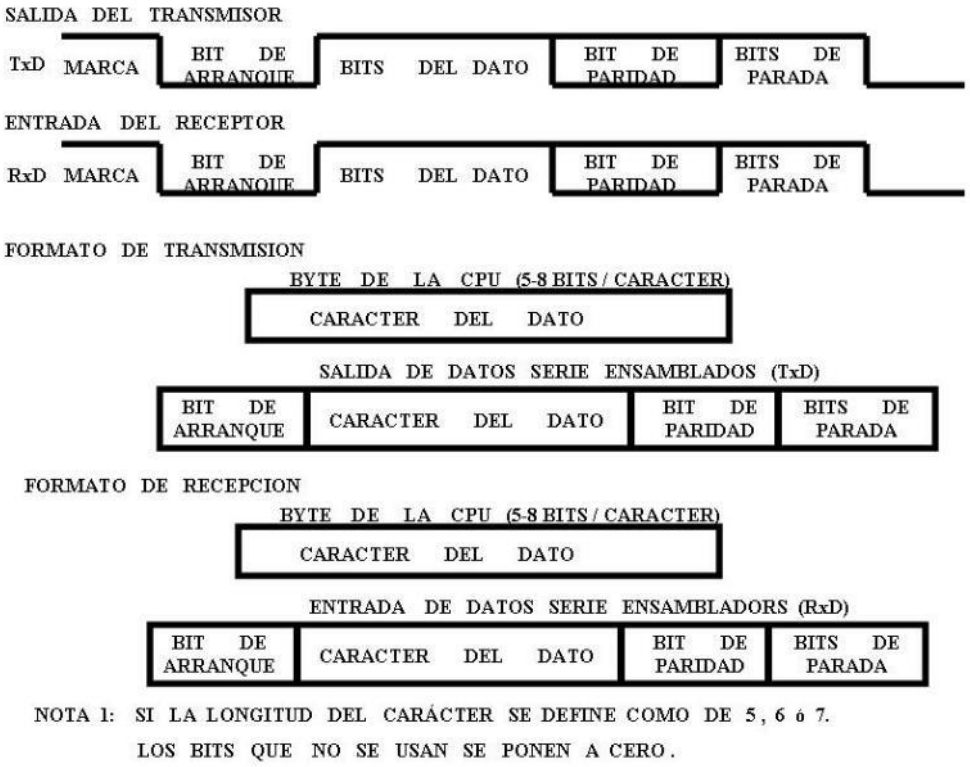


Figura Modo Asíncrono

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 anterior figura 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 Formato para Lectura de los Estados 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, que 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 va 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.

RESUMEN

En esta unidad se revisaron las principales partes que integran tanto al CPU internamente como los dispositivos de almacenamiento externo, es decir los discos duros. Las unidades anteriores nos han proporcionado herramientas para comprender características, parámetros y propiedades con las que identificamos estos dispositivos. Hemos revisado el proceso de encendido de una computadora y comprendido el papel de la información almacenada tanto en las memorias ROM como en las RAM y cómo podemos obtener esta información de las computadoras en funcionamiento. Es importante hacer notar que lo revisado en este capítulo es solo una introducción que busca establecer la arquitectura básica de una computadora partiendo desde la forma como codifica la información hasta procesos de transformación y almacenamiento de la misma y considerando la parte teórica vinculada con la tecnológica que hacen posible la existencia de las computadoras como las conocemos actualmente.

BIBLIOGRAFÍA

SUGERIDA

Autor	Capítulo	Páginas
Quiroga (2010)	8 8 13	Interrupciones 188-196 BIOS 192-196 DMA 300-316
Stallings (2006)	3 6 7	Interrupciones 63-75 Memoria externa 175-204 DMA 229-234
Tanenbaum (2000)	5	DMA 357-358 Interrupciones 379-383
N/A	Organización física de los discos	aquí

Mano, Morris. (1986). *Lógica Digital y diseño de computadores.* México: Prentice Hall Hispanoamericana.

Quiroga, Patricia. (2010). *Arquitectura de computadoras.* México: Alfaomega.

Stallings, Williams. (2006). *Organización y arquitectura de computadores.* Madrid: Prentice Hall.

Tanenbaum, Andrew S. (2000). *Organización de computadoras. Un enfoque estructurado*. México: Prentice Hall

Plan 2012

2016

actualizado